



Semi-Automatic Assessment of Basic SQL Statements

Aisha Nasser Saif AL-Salmi

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of
Doctor of Philosophy Degree of Loughborough University

August 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

AISHA NASSER SAIF AL-SALMI
August 2018

Acknowledgments

I am Thanking Allah for everything has blessed me. (Alhamdulillah (All praise be to Allah))

I would like to give special thanks to; my supervisors, Professor Eran and Dr. Shaheen for their continual support and encouragement throughout my Ph.D work. In addition, I would like to thank both Dr. Firat Batmaz and Dr. Gerald Schaefer, for their support and supervision in my last four years. I would also like to thank my second marker, Professor Ray Dawson, for providing key insights into the project. My grateful thanks to Professor. Sergey Saveliev; Associate Dean Research (Science); at Loughborough University for his kindness help to overcome all difficulties during my studies. I would also like to thank the Computer Science Department, IT Services, Mathematics Learning Support Centre and Library at Loughborough University for all the facilities and support.

I express my sincere gratitude to; Ministry of Manpower and Ministry of Higher Education in Sultanate of Oman for making the Ph.D. scholarship possible. I was thrilled to learn of my selection for this honour and I am deeply grateful of continues support. In addition, I would like to thank Mrs. Finoon Saleh, Academic Advisor from Cultural Attaché Office, Embassy of the Sultanate of Oman; for her kindness help to overcome all difficulties during my studies.

I would like to thank also the people who assisted me in my research either by opinion or undertaken the experiment in their Colleges. Dr. Huda AL-Shuily from Higher College of Technology, Muscat, Oman. Dr. Smitha Sunil Kumaran Nair from Middle East College, Muscat, Oman. Mr. Hamood AL-Hadhrami and Mr. Manish from Modern College for Business and Science, Muscat, Oman. Mr. Noor Khan from Alraffd Company, Muscat, Oman.

Finally, I would like to thank all the members of my family as I would never finish this thesis and I would never find the courage to overcome all difficulties during this work without them. I would like to thank; my husband “Ali AL-Maani” and my dearest son “Qais AL-Maani” for their support and motivation. Enormous thanks to my lovely family (Father, Brothers, Sisters, Sister-in-laws, Brother-in-laws, Nieces and Nephews) for their understanding and supports. I would like to thank my friends who have always given their full support throughout my time spent at university.

Abstract

Learning and assessing the Structured Query Language (SQL) is an important step in developing students' database skills. However, due to the increasing numbers of students learning SQL, assessing and providing detailed feedback to students' work can be time consuming and prone to errors. The main purpose of this research is to reduce or remove as many of the repetitive tasks in any phase of the assessment process of SQL statements as possible to achieve the consistency of marking and feedback on SQL answers.

This research examines existing SQL assessment tools and their limitations by testing them on SQL questions, where the results reveal that students must attain essential skills to be able to formulate basic SQL queries. This is because formulating SQL statements requires practice and effort by students. In addition, the standard steps adopted in many SQL assessment tools were found to be insufficient in successfully assessing our sample of exam scripts. The analysis of the outcomes identified several ways of solving the same query and the categories of errors based on the common student mistakes in SQL statements.

Based on this, this research proposes a semi-automated assessment approach as a solution to improve students' SQL formulation process, ensure the consistency of SQL grading and the feedback generated during the marking process. The semi-automatic marking method utilises both the Case-Based Reasoning (CBR) system and Rule-Based Reasoning (RBR) system methodologies. The approach aims to reduce the workload of marking tasks by reducing or removing as many of the repetitive tasks in any phase of the marking process of SQL statements as possible. It also targets the improvement of feedback dimensions that can be given to students.

In addition, the research implemented a prototype of the SQL assessment framework which supports the process of the semi-automated assessment approach. The prototype aims to enhance the SQL formulation process for students and minimise the required human effort for assessing and evaluating SQL statements. Furthermore, it aims to provide timely, individual and detailed feedback to the students. The new prototype tool allows students to formulate SQL statements using the point-and-click approach by using the SQL Formulation Editor (SQL-*FE*). It also aims to minimise the required human effort for assessing and evaluating SQL statements through the use of the SQL Marking Editor (SQL-*ME*).

To ensure the effectiveness of the *SQL-FE* tool, the research conducted two studies which compared the newly implemented tool with the paper-based manual method in the first study (pilot study), and with the SQL Management Studio tool in the second study (full experiment). The results provided reasonable evidence that using *SQL-FE* can have a beneficial effect on formulating SQL statements and improve students' SQL learning. The results also showed that students were able to solve and formulate the SQL query on time and their performance showed significant improvement.

The research also carried out an experiment to examine the viability of the SQL Marking Editor by testing the SQL partial marking, grouping of identical SQL statements, and the resulting marking process after applying the generic marking rules. The experimental results presented demonstrated that the newly implemented editor was able to provide consistent marking and individual feedback for all SQL parts.

This means that the main aim of this research has been fulfilled, since the workload of the lecturers has been reduced, and students' performance in formulating SQL statements has been improved.

Keywords: semi-automated, case-based reasoning systems, rule-based reasoning system, partial marking, formative assessment, feedback, online, SQL.

Dedication

To my father and my husband for their ongoing love and support
and to my mother who could not see this thesis completed.

List of Abbreviations

Abbreviation	Full Form
CAA	Computer Assisted Assessment
CBA	Computer Based Assessment
ABC	Access by Computer Approach
IOS	International Organization for Standardization
DDL	Data Definition Language
DML	Data Manipulation Language
SSMS	SQL Server Management Studio
RDBMS	Relational Database Management System
SQL	Structured Query Language
AI	Artificial Intelligence
SQL-FE	SQL Formulation Editor
SQL-ME	SQL Marking Editor
RBR	Rule-Based Reasoning
CBR	Case-Based Reasoning

Table of Content

DECLARATION	I
ACKNOWLEDGMENTS	II
ABSTRACT	III
DEDICATION	V
LIST OF ABBREVIATIONS	VI
TABLE OF CONTENT	VII
LIST OF TABLES	XI
LIST OF FIGURES.....	XII
CHAPTER 1.....	1
INTRODUCTION	1
1.1. OVERVIEW AND MOTIVATION	1
1.2. AIM AND OBJECTIVES	3
1.3. APPROACH	4
1.4. OUTLINE OF THE THESIS	4
1.5. CONTRIBUTIONS.....	7
1.6. PUBLICATION	9
CHAPTER 2. BACKGROUND	10
ASSESSMENT IN EDUCATION: AN OVERVIEW.....	10
2.1. INTRODUCTION	10
2.2. ASSESSMENT IN EDUCATION	10
2.3. COMPUTER-ASSISTED ASSESSMENT (CAA).....	11
2.3.1. Diagnostic Assessment.....	13
2.3.2. Formative Assessment	14
2.3.3. Summative Assessment	14
2.4. COMPARISON BETWEEN ASSESSMENT TYPES.....	15
2.5. MANUAL AND COMPUTER-ASSISTED ASSESSMENT.....	15
2.6. SEMI-AUTOMATED AND FULLY-AUTOMATED ASSESSMENT	16
2.7. TYPES OF AUTOMATED ASSESSMENT	17
2.7.1. Diagram Assessment	17
2.7.2. Programming Language Assessment.....	18
2.8. SUMMARY.....	19
CHAPTER 3. LITERATURE REVIEW.....	21
AUTOMATIC ASSESSMENT OF SQL.....	21
3.1. INTRODUCTION	21
3.2. STRUCTURED QUERY LANGUAGE (SQL)	21
3.2.1. SQL Assessment	24
3.2.2. SQL Assessment Grading.....	24
3.3. DIFFICULTIES IN LEARNING AND ASSESSING SQL.....	25
3.4. EXISTING SQL LEARNING AND ASSESSMENT TOOLS	26

3.4.1.	SQL Tester	26
3.4.2.	SQLg (SQL-Statement Grader)	27
3.4.3.	SQL-KnoT (Knowledge Tester)	29
3.4.4.	SQLify	30
3.4.5.	ActiveSQL	32
3.4.6.	SQLator	33
3.4.7.	AsseSQL	33
3.5.	SUMMARY OF EXISTING SQL TOOLS	35
3.6.	ARTIFICIAL INTELLIGENCE IN EDUCATION	38
3.6.1.	Case-Based Reasoning (CBR)	38
3.6.2.	Rule-Based Reasoning (RBR)	40
3.6.3.	Integration of Rule-based Reasoning and Case-based Reasoning	42
3.7.	SUMMARY	43
CHAPTER 4.....		44
RESEARCH METHODOLOGY		44
4.1.	INTRODUCTION	44
4.2.	RESEARCH APPROACHES	44
4.3.	RESEARCH DESIGNS	45
4.4.	DATA COLLECTION	47
4.4.1.	Surveys	48
4.4.2.	Questionnaires	49
4.5.	DATA ANALYSIS	50
4.5.1.	Existing Exam Scripts Analysis	50
4.5.2.	Data Analysis using t-test	51
4.6.	ETHICAL REQUIREMENTS	52
4.7.	SUMMARY	53
CHAPTER 5.....		54
ANALYSIS OF THE EXISTING SQL EXAMINATION SCRIPTS		54
5.1.	INTRODUCTION	54
5.2.	DATA COLLECTION	55
5.2.1.	Existing SQL Examination Scripts Data Collection	55
5.2.2.	Common Mistakes in SQL Exam Scripts	56
5.2.2.1.	Discussion of Common Mistakes	57
5.2.3.	Model Answers to Each Query	58
5.2.3.1.	Discussion of the Different Model Answers	59
5.2.4.	Errors Categories	60
5.2.4.1.	Discussion of Error Categories	62
5.2.4.2.	Analysis of Each Error Category	63
5.3.	THE IDEAL SQL MARKING PROCESS	65
5.4.	SUMMARY	65
CHAPTER 6. DESIGN, IMPLEMENTATION AND EVALUATION		67
SQL FORMULATION EDITOR (SQL-FE)		67
6.1.	INTRODUCTION	67
6.2.	THE SQL FORMULATION EDITOR (SQL-FE)	68
6.2.1.	Requirements	69
6.2.2.	Components	73

6.2.3.	<i>Technologies used in the development</i>	78
6.2.3.1.	Software Tools	78
6.2.3.2.	Software Source Code.....	80
6.3.	PILOT STUDY	84
6.3.1.	<i>Participants</i>	84
6.3.2.	<i>Study Procedure</i>	85
6.3.3.	<i>SQL Questions</i>	86
6.3.4.	<i>Study Analysis and Discussion</i>	86
6.3.5.	<i>Results</i>	88
6.4.	EXPERIMENT.....	94
6.4.1.	<i>SQL Formulation Editor (SQL-FE)</i>	95
6.4.2.	<i>Participants</i>	96
6.4.3.	<i>SQL Questions</i>	97
6.4.4.	<i>Design of the Experiment</i>	102
6.4.5.	<i>Statistical Analysis</i>	105
6.4.6.	<i>Mean Time Hypotheses</i>	106
6.4.7.	<i>Marks/Performance Hypotheses</i>	106
6.4.8.	<i>Results and Discussion</i>	107
6.5.	SUMMARY.....	112
CHAPTER 7.....		114
A NEW SEMI-AUTOMATIC SQL ASSESSMENT FRAMEWORK.....		114
7.1.	INTRODUCTION	114
7.2.	APPROACH DESCRIPTION	115
7.2.1.	<i>Pre-processing</i>	117
7.2.2.	<i>Normalisation Operation</i>	117
7.2.2.1.	The Remove Normalisation Operation	118
7.2.2.2.	The Replace Normalisation Operation	120
7.2.2.3.	The Sort Normalisation Operation	121
7.2.2.4.	Normalisation operation applied in real data of SQL statements	126
7.2.2.5.	Analysis and Discussion.....	133
A.	SQL statements retrieved from existing exam scripts (2014).....	133
B.	SQL Statements Retrieved from SQL-FE (2016).....	134
7.2.3.	<i>Grouping Process</i>	136
7.3.	MARKING PROCESS OF SQL STATEMENTS	137
7.3.1.1.	SQL Generic Marking Rules	138
7.3.1.2.	SQL Partial Marking.....	138
7.3.1.3.	Propagation of Marks and Feedback	140
7.3.2.	<i>Generic Marking Rules of Semi-Automatic SQL Assessment</i>	141
7.3.2.1.	SQL Marking Rules Classifications	141
7.3.2.2.	Marking Rules Procedure	142
7.3.3.	<i>SQL Marking Process and Marking Rules</i>	143
7.3.3.1.	SELECT, FROM, (WHERE or/and ON) and ORDER BY Clauses	144
7.3.3.2.	SELECT, FROM, WHERE and GROUP BY (Aggregate Functions)	150
7.3.3.3.	SELECT, FROM, GROUP BY and HAVING	153
7.4.	SUMMARY.....	155
CHAPTER 8. DESIGN, IMPLEMENTATION AND EVALUATION		157
SQL MARKING EDITOR (SQL-ME)		157
8.1.	INTRODUCTION	157

8.2.	THE SQL MARKING EDITOR (SQL-ME)	157
8.2.1.	SQL-ME Requirements	158
8.2.2.	SQL-ME User Interface	158
8.3.	SQL MARKING PROCESS (GENERIC RULES)	162
8.4.	THE MARKING PROCESS EXPERIMENT	163
8.4.1.	Participants	165
8.4.2.	Questions	165
8.4.3.	Measurements	167
8.4.4.	Data Collection	167
8.4.5.	Experimental Results and Discussion	168
8.5.	FINDINGS	172
8.6.	SUMMARY	173
CHAPTER 9		174
CONCLUSION AND FUTURE WORK		174
9.1	INTRODUCTION	174
9.2	SUMMARY OF EACH CHAPTER	174
9.3	CONTRIBUTIONS	175
9.4	LIMITATION AND FUTURE WORK	177
9.5	SUMMARY	178
REFERENCES		179
APPENDICES		189

List of Tables

TABLE 3-1: INSTRUCTOR PROCEDURE TO APPLY THE MARK SUGGESTED BY SQLIFY	30
TABLE 3-2: TWO CORRECT QUERY SOLUTIONS AND ONE INCORRECT IN CQ CLASS AND THEIR EVALUATION	31
TABLE 3-3: EXAMPLE OF ACTIVESQL MARKING GRADING SYSTEM	32
TABLE 3-4: FEATURES EVALUATION OF EXISTING SQL ASSESSMENT AND LEARNING TOOLS	36
TABLE 3-5: RULES CONDITIONS	41
TABLE 5-1: EXAMPLE OF COMMON SQL MISTAKES THAT STUDENT MADE IN THE DATABASE EXAM (JUNE 2013)	56
TABLE 6-1: SQL- <i>FE</i> USER INTERFACE DESIGN REQUIREMENTS	70
TABLE 6-2: PARTICIPATING STUDENTS SOLVING SQL QUESTIONS USING BOTH MODES	85
TABLE 6-3: LIST OF SQL QUIZ QUESTIONS	86
TABLE 6-4: SAMPLE DATA OF TIME SPENT BETWEEN PAPER PENCIL METHOD AND SQ- <i>FE</i> METHOD	89
TABLE 6-5: DESCRIPTIVE STATISTICS OF TIME.....	90
TABLE 6-6: DESCRIPTIVE STATISTICS FOR RESPONSE TO FEEDBACK QUESTIONS	92
TABLE 6-7: RESULTS OF T TEST FOR RESPONSE TO FEEDBACK QUESTIONS	93
TABLE 6-8: RESULTS OF THE <i>t</i> -TEST FOR THE RESPONSE TO THE FEEDBACK QUESTIONS	94
TABLE 6-9: THE LECTURER TABLE	97
TABLE 6-10: THE COURSE TABLE	98
TABLE 6-11: SQL QUESTIONS AND THEIR MODEL ANSWERS: SET A.....	99
TABLE 6-12: EMP TABLE.....	100
TABLE 6-13: DEPT TABLE.....	100
TABLE 6-14: SQL QUESTIONS WITH THEIR MODEL ANSWER: SET B	101
TABLE 6-15: THE CROSSOVER EXPERIMENTAL DESIGN DISTRIBUTION	103
TABLE 6-16: PAIRED SAMPLES TEST OF THE TWO TOOLS	109
TABLE 6-17: DESCRIPTIVE STATISTICS OF THE MEAN MARKS OBTAINED USING THE TWO TOOLS	109
TABLE 7-1: ORIGINAL SQL STATEMENTS.....	128
TABLE 7-2: NORMALISATION OPERATION APPLIED ON THE SELECT CLAUSES	129
TABLE 7-3: NORMALISATION OPERATION APPLIED ON THE FROM CLAUSES	130
TABLE 7-4: NORMALISATION OPERATION APPLIED ON THE WHERE CLAUSES	131
TABLE 7-5: NORMALISATION OPERATION APPLIED ON THE AND OPERATOR.....	132
TABLE 7-6: DIVISIONS OF SQL CLAUSE PARTS USING A SPREADSHEET	133
TABLE 7-7: NUMBER OF SQL STATEMENT OCCURRENCES IN EACH GROUP	136
TABLE 7-8: SQL STATEMENTS PARTS.....	139
TABLE 7-9: MARKING RULES CLASSIFICATION (SAMPLE).....	141
TABLE 8-1: THE SQL QUESTIONS USED IN THE EXPERIMENT WITH THEIR MODEL ANSWERS.....	166
TABLE 8-2: THE PARTICIPANTS' RESPONSES ON Q1, 2 AND 3	169
TABLE 8-3: THE PARTICIPANTS' RESPONSES ON Q4.....	169
TABLE 8-4: THE PARTICIPANTS' RESPONSES ON FEEDBACK QUALITY (Q1, 2 & 3)	171

List of Figures

FIGURE 1-1: STRUCTURE OF THE THESIS	5
FIGURE 1-2: CONTRIBUTIONS AND SUB-CONTRIBUTION DIAGRAM	8
FIGURE 2-1: BLOOM'S TAXONOMY BY (BLOOM, 1956)	12
FIGURE 3-1: THE RELATIONSHIP BETWEEN EMP AND DEPT TABLES	22
FIGURE 3-2: THE CBR CYCLE ACCORDING TO (AAMODT AND PLAZA, 1994).....	39
FIGURE 3-3: FORWARD-CHAINING WITH "FIRST COME, FIRST SERVED" (HOPGOOD, 2012).....	41
FIGURE 4-1: SAMPLE OF SQL EXAM SCRIPTS USING SPREADSHEET	50
FIGURE 5-1: THE RELATIONSHIP BETWEEN A STUDENT AND ERRORS (ONE-TO-MANY)	63
FIGURE 5-2: THE RELATIONSHIP BETWEEN STUDENTS AND ERRORS (MANY-TO-ONE)	63
FIGURE 5-3: SQL ERRORS CATEGORIES BREAKDOWN FOR 2013 AND 2014 STUDENTS' EXAM SCRIPTS.....	64
FIGURE 6-1: USE CASE DIAGRAM OF THE CORE FUNCTIONALITIES OF SQL-FE	68
FIGURE 6-2: SQL FORMULATION EDITOR (SQL-FE).....	71
FIGURE 6-3: QUESTION PAN	74
FIGURE 6-4: LEFT NAVIGATION BAR (SQL COMMANDS AND FUNCTIONS)	74
FIGURE 6-5: LEFT NAVIGATION BAR (TABLE SCHEMA)	75
FIGURE 6-6: RIGHT NAVIGATION BAR (SQL KEYWORDS AND OPERATORS)	75
FIGURE 6-7: ENTERING THE SQL STATEMENT USING THE MOUSE POINTER IN SQL-FE.....	76
FIGURE 6-8: TEXT-AREA PANE (USED TO ENTER STRING AND NUMERIC DATA).....	76
FIGURE 6-9: ENTERING DATE VALUES USING THE STRING DATA TYPE	77
FIGURE 6-10: THE TWO TYPES OF CONTROL BUTTONS	77
FIGURE 6-11: THE LIFECYCLE OF PHP REQUEST PROCESSING DIAGRAM.....	79
FIGURE 6-12: PRINT SCREEN OF PHPMYADMIN DATABASE.....	80
FIGURE 6-13: SQL-FE REGISTRATION FORM.....	80
FIGURE 6-14: AN EXAMPLE OF A SQL STATEMENT ANSWER	83
FIGURE 6-15: BOXPLOT OF TIME TAKEN TO COMPLETE THE TEST FOR TWO MODES	90
FIGURE 6-16: HISTOGRAM OF DISTRIBUTION OF TIME TAKEN TO COMPLETE THE TEST FOR PEN AND PENCIL MODE	91
FIGURE 6-17: HISTOGRAM OF DISTRIBUTION OF TIME TAKEN TO COMPLETE THE TEST FOR SQL-FE MODE	92
FIGURE 6-18: EXECUTED SQL STATEMENTS USING THE SSMS TOOL	95
FIGURE 6-19: THE RELATIONSHIP BETWEEN THE LECTURER AND COURSE TABLES	98
FIGURE 6-20: THE RELATIONSHIP BETWEEN THE DEPARTMENT AND EMPLOYEE TABLES.....	101
FIGURE 6-21: BOXPLOT OF THE TIME TAKEN TO COMPLETE THE TEST USING THE TWO TOOLS.....	107
FIGURE 6-22: HISTOGRAM OF THE DISTRIBUTION OF TIME TAKEN TO COMPLETE THE TEST USING SQL-FE	108
FIGURE 6-23: HISTOGRAM OF THE DISTRIBUTION OF TIME TAKEN TO COMPLETE THE TEST USING SSMS.....	108
FIGURE 6-24: BOXPLOT OF PERFORMANCE MARKS OF BOTH SQL-FE AND SSMS	110
FIGURE 6-25: HISTOGRAM OF THE DISTRIBUTION OF MARKS OBTAINED USING SQL-FE	111
FIGURE 6-26: HISTOGRAM OF THE DISTRIBUTION OF MARKS OBTAINED USING SSMS	111

FIGURE 7-1: THE PROPOSED SEMI-AUTOMATIC APPROACH	116
FIGURE 7-2: NORMALISATION PROCESS APPLIED ON THE SQL STATEMENTS OF THE EXAM SCRIPTS	134
FIGURE 7-3: NORMALISATION PROCESS APPLIED TO SQL STATEMENTS OF SQL-FE.....	135
FIGURE 7-4: PROPAGATION OF SQL STATEMENT PARTS.....	140
FIGURE 7-5: AN ILLUSTRATION OF THE MARKING PROCESS AFTER APPLYING THE RULES.....	145
FIGURE 7-6: THE MARK PROPAGATION PROCESS WITH OTHER GROUPS.....	146
FIGURE 7-7: E.G. THE FROM CLAUSE IN G11 IS NOT IDENTICAL TO THAT IN G12.....	147
FIGURE 7-8: THE FROM CLAUSE SHOULD BE MARKED AS A SEPARATE PART	147
FIGURE 7-9: AN SQL ANSWER CONTAINING ON AS A JOIN STATEMENT	148
FIGURE 7-10: AN SQL ANSWER USING WHERE AS JOIN.....	149
FIGURE 7-11: AN SQL ANSWER USING A GROUP BY CLAUSE	150
FIGURE 7-12: THE MARKING PROCESS OF A GROUP BY CLAUSE WITH MULTIPLE FIELDNAMES.....	151
FIGURE 7-13: THE WHERE CLAUSE MARKING PROCESS WITH A GROUP BY CLAUSE	152
FIGURE 7-14: MARKING GROUP BY AND HAVING CLAUSES AS A GROUP	153
FIGURE 7-15: AN SQL STATEMENT WITH A HAVING CLAUSE MARKED SEPARATELY.....	154
FIGURE 8-1: THE SQL MARKING PROCESS ARCHITECTURE	159
FIGURE 8-2: THE USER INTERFACE OF THE SQL-ME (PARTIAL MARKING INTERFACE).....	161
FIGURE 8-3: SQL STATEMENTS IN GROUPS (GENERIC MARKING RULES).....	164
FIGURE 8-4: THE PARTICIPANTS' RESPONSES' ON Q4	170
FIGURE 8-5: THE PARTICIPANTS' RESPONSES ON FEEDBACK QUALITY (Q4)	171
FIGURE 8-6: THE PARTICIPANTS' RESPONSES ON USEFULNESS OF SQL-ME	172

Chapter 1.

Introduction

1.1. Overview and Motivation

Computer-Assisted Assessment (CAA) has turned into an essential technique that can provide a comprehensive formulation and marking environment (Adesina, 2016). Simultaneously, it can be utilised to reduce the review and assessment load on lecturers (Pardo, 2002). Despite assessment being critical to student learning and certification where several automatic assessment frameworks have been developed, the adoption has been inconsistent (Bennett *et al.*, 2017). Fully-automatic assessment covers just a part of the general assessment requirements in computer science courses (Adesina *et al.*, 2015). According to Bloom (1956), designing an assessment tool should match the learning objective along with the commonly used question types. In this case, CAA can support different types of questions, where it is categorised as either as fully-automatic assessment or semi-automatic assessment (O'Reilly and Morgan, 1999). The fully-automatic assessment evaluates the submitted answers automatically and lecturers do not have to grade each submission individually (Weinberger, 2011). This type of assessment can provide consistent feedback and reduce the lecturers' workload. However, it often ignores the main parts of students' answers when providing feedback. On the other hand, semi-automated assessment approach is a partially automatic evaluation of the submitted work. It provides each part of the assessed work with a final score while lecturers do the final grading. This means that each part of the solution is marked and provided with feedback through the help of human markers (Tremblay and Labonté, 2003).

Lecturers and educators often resort to setting fewer assessment tasks or accepting the significant increase in their manual marking load, which can affect the feedback quality provided. The semi-automated assessment approach has thus become vital for coping with this increased workload. Furthermore, for these reasons, this research considers the semi-automated assessment approach for implementation since it is used in computer science class assessments, where computer programs or source-codes are automatically evaluated and then manually revised by lecturers (Ala-Mutka, 2005).

Structured Query Language (SQL) is the leading database language in teaching and assessment environments. However, to formulate and assess useful SQL queries, various difficulties and challenges are often faced, which requires more practice from students and further assessment efforts from lecturers (Ahadi *et al.*, 2016). Research by Tropashko and Burleson (2007) stated that “SQL is a declarative language; with no mechanisms for flow control, loops, variables and no methods for storing intermediate results”. Although SQL contains simple syntax, marking and assessment of its coursework can be very difficult for lecturers. The reason for that is that SQL statements need to be tested and evaluated individually according to the syntax structure, style and datasets (output data). A large number of studies have aimed to reduce the lecturers’ workloads and increase the efficiency of the feedback submitted to the students (e.g. Brusilovsky *et al.*, (2008); Sadiq *et al.*, (2004); Prior and Lister (2004); Kleiner *et al.*, (2013); Raadt *et al.*, (2007), and Mitrovic (1998)). However, SQL offers many ways to solve the same query, and most of the aforementioned studies rely on comparisons of datasets without checking the ways students tried to solve the query. In order to solve these problems, a stronger analytical tool is needed to evaluate the structure of the whole query and give consistent marks to the students. The proposed tool will not only look at the structure of the SQL query, but will also mark different (i.e. alternative) ways of solving the query without any restrictions on the lecturer’s solutions. In addition, the tool will provide marks for every correct statement submitted by the student and provide students with visual feedback on the errors they made.

This research is based on the Semi-Automatic Assessment approach, which utilises both the Case-Based Reasoning (CBR) system and the Rule-Based Reasoning (RBR) system techniques. The approach aims to improve SQL learning and assessment by enhancing the learning approach of SQL queries for students, reducing the marking workload of lecturers, enhancing the consistency of grades provided to students, and delivering an effective and timely feedback to them. In addition, the research only focuses on solving problems of basic SELECT clauses, which cover the following clauses;

```
SELECT < list of columns>  
FROM <table list>  
WHERE <row condition>  
GROUP BY <group list>  
HAVING <group condition>  
ORDER BY <sort list>
```


This chapter begins with an overview of Computer-Assisted Assessment and the motivation of this work. Following this, the aims and objectives of the research are outlined in Section 1.2. The chapter then discusses the research approach and outline the structure of this thesis in Section 1.3 and 1.4, respectively. The novel contribution and sub-contributions of the thesis are listed in Section 1.5. Finally, the chapter concludes with publication details in Section 1.6.

1.2. Aim and Objectives

This research proposes a semi-automated assessment framework that supports human markers. The main purpose of this research is to reduce or remove as many of the repetitive tasks in any phase of the marking process of SQL statements as possible. As identical tasks are performed less frequently (possibly only once) by examiners, consistency of marking and feedback on SQL answers can be achieved. In other words, the primary target of this research is to reduce the time and effort associated with the SQL evaluation process.

There are several objectives, which the Semi-automated Assessment of SQL Statements research aims to achieve, including:

1. Identifying the problems with existing SQL learning and marking systems. This includes defining the problems and limitations caused by using manual marking, as well as examining the existing SQL assessment tools and analysing them in terms of how they work and what features are used to mark SQL statements.
2. Analysing different common errors made by students. This involves identifying the common mistakes in students' answers and analysing them to implement an accurate marking environment that can help identify the similarities between SQL statements and mark them automatically.
3. Providing a detailed rationale of the requirements and components of the developed SQL Formulation Editor (*SQL-FE*), as well as performing an appropriate experimental study to evaluate the time saving and the students' performance using the SQL formulation Editor (*SQL-FE*). Furthermore, a hypothesis of the fundamental relationship between the experiments and surveys of the research methodology designs should also be tested.
4. Developing a novel framework that provides a platform where different intelligent techniques work together to support the assessment process of SQL statements by utilising the case-based and rule-based reasoning systems.

5. Developing techniques (such as normalisation operations and grouping of statements) to reduce the repetitive tasks or eliminate them completely where possible. Furthermore, the common repetitive tasks in the assessment process should also be identified.
6. Providing a detailed rationale of the requirements and components of the developed SQL Marking Editor (*SQL-ME*) and perform an appropriate experimental study to evaluate the feasibility of the Semi-automatic Assessment approach (*SQL-ME*) and analysing its results.

1.3. Approach

This research focuses on the semi-automated SQL assessment approach. The aim of semi-automation is to reduce the number of SQL statement clauses marked by examiners. This requires identifying and grouping identical clauses in students' solutions by finding their identical components using different SQL statements clauses attributes (e.g. commands, functions, operators and keywords). At the same time, the marking process goes through four main stages, which are the normalisation, partial marking, grouping the identical statements, and applying the new formulated SQL marking rules which utilise both the Case-Based Reasoning (CBR) and the Rule-Based Reasoning (RBR) systems.

The semi-automated assessment approach is a solution to ensuring the consistency of the SQL marking and feedback generated during the marking process. It uses the string matching method, which does not involve matching students' answers with the model answers. Rather, it groups the matching clauses of students' SQL statements and then asks the examiners to approve the correctness of SQL clauses from each of the different groups. To evaluate the proposed approach, this research implements a complete SQL learning and assessment framework that supports the process of the semi-automated assessment approach.

1.4. Outline of the Thesis

This thesis investigates the use of Semi-automated Assessment of SQL Statements as a solution to reducing examiners' marking workload. It describes previous work carried out in the field of automated assessment and presents arguments for using a new framework environment for practicing and assessing SQL statements. The thesis comprises of nine chapters as illustrated in Figure 1-1, which provides an overview of the thesis structure.

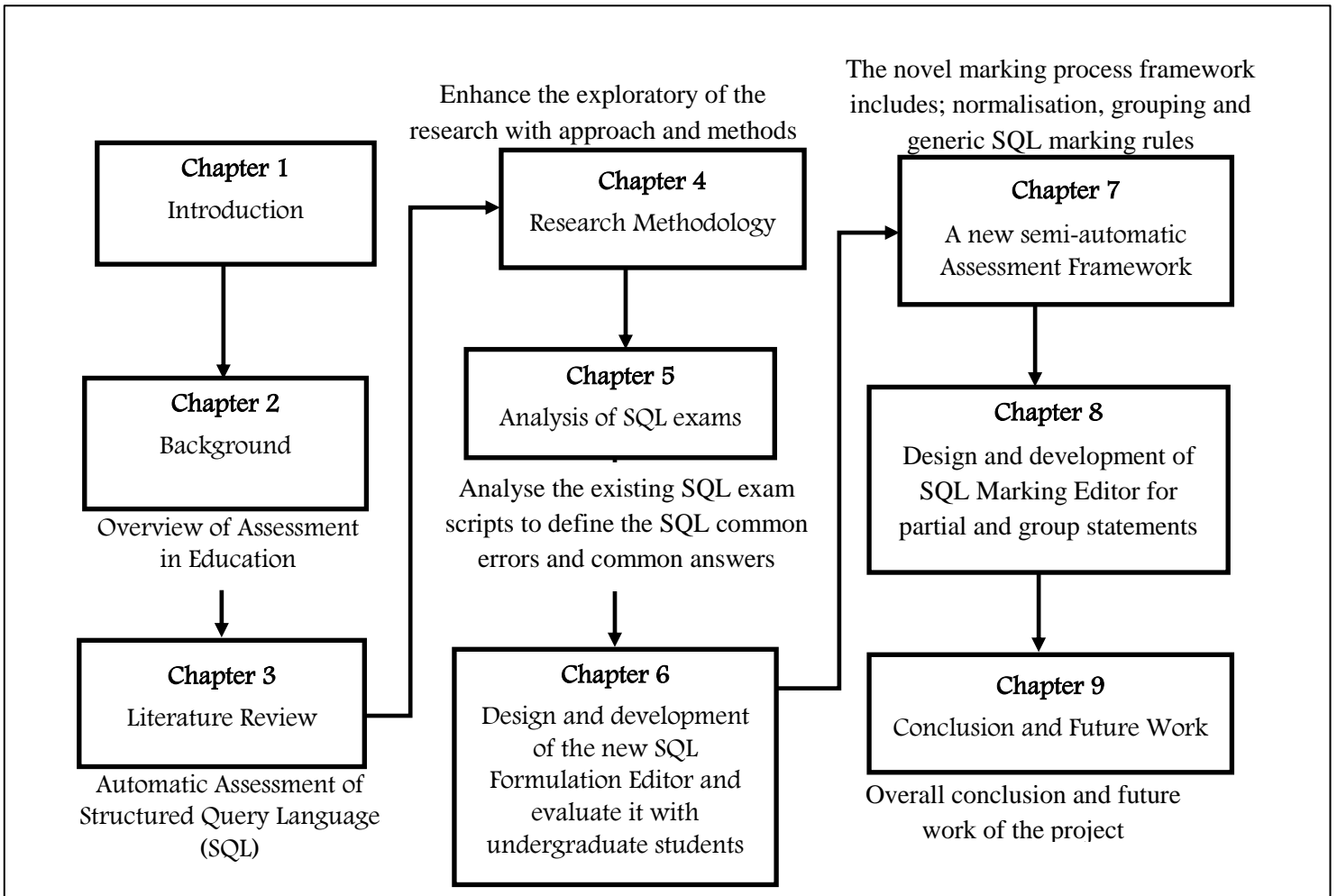


Figure 1-1: Structure of the Thesis

The main body of this thesis is organised as follows:

Chapter 2: Presents a background on assessment approaches in education, an overview of the computer and its use in education, and an introduction to computer-assisted assessment (CAA). Furthermore, it discusses the difference between the fully-automatic and semi-automatic assessment approaches and outlines the various types of automated assessment.

Chapter 3: Presents a literature review of the marking and grading of SQL statements. It briefly discusses the main difficulties of SQL learning and assessment and presents a survey of existing SQL learning and assessment tools and the related state-of-the-art approaches of automated assessment. Moreover, it highlights multiple SQL learning and assessment tools that have been developed for learning and assessing SQL statements. Finally, it proposes a new solution to overcome the current challenges by integrating the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems.

Chapter 4: Describes the methodology of this research. The main objective of this chapter is to enhance the exploration of the research. It discusses the research approaches, designs, data collection and analysis methods and techniques used to simplify the research design framework.

Chapter 5: Overviews the data collection process used to collect data from the exam scripts of the Database module. In addition, it analyses the common errors in SQL exam scripts and develops a simple grading scheme. This chapter's intention is to explore the difficulties and challenges, which students and examiners face in the manual assessment of SQL statements and what errors students frequently make when they use the current system.

Chapter 6: Presents the design and implementation of the SQL Formulation Editor (*SQL-FE*), which uses the point-and-click method. The objective of this chapter is to provide an overview of the newly implemented system and explain how the system works. In addition, it introduces two different studies to evaluate the new *SQL-FE* tool. The purpose of this evaluation is to test the new specialised tool in terms of formulating SQL statements. The evaluation part is divided into two sections, a pilot study and a full experiment, both of which involve students testing the tool to evaluate the time spent in formulating SQL statements and the students' performance. The chapter also presents the evaluation results and the students' feedback after formulating SQL statements with the implemented tool.

Chapter 7: Discusses the details of the semi-automated assessment framework that aims to reduce the workload of examiners. In addition, the approach aims to enhance students' SQL learning experience and provide them with distinct and detailed feedback. The main purpose of this chapter is to provide an overall explanation of the new approach and discuss how it can solve the main challenges of the current learning and assessment systems highlighted in Chapter 3. In addition, it explains the marking process of SQL statements by formulating new generic marking rules for the SQL statements using both the case-based reasoning and rule-based reasoning systems. The chapter then proceeds to discuss how to enhance the marking propagation technique between SQL clauses and decrease the number of the SQL statements that should be marked by examiners. Finally, it concludes the distribution of the SQL clauses parts feedback by receiving a consistent feedback for all the identical parts.

Chapter 8: Presents the design and implementation of the SQL Marking Editor (*SQL-ME*) using the semi-automatic assessment approach. The objective of this chapter is to provide an overview of the newly implemented system and explain how the approach will reduce the SQL statements marked by the examiners. The chapter then describes the evaluation of the newly implemented editor, where examiners test the *SQL-ME* tool. In this context, two studies are carried out; one to test the normalisation operation of SQL statements and how it can increase the similarities across SQL statements, and the other to test the marking propagation of SQL statements and how it can increase after applying the generic rules.

Chapter 9: Presents the conclusion of this research and recommendations for future work directions. It highlights the reasoning and judgments on the findings of this research in terms of the results and outcomes. In addition, in the future plan section, the chapter lists the upcoming tasks that will take place in the coming years as an extension to this research.

1.5. Contributions

The main novel contribution of this research is the development of a novel framework that provides a platform to support the assessment process of SQL statements, which supports the integration of both the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems that use application of the Artificial Intelligence (AI) methodology. Such a framework advantages are:

- Enables human and computer association during the assessment process.
- Analyse beginner students' SQL statements in terms of SQL clauses to provide consistent feedback,
- Reduce the overall SQL statement clauses marked by examiners. This means to reduce the human intervention on marking and reuse the comments given for similar SQL parts.
- Enhances the accuracy of marking and provides students with immediate feedback.

This results in reducing or removing as many of the repetitive tasks in all phases of the marking process of SQL statements as possible.

The following are the sub-contributions involved:

To achieve the objectives of this research, there are several sub-contributions involved, which illustrated in Figure 1-2. The figure outlines the sub-contributions with arrows to show how the contributions are connected, and leads to achieving the main novel contribution, which represents “No. 4” in the following figure.

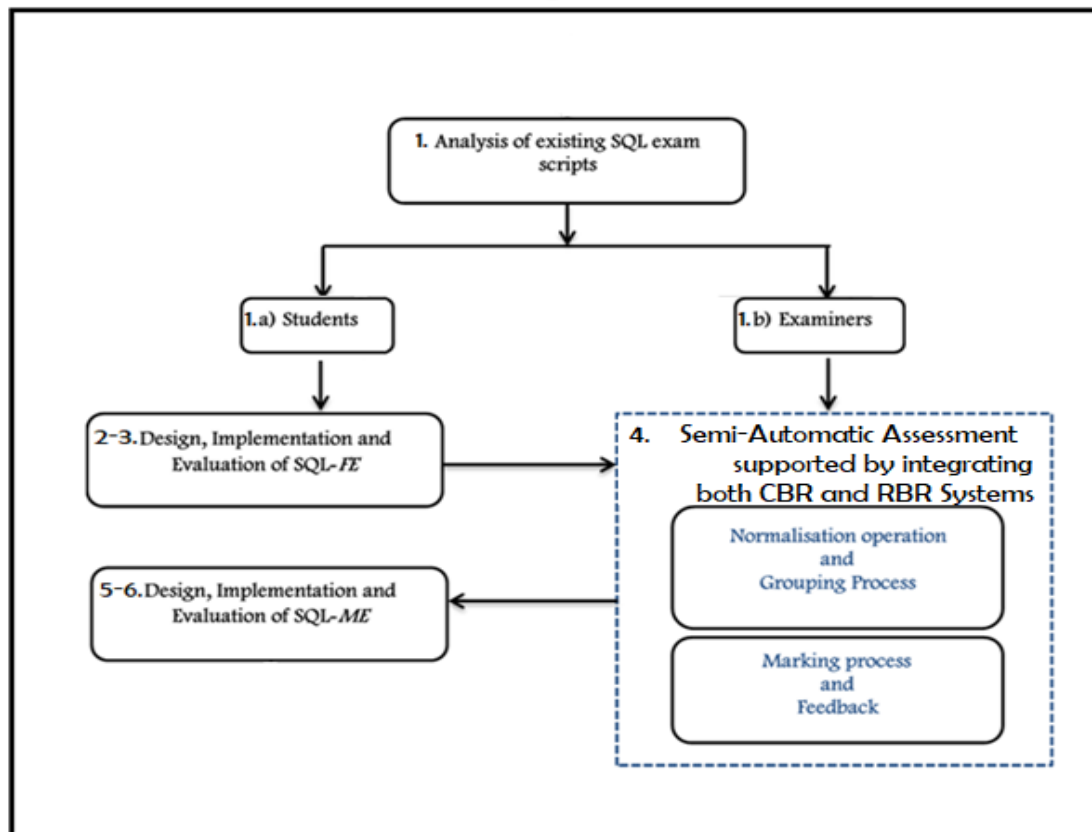


Figure 1-2: Contributions and Sub-contribution Diagram

1. To identify the common mistakes committed by students and find the alternative ways of solving the same SQL query, the researcher has collected and analysed previous SQL exam papers. The analysis has gone through different phases to identify them. Figure 1-2 shows that this analysis became the foundation of this research, which is used as a tool for supporting the design and implementation of the new editors;

1. a) Students: to formulate the SQL statements using the SQL Formulation Editor (SQL-FE). This tool has been implemented to help students formulate their SQL query using point-and-click approach. Using this approach assisted the student not to attempt any spelling mistakes and add unnecessarily elements in the query.

1. b) Lecturers: to mark the SQL statements and submit feedback to students which submitted by SQL-FE using the SQL Marking Editor (SQL-ME).

2. To formulate SQL statements that eliminate adding unnecessary elements to SQL statements and prevent students from making minor and avoidable mistakes, the researcher has designed and implemented a new SQL Formulation Editor named as *SQL-FE*.
3. To obtain the students feedback of the new implemented editor and to test the editor performance that reduce the errors while solving SQL statements, the researcher has evaluated the *SQL-FE* from several college students and collect their opinions of how to enhance it.
4. To reduce the repetitive marking in duplicated SQL answers or remove them completely where possible, the researcher has applied the normalisation operation, which is based on the proposed semi-automatic SQL assessment framework. This lead to develop a new technique for marking process using the SQL generic marking rules. The SQL marking process is an integration of both Rule-based Reasoning (RBR) and Case-based Reasoning (CBR) systems. This method shows how efficiency and savings in marking time may be obtained by reducing repetitive activities.
5. The marking process of the SQL statement has proposed a new semi-automatic assessment framework to mark the identical SQL statements using a new SQL Marking Editor named as *SQL-ME*.
6. To obtain the lecturers feedback of the new implemented editor and to evaluate the feasibility of the editor performance, the researcher has performed an appropriate experimental study to evaluate the feasibility of the semi-automatic assessment approach using the new implemented *SQL-ME* through several SQL experienced lecturers and collect their opinions of how to enhance it.

1.6. Publication

The above contributions have resulted in the following conference paper:

PNo	Publication	Relevant Chapter	Appendix
1	AL-Salmi A. (2018). A Web-based Semi-Automatic Assessment Tool for Formulating Basic SQL Statements: Point-and-Click Interaction Method . In Proceedings of the 10th International Conference on Computer Supported Education - Volume 2: CSEDU, ISBN 978-989-758-291-2, pages 191-198. DOI: 10.5220/0006671501910198.	4,5,6	1

Chapter 2. Background

Assessment in Education: An Overview

2.1. Introduction

Automated assessment of programming has become an important method for grading students' work and providing effective feedback to an enormous number of students (Buyrukoglu, Batmaz and Lock, 2016). Computer-Assisted Assessment (CAA) is a field of learning technology that studies the use of computers (Higgins *et al.*, 2002). CAA may be used for both formative and summative assessments to deliver, analyse and mark student assessments (Bull and Danson, 2004). This chapter defines Computer-Assisted Assessment (CAA) and describes the techniques of CAA, providing detailed information on formative, summative and diagnostic assessments. It illustrates Bloom's Taxonomy and the types of assessment, namely diagram and programming language assessments.

The rest of this chapter is organised as follows. Section 2.2 discusses assessment in education, while Section 2.3 describes the process of computer-assisted assessment and introduces three different techniques of CAA, whose features are then compared in Section 2.4. Section 2.5 provides the definitions of and specifies the difference between manual and automated assessments. A comparison between semi-automated and fully-automated assessments is presented in Section 2.6, while Section 2.7 discusses automated assessment in fine detail. Finally, Section 2.8 concludes the chapter by providing a summary of its contents.

2.2. Assessment in Education

Harlen *et al.* (1992) defines assessment as the process of gathering information about students' answers in educational tasks. A study by Taras (2005) declared that there are a number of reasons why lecturers assess their students. Among those reasons, assessment can shed light on how students have developed and where they have progressed. Furthermore, it can help lecturers to make modifications to their teaching practices to improve the learning experience for their students.

In addition, it can provide lecturers with information about what they have taught to students, and in what other areas they should assess them. According to the USA National Institute of Education (1997, p.160):

“If assessment is to be a positive force in education, it must be implemented properly. It cannot be used to merely sort students or to criticise education. Its goals must be to improve education. Rather than 'teach to the test', we must 'test what we teach'”.

Therefore, the main purpose of conducting assessments is to improve learning and teaching quality by extracting the positive power of students' knowledge (Harlen *et al.*, 1992). In addition, assessment provides lecturers with information on students' progress and improvement, and helps them to enhance the teaching and learning experience for future and present students (Taras, 2005). James *et al.*, (2002, p.8) argued that assessment should be a strategic tool to enhance teaching and learning due to the fact that students often *"work backwards through the curriculum, focusing first and foremost on how they will be assessed and what they will be required to demonstrate they have learned"*.

2.3. Computer-Assisted Assessment (CAA)

“CAA is a common term for the use of computers in the assessment of student learning. The term encompasses the use of computers to deliver, mark and analyse assignments or examinations.” (Bull and McKenna, 2004, p.8)

CAA refers to the process of assessing students' progress using computers (Conole and Warburton, 2005). CAA is used mainly for a range of activities such as delivering marks, analysing assignments or examinations and providing effective feedback (Stephens *et al.*, 1998). Dalziel (2001) stated that computer-assisted assessment might significantly enhance the overall learning outcomes by providing learners with efficient exams and useful feedback. There are a number of benefits associated with the use of CAA (Bull and McKenna, 2004). These include motivating and encouraging students to practice skills by providing opportunities for formative assessment, broadening the range of the knowledge assessed (e.g. creating websites or complex diagrams) and offering opportunities for more immediate feedback, as well as allowing feedback to be delivered in different ways. Bloom's Taxonomy provides a framework that aids thinking about the purpose of assessment.

The taxonomy classifies six levels of learning objectives, which are Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation (Bloom, 1956). Figure 2-1 shows Bloom's Taxonomy, which was created by Benjamin Bloom during the 1950s, and is a way to categorise the levels of reasoning skills required in classroom situations.

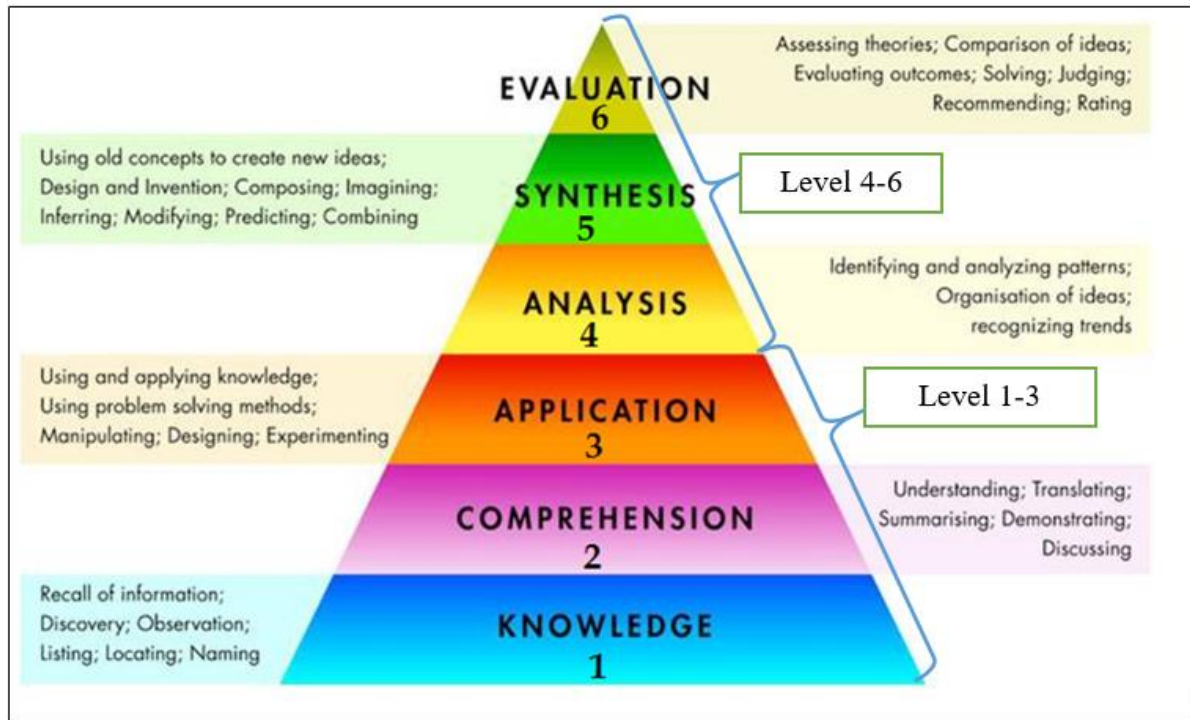


Figure 2-1: Bloom's Taxonomy by (Bloom, 1956)

This hierarchical taxonomy structure lists six levels of thinking and learning skills that range from basic learning objectives such as knowledge of content through higher-order learning such as synthesis, evaluation, and creativity. The six levels as illustrated in Figure 2-1, each requiring a higher level of abstraction from the students than the last.

- **Levels 1-3:** are sometimes described as relating to "shallow" or "surface" learning.
- **Levels 4-6:** are associated with "deep" learning.

From the figure above, assessment of level 1 is quite simple and can frequently be accomplished by Multiple Choice Questions (MCQs) or questions requiring simple responses. However, it becomes gradually more difficult to measure a student's competence as the higher-level objectives are addressed. Carter et al. (2003) listed different types of questions that are often used in CAA tools to test surface learning, which include:

- A. Multiple Choice Questions (MCQ):** a single choice of response is made. MCQs potentially have high reliability, validity, and fast feedback submission to the students.

- B. Multiple Response Questions (MRQ):** similar to MCQs but with multiple selections of response.
- C. True/False Questions:** a test consisting of a series of statements to be marked as true or false.
- D. Short Answer Questions:** require a response in the form of text.
- E. Essay Questions:** test a wide range of abilities including the capacity to draw on a wide range of knowledge. Marking (i.e. grading) is made relatively fast by providing marking schemes before the lecturers start to mark and sharing them with students.
- F. Numerical Questions:** require a numerical response.

These classes help educators distinguish more closely what they teach and, by implication what they should assess and provide feedback on. By providing the hierarchy of levels, this taxonomy can assist teachers in designing performance tasks, crafting questions for engaging with students, and providing feedback on student work.

It has been argued that such simple question types cannot be used to measure students' deep learning skills (Entwistle, 2000). Therefore, computer-assisted assessment software tools aim to encourage newer question types which are not restricted only to MCQs and True/False questions, but cover various other types, like computer programming and computer diagrams, which constitute a growing interest area for many researchers (Rawles *et al.*, 2002). Research by O'Reilly and Morgan (1999) and Bull and McKenna (2004) stated that CAA can be categorised into three types: *Diagnostic Assessment*, which is used by tutors to define their students' knowledge, *Formative Assessment*, which provides feedback to assist the learning process, and *Summative Assessment*, which is used for grading purposes. The three techniques are explained in detail in the subsequent subsections.

2.3.1. Diagnostic Assessment

According to the University of Northern Illinois (2004), diagnostic assessment (assessment as learning) is used to learn about students' strengths and weaknesses, which can help lecturers to plan what to teach and how to teach it. It is used to define students' knowledge, usually at the beginning of the year (i.e. before the course starts), to assess the effectiveness of the teaching (Sclater and Howie, 2003). Diagnostic assessment can analyse different features of difficulties that students face, such as students with a lack of knowledge, students with difficulties in understanding and students with weaknesses in skills (Conole and Warburton, 2005).

There are different types of diagnostic assessment, such as pre-tests, which highlight the abilities of students, self-assessments, which identify skills and capabilities, and interviews, which should be brief and private (University of Northern Illinois, 2004).

2.3.2. Formative Assessment

Formative assessment refers to student involvement in the assessment and learning practice which involves the collaboration between teacher and students aimed at improving the learning process (López-Pastor and Sicilia-Camacho, 2017). It is the process by which teachers provide information to students during the learning process to modify their understanding which also named as (assessment for learning) (Pieterse, 2013).

It is a process in which lecturers use various tools to define what students know and the gaps in their understanding, and plan future instructions accordingly to improve learning (Pinckok and Brandt, 2009). The Council of Chief State School Officers (2008) stated that, based on feedback about students' performance, formative assessment is used as a process to enhance students' education. Formative feedback helps student to develop a deeper understanding of their learning, since it is an essential component of the formative assessment process (Clark, 2011).

“Formative assessment refers to assessment that is specifically intended to provide feedback on performance to improve and accelerate learning” (Sadler, 1998, p. 77).

Rowntree (1987) stated, "Feedback or 'knowledge of results' is the lifeblood of learning". According to Bedford and Price (2007), successful feedback should focus on learning rather than on marks and should be understandable. In other words, if students know exactly what went wrong with their submissions and exactly what their mistakes were, they can use the feedback information to learn and revise their answers. Clark (2011) stated that formative feedback cannot involve simply telling a student to either "try again" or "reconsider your work", since this does not guide the student with appropriate instructions. However, feedback becomes formative when students are provided with supportive instructions which help them to improve their thinking and enhance their learning process (Clark, 2011).

2.3.3. Summative Assessment

Summative assessment (assessment of learning) usually occurs at the end of a course of study (Taras, 2001). The main reasons for using summative assessment are to identify what has been learned over a period of time and to summarise students' performance by sending progress reports to them (Harlen and James, 1997).

Research by Chalmers and McAusland (2002) indicated that summative assessment is conducted as an official evaluation, where students are informed in advance and can prepare. In addition, they specify that it should be held in a supervised location, with specific timing, and the results should be either hidden from students or displayed at the end of their studies. The evaluation techniques used in summative assessment include projects, interviews and analysis of work samples (Chalmers and McAusland, 2002).

2.4. Comparison between Assessment Types

Diagnostic assessment performs well in measuring students' performance before starting their studies; however, it cannot fulfil the aim of students getting their annual grades and receiving feedback, which is achieved using either formative or summative assessment (Sclater and Howie, 2003). While formative assessment can allow students to be automatically directed, through feedback, to follow-up references and resources, summative assessment needs to be formal, structured and supervised, and therefore requires more effective co-ordination between academic departments and central services than formative assessment (Clark, 2011). Generally, summative assessment systems do not provide feedback or suggestions (Stephens et al., 1998).

Summative assessment refers to the assessment of participants and summarises their development at a particular time. In contrast to formative assessment, the focus is on the outcome of a programme.

2.5. Manual and Computer-Assisted Assessment

Assessment in higher education can be either paper-based or automated system. Paper-based assessment has shown a number of problems, especially when high numbers of students are enrolled in one class, because it is conducted manually (Carter *et al.*, 2003). Manual assessment might affect lecturers' time management, as the marking load is increased, which can lead to them either setting the students fewer assessment tasks (e.g. mid-terms, quizzes and assignments) or adding additional marking time to their schedules (Carter *et al.*, 2003). As such, large class sizes, limited time for marking assessments and non-effective feedback have led educators to think about computerised assessment.

Automated assessment has recently become more useful for both students and staff since network computer technology can now support teaching and learning in higher education. Peat and Franklin (2002) stated that online assessment has become more popular for supporting the improvement of both teaching and learning.

A study done by Woit and Mason (2003) showed that automated assessment may improve students' motivation and programming efficiency when it is implemented securely and efficiently. In addition, online assessment provides students with appropriate feedback that can help them enhance their learning progress (Ihantola *et al.*, 2010). Manual assessment leads to a less efficient learning process and a difficulty in assessing students' work, whereas automated assessment can achieve an improvement in the learning and teaching process, since it can reduce marking workloads, enhance grading accuracy and encourage interaction between lecturers and students via feedback.

2.6. Semi-Automated and Fully-Automated Assessment

Computers can be used for assessment in two different ways. The first approach is the semi-automated assessment, which is a partially automatic evaluation of the submitted work, providing parts of the final score while lecturers do the final grading. Second approach is the fully-automated assessment, which fully evaluates submissions so lecturers do not have to grade each submission individually (Weinberger, 2011). Kakkonen *et al.* (2004) defined semi-automated assessment as a system that takes responsibility for more powered aspects of assessment to prepare submissions, compilation, testing, style analysis and report generation.

A semi-automated system needs to provide some kind of automation that is not completely dependent on human interaction to assess each assignment, but leaves grading and feedback to the lecturer (Weinberger, 2011). Saikkonen *et al.*, (2001) has mentioned several benefits of the fully-automated assessment such as;

- The fact that the assessment is carried out online so the students can get their grades immediately and resubmit their wrong answers after considering their mistakes.
- Easy analysis of the structure of students' code.
- Avoiding the use of the comparison stage between the expected result and the output from students' codes.

While semi-automated assessment is often used in computer science class assessments, where programs or source code are automatically evaluated and then manually reviewed by lecturers; fully-automated assessment does not require human interaction to produce a final grade for students' work, although there is some necessary preparation involving setting up and initiating the grading process (Tremblay and Labonté, 2003).

In addition, even though fully-automated assessment can be performed at the lower levels of blooms' taxonomy; (because these levels require at least one correct answer, e.g. multiple-choice questions (Clark, 2011)); it cannot be applied on the higher levels because students' solution are generally written answers including computer programming codes or essay assessments (Wong *et al.*, 2012).Semi-automatic assessment is used for evaluating students' learning and submitting grades with meaningful feedback, such as those of a midterm exam, final project or final exam (Douce *et al.*, 2005). It generates immediate feedback on the validity of students' solutions to guide them with regard to what corrections they need to make to their answers.

Under these circumstances, semi-automatic assessment that supports a computer assessment approach should be utilised to assess students' answers that are based on any computer programming code (e.g. JAVA or a declarative language such as Structured Query Language (SQL)). Kakkonen et al. (2004) used a different distinction between fully- and semi-automated assessments, where they stated that fully-automated assessment only provides a score (summative assessment), whereas a semi-automated system provides a grade and more details to support learning (formative assessment).

2.7. Types of Automated Assessment

Many universities are currently aiming to enhance the student assessment process, especially for first-year courses that include high numbers of enrolled students. Growing student numbers in computer science courses have resulted in rising efforts to develop automated assessment systems that can reduce the workload of lecturers and enhance student feedback. According to Tshibalo (2007), academic workload is increased in higher education, and automated assessment may help reduce this workload by helping lecturers manage the large volume of marking. Several researchers have focused on the automatic assessment of diagrams and programming languages. To follow, two types of automated assessment; diagram assessment and programming language assessments.

2.7.1. Diagram Assessment

Numerous researchers have demonstrated an intention of developing projects to assess database diagrams. Tselonis et al. (2005), Batmaz and Hinde (2007), and Higgins et al. (2009) focused on implementing a semi-automated approach, where a computer takes part in assessing students' diagrams using the CBR method.

CBR is a method of solving new problems by utilising the solution(s) of identical past problems (Kolodner, 2014). It selects diagrams that are the same as the diagram being marked by comparing them against each other. The target diagram is then given the same mark as the identical diagrams found in the diagram body. If no similar diagrams are found, the target diagram is passed to a human for marking. Higgins et al. (2009) presented a Computer-Based Assessment (CBA) technology, which refers to the delivery of materials for teaching, assessment, student solutions and feedback. They evaluated the feasibility and usefulness of developing and deploying diagram-based exercises by using the DATsys and CourseMarker approaches (Higgins *et al.*, 2009).

Some automated assessment tools are designed mainly for summative assessment (e.g. BOSS (Luck and Joy, 1999)), while others show the student the results of the automatic assessment and allow resubmissions if the student is not satisfied with the results (e.g. CourseMaker (Higgins *et al.*, 2003)). At the University of Manchester, specifically in the Computer Science department, they established the Access By Computer (ABC) approach, which defines identical components by using those component's attributes (e.g. label, type, adjacent boxes) (Tselonis *et al.*, 2005).

2.7.2. Programming Language Assessment

Automated programming assessment has recently become an important method of assisting the lecturers of programming courses in automatically marking and grading students' programming exercises, as well as providing useful feedbacks on their programming solutions (Romli *et al.*, 2010). The majority of these systems have been developed to assess objected-oriented programming languages, such as Java, C/C++ and Pascal (Pribela *et al.*, 2014). Tremblay and Labonté (2003) introduced a semi-automated marking system for Java programs using JUnit (a public test suite specified by the lecturer for a given assignment that is used to provide students with early feedback). It allows students to submit their solutions, after which the system tests the submissions on the public test suite, and at the end, the appropriate results are sent back to the students, indicating either success or failure. Benford et al. (1993) introduced the Ceilidh system, which uses string matching to compare the output of students' programs with the model output set by the lecturer. String matching algorithms is to find all the occurrences of strings (also called patterns) within a larger string or text (Shah and Oza, 2018).

Research by Saikkonen et al. (2001) used the Ceilidh system to assess exercises written in C or Java for a basic programming course. On the other hand, marking using the ASSYST system is done through automated testing (based on the context-free grammar specification of the expected output). It allows students to submit their programs by email, after which the lecturer tests and marks them and sends back an evaluation report (Jackson and Usher, 1997). Another example of automated programming assessment is the BOSS system, presented by Joy et al. (2000), which supports both the submission and the testing of the program code using textual output comparison techniques.

In addition to the aforementioned examples, McQuain (2003) developed a web technology called Curator, which allows students to submit different types of assignments (i.e. not only programs), and which uses textual comparison as an automatic marking method.

2.8. Summary

This chapter presented an overview of assessment in education and outlined the different aspects of each assessment type. It provided an introduction to computer-assisted assessment (CAA) and defined the various types of CAA, dedicating significant attention formative, summative and diagnostic assessments. Furthermore, it illustrated the Bloom's taxonomy and highlighted the different types of assessments, where it discussed diagram and programming language assessments in some detail. It also examined the general strategy for automatic marking based on meaningful components and further examined the construction of automated marking assessments. Several research study focusing on diagrams and programming languages assessments were highlighted, which aim to help reduce the workload of lecturers and enhance the feedback delivery.

There are many opportunities offered by computer-assisted assessment for both formative and summative assessments. The students benefits from timely and specific feedback on their learning and get chances to practice skills. Lecturers can use CAA to enhance assessment methods, whether a paper-based or automated approach is adopted. In this chapter, the significance of semi-automatic assessment role was described, along with formative assessment, which in basic term, is dependent on computer-assisted assessment. Although, the semi-automatic assessment approach provides an improved individual and detailed feedback on formative assessment comparable to fully-automatic assessment; the existing semi-automatic assessment systems have suffered from poor feedback consistency.

This is because the large number of students in the classroom can often cause human markers to generate inconsistent marking and feedback. Alternatively, the human marks should target to reduce or remove as many of the repetitive tasks in any phase of the marking process as possible to provide consistent and effective feedback to students. One of the repetitive tasks is the re-use the same mark for identical parts of students' solutions. The Structured Query Language (SQL) shares common features with other programming languages that make it acceptable to be marked automatically. In this research, SQL was selected as the basis of this research, as described in detail in Chapter 3. The chapter provides a literature review on the automated assessment of SQL and analyses the ideal SQL marking system.

Chapter 3. Literature Review

Automatic Assessment of SQL

3.1. Introduction

Numerous theories have been proposed to explain what motivates the SQL assessment process for example, (Fehily, 2010; Kleiner, Tebbe and Heine, 2013; Kleerekoper and Schofield, 2018). This chapter reviews the literature on the manual SQL system process and analyses the required SQL marking system that should be used in this research. In addition, a survey of existing automated SQL assessment tools is presented, demonstrating the difficulties in learning and assessing SQL queries. Review of existing SQL learning and assessment tools and their features is also provided. Furthermore, this chapter also examines various types of knowledge bases for more efficient problem solving methods. The methodologies used in these knowledge-based systems include the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems. The literature suggests that the integration of both systems results in a suitable environment for the semi-automatic assessment of the SQL statements.

This chapter starts by providing an overview of the Structured Query Language (SQL) in Section 3.2, where this section demonstrates the process of acceptable SQL assessment marking and SQL grading techniques. Section 3.3 explains the various difficulties in learning and assessing SQL queries, while Section 3.4 reviews the existing SQL learning and assessment tools and their features in detail. Subsequently, a summary of the existing SQL tools is presented in Section 3.5, before introducing the two types of Artificial Intelligent (AI) systems used in education in Section 3.6. Finally, Section 3.7 provides a short summary of the chapter.

3.2. Structured Query Language (SQL)

A relational database management system (RDBMS) shifts and stores data into a database and retrieves it so that applications can manipulate it (Bruno, 2003). According to Rob et al. (2008), RDBMS is a set of both logical and physical operations. The logical operations are applications, which specify the required content; for example, an application requests an employee's name from a table.

However, the physical operations define how things should be performed and built in the database. For instance, foreign aspects are used to identify relationships between tables. An RDBMS allows users to specify queries through the use of high-level declarative languages, such as “SQL” the abbreviation for Structured Query Language. The SQL is the standard querying language for relational databases (Litoriya and Ranjan, 2010). According to Kleerekoper and Schofield (2018) SQL is easier to learn than languages like Java or Python, where it is syntactically smaller and more structured. Abelló et al. (2008) argued that SQL, which is comprised of commands to define schema structures (i.e. tables), is the main database (DB) language that is used to perform tasks such as update data on a database or retrieve data from a database. A database mostly contains one or more tables, and each table is identified by a name (e.g. "EMP" or "DEPT"). Furthermore, each table contain columns (fields) and records (rows) of data relationships, as illustrated in Figure 3-1.

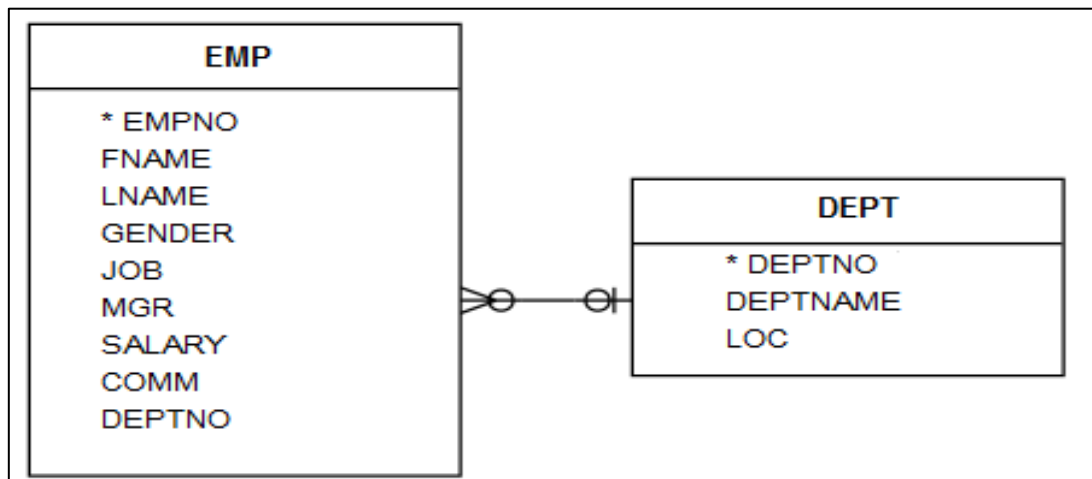


Figure 3-1: The relationship between EMP and DEPT tables

Figure 3-1 shows the primary key of the EMP table is the employee number (EMPNO), and the primary key of the DEPT table is the department number (DEPTNO). In addition, the department number (DEPTNO) in the EMP table is a foreign key that references the primary key of the DEPT table (DEPTNO). The SQL statements for creating both tables are as follows:

- First, create **Department** “DEPT” table as:

```

CREATE TABLE DEPT
(DEPTNO NUMBER CONSTRAINT DEPT_DEPTNO_PK PRIMARY KEY,
DEPTNAME VARCHAR2(15),
LOC VARCHAR2(30));
  
```

- Second, create **Employee “EMP”** table as:

```
CREATE TABLE EMP
(EMPNO NUMBER CONSTRAINT EMP_EMPNO_PK PRIMARY KEY,
FNAME VARCHAR2(15),
LNAME VARCHAR2(15),
GENDER VARCHAR2(10),
JOB VARCHAR2(20),
MGR VARCHAR2(15),
SALARY DECIMAL(7,2),
COMM NUMBER,
DEPTNO NUMBER,
CONSTRAINT EMP_DEPTNO_FK FOREIGN KEY (DEPTNO)
REFERENCES DEPT (DEPTNO));
```

The SQL create statements illustrates that every foreign key value in the DEPTNO column of the EMP table matches a primary key value in the DEPTNO column of the DEPT table. This relationship can be explained as follows:

"Each employee works for utmost one department, but many employees may work for the same department."

SQL is used to access and manipulate data in a database (Raadt *et al.*, 2007) and has become the most widely used relational database language (Melton, 1993). It was released in the early 1970s by (Codd, 1970), who proposed a new model for database systems called the “Relational Model”. SQL was standardised by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) in the early 1980s. The standard SQL syntax that is used to interact with relational databases contains different clauses, functions and expressions. SQL clauses in relational databases are; CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. SELECT and FROM are essential components of SQL statements and include WHERE, GROUP BY, HAVING and ORDER BY as optional clauses. Expression produces scalar values or tables consisting of columns and rows of data. SQL statements can retrieve data from different database tables using the following SQL syntax (Donahoo and Speegle, 2010):

SELECT < list of columns>	Mandatory
FROM <table list>	Mandatory
(WHERE <row condition>)	Optional
(GROUP BY <group list>)	Optional
(HAVING <group condition>)	Optional
(ORDER BY <sort list>)	Optional

3.2.1. SQL Assessment

Many researchers are trying to address the issue of the manual assessment process of SQL queries. Ke et al. (2009) listed the different challenges of manual SQL assessment, which include the fact that students cannot get the feedback on their work immediately. In addition, Ke and other follow researchers stated that manual grading wastes a lot of time for lecturers and might cause human mistakes. SQL marking process might share common features with other programming languages, which can be utilised to make it acceptable and useful. These features mainly support the function and performance of the SQL marking process. Furthermore, these features include output comparison, which is the most popular feature and is used in various systems. Ala-Mutka (2005) indicated that output comparison could include running the model solution and students' code. Checking SQL syntax is the most commonly reported way to define tests, and is also the most important part of the process (e.g. compiling the program, running the code and comparing the output with the expected (model) output) (Tremblay and Labonté, 2003).

A feedback mechanism is used to provide individual students with information that is focused on their SQL learning performance (Walker, 2011). There are many benefits of an effective feedback, such as improving the students' progress, motivation and confidence, and enhancing their achievements. In addition, consistent marking and grading can accurately indicate the level of performance which has been achieved by a student (Thompson and Ahn, 2012). In addition, the use of grades might affect the students' learning, since they can provide a standardized measure of student's performance, certify that a course of study has been completed and particular standards have been achieved (Thompson and Ahn, 2012).

3.2.2. SQL Assessment Grading

In computer science education, communication between the lecturer and the students is an important component that should involve an effective feedback process and consistent grading (Noonan, 2006). Frequently, students direct their efforts based on what is assessed and how it affects the final grade (Ihantola *et al.*, 2010). Multiple sclerosis researchers have implemented an improved SQL grading process that could satisfy both students and lecturers. There are some requirements that can enhance the SQL grading process, such as feedback quality, response time, accuracy, consistency and flexibility (Bruno, 2003).

Feedback is an important component of formative assessment which helps students to develop a deeper understanding of their learning (Clark, 2011). A model feedback process allows students to receive an accurate score of their work shortly after submitting it, with detailed breakdown of areas of improvement and non-functionalities (Bruno, 2003). Accuracy and fairness in grading are other obvious requirements since they increase the students' motivation once they receive accurate marks on their submissions (Karavirta *et al.*, 2007). Consistency in grading is another important factor in enhancing the SQL grading process, since it can increase the consistency of marks allocation and help to reduce the marking load of lecturers by granting them more effective teaching tasks (Dekeyser *et al.*, 2007). The flexibility of grading SQL-based exercises is an essential feature that allows students to resubmit their work and provide them with the ability to check the correct solutions that are stored in a database (Prior and Lister, 2004).

3.3. Difficulties in Learning and Assessing SQL

Several researchers have recognised a number of difficulties involved in learning and assessing SQL. Some of these difficulties can be summarised as follows. SQL is different from some other query languages in that it is non-procedural (Dekeyser *et al.*, 2007). This means that students only have to specify what data they want to extract from the database, and do not have to worry about how the data is stored, or how to go about retrieving it. Incorrect tables and attributes lead students to memorise the full table schema, resulting in problems while practicing SQL statements (Kearns *et al.*, 1997). These problems can mislead the students to focus on the SQL syntax, and guide them to a different direction that does not give them the right answer. Understanding the basic of SQL syntax is of great significance, and is considered to be the first step of learning SQL (Kenny & Pahl 2005).

However, students may misunderstand the basic elements of SQL. The reason behind that could be that they have trouble in mastering the basic SQL concepts, such as joining functions, aggregation and grouping, and operators (Mitrovic, 1998). To practice SQL statements, students need to understand the requirements of the SQL questions. However, they may still incorrectly express the final output of the queries. These difficulties have motivated several researchers to develop numerous SQL tutoring and assessment systems, which are listed in the next section. Tools that provide various forms of support for assessing SQL statements do exist, some of which are subsequently discussed in detail. Such tools can help this research to find the ideal automated marking process and provide better grading schemes for a new SQL assessment environment.

3.4. Existing SQL Learning and Assessment Tools

This section introduces various tools for learning and teaching SQL that have been implemented to assess SQL statements. These tools have been classified into two categories such as formative assessment that used to monitor student learning and summative assessment which used to evaluate student learning and submit grading with feedback for example a midterm exam, final projects and final exams. This research focuses on different types of tools that might be used as learning or assessing the SQL. However, most of those tools focus on the functionality of the tool itself not on the student common mistakes when they are writing the SQL queries. The following are combination of summative and formative assessment SQL tools; for instance:

1. **SQL Tester** "An online SQL assessment tool and its impact" by (Kleerekoper and Schofield, 2018).
2. **SQLg** "Automated grading and tutoring of SQL statements to improve student learning" by (Kleiner *et al.*, 2013).
3. **SQL-KnoT** "An Open Integrated Exploratorium for Database Courses" by (Brusilovsky *et al.*, 2008).
4. **SQLify** "*Do students SQLify?*" titled as "*Improving learning outcomes with peer review and enhanced computer assisted assessment of querying skills*" (Raadt *et al.*, 2006)
5. **ActiveSQL** "Automatic Checking of SQL: Computerised Grading" (Cumming and Russell, 2005).
6. **SQLator** "SQLator: an online SQL learning workbench" (Sadiq *et al.*, 2004).
7. **AsseSQL** "Online Assessment of SQL Query Formulation Skills" (Prior, 2003).

3.4.1. SQL Tester

SQL Tester tool was introduced by Kleerekoper and Schofield (2018). It has been implemented to reduce plagiarism, motivate deep learning and provide students with accurate tasks and assess their performance in person to provide timely formative and summative feedback. The SQL Tester marks an answer as correct if it exactly matches the desired output. The students may make as many attempts as they wish in that time, and after every attempt they are shown the output of their query.

The main strength of the SQL Tester is students have been engaged strongly with new tool for learning and has motivated them to revise where it has affected their final marks. On the other hand, there are two main drawbacks when using the SQL Tester; first, match exactly the student answer with the reference answer which doesn't give chance for student to think about different way of solving same query. Second, when marking the SQL statements it gives only error message from the Relational Database Management System (RDBMS) which no extra feedback was given to students to understand the error message.

3.4.2. SQLg (SQL-Statement Grader)

SQL-Statement Grader has been introduced by Kleiner et al. (2013). It is an automatic grading and tutoring of SQL statements used to improve student learning. Based on the feedback generated by the SQLg, the student can repeat the solution as much as they want until they get satisfy with the resulted quires. After the completion of the grading process the reporter generates an XML formatted report. This report can be converted by an XSLT style sheet. By default the report is converted to XHTML, but the WebCAT plug-in package comes with a specialised transformation file to embed the reports into the WebCAT user interface. The following example demonstrates the process of SQL Grader evaluation which has been taken from Loughborough University database module exam script specifically SQL questions.

Question: Display “Employee name of department earned commission comm” for each salesman in reverse commission order and the year of the hire date as Hired.

Model Answer:

```
SELECT EMPNAME || ' OF DEPARTMENT ' || DEPTNO || ' ERRAND COMMISSION  
' || COM, TO_NUMBER (TO_CHAR(HIREDATE, 'YYYY')) HIRED  
FROM EMP  
WHERE JOB= 'SALESMAN'  
ORDER BY COMM DESC;
```

Student Answer:

```
SELECT EMPNAME || ' OF DEPARTMENT ' || DEPTNO || ' ERRAND  
COMMISSION ' || COM, TO_CHAR(HIREDATE, YEAR) HIRED  
WHERE JOB= 'SALESMAN'  
ORDER BY COMM;
```

The following are SQLg evaluation steps of individual solution:

- A. The SQL statement will be loaded with the model answer.
- B. The student solution does not have any forbidden elements however; it doesn't match exactly the model answer.
- C. From the above example, the student answer has different syntax and result set from the model answer, therefore SQLg will count the syntax as it contains errors which mean the statement will be discarded. The student will receive a message including the original database *Error: ORA-00904: "YEAR": invalid identifier.*

- The student will be given another attempt to fix the syntax where she/he need to change:

From: To_Char (Hiredate, Year) **to:** To_Char (Hiredate, 'YYYY')

- D. Now the syntax check succeeded so the grader proceeded with the cost check. After that, SQLg confirm that the column count and data type were corrected. Another error is deducted when character value returned instead of numeric value.

- The following message was shown:

➤ *Datatype of column 2 is wrong. Expected: Number, your solution: Varchar2*

- The student will change the error to: To_Number (To_char (Hiredate, 'YYYY'))
- The last error the SQLg has identified is the student didn't sort the comm in descending order, he/she sorted with ascending therefore, the student will be given another chance to change from:

From: ORDER BY COMM **to:** ORDER BY COMM DESC

- E. The student will receive full mark after he/she made the changes needed. In any case, there were still other changes then that would be marked manually by the instructor.

A major strength of SQLg is that this system has high quality evaluation process which has follow different steps to evaluate and analyse the solution and give accurate marks for the student. In addition, the feedback concerning syntax was very clever to help student identify their errors and give them chance to update them. On the other hand, it might have been helpful to provide more details of manual marking (Part (E)) and update point since the system is working as semi-automated marking. Also, the tool only focus on helping student practice the SQL statements before the real assessment can be conducted. It doesn't help instructors to conduct exams.

3.4.3. SQL-KnoT (Knowledge Tester)

Knowledge Tester is an integrated tool for SQL learning that generates questions which require a student to write an SQL query for a sample database which evaluates the correctness of students' answer and provides a student with feedback. Every time a student accesses a SQL-KnoT question, the actual question text is generated by corresponding template from the predefined database (Brusilovsky et al. 2008). Brusilovsky et al. (2010) stated that to be evaluated as correct, the student solution must always produce the same result as the model solution. For that reason, SQL-KnoT compares the result produced by the student solution with the result produced by the pre-stored correct model answer. If a student fails to answer an SQL-KnoT question, he/she can open SQL-Lab to run and debug previous solution. The SQL-Lab allows students to formulate and execute queries, observe their results, and test performance of SQL scripts. The following example demonstrates the process of SQL-KnoT evaluation which has been taken from Loughborough University database module exam script specifically SQL questions.

Question: "Display the department number and total salary of employee in each department that employs five or more people".

Model Answer:

```
SELECT DEPTNO, SUM(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING COUNT (EMPNO)>=5;
```

Student 1 Answer:

```
SELECT D.DEPTNO, SUM(E.SAL) AS "TOT SAL"
FROM DEPT AS D EMP AS E
ON D.DEPTNO= E.DEPTNO
GROUP BY D. DEPTNO
HAVING COUNT (EMPNO)>=5;
```

Comments:

- The SQL-KnoT grading system will show as: Correct
- Reason: result data set of the student 1 answer is exactly same as the model answer.

Student 2 Answer:

```
SELECT DEPTNO, TOTAL (SAL)
FROM EMP
GROUP BY DEPTNO
HAVING COUNT (DEPTNO) >=5;
```

Comments:

- The SQL-KnoT grading system will show as: Incorrect
- Reason: The student 2 has added an incorrect function name (Total) on the SQL syntax and all the answer has been counted as wrong.

From the above example, the SQL-KnoT has maintained the main aim of marking system which stated as to be evaluated as correct, the student solution must always produce the same result as the model solution and for that student 1 has graded as correct and student 2 as incorrect. On the other hand, SQL-KnoT has shown fairly limited grading system since the evaluation should contain the SQL syntax with the result dataset. The student 2 has tried to answer the query but since the grading system is just constraining on the dataset the syntax has not been checked and student 2 should repeat the full process to know his/her error.

3.4.4. SQLify

It has been developed by Raadt, Dekeyser and Lee (2006) to enhance automatic assessment and semantic feedback. The aims of introducing SQLify are deliver high quality learning experience for students, consistent assessments grades and reduce instructors' marking load. The SQLify is evaluating the SQL queries by following the determining value of Conjunctive queries. Table 3-1 describe the instructor procedure to apply the mark suggested by SQLify.

Table 3-1: Instructor procedure to apply the mark suggested by SQLify

Level	Conjunctive Queries	Description
L0 or L1	$sys \leq L1$	The submission is incorrect
L2, L3, L4	$sys = L2 \wedge L2 \leq std1 \leq L4 \wedge L2 \leq std2 \leq L4$	The submission is largely incorrect
L2, L3, L4 or L5	$sys = L2 \wedge \neg(L2 \leq std1 \leq L4 \wedge L2 \leq std2 \leq L4)$	There is a conflict between reviewers and the system
L0, L2, L6 and L7	$sys = L6 \wedge (std1 \leq L4 \vee std2 \leq L4)$	The system suggests the answer is probably correct but the reviewers disagree.
L7	$sys = L6 \wedge std1 \geq L5 \wedge std2 \geq L5$	The system thinks the query is probably correct and the reviewers agree
L7	$sys = L7$	The system indicates that the answer is certainly correct

The following example demonstrates the process of SQLify evaluation which has been taken from Loughborough University database module exam script specifically SQL questions.

Question:

Display the names and jobs of all Employees who have the same jobs as Employees in the sales department and earning more than 800 Pounds.

Model Answer:

```
SELECT EMPNAME, JOB
FROM EMPE, DEPARTMENTD
WHERE E.DEPTNO = D.DEPTNO
AND DEPTNAME = 'SALES' AND SAL>800;
```

Table 3-2: Two correct query solutions and one incorrect in CQ class and their Evaluation

Student No	Student Answer	Grade	Sys	Std1	Std2
Student 1	SELECT EMPNAME, JOB FROM EMP JOIN DEPT ON DEPTNO WHERE DEPTNAME= 'SALES' AND SAL>800;	Correct	L7	L6	L7
Student 2	SELECT EMPNAME, JOB FROM EMPE WHERE SAL>800 AND EXISTS (SELECT * FROM DEPTD WHERE E.DEPTNO= D.DEPTNOAND DEPTNAME= 'SALES');	Correct	L7	L7	L4
Student 3	SELECT EMPNAME, JOB FROM EMPE WHERE DEPTNAME= 'SALES' AND SAL> 300;	Incorrect	L2	L6	L4

Table 3-2 shows that student 1 and student 2 have two different answers which are not exactly like the model answer. However, both of them are semantically correct. The evaluation has been performed using the conjunctive queries and three variables are declared per submitted query for each student which they are sys, and two correctness marks from peers student 1 and student 2. Whereas student 3 has got problem solving the query and his solution is incorrect to the above query. These results are consistent with the aims of the SQLify research which targeted as increase the quality of learning experience, consistent assessments grades and reduce instructors' marking load. The research example has discussed the professional criteria of evaluating the SQL statements with high consistency between the students.

However, it might have been more accurate while marking the student answer since it should go through different reviewers who are having different criteria in solving SQL queries. Moreover, there is no explanation of evaluating the data set of the SQL queries and if the system can do perform that task or not.

3.4.5. ActiveSQL

Cumming and Russell (2005) introduced ActiveSQL as an integrated learning environment that provides SQL tutorials, supports online assessment and offers immediate feedback after analysing results. A percentage is used for measuring performance by the ActiveSQL grading system. It starts at 0% if the student did not attempt an answer, and 100%, if the student answered the question perfectly. To calculate the student performance, Russell and Cumming (2004; 2005) compared the student’s output solution with the model output solution. Subsequently, the rows and columns that specify additional data are highlighted, and the percentage is calculated as; the proportion of correct cells (without including the header) against the higher of either the total cell count of the sample solution of the total cell count of the student answer. It can be measured by using the example in Table 3-3:

Table 3-3: Example of ActiveSQL Marking Grading System

StdNo	StdName	LectCode
100	ABC	1
924	CEW	1
325	JOL	2
123	ANY	3

Total cell count of correct student answer / (Total cell count of the student answer * Total cell count of the model answer)	
The accuracy measure:	$= 2/(3*4)$ $= 0.1666*100$ $= 16.6$ $= 17\%$

The advantage of the percentage approach is that as more filtering is added to the student query, the number in general will rise. However, ActiveSQL lacks additional grading criteria that follow the syntax and find out the difference between the students and model solutions.

The percentage grading system does not identify the level of understanding of the student since the feedback is given to the student is not enough and does not provide the correct answer of the SQL query.

3.4.6. SQLator

SQLator is a web-based interactive tool for learning SQL. It has been introduced by (Sadiq et al. 2004). The evaluation process of the SQLator does not check or analyse the SQL syntax but it will generate a direct estimation which is either a correct or incorrect statement. The student will have the chance to do various attempts until finding the correct statement otherwise they can access the correct solutions from the SQLator database. It reduces the marking time, increases the efficiency since it's marking them automatically and provides immediate binary feedback to the learners. The learner selects a query to work on and writes an SQL statement to solve the selected query. SQLator evaluates the SQL statement and provides the result; either correct or incorrect (Sadiq et al. 2004). The feedback which has been given to student is only says correct which means the syntax and data set are exactly like the model answer. If the student made any different way of writing the syntax or the data set has slightly been changed so the result will be as incorrect which leads to unknown feedback that will be sent to the student. In addition, the student can choose the queries depend on their difficulties for instance simple, advanced and hard. The system doesn't mentioned what type of questions are classified as simple or advanced or hard which makes the student try all of them to know what is suitable for their abilities.

3.4.7. AsseSQL

It has been introduced by Coleman and Lister (2004) to examine the effect of grading of queries submitted by students. It generates immediate feedback on validity of the students' solution that would guide them to what corrections they need to make to their query. The programme marks the student's answer using a pattern matching system where it compares the data set produced by the execution of model answer to the data set that results from the execution of student's answer. If the data set are exactly the same, the student's answer is flagged as correct otherwise it is flagged as unsuccessful attempt. Also if the submitted answer is syntactically incorrect, an error message is displayed. Coleman (2004) has detailed some of challenges and issues using the AsseSQL such as; the marking of the test is binary which means either correct or it's incorrect.

If the student's answer is partly correct then no marks are allocated. The RDBMS error messages are unclear or unhelpful means doesn't give detailed feedback about what kind of error they made. The following example demonstrates the process of SQLator and AsseSQL evaluation which has been taken from Loughborough University database module exam script specifically SQL questions.

Question: "Display the names of all employees who work in a department that employs an analyst".

Model Answer:

```
SELECT EMPNAME
FROM EMP
WHERE DEPTNO IN (SELECT DISTINCT EMP.DEPTNO
                 FROM EMP INNER JOIN DEPT
                 ON EMP.DEPTNO=DEPT.DEPTNO
                 WHERE JOB='ANALYST');
```

Student 1 Answer:

```
SELECT EMPNAME
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
                WHERE JOB='ANALYST');
```

Comments:

- The SQLator and AsseSQL grading system will show as: Correct
- Reason: result data set of the student 1 answer is exactly same as the model answer.

Student 2 Answer:

```
SELECT EMPNAME, DEPTNO
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
                WHERE JOB='ANALYST');
```

Comments:

- The SQLator and AsseSQL grading system will show as: incorrect
- Reason: the student 2 has data set same as the model answer however, additional data will be displayed which means it doesn't match the original dataset therefore the evaluation will be presented as incorrect.

The demonstrations of the above examples have shown that AssessSQL and SQLator tools might affect the evaluation process since the system doesn't give fare marks distribution among the students. That because the requirement of the question is asking for IN and INNER JOIN and that student should be given different mark from others who didn't add them. Furthermore, only the difference between first answer and second answer is the data set will have more additional data even if the original answer is there. The student will get incorrect answer because his/her answer doesn't match exactly the model answer. They didn't give any detailed feedback why it's wrong which will lead the student to try different attempts till they know what errors they attempt.

3.5. Summary of Existing SQL Tools

It is clear from the above literature that most research papers have limited their descriptions of the tools' features. Therefore, the literature review only points out and compares the features that have been highlighted by the authors. Table 3-4 demonstrate the summary of existing SQL assessment and learning tools which divided into fully-automated and semi-automated marking systems. It displays the evaluation of several tools with different features starting from 2018 going back to 2003.

Table 3-4 shows that matching reference solution features are common between most of the systems except the SQLify system. It also shows that checking the forbidden element in the SQL statement is an important feature, however; only the SQLg is implementing it. This would help to identify unnecessary elements of the SQL query which can effect on the syntax or dataset. On the output result feature only SQLg and SQL Tester which does not use it since both of them are not considered only the last output answer; however, they care about the way student answer the SQL statements. The grading process of each system differs, for example the SQL-KnoT, AsseSQL and SQLator are using the binary technique which gives either correct or incorrect final result. Checking the SQL syntax is major process in SQL grading process since the query cannot be evaluated and tested without making sure that SQL syntax is working perfectly.

Table 3-4: Features Evaluation of Existing SQL Assessment and Learning Tools

Tools \ Features	Check Forbidden Elements	Check SQL Syntax	Matching Reference Solution	Output Result	Grading/ Feedback	Advantages	Disadvantages
1. SQL Tester (Kleerekoper and Schofield, 2018)	✗	✓	✓	✓	Correct /Incorrect	<ul style="list-style-type: none"> Engaged strongly with SQL Tester tool. Motivated students to revise and practice before the real assessment. 	<ul style="list-style-type: none"> Match exactly the student answer with the reference answer. No extra feedback was given to students.
2. SQLg (Kleiner, Tebbe and Heine, 2013)	✓	✓	✓	✗	Score	<ul style="list-style-type: none"> System has high quality evaluation process. Give accurate marks for the student. Feedback concerning syntax to help student identify their errors 	<ul style="list-style-type: none"> The SQLg tool focuses on helping student practice before the real assessment. Might be helpful to provide more details of manual marking.
3. SQL-KnoT (Brusilovsky <i>et al.</i> , 2010)	✗	✗	✓	✓	Correct /Incorrect	<ul style="list-style-type: none"> If a student fails to answer an SQL-KnoT question, they can open SQL-Lab to run and debug previous solution 	<ul style="list-style-type: none"> Shown fairly limited grading system. Constraining on the dataset the syntax.
4. SQLify (Raadt, Dekeyser and Lee, 2006)	✗	✓	✗	✓	Conjunctive Queries	<ul style="list-style-type: none"> Deliver high quality learning experience for students. Consistent assessments grades. Reduce instructors' marking load 	<ul style="list-style-type: none"> Go through different reviewers who are having different criteria. No explanation of evaluating the data set of the SQL queries
5. ActiveSQL (Cumming and Russell, 2005)	✗	✓	✓	✓	Percentage	<ul style="list-style-type: none"> Percentage approach filtering is added to the student query. 	<ul style="list-style-type: none"> Percentage grading system does not identify the level of understanding. Feedback given to the student is not enough.
6. SQLator (Sadiq <i>et al.</i> , 2004)	✗	✗	✓	✓	Correct /Incorrect	<ul style="list-style-type: none"> Reduces the marking time. Increases the efficiency of marking. Provides immediate binary feedback to the learners. 	<ul style="list-style-type: none"> The system doesn't mentioned what type of questions. The feedback which has been given to student is only says correct.
7. AsseSQL (Prior, 2003)	✗	✗	✓	✓	Correct /Incorrect	<ul style="list-style-type: none"> Generates immediate feedback The programme marks the student's answer using a pattern matching system. 	<ul style="list-style-type: none"> Marking of the test is binary which means either correct or incorrect. The RDBMS error messages are unclear or un-descriptive

A review of some of existing SQL tools shows that SQL-KnoT, AsseSQL and SQLator have the same main feature, which is that a student's solution must always produce the same result as the model solution. This feature is implemented as part of the grading process, where students' solutions can be graded either as correct or incorrect answers. ActiveSQL and SQLator are more similar to one other, where both of them cannot check the syntax of the SQL and do not check for forbidden elements. Dekeyser et al. (2007) stated that SQLator and AsseSQL are apply only binary grading to queries submitted by students and do not provide comments or suggestions for improvement. Also, they declared that both AsseSQL and SQLator create only single channel of communication between the student and the instructor via the system.

On the other hand, SQLg, SQLify and ActiveSQL tools exhibit more professional features with higher quality of grading and student feedback. SQLg offers more features with much enhanced qualities since the statements are checked starting from forbidden elements until matching reference solutions. Furthermore, SQLator employs a manual check after finishing the grading process to ensure consistency while marking. SQLify and ActiveSQL have some similarities, however; SQLify offers higher standards in checking semantic SQL statements before matching it with the reference model. However, the above systems focus on the final submitted answers without providing any detailed feedback as to why it an answer is wrong.

This often leads the student to make different attempts until they know what errors they have been making as with AsseSQL and SQL Tester students may address questions in any order and makes as many attempts as they want within the time. The SQL Tester have many similarities with AsseSQL tool for example choice of categories of SQL questions, number of attempts and students' solutions can be graded either as correct or incorrect answers. Despite the SQL Tester is displaying the schema of the relevant database along with questions and model answer output, the AsseSQL is not showing the schema. Prior (2003) stated that one of the challenges of using AsseSQL is that marking of tests is binary, which means that solutions are evaluated as either correct or incorrect. If the student's answer is partly correct, then no marks are allocated. As such, AsseSQL might affect the evaluation process since the system does not distribute marks fairly among students.

3.6. Artificial Intelligence in Education

Artificial intelligence (AI) in education has attracted significant research attention, as it promises to improve education quality and enhance traditional teaching and learning methods (Luger and Stubblefield, 1998). AI programs, referred to as Intelligent Tutoring Systems (ITS), can imitate the reasoning of human behaviour in solving a knowledge intensive problem of teaching and learning (Mitrovic, 2003).

They therefore have the potential to make a significant effect on learning by performing routine education tasks such as marking assessments and providing students with feedback (Jackson, 1996). The two main AI-based education technologies included in this research are the Case-Based Reasoning (CBR) and the Rule-Based Reasoning (RBR) systems. Bichindaritz *et al.* (1998) argued that CBR and RBR have emerged as two important and complementary reasoning methodologies in artificial intelligence (AI). CBR and RBR use knowledge and problem solving skills along with previous design experience to solve design problems in education. CBR and RBR are explained in detail in the following subsections.

3.6.1. Case-Based Reasoning (CBR)

"A case-based reasoner solves new problems by adapting solutions that were used to solve old problems." (Riesbeck and Schank, 1989)

Case-Based Reasoning (CBR) is one of the several computational models in the field of artificial intelligence (AI) being considered for solving design problems (Aamodt and Plaza, 1994). The aim of this model is to find solutions for various design problems by exploring databases that store similar design cases and models (Riesbeck and Schank, 1989). As a technique to solve new problems based on previous successful cases, CBR represents significant prospects for improving the accuracy and effectiveness of solving unstructured decision-making problems. The reasoning process can be summarised using the following four basic stages in the CBR cycle: **Retrieve**, **Reuse**, **Revise**, and **Retain**. In CBR, the handled structures are known as cases that represent a problem situation. The four basic stages are known as the CBR cycle, as illustrated in Figure 3-2.

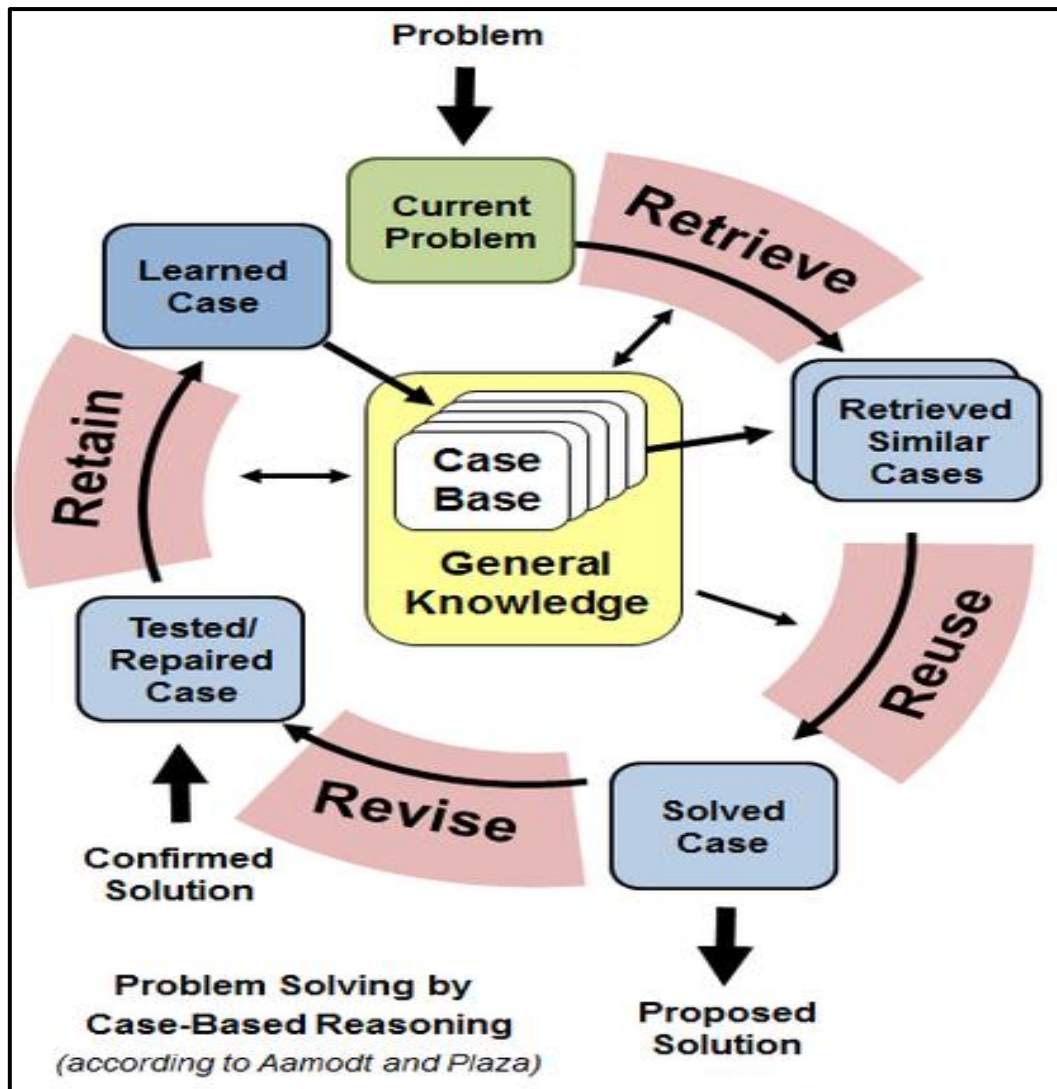


Figure 3-2: The CBR cycle according to (Aamodt and Plaza, 1994)

Aamodt and Plaza (1994) presented CBR using four stages; first, a new case (problem) is executed in the system, before it proceeds to identify the matched cases and “retrieve” the most similar cases from the knowledge base. As soon as the cases are obtained, the system may solve the new problem by adapting their existing solutions to solve the current problem, in a stage named as “reuse”. However, the differences between the two problems must be considered before adapting the same solution. Subsequently, the “revise” stage can be proposed once the reuse stage is done, to suggest a new solution on which a new case can be built. Finally, in the “retain” stage, the new case can be retained for future problem solving.

According to the CBR cycle, the reuse of similar problems from previous experiences can be adopted in the marking process. This means that human markers can utilise the same marking and feedback using similar previous problems and their solutions. Therefore, the recent and previous problems are matched and compared to find similarities, and the most suitable answer found is then adopted. CBR increases problem solving efficiency since it compares the stored solutions with similar cases and stores them for future use (Kolodner, 2014). Some research studies have adopted the CBR approach in the marking process, resulting in an increased system efficiency of the marked components and a reduction of lecturers' workload as a result of avoiding marking identical components. For example, Batmaz (2011) adopted the CBR method for a semi-automatic diagram marking process that focuses on partial marking. Batmaz focused on semi-automated diagram marking, which can reduce the number of diagrams marked by the examiner by identifying the identical components in different student diagrams. Adesina *et al.* (2013) used a multi-touch drag-and-drop style tool to solve basic arithmetic problems. In this research, CBR was used to compare the previous solutions with the current problem to mark the student's mathematical answers. Buyrukoglu (2018) adopted CBR to utilise the similarities of Python programming scripts by adopting previous student answers and comparing them to the current student script.

Overall, these research studies show that the CBR system has been adopted to allow a consistent marking process for different types of assessments, such as diagrams, mathematics and computer programming.

3.6.2. Rule-Based Reasoning (RBR)

The Rule-Based Reasoning (RBR) represents knowledge in terms of a set of rules that provide instructions to express what to do or how to conclude different situations (M. Cabrera and Edye, 2010). It is a framework imitating human reasoning in solving different knowledge intensive problems (Bichindaritz *et al.*, 1998). Hopgood (2012) stated that RBR examines and analyses a certain form of all rules, where it will be activated and executed at the same time. This means "if a rule's condition is met, then the rule will take place and be applied. If there is any rule condition that did not receive any action, then other rules will be considered, where a further check for more rules will commence", as illustrated in Figure 3-3

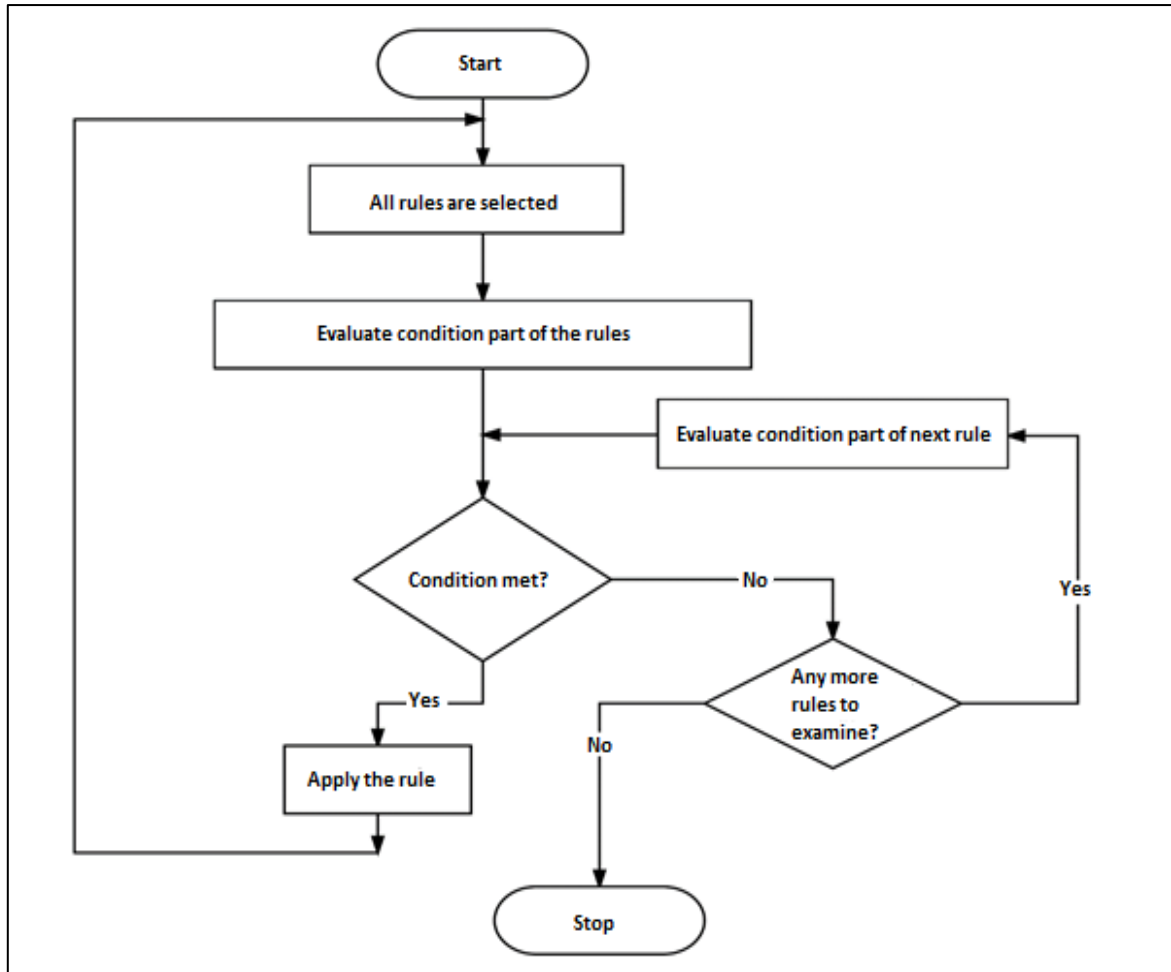


Figure 3-3: Forward-chaining with “first come, first served” (Hopgood, 2012)

A rule in artificial intelligent (AI) can be defined as an **If/Then** structure that provides some description of how to solve a problem. Such a rule consists of two parts: the **If** part, which is called the condition or antecedent, and the **Then** part, which is called conclusion or action. Table 3-5 shows the general form of a rule (Sasikumar et al., 2007).

Table 3-5: Rules Conditions

AND		OR		Both AND/OR	
If	condition 1	If	condition 1	If	condition 1
AND	condition 2	OR	condition 2	AND	condition 2
AND	condition 3	OR	condition 3	OR	condition 3
.....		
THEN	action 1, action 2, ...	THEN	action 1, action 2, ...	THEN	action 1, action 2, ...

This can be explained as “If a certain condition is true, then a particular result happens”. A rule can have multiple conditions joined by the keywords AND (conjunction), OR (disjunction), or a combination of both. The condition list (condition 1, condition 2, condition 3, etc.) is evaluated based on what is currently known about the problem that should be solved. This means that each action of a rule typically checks if the particular problem occurrence satisfies some condition.

3.6.3. Integration of Rule-based Reasoning and Case-based Reasoning

There has been very little research in combining Rule-based Reasoning Systems and Case-based Reasoning. The only researches in this area are either medical systems or problem solving system which their approaches are very different from this research. However, some of these researches can be explained as examples of different areas. For example, in medical researches by (Berka, 2011; Cabrera and Edye, 2010; Bichindaritz *et al.*, 1998) where they have carried out on the development of a medical diagnostic system, using the Case-based Reasoning methodology. These researches were focused on the implementation of the adaptation stage, from the integration of Case-based Reasoning (CBR) and Rule-based Reasoning (RBR) Systems that allows reutilizing rules, contexts, integrity constraints, and cases and reasoning. Another research area is for problem solving in complex, real world situations, it is useful to integrate RBR and CBR. This research presents an approach to achieve a compact and seamless integration of RBR and CBR within the base architecture of rules (Dutta and Bonissone, 2013). Since there are limited numbers of researches discussing the integration between those two techniques (CBR and RBR), and most researches concentrating on either CBR or RBR for that, the primary contribution of this research is to integrate both Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems.

This can allow the adoption of a new marking technique that is based on reusing previous solutions for similar cases and formulating new list of generic marking rules. This may contribute towards providing students with consistent marks and feedback. Both RBR and CBR are very important reasoning methodologies to identify the similarities in SQL statements parts and enhance the marking consistency. The prototype of this thesis has the ability to identify several types of SQL assessment.

More specifically, it can distinguish the similarities between the SQL statements parts, allow the adoption of a new marking technique that is based on reusing previous solutions for similar cases using Case-based Reasoning, implement new marking rules for solving the query with different ways without depending on reference model using the Rule-based Reasoning and provide instant feedback for the students.

3.7. Summary

This chapter introduced several features of advanced marking and grading systems to enhance the SQL learning and teaching. Different features of existing SQL learning, assessing systems were discussed and the ideal features of the proposed solution were defined. Furthermore, the chapter examined the general strategy for automatic SQL marking based on formative assessment, and how automated marking assessments are constructed. One of the objectives of this research is to identify the problems with existing SQL learning and marking systems. This includes highlighting the limitations caused by manual marking, as well as defining the new proposed evaluation criteria.

The proposed solution is to integrate Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems, both of which need to be adopted in the new marking technique for reusing previous SQL solutions for similar cases in order to contribute towards providing students with consistent feedback. The chapter concludes with the results of applying SQL assessment approaches to enhance student SQL learning, and illustrates how the consistency afforded by automatic marking can overcome some of the drawback of human marking.

The following chapter describes the research methodology that was utilised to conduct the investigations detailed in this research.

Chapter 4.

Research Methodology

4.1. Introduction

This chapter describes the research methodology of this research. The main objective of this chapter is to define and enhance the scope of the research. Research methodology is an important part of answering research questions (Bryman *et al.*, 2011). The research methodology supports researchers in understanding the reasons and motivation of the research (Rowley, 2014). This chapter involves the selection of techniques used to gather and analyse data. It presents different research approaches, designs and data analysis methods that could be used in order to provide solutions for research objectives. In addition, the advantages and disadvantages of these approaches and methods are briefly examined in order to choose the ideal methodology for this research.

This chapter opens by discussing the three approaches to research in Section 4.2. Section 4.3 introduces the types of the research design; and which design was selected for this research. The chapter then discusses the research data collection and data analysis methods and their merits and drawbacks in Section 4.4 and 4.5, respectively. Section 4.6 discusses the ethical requirements standards of the research. Finally, Section 4.7 provides a summary of the chapter.

4.2. Research Approaches

Bryman *et al.* (2011) stressed the importance of selecting an appropriate research approach when conducting a research. The research approach helps the research to develop ideas and reasoning that can be adopted in the study and to simplify the research design framework (Slevitch, 2011). The three common research approaches are briefly defined below to clarify the selected approach of this research. The three research approaches are (a) deductive approach, (b) inductive approach, and (c) abductive approach.

- a) **Deductive approach:** is the approach that searches for evidence to prove or disprove a hypothesis. It begins by observing a pre-existing theory, from which the hypothesis, which depends on that theory, should be produced. Finally, the processes proceeds to test that theory (Greener, 2008).
- b) **Inductive approach:** this approach is different from the deductive approach since it starts from the specific to the general. This means that the inductive approach navigates the focus of the research from the particular to the general by investigating various research methods to generate a theory from the research (Slevitch, 2011).
- c) **Abduction approach:** this approach involves both inductive and deductive reasoning, and starts with an observation, before attempting to find the simplest and most likely explanation to be adopted (Bryman *et al.*, 2011).

➤ This thesis research approach:

This approach was chosen because it integrates between two reasoning approaches (inductive and deductive). As such, this should increase the flexibility abductive for adjusting any modifications needed and provide new ideas for the research (Cresswell, 2014). In addition, selecting this approach can assist the research in achieving the objectives of designing and testing the new formulation and marking SQL tools framework. As this research involve the design, implementation and testing of SQL Formulation and Marking tools, then the inductive approach will be adopted which initially develops the conceptual model to form the base for SQL data collection and analysis as discussed in details in Chapter 5. In this case, both the deductive and inductive approaches will be utilised for the implementation process of new tools as mentioned in Chapter 6 and Chapter 8. At the same time, both chapters will establish a series of hypotheses using the same approach so it can be tested and evaluated.

4.3. Research Designs

Researchers need to decide which type of study to conduct before selecting either a qualitative, quantitative, or mixed methods research to conduct (Bryman *et al.*, 2011). This research is based on two main research study designs; experimental research and survey research. The three research methodology designs used in this research; namely (a) qualitative, (b) quantitative, and (c) mixed methods research, are explained briefly below (Cresswell, 2014).

- a) **Qualitative research** is an approach of exploring the problem by relating it to individuals or a social or human group. It is the method that uses observation to gather non-numerical data (e.g. sound and images), which is usually obtained from case studies, interviews and observations.
- b) **Quantitative research** is the approach that emphasises the measurement of numerical data, which is analysed using statistical procedures, before being able to generalise and replicate the findings. The popular quantitative research methods include experiments and non-experimental approaches (e.g. surveys).
- c) **Mixed methods research** is an approach based on collecting data by using a combination of quantitative and qualitative research designs. This approach starts by collecting quantitative data from statistical data using a quantitative method (e.g. survey), and then proceeding to gather qualitative data using a qualitative method (e.g. interview).

➤ This thesis research design:

The multiple method will be used in this research. This is because the multiple method utilises the advantages of both the quantitative and qualitative methods, which offers the researcher a chance to examine the research problem through a variety of ways. Furthermore, interpretation is continual and influences all stages in the research process. This permits the provision of detailed information on the examined study. The following explains the research methods selected and how they affect the results of each research stage.

The quantitative method was used in the first SQL data collection survey based on existing Loughborough University SQL exam scripts. The first survey attempted to explore the difficulties and challenges when students manually formulate SQL statements. In addition, the survey attempted to identify all types of common mistakes and the different ways of answering the query. Furthermore, it also aimed to find a proper solution of learning and marking SQL statements. As such, in this case, the survey study design provided a qualitative and a numerical description by studying students' SQL answers.

Data was then collected from the second set of SQL statements after testing the newly implemented SQL *Formulation Editor* (SQL-FE), where the research used both qualitative and quantitative approaches (mixed methods research). The qualitative research was used to quantify the problem by generating numerical data that can be transformed into usable statistics.

This was used in the pilot study, which utilised an online feedback survey to obtain students' feedback about the new *SQL-FE* tool and compared its features to paper-based methods. In the experimental study of the *SQL-FE* and SSMS tools, the qualitative research method was used to obtain the opinions of the participants by using a questionnaire. In this case, the experimental research design evaluated the impact of the *SQL-FE* tool on students' SQL solution outcomes and how it improved their formulation performance by using the point-and-click method. The third set of SQL data collected from new *SQL-FE* tool was normalised grouped and marked using the Case-based Reasoning System (CBR) and the Rule-Based Reasoning System (RBR). The normalisation process employed a survey research using the quantitative approach as a first step to determine the impact of the data normalisation and the level of similarity between SQL statement parts after applying the normalisation process. The experiment on the newly implemented *SQL Marking Editor (SQL-ME)* employed a qualitative approach as a first step to measure the lecturers' individual satisfaction of the *SQL-ME* prototype interface. The third step involved a study that followed a qualitative approach to collect observations with regards to testing the *SQL-ME* by several participants. In this case, the survey and experimental research design both scored the outcome of using the new semi-automatic assessment approach by students and lecturers positively.

4.4. Data Collection

Data collection is a procedure of gathering data from all related sources to find answers to the research problem. Furthermore, data collection can help to check the validity of the research method by encouraging various participants to get involved in surveys, questionnaires or interviews (Greener, 2008). In this research, data were gathered by utilising both surveys and questionnaire studies. These studies have carried out several advantages, which can be listed as follows.

1. Questionnaires and surveys are the most affordable ways to gather quantitative data. In addition, quick and easy to collect results with online and portable devices. This is because the research has provided an online data collection through a hyperlink, which linked to an online page (<https://www.surveymonkey.co.uk/>). It is a built-in online tool which creates and posts questions to different participants. This has reduced time and expenses of the research.

2. Questionnaires and surveys have allowed the research to numerous data from a large number of participants. Those participants are either educators or students which the research is interested to collect their feedback and opinions about new implemented formulation and making editor.
3. Since most survey and questionnaire providers are quantitative in nature and allow easy analysis of results, the research has used survey monkey tool which has provided an easy to analysis of all data collected of the research experiments

In contrast, the research has faced two main disadvantages by using the survey and questionnaire such as;

1. Since the research has used an online tool to gather information, the participants (mostly students) have no time limits which they took their time to complete the questionnaire at their own leisure. This has limited the time to retrieve and analyse the data and submit them on time.
2. When the research has used an open-ended questions in the questionnaires, students has left them unanswered. This has effected the results in case of enhancing the new implemented editor.

➤ **This thesis Data Collection:**

To follow, each of these data collection methods are briefly described, along with an explanation of how they were applied in this research.

4.4.1. Surveys

Survey research is one of the most important data collection methods, and is produce a variety of quantitative data. In this research, two main surveys took place as described below.

- a) The first SQL survey from an existing SQL exam script attempted to explore the difficulties and challenges encountered by students when they manually formulate SQL statements, as well as identify all types of common mistakes and the different ways of answering the query. The survey study design provided a qualitative and a numerical description by studying the students' SQL answers.

- b) The normalisation process employed a survey research using the quantitative approach as a first step to determine the impact of the data normalisation and the level of similarity between SQL statement parts after applying the normalisation process.

4.4.2. Questionnaires

Questionnaires are typically used for collecting data related to the research in survey-type situations. The main purpose of using a questionnaire in a research is to allow a relatively large number of people to participate in a quantitative research study (Rowley, 2014). This can support the researcher to obtain responses from a large number of participants, where the data collected can generate advanced and accurate research findings. However, on the other hand, a large number of participants would result in a huge amount of data that needs to be collected and analysed (Adams and Cox, 2008). Questionnaires might be created as printed copies of paper-based questions or distributed online where they can be sent through web media (e.g. email or via any website or professional networks for researchers). In both media, the participants are asked to answer the questions, and after completing them, send them back to the researcher (Rowley, 2014). In this research, a questionnaire was designed and employed for each study. This is because the questionnaire was deemed to be an appropriate technique to collect data related to SQL learning and assessment from both student and lecturer participants. The questionnaires used are described below.

- a) **Pilot study questionnaire:** A pilot study was conducted to measure the performance of students and the time they took while formulating SQL statements by using the new SQL-*FE* tool and the paper-based methods. The questionnaire was provided to the participants through a hyperlink, which linked to an online page <https://www.surveymonkey.co.uk/> on the same website of the SQL-*FE* interface. There were ten different questions, including multiple-choice and fill blanks questions, to evaluate the overall participants' satisfaction by the new SQL-*FE* editor.
- b) **Experiment questionnaires:** a questionnaire-based experiment was conducted on the newly implemented SQL Marking Editor (SQL-*ME*). The questionnaire was designed as a paper-based tool that contained three measurement categories; SQL-*ME* user interface and time spent on marking statements, SQL-*ME* feedback quality, and usefulness of the SQL-*ME* tool.

A number of different questions were asked, including multiple-choice and fill blanks questions, to evaluate the overall satisfaction associated with the new SQL-*ME* editor with different marking processes (fully or partially marking processes).

4.5. Data Analysis

In this research two different data analysis has been applied which explained in details in multiple chapters such as; chapter 5, chapter 6 and chapter 8. The following are summary of different analysis methods which have been used and how they have been evaluated. The first analysis is using manual analysis, which collected existing exam scripts and compares the errors and different ways of solving same SQL statements by using the spreadsheet. The second approach is using the t-test paired analysis method which has evaluated the full implemented SQL formulation and marking environment.

4.5.1. Existing Exam Scripts Analysis

Chapter 5 discuss the analysis of existing SQL examination scripts which has been collected using a spreadsheet. It contains the SQL questions, model answers and different students' answers with their grades. The sheet was used to filter the students' answers and find the common errors made by students. The study focuses on SQL questions; all the SQL answers have been listed in a spreadsheet along with the grades which the students got as illustrated in Figure 4-1.

QNo	Question A	Question B	Question C	Question D	Total Grade
Question	List the names and hire dates of all employees in the order they were hired.	List the department number and total salary of employees in each department that employs four or more people.	Give a list of ALL department names with the employees in each department.	Produce a list the names of salesmen together with their department names. List only those salesmen that work in an existing department.	
Model Answer	SELECT ename, hiredate FROM emp ORDER BY hiredate;	SELECT deptno, SUM(sal) FROM emp GROUP BY deptno HAVING COUNT (empno) >= 4;	SELECT dname, ename FROM dept LEFT OUTER JOIN emp ON dept.dept = emp.deptno;	select dname, ename from emp inner join dept on emp.empno=dept.deptno where job="salesman" and deptno is not null;	
Marks	2 Marks	4 Marks	2 Marks	2 Marks	20
926_1	select ename, hiredate from emp list by hiredate asc;	select deptno, sum(sal) from emp where count(deptno)>=4 group by deptno;	select dname, ename from emp, dept where emp.deptno=dept.deptno;	select ename, dname from emp, dept where emp.deptno=dept.deptno;	
Marks	1.5	3	1.5	1	13
872_2	select ename, hiredate from emp order by asc (hiredate);	select deptno, sum(sal)from emp group by deptno where count(empno)>=4;	select dname, ename from emp join dept group by deptno;	select ename, dname from emp join dept where job="salesman" and emp.deptno is not null;	

Figure 4-1: Sample of SQL Exam Scripts using spreadsheet

4.5.2. Data Analysis using t-test

The data analysis using t-tests to check the viability of the new formulation and marking SQL statements editors for students and examiners. In other words, they assessed if students could formulate basic SQL using SQL-*FE* and if the examiners struggled to mark the SQL statements; and how their feedback and marking experience can be improved.

“The t-test is a statistical test for the mean of a population and is used when the population is normally or approximately normally distributed and σ is unknown” (Bluman, 2012, p.427).

Grange (2011) explained in his tutorial three different types of the *t*-test technique namely;

- a) **Paired t-test type:** is used when data comes from the same participant, which means that each participant took both conditions of the test. It is used to compare two population means where participants have two samples in which observations in one sample can be paired with observations in the other sample.
- b) **Two-sample equal variance:** is used when the mean comes from different groups and the variances associated with each group are the same. This means that the variance of two groups is equal variance.
- c) **Two-sample unequal variance:** is used when the mean comes from different groups. In other words, if the variances of the two groups are not equal, the test will use the third type.

➤ This thesis Data Analysis Type:

This research selected the paired *t*-test for all experiments data analysis either for student to formulate the SQL queries or for the examiners to mark them.

- A. Each participant (student) had to do the quiz using the paper-and-pencil and SQL-*FE* editor modes. Therefore, a paired t-test for two related samples was used to test the significance of the difference in the meantime taken to complete the experiment between SQL-*FE* and the SSMS tool as explained in details in Section 6.4.
- B. Each session involved one participant (examiner), who performed two tasks during a one hour session. This experiment measured the feasibility of the semi-automatic approach, focusing on the assessment aspects by using both marking system pages of the SQL-*ME*.

The objective was to gain insight into the quality of the two different environments by measuring the number of students appearing on the list and the groups available as explained in details in Section 8.4.

SPSS is an IBM open source software which offers advanced statistical analysis, text analysis and open-source extensibility. Its ease of use, flexibility and scalability make SPSS accessible to users with all skill levels and outfits projects of all sizes and complexity to help users find new opportunities and improve efficiency. To achieve this, a one-sample t-test using SPSS statistics was used to measure the variance of the statistical analysis procedures of three parts using the following formula.

$$t = \frac{\sum d}{\sqrt{\frac{n(\sum d^2) - (\sum d)^2}{n-1}}}$$

d = difference per paired value
 n : number of samples

Each part is associated with one or two explored measurements; satisfactory, qualitative and quantitative. The first part reported the analysis of the examiners' attitudes towards the use of the SQL-*ME* tool, while the second part reported the relationships between the examiners' marking and their qualitative feedback provided using the SQL-*ME* tool. The third part analysed the quantitative feedback using the SQL-*ME* tool.

4.6. Ethical Requirements

The following steps were taken to ensure that the study complied with the high ethical standards required of such research study:

- a) An approval was obtained from the Research Ethics Committee of Loughborough University. In this research, all experiments have involved human participants (students and examiners) to solve the SQL questions and mark the SQL statements in several education institutes. For this reason, Loughborough University has maintained some requirements to be used in case human participants are involved to ensure that the researcher is meeting the required ethical standards.

For both experiments, the researcher has fill-up a form named as “**Ethical Clearance Checklist (for student involving Human Participants)**” as illustrated in *Appendix 2*. Then submit it to the ethics approvals sub-committee to be approved and start the experiment. Once the approval has been received, then the human participants have started the experiment.

- b) The informed permission of participants (examiners) was obtained before involving them in the study as illustrated in *Appendix 3*.
- c) Details of the instructions of the study were clearly explained to all participants (Examiners) as illustrated in *Appendix 4*.
- d) All participants were informed of their freedom to choose whether to participate in the study without any consequence.
- e) The privacy of the research participants was ensured so that no personal data was collected from respondents. In this case, the research has not asked to use any of personal details of the students or examiners. However, in *SQL-FE* experiment, student where asked to register through the SQL formulation editor and write their preferred email address without mentioning their name or any other details to ensure privacy of the participants.
- f) The participants were briefed about the aims and objectives of the study before the primary data collection process. This has encouraged the research to rich higher number of participants and motivates to get accurate solutions from them.

4.7. Summary

This chapter discussed the general research methodologies, designs and approaches used for the work conducted by this research. First, it introduced the research approaches along with the justification of choosing the selected approach. Subsequently, a comprehensive explanation of the various research designs was provided. A discussion of the data collection and data analysis processes was then presented. Finally, the chapter highlighted the main ethical requirements and the rules that should be followed before and after conducting any research methodology process.

Chapter 5.

Analysis of the Existing SQL Examination Scripts

5.1. Introduction

Learning the Structured Query Language (SQL) is an important step in developing database skills (Patel, 2012; Litoriya and Ranjan, 2010; Lans, 2007). This is verified by the fact that the numbers of higher education students learning SQL are constantly increasing. Early tools were only designed for teaching and offered increased feedback and personalised learning, but not summative assessment (Kleerekoper and Schofield, 2018). In addition, most research studies focus on marking and providing feedback on the final query output rather than the formulation of the SQL statement clauses as discussed in details in both Sections 3.3 and 3.4. Focusing on statement formulation can assist the examiners to diagnose the strengths and weaknesses and provide detailed feedback on SQL statements after they have been submitted for marking.

This chapter aims to achieve one of the main objectives of this research which listed in objectives list on section 1.2. It is to analyse different common errors made by students. This involves identifying the common mistakes in students' answers and analysing them to implement an accurate SQL formulation and marking environment that can help identify the similarities between SQL statements and mark them automatically. In addition, explore the difficulties and challenges that examiners face in manual assessment of SQL, and how such challenges can be addressed and solved.

The rest of this chapter is organised as follows. Section 5.2 explains the data collection methodology, while Section 5.2.2 highlights the common mistakes in SQL scripts. The various model answers for each query are discussed in Section 5.2.3. In Sections 5.2.4, error categories are introduced, and each student's error(s) is grouped under the appropriate error category. Section 5.2.4.1 and 5.2.4.2 discuss the error categories and their analysis, while Section 5.3 discusses the ideal SQL learning and marking process. Finally, Section 5.4 provides a summary of the chapter.

5.2. Data Collection

As discussed in Section 4.4, data collection has supported this research to check the validity of the research method by encouraging various participants (students and educators) to get involved in surveys and questionnaires or even by collecting previous year's exam scripts. For that, this chapter aims to provide a broad investigation and discussion of the research results. It discusses the data collection method, which was used to collect data from the Database module's exam scripts, and highlights different aspects of common mistakes.

5.2.1. Existing SQL Examination Scripts Data Collection

The conducted study consists of exam scripts for semester two (June 2013 and June 2014) of the Database module. The study identifies the common errors in SQL statement questions attempted by the undergraduate students of Loughborough University. The Database module exam scripts had four different question types, and the students had the flexibility to choose three questions. After filtering the exam scripts, 78 students attempted the SQL questions in 2013 and 72 students in 2014. These numbers correspond to 71% and 60% of the students attempting to solve the SQL questions in 2013 and 2014, respectively. The data collected contained the SQL questions, model answers and the students' answers along with their grades. The study focused on SQL questions only, and as such, all SQL answers were listed in a spreadsheet along with the grades that the students obtained.

In this analysis, there were seven questions from year 2013 (see Appendix 5) and seven questions from year 2014 (see Appendix 6). This means that 14 different questions were retrieved from the existing exam scripts with their answers. Those questions were a combination of DML (Data Manipulation Language) and DDL (Data Definition Language) statements. This is because the research had an interest in knowing all types of common mistakes and the different ways of answering a query. Furthermore, to find a proper solution of learning and marking SQL statements, one must start by studying and analysing different students' SQL answers. Therefore, this chapter analyses different SQL statements related to only solving the problems of the basic SELECT clauses, which cover SELECT, FROM, WHERE, JOIN, GROUP BY, HAVING and ORDER BY.

5.2.2. Common Mistakes in SQL Exam Scripts

As mentioned above, 71% of the students tried to solve the SQL questions in 2013, whereas only 60% tried to solve them in 2014. These figures show that not all students have the confidence to solve SQL queries. This might be because of the difficulties students face while solving SQL questions. Research by Renaud and van Biljon (2004) stated that the difficulties of solving SQL questions are “...due to the nature of SQL, and the fact that it is fundamentally different from the other skills students master during their course of study”.

Table 5-1 uses an example from Appendix 5 to provide a further explanation about the analysis of the results of students’ exam scripts. It provides a description of the questions, model answers and the common errors students made while attempting to solve the SQL questions. In addition, it highlights different examples of students’ errors and shows how many students made the same error. The tables in Appendix 5 and Appendix 6 show lists of exam scripts for semester 2 of the Database module (June 2013 and June 2014). They indicate the common errors in SQL statement questions attempted by undergraduate students of Loughborough University. Each question on the exam script was analysed individually to find the common errors and the number of students who made the same error.

Table 5-1: Example of common SQL mistakes that student made in the Database exam (June 2013)

QID	Question Description	Model Answer	Common Mistake	Examples of Students’ Mistake	Common Mistakes Made / 78
1.	Display the department number and total salary of employees in each department that employs five or more people.	SELECT DEPTNO, SUM(SAL) FROM EMPLOYEE GROUP BY DEPTNO HAVING COUNT(EMPNO) >=5;	a) Use of WHERE instead of HAVING clause.	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO WHERE COUNT(DEPTNO) >=5;	29
			b) Missing SQL function SUM()	SELECT DEPTNO, SAL FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO) >5;	10
			c) Use of TOTAL instead of SUM	SELECT DEPTNO, TOTAL(SAL) FROM EMP GROUP BY DEPTNO HAVING (COUNT(DEPTNO) >=5);	4
			d) Use of COUNT instead of SUM	SELECT DEPNO, COUNT(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(SAL) >=5;	3

5.2.2.1. Discussion of Common Mistakes

The number of common mistakes made by the students in both years suggests that students may have found understanding the queries a challenge, because most of them made similar mistakes. In common mistake “a” of Table 5-1, many students tried to solve the first question using the WHERE clause instead of the HAVING clause, when there cannot be an aggregate function in a WHERE clause.

In common mistake “b”, students attempted the query, however, they failed to add an important component of an SQL query into their solution – the “SUM ()” function. On the other hand, common error “c” shows that some students could understand the requirement of the query, that is, that they needed to use a function. However, they used “TOTAL” instead of “SUM ()”, which causes errors in the query. The last common mistake “d”, shows another example of changing the keyword, whereby students attempted the query using “COUNT” instead of the “SUM ()” function. As is clear from Table 5-1, the last three common mistakes are based on functions, which indicate that students might have had some confusion or lack of awareness of functions and their use. In addition, these mistakes were repeated in several questions in different years, as demonstrated by the answers to Questions 5 and 7 in Appendix 5 and Appendix 6.

The common mistake tables in Appendices 2 and 3 highlight further common mistakes and contain a much larger sample of student attempts. In Appendix 5, 30 students attempted Question 4 without adding “Data Type” or “Values” to their answers. For example:

```
CREATE TABLE EMP1  
(EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE);
```

The above statement indicates that students were able to create tables, since they provided most of the requirements to do so, but missed significant sections (i.e. the data type of each field name and values). Furthermore, Question 3 in Appendix 5 and Questions 10 and 14 in Appendix 6 specify another common mistake made by many students. The mistake shows that many students may have had difficulties when it came to using the GROUP BY and HAVING clauses, as they either forgot to add them or added them incorrectly. An example of this is:

```

SELECT EMPNAME
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
                WHERE JOB = 'ANALYST'
                GROUP BY DEPTNO);

```

As the example shows, the GROUP BY clause was added inside the sub-query, which is a misplacement that affected the students' results.

5.2.3. Model Answers to Each Query

The tables in Appendix 7 and Appendix 8 show the model answers for each SQL question of the 2013 and 2014 exam scripts. The tables also present the number of students who were able to correctly solve the query with a solution that was not one of the lecturers' answers. The models answers are split into different groups, where (i) and (ii) present the model answers by lecturers and (iii) presents the answers by students. It is clear from the percentages that the number of students who answered the SQL question varies from one question to another and depends on the requirements of each SQL question. The students had the option solve the query using either the lecturers' solution or their own different but correct answer. For example, in **Question 3**: "Display the names of all employees who work in a department that employs an analyst". Two different solutions result in the same correct answer for example:

(i) SUB-QUERY

(ii) JOIN

```

i) SUB-QUERY
SELECT EMPNAME
FROM EMP
WHERE DEPTNO IN (SELECT DISTINCT DEPTNO
                FROM EMP
                WHERE JOB = 'ANALYST');

ii) JOIN
SELECT DISTINCT E1.EMPNAME
FROM EMP1 , EMP2
WHERE E1.DEPTNO = E2.DEPTNO
AND E2.JOB = 'ANALYST';

```


On the other hand, there were number of students who failed to answer the query correctly and caused different types of errors, which are explained in detail in Appendix 5 and Appendix 6.

5.2.3.1. Discussion of the Different Model Answers

Questions 1, 2 and 4 in Appendix 7 show that students understood the requirement of the queries and tried to solve them with different kinds of solutions, and a large number of students answered them correctly.

For example, in **Question 2**: “Display the name of each employee with his/her department name”. There are two different correct solutions:

- The first correct answer (i) was given by 15 students, who used a JOIN statement:

```
SELECT DEPTNAME, EMPNAME
FROM DEPT INNER JOIN EMP
ON DEPT.DEPTNO = EMP.DEPTNO;
```

- The second correct answer (ii) was given by 38 students, who used a WHERE clause:

```
SELECT EMPNAME, DEPTNAME
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO;
```

However, Questions 3, 5, 6 and 7 were answered incorrectly by a higher number of students; as although some of the students did manage to find a different way of answering the query than that provided by the lecturer, most of them failed. Additionally, the number of students attempting the SQL question decreased dramatically from Question 5 to Question 7. The reason for this might be due to the fact that constraints and DML commands are more difficult for students to master. For example, in Question 7: “Configure the EMP1 table such that if a department is deleted from the DEPT table any associated employees are automatically deleted from the EMP1 table”, even though there were 33 students who attempted this question, only one provided a correct answer. The correct answer is:

```
ALTER TABLE EMP1 ADD CONSTRAINT FKEY FOREIGN KEY (DEPTNO)
REFERENCES DEPT (DEPTNO) ON DELETE CASCADE;
```

This answer shows that only one student understood the FOREIGN KEY concept and how to add it to SQL statements correctly.

On the other hand, the 2014 exam scripts show a significant increase in the number of correct answers compared to those from 2013, except for two questions (3 and 7), which had a very low number of attempts. Those two questions raised the percentage of incorrect answers to 85%. For example, in **Question 7** (Appendix 8): “Create a view called BOSS which has the name and number of each employee with the name and number of his or her manager (with blanks alongside any employee that has no manager”.

While there were eight students who answered this correctly in two different ways, many students failed to attempt it and there were many mistakes. The two correct methods of answering the question were identified by the lecturer as:

- (i) using a JOIN statement
- (ii) using a WHERE clause:

```
i) JOIN Statement
CREATE VIEW BOSS AS SELECT A.EMPNAME AS EMPNAME, A.EMPNO AS
EMPNO, B.EMPNAME AS BNAME, B.EMPNO AS BOSSNO
FROM EMPA LEFT OUTER JOIN EMPB
ON A.MGR = B.EMP;

ii) WHERE Clause
CREATE VIEW boss AS SELECT EMPNO, EMPNAME, JOB, MGR, HIREDATE,
DEPNAME
FROM EMP , DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

The rest of the questions show that the students managed to solve the query with different solutions, and that the percentage of students who answered correctly was high.

5.2.4. Errors Categories

After analysing all SQL script questions and their answers, this research initially categorised the students’ common errors as synonyms, syntax errors, incorrect keywords/functions and incomplete SQL statements. More details about these categories are presented below.

1. Synonyms Errors

SQL is an English-based programming language, a fact that causes some students to use words or phrases that mean exactly or nearly the same as other words or phrases used in SQL, thinking that they might provide the same results.

Students sometime also forget the name of a clause or think that they could obtain the output by using an incorrect but similar keyword/function of the clause. The example below shows a student using “SORT BY”, which is a synonym of “ORDER BY”, but cannot be accepted in SQL syntax.

```
SELECT EMPNAME, HIREDATE
FROM EMP
SORT BY HIREDATE;
```

2. Incorrect Keywords/Functions

Students might think that by using more complex commands or clauses in their answers, they will come to a more accurate solution and gain more marks. This example shows that a student used a “GROUP BY” clause, which is not required in the solution and results in an incorrect answer.

```
SELECT EMPNAME
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
                WHERE JOB='ANALYST'
                GROUP BY DEPTNO);
```

3. Syntax Errors

SQL is a structured language, with rules and regulations that must be followed. Changing the names of clauses or exchanging them with other functions’ names does not result in correct answers. The following example shows a student who used a “WHERE” clause instead of a “HAVING” clause. In such a case, the system will fail to run the query and an error will be generated: “Can’t have aggregate function in ‘WHERE’ clause”.

```
SELECT DEPTNO, SUM (SAL)
FROM EMP
GROUP BY DEPTNO
WHERE COUNT (DEPTNO) >=5;
```

4. Incomplete SQL Syntax

Students sometimes think that if they write short answers without mentioning all the required SQL syntax, they will reach the right solution or approach the correct answer. The example below shows a case in which a student forgot to add “Date Type” and “Values” to their answer, which resulted in inaccurate table creation. This led to the loss of significant marks, since the SQL statement was not effectively solved or completed.

```
CREATE TABLE EMP1  
(EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE);
```

5.2.4.1. Discussion of Error Categories

The initial error categories were identified based on common student mistakes, where most errors can be classified under one of these categories. However, it should be noted that this research found a large number of empty answers – questions that students left without any solutions. In the 2013 exam scripts, there were 81 empty answers, compared to 30 empty answers in the 2014 scripts.

These cannot be classified under any of the error categories. However, they cannot be ignored either, since they constitute a very serious problem that research should investigate in detail in order to fully understand it and find ways by which it can be resolved. The tables in Appendices 6 and 7 show the classification of students’ errors under each error category. The tables also show how the different error categories were attached to each error made by a student. It is clear from these tables that different categories of error could be found in the answers of the same question. For example, in **Question 1**, four different error categories were found in the students’ answers. In addition, some of these answers included an error that can be considered belonging to two different categories. For instance, one answer to **Question 1** was:

```
SELECT DEPTNO, TOTAL(SAL)  
FROM EMP  
GROUP BY DEPTNO  
HAVING (COUNT(DEPTNO)>=5);
```

This answer could be categorised as a wrong use of clauses or functions, since instead of using “TOTAL ()”, it should use the “SUM ()” function. Additionally, the answer could be categorised under an incorrect use of keyword/function, since “TOTAL ()” and “SUM ()” have the same meaning. It can also be clearly seen from the tables that the number of committed errors by students is greater than the number of students answering incorrectly. The reason behind this is that one student can make many errors in the same question, and many students can make the same error in different questions, as illustrated by Figure 5-1 and Figure 5-2.

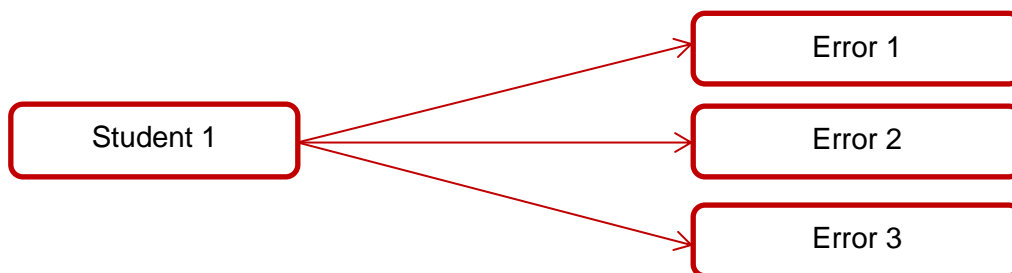


Figure 5-1: The relationship between a student and errors (One-to-Many)

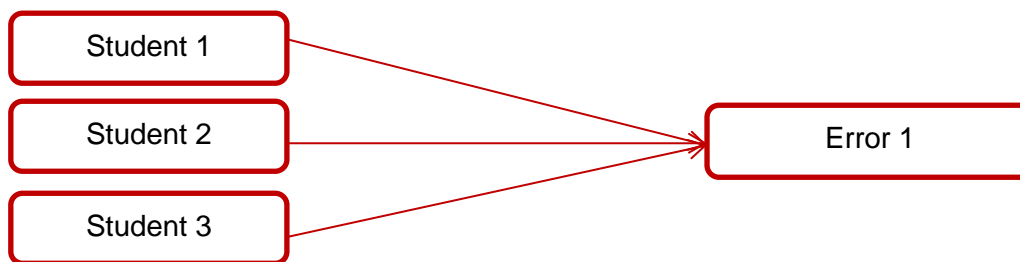


Figure 5-2: The relationship between students and errors (Many-to-One)

5.2.4.2. Analysis of Each Error Category

Figure 5-3 shows the breakdown of errors in terms of frequency from the 2013 and 2014 exam scripts. From the figure, it is clear that the incomplete SQL syntax errors represented the highest amount of errors in both years, since most of the students attempted the questions but failed to complete them. The error category with the second largest number of students committing it in 2013 is the incorrect keyword function, with 70 students. However, the number of students who made this type of error decreased in 2014 to 45 students.

On the other hand, the synonyms and syntax errors categories represented the lowest amount of errors committed by students. Generally, the statistics shows that the number of errors made in each category were similar in 2013 and 2014.

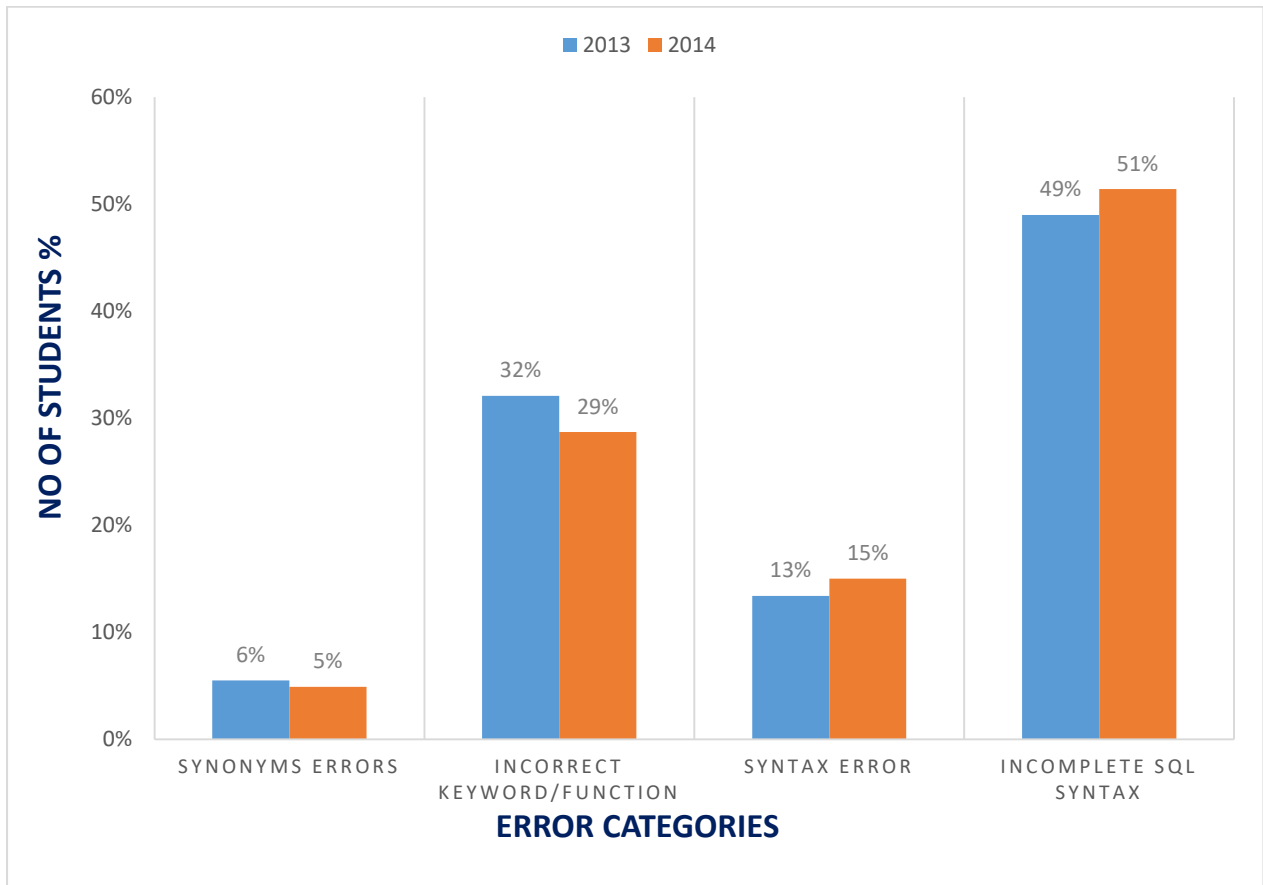


Figure 5-3: SQL Errors Categories breakdown for 2013 and 2014 students' exam scripts

Figure 5-3 also shows that the incomplete SQL syntax errors increased between 2013 and 2014, with a very high percentage of students committing such errors in both years: 49% in 2013 and 51% in 2014. In addition, a high percentage of students committed syntax errors or and incomplete SQL syntax. The reason behind this might be due to the students' weakness in solving SQL syntax functions. The percentage of students making incorrect keyword or function errors was 32% in 2013 and 29% in 2014. On the other hand, the percentage of students making synonyms errors was only 6% in 2013 and 5% in 2014. This indicates that only a minority of students struggled with synonym errors.

5.3. The Ideal SQL Marking Process

The proposed methodology that research might use is Case-Based Reasoning (CBR), which identifies how to solve research problems based on the solutions to similar previous problems (Watson and Marir, 1994), and Rule-Based Reasoning (RBR), which is a framework that copy the thinking of a human expert in solving a knowledge intensive problem (Grosan and Abraham, 2011). To ensure marking consistency and a reduced workload for lecturers, the proposed semi-automatic assessment system will incorporate the best features that have already been used by most existing systems, as well as include new features. The features of existing features of current systems include restriction of prohibited elements while formulating the SQL statements by the students and applying partial marking to provide immediate feedback. In addition, students will be allowed to solve the query in a different way to that of the lecturers as long as it provides the same output. This will be realised by applying fewer restrictions on students' SQL statements. At the same time, SQL syntax will be checked, which is the most commonly reported way to define tests, and also the most important part of the process (e.g. compiling the program, running the code and comparing the output with the expected output) (Tremblay and Labonté, 2003). Furthermore, feedback will be sent to provide individual students with information that focuses on their SQL learning performance (Walker, 2011).

There are numerous benefits of effective feedback, such as improving students' progress, boosting their achievements, enhancing their punctuality with which they hand in their work, and improving motivation and confidence. Similarly, marking and grading can indicate the level of performance that has been achieved by the student. The use of grades might affect students' learning, since they can provide a standardised measure of a student's performance, and certify that a course of study has been completed and particular standards of accomplishment have been achieved (Thompson and Ahn, 2012).

5.4. Summary

The main objective of this chapter was to identify the common mistakes in students' SQL and analyse them to implement an accurate marking environment that can help identify the similarities between SQL statements and mark them automatically.

This chapter summarised the different common SQL mistakes and the various model answers that can be provided to solve the same SQL query. It also categorised errors in four types such as synonyms errors, incorrect keywords/functions, syntax errors and incomplete SQL statements. It presented a study that grouped SQL errors in terms of the mistakes made in different SQL exam scripts. A new approach and framework was introduced to minimise or remove the dissimilarities between the SQL answers, while at the same time, enhancing the marking consistency and delivering context feedback. The new proposed semi-automated approach and framework are explained in detail in Chapters 7 and 8 with specific examples.

The next chapter describes the implementation of a new SQL Formulation Editor (*SQL-FE*), which is a specialised system that allows students to formulate SQL statements without any prohibited elements or errors.

Chapter 6. Design, Implementation and Evaluation

SQL Formulation Editor (SQL-*FE*)

6.1. Introduction

Chapter 5 identified the different SQL syntax errors after analysing a number of manual SQL scripts. Some syntax errors were classified as insignificant, while others were classified as significant. The insignificant SQL errors, such as spelling errors, can be excluded while making an SQL quiz. Spelling errors might occur as a result of wrong column names, wrong table names, or wrong syntax in one or more clauses of the SQL statement (Ahadi *et al.*, 2016). Although spelling mistakes could be categorised as insignificant error, this research considers it as one of the main issues to be addressed before implementing the new SQL-*FE* editor (Al-Salmi, 2018). The research also identified other syntax errors that can be categorised under significant errors that can affect the full SQL statement, such as reserved words errors (i.e. name, and, of) and the wrong use of aggregation functions (i.e. Average instead of AVG) (see Appendix 5 & 6). These errors might affect students' SQL answers and reduce their performance by wasting significant time, which leads to losing marks.

Learning the Structured Query Language (SQL) is an important step towards developing advanced database skills. As such, recently, the number of higher education students learning SQL has been constantly increasing. In this context, most researches focus on marking and providing feedback on the final query output rather than on the formulation of the SQL statement clauses as discussed in Chapter 3. Focusing on statement formulation can assist the examiners in diagnosing the strengths and weaknesses of students' answers and provide detailed feedback on SQL statements that have been submitted for marking. This chapter proposes a new semi-automatic assessment tool, called SQL-*FE*, for higher levels of education. The tool allows students to formulate SQL statements using the point-and-click interaction method.

The results have provided reasonable evidence that using *SQL-FE* can have a beneficial effect on formulating SQL query on time, and demonstrated a significant improvement in students' performance. The rest of this chapter is organised as follows. Section 6.2 describes the SQL formulation editor's requirements and components. It also provides a simple example to illustrate the process of formulating SQL statements using the editor. The pilot study is discussed in detail in Section 6.3. To ensure the effectiveness of the tool, the research conducted an experiment that compares *SQL-FE* with the SQL Server Management Studio (SSMS) tool, which is reported in Section 6.4. Finally, Section 6.5 concludes the chapter by presenting a summary of its findings.

6.2. The SQL Formulation Editor (*SQL-FE*)

SQL-FE was developed to enable students to formulate SQL statements, execute or run the queries and submit the SQL statements for marking. The tool was designed for the web to provide an effective avenue for testing students' SQL statements, as well as to provide quick feedback responses after marking students' SQL statements using the automated system. Figure 6-1 shows the use case diagram, which displays the core functionalities of the *SQL-FE* tool.

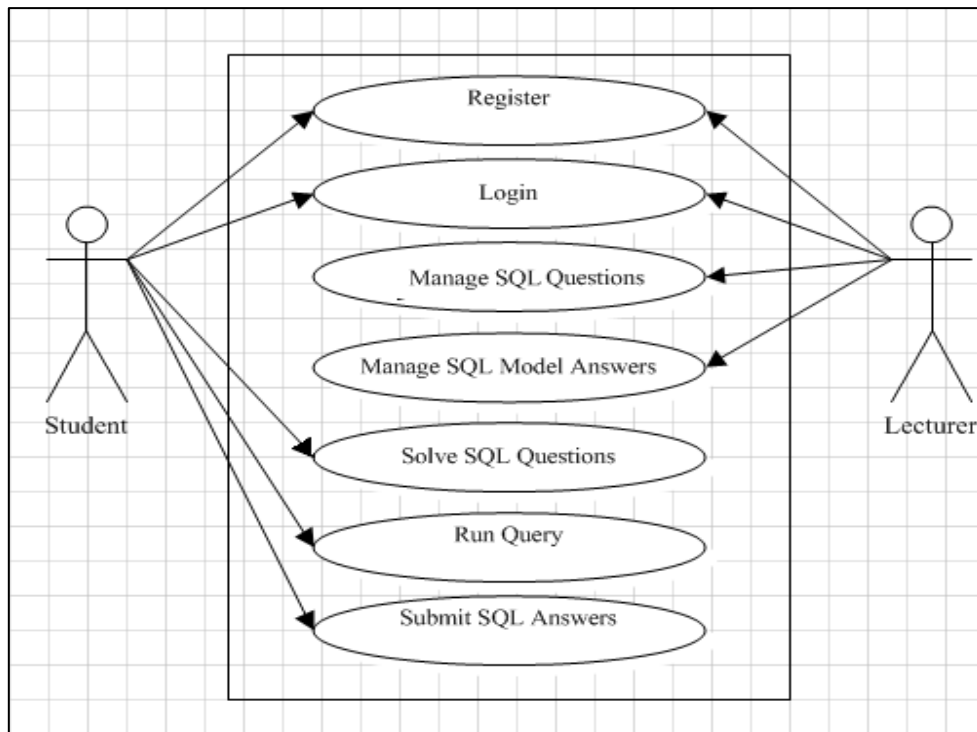


Figure 6-1: Use case diagram of the core functionalities of *SQL-FE*

The use case identifies the primary actors (users) of SQL-*FE*, along with the key use cases. Two types of actors use the tool: lecturers and students. In order to enforce proper security, each actor must first register into the editor before he/she can use any of the editor's functionalities. Registration ensures that a proper email address and password are created for each new user. The two actors—lecturer and students— will have access to different functionalities using the editor. The first step for the lecturer is to handle a given SQL assignment by creating and managing the SQL questions. Subsequently, SQL answers are assigned for each question, with multiple options (methods) of solving the same question. Once the student logs in to the editor, the time count will start automatically for each submitted SQL answer. The student will then solve the SQL questions and try to run them before submitting them for marking.

6.2.1. Requirements

SQL-*FE* needs to have a number of different functionalities, such as inserting, updating and deleting SQL components. These functionalities have been added as buttons in the SQL-*FE* tool, which allows students to modify SQL statements easily. SQL-*FE* uses a point-and-click user interface. The point-and-click approach can be used with different input devices, such as a computer mouse, touch pads and touch screens. However, there are two questions related to the selection of the point-and-click user interface style, which are:

- a) *Why has this tool chosen to use point-and-click interaction style rather than drag-and-drop interaction style or typing using the keyboard?*
- b) *Does using this style lead to an enhanced student SQL assessment performance?*

Several researchers have examined the differences in speed and accuracy between the two styles— point-and-click and drag-and-drop — on various tasks (Boritz *et al.* 1991; Gillan *et al.* 1990; MacKenzie, 1992). The MacKenzie (1992) study found that using the pointing method was faster than the dragging method, while the dragging style led to more errors than the pointing style. Another experiment for an educational game by Inkpen (2001) showed that using the drag-and-drop style creates more errors compared to point-and-click, and that the point-and-click was preferred by students. However, the decision to select either the drag-and-drop style or point-and-click style depends mostly on the task to be completed. For example, Adesina *et al.* (2013) used the multi-touch drag-and-drop style to solve basic arithmetic problems.

This allowed students to drag the numbers from the problem and drop them in the solution pad. Subsequently, via using multiple gestures, the mathematical operation can be computed using the arithmetic operators. The study of Adesina et al. showed improvements in the student mathematical performance of solving problems and provided more functionality to the process. In this research, the drag-and-drop style would not be useful in creating SQL statements since SQL needs to have structured syntax and changing the order might create errors. For this reason, the *SQL-FE* tool was designed to be compatible with the point-and-click interaction style. Moreover, the difference between *SQL-FE* and previous editors (Raadt et al., 2006; Sadiq et al., 2004; Abelló et al., 2008) is that *SQL-FE* will not allow keyboard typing. This means that the editor restricts students from writing the SQL statements using the keyboard, except for some special cases (as explained in the components subsection (6.2.2)). The reason for this is to avoid any trivial mistakes such as spelling mistakes, unnecessary words and synonyms, as described in detail in Chapter 5. Furthermore, the point-and-click interaction styles are compatible with different touchscreen technology devices such as tablets. These technologies have improved the effectiveness of student performance in different education aspects (Bonastre et al., 2006; Murray & Olcese, 2011; Moran et al., 2010; Adesina et al., 2015). This means the students might find it easier to utilise touchscreen interactions to complete the syntax using tablet devices. The user interface design requirements of the SQL Formulation Editor are listed in Table 6-1 and described in details in Section 6.2.2:

Table 6-1: *SQL-FE* user interface design requirements

RNo.	Design Requirements	Functionality	Reason
1.	Point-and-Click attraction method	This would allow student to click on the links provided rather than writing using keyboard.	<ul style="list-style-type: none"> • To avoid any spelling mistakes • To avoid adding unnecessarily elements to the SQL statements.
2.	Commands, Functions and Table Schema, Keywords and Operators	This list of main components to formulate the SQL statement.	<ul style="list-style-type: none"> • Table schema (to retrieve the table name and fieldnames easily) • To avoid using synonyms of functions. • Commands should be in a correct order to be executed and retrieve data.
3.	Text area	To allow students search for numeric or string data (text or date).	<ul style="list-style-type: none"> • If a query asks for numeric or text data to be searched for, the text area is allowing student to write using the keyboard and add it in the statement.
4.	Undo, Redo and Delete buttons	To manipulate the SQL statement	<ul style="list-style-type: none"> • Student can't use keyboard to go back and retrieve what they added for that certain buttons have been added to help them manipulate their statements.
5.	Run Query and Submit buttons	To executed the query and submit it to the examiner for marking	<ul style="list-style-type: none"> • After formulating the query, the run button allow student to retrieve the data needed and then submit it to the examiner to get mark and feedback.

WELCOME TEST [HELP - CONTACT US - FEEDBACK- LOGOUT]

b. Left Navigation bar

COMMAND
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
FUNCTIONS
SUM(
AVG(
MAX(
MIN(
COUNT(
TABLE SCHEMA
EMP
EMPNO int(11)
DEPTNO int(11)
FNAME varchar(255)
LNAME varchar(10)
GENDER text
JOB varchar(255)
SALARY decimal(8,2)
DEPT
DEPTNO int(11)
DEPTNAME varchar(255)
LOC varchar(255)

c. Table Schema

Question: Marks (2)

1. Find the first names of all employees who are work as clerks and earn a salary of more than 2500.

a. Question pane

SQL Statement

e. SQL Answer pane

f. Text area pane

g. Control buttons

d. Right Navigation bar

KEYWORDS				
ASC	DESC	ALL		
AND	ANY	IN		
LIKE	NOT	OR		
IS NULL	IS NOT NULL			
BETWEEN	AS	ON		
OPERATORS				
,	;	+	-	*
/	%	=	<>	
>	<	>=	<=	
#	'	()	

Figure 6-2: SQL Formulation Editor (SQL-FE)

(Source Code: studentexam.php): “SQL Formulation Editor User Interface”

```
<div class="col-md-8 form-group" style="padding:3px !important;">
  <div class="panel panel-default">
    <form name="frmX" method="post" id="frmX" autocomplete="off">
      <input type="hidden" id="qid" name="qid" value="<?PHP echo $results->qid; ?>">
      <input type="hidden" id="qaid" name="qaid" value="<?PHP echo $qaresult->qaid; ?>">
      <input type="hidden" id="submittime" name="submittime" />
      <div class="panel-body" style="padding:0px !important;">
        <div class="myQuestionBox"> <div class="col-md-12 myheading">Question:
          <span class="count">Marks (<?PHP echo $results->marks; ?>)</span></div>
          <div class="col-md-12 myquestion"><?PHP echo $results->questions; ?></div>
        <div class="col-md-12 myquestion" style="text-align:right;"><?PHP echo($links); ?></div>
      <br clear="all" /> </div>
      <div class="col-md-12" style="text-align:right;"></div>
      <div class="col-md-8"><label class="text-warning">SQL Statement</label>
      <textarea name="QueryPanel" style="resize:none; color: #FFF !important;
      height: 110px; font-size: 11px; letter-spacing: 1px;" rows="8"
      class="form-control" id="QueryPanel"><?PHP //echo $qaresult->ans; ?></textarea>
      <br clear="all" /><div class="col-md-12 myquestion" style="text-align:right; padding:0px">
        <button type="button" id="mytime" style="border: 0px; font-size: 14px;
        letter-spacing: 1px; display: none;" class="btn-outline btn-success"></button>
        <button type="button" id="undo" class="btn btn-success margin undo">Undo</button>
        <button type="button" id="redo" class="btn btn-primary margin redo">Redo</button>
        <button type="button" id="reset" class="btn btn-danger margin">Reset</button>
        <button type="button" id="runQuery" class="btn btn-warning margin">Execute
        Query</button>
        <button type="button" id="submitbutton" class="btn btn-primary margin">Submit</button>
        <button type="button" id="showans" style="display:none !important;"
        class="btn btn-warning margin">Suggestion</button>
        <button type="button" style="display:none !important;" id="getallanswers"
        class="btn btn-warning margin">Answer Log</button> </div>
      </div>
    <div class="col-md-4" style="padding-top: 25px;">
    <style>
    .mybtn { font-size: 12px !important;
      font-weight: normal !important;
      letter-spacing: 1px !important; }
      .inputmargin { margin-top:5px !important; color:#FFF !important;
      font size:12px !important; }
      #notification { font-size:10px !important;
      letter- spacing:1px !important;color:#FFF !important; }
    </style>
    <button class="btn btn-warning mybtn datatype" id="String" type="button">String</button>
```

```

<button class="btn btn-warning mybtn datatype" id="Numeric"
type="button">Numeric</button>
<input type="text" class="form-control inputmargin" id="myvalueforbox" />
<button class="btn btn-danger mybtn inputmargin" id="confirmvalue"
type="button">Confirm</button>
<div class="col-md-12" style="padding:0px; margin:8px 0px;" id="notification"></div>
</div><div class="col-md-12 form-control" id="suggestionans"
style="resize:none; color: #FFF !important;"><div class="col-md-11 setcommand"
id="<?PHP echo $results->ans; ?>"><?PHP echo $results->ans; ?></div></div>
<div class="col-md-12 myquestion" id="queryresult">
</div> <div class="col-md-12" id="myDiagram" style="background-color:#FFF !important;
overflow:scroll; width:100%; height:300px; display:none; text-align:center;"></div>
    <textarea id="mySavedModel" style="display:none;">
    </textarea> </div>
    </form>
<div class="col-md-2 form-group panel panel-default">
<?PHP
foreach($this->home->get_enum_values('hd_sqlcommands','commandstype') as $command)
{ if($command=='Operators' || $command=='Keywords')
{
$childList = $this->home->getcommandsList($command);
if($childList['count'] > 0)
{echo '<div class="col-md-12 schema allpadding" id="'. $command. ' ">'. $command. '</div>';
foreach($childList['data'] as $gcl)
{ echo '<li class="sfieldname setcommand" id="'. $gcl->commandtext. '
">'. $gcl->commandtext. '</li>';}}}}?>
    <!--/.main-->
<?php $this->load->view('common/footer');?>
<script> $(function(){ var fiveSeconds = new Date().getTime() + 0000;
$('#mytime').countdown(fiveSeconds, {elapse: true}) .on('update.countdown',
function(event) { var $this = $(this); $this.html(event.strftime('<span>%H:%M:%S</span>'));
$('#submittime').val(event.strftime('%H:%M:%S'));
});
});
</script>

```

6.2.2. Components

The *SQL-FE* tool is designed to achieve the requirements of SQL assessment using the semi-automated approach. The editor is based on automatic SQL formulation. This section explains the components of *SQL-FE*, as illustrated in Figure 6-2.

The editor contains seven main components which identifies the core functionalities of the tool. The main functionalities are the navigation bar, table schema, function buttons and SQL question & answer pane.

- a. **Question pane:** Figure 6-3 illustrates the question pane which serves to show the SQL question scenario and identify the query requirements needed to solve the SQL statements. Placing the SQL question in the same SQL-*FE* web page makes it more convenient for students to solve the SQL statements. In addition, it saves on the printed-paper otherwise needed for listing the SQL questions manually.

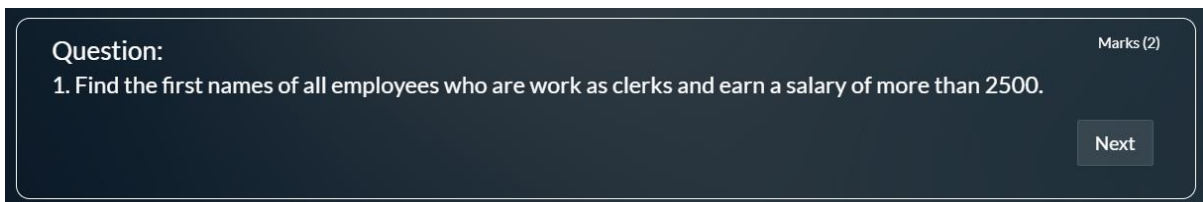


Figure 6-3: Question Pan

- b. **Left Navigation bar:** The left navigation bar consists of two main parts, commands and functions as illustrated in Figure 6-4. The commands list assists students while solving the SQL statements, whereas functions have been added to allow performing calculations on data. The commands and functions are placed on the left hand side, where students can easily access them to solve the queries. The editor lists the basic SELECT commands and functions; however, they can be modified and expanded depending on the question’s requirements.

COMMAND
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
FUNCTIONS
SUM(
AVG(
MAX(
MIN(
COUNT(

Figure 6-4: Left Navigation bar (SQL commands and Functions)

- c. **Table Schema:** The table schema displays the table name, field names and their data-type to be used while solving the SQL questions as shows in Figure 6-5. This means that there is no need for a printed-paper to display the table schema for the student as it is already viewable on the web page.

TABLE SCHEMA	
EMP	
EMPNO	int(11)
DEPTNO	int(11)
FNAME	varchar(255)
LNAME	varchar(10)
GENDER	text
JOB	varchar(255)
MGR	int(11)
SALARY	decimal(8,2)
COMM	int(11)
DEPT	
DEPTNO	int(11)
DEPTNAME	varchar(255)
LOC	varchar(255)

Figure 6-5: Left Navigation bar (Table schema)

- d. **Right Navigation bar:** The right navigation bar consists of reserved SQL keywords used for defining, manipulating and accessing the database as shows in Figure 6-6. In addition, it contains a set of operators used in the WHERE command to perform operations such as comparisons and arithmetic calculations. Separating the navigation bar to two separate left and right bars serves to provide more vertical space for the main content such as the SQL question and the SQL statement answer bars.

KEYWORDS					
ASC	DESC	ALL	AND		
ANY	IN	LIKE	NOT		
OR	IS NULL	IS NOT NULL			
BETWEEN	AS	ON			
OPERATORS					
,	;	+	-	*	/
%	=	<>	>	<	
>=	<=	#	'	(
)					

Figure 6-6: Right Navigation bar (SQL keywords and operators)

- e. **SQL Answer pane:** The SQL answer pane is used to enter the SQL answer using the left and right navigation bars. The point-and-click interaction style allows students to point on the navigation bar and click using the mouse pointer to complete the SQL answers without the need for using the keyboard, as illustrated in Figure 6-7.

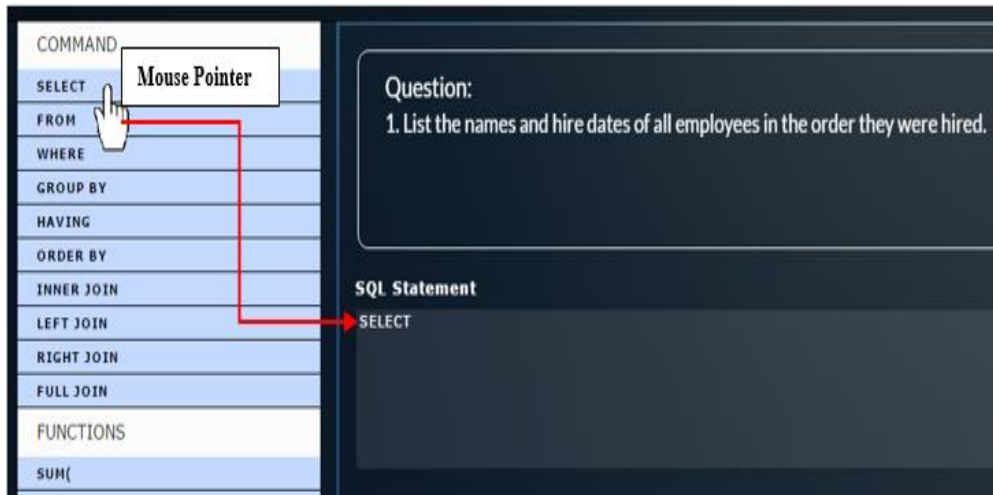


Figure 6-7: Entering the SQL statement using the mouse pointer in SQL-FE

- f. **Text area pane:** The text area pane helps students to add different numerical or string values to limit the data retrieved, which cannot be done by using the available navigation. The reasons for not using the keyboard were described in the requirement section (6.2.1). The text area provides an exception to keyboard use by allowing students to enter either string or numerical values depending on the question's requirements, as demonstrated in Figure 6-8. To insert any values, the student should choose either string or numerical values, where the tool will present a clear message for students about which data should be added. This message will appear under the confirm button. Subsequently, using the keyboard, they can enter the desired value and then hit the confirm button, where the value will be transferred to the SQL statement bar.

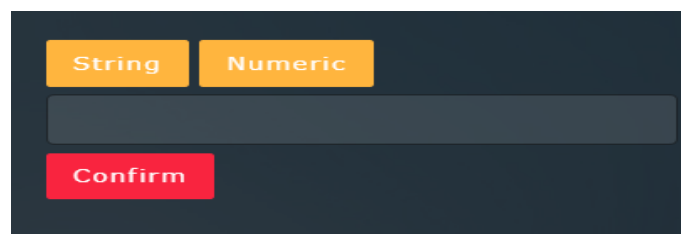


Figure 6-8: Text-area pane (used to enter string and numeric data)

As the editor tests the basic SQL statements, only string and numerical values are allowed to be entered as values to the SQL statements. Therefore, the date data-type values can be retrieved using the string values as an initial step, as illustrated in Figure 6-9.

The screenshot shows a SQL query execution interface. At the top, a question asks to display the first names and hire dates of all employees hired between 2010 and 2012. The SQL statement entered is: `SELECT EMP.FNAME , EMP.HIREDATE FROM EMP WHERE EMP.HIREDATE BETWEEN '2010-01-01' AND '2012-12-31'`. The interface includes buttons for 'String' and 'Numeric' data types, a date input field containing '2012-12-31', and a 'Confirm' button. Below the query bar are buttons for 'Undo', 'Redo', 'Reset', 'Run Query', and 'Submit'. A green success message states: 'Success#: Your query has been executed successfully and we found 3 records'. Below this is a table with the following data:

FNAME	HIREDATE
Smith	2012-07-17
Laura	2011-10-18
Clark	2010-10-28

Figure 6-9: Entering date values using the string data type

- g. Control buttons:** The control buttons are divided into two categories, as shown in Figure 6-10. The first category (1) is used to make any amendment in the SQL statements, such as to redo, undo and reset the SQL statements. Since students are prevented from using the keyboard, they are not able to use the backspace button to delete or navigate inside the SQL statement bar. To solve this issue, different buttons have been added to redo, undo and reset the SQL statements in order to help students to navigate using the mouse easily.

The screenshot shows a SQL query execution interface with the SQL statement: `SELECT EMP.EMPNO FROM EMP WHERE EMP.EMPNO = 5 ;`. Two groups of control buttons are highlighted with numbered boxes:

- Group 1:** A box containing a '1' icon and three buttons: 'Undo' (green), 'Redo' (blue), and 'Reset' (red).
- Group 2:** A box containing a '2' icon and two buttons: 'Run Query' (orange) and 'Submit' (blue).

Figure 6-10: The two types of control buttons

The second set of control buttons (2) deal with running the SQL query to show the SQL result output. In addition, a submit button is used to save students' SQL answers for marking. In *SQL-FE*, the answers are saved automatically in the created database after submitting (using the submit button) each SQL statement. After the exam, the students' answers are easily retrieved to be marked by the lecturer. In contrast, users of other existing SQL tools have to save the SQL statement answers manually in a folder or an external device to be later marked by the lecturers.

6.2.3. Technologies used in the development

To achieve the design goals appropriate technologies were employed to implement the new formulation tool, which are the software tools and software source code.

6.2.3.1. Software Tools

The dynamic Web page and how PHP interacts with the other applications involved in the process is illustrated in Figure 6-11. The figure shows the lifecycle of PHP request and the main parts, scripting tools with other tools which are commonly used with them. It displays the client (web browser) submits an HTTP request to the Apache web server to find the main page that contains HTML, PHP, JavaScripts and Database; then the server returns a response to the client. The HTTP works as a request-response protocol between a client and server. Each of those parts is described in details as follows.

- 1. Client Side:** It refers to everything in a web application that is displayed or takes place on the client (end user device). A web browser may be the client, and an application on a computer that hosts a web site may be the server. In this research, the *SQL-FE* is the client side which illustrated previously in Figure 6-2.
- 2. Network:** is a collection of computers, servers, mainframes, network devices, peripherals, or other devices connected to one another to allow the sharing of data.
- 3. Apache Web Server:** is the open source web server used to serve the pages from the Marking Assistant.
- 4. PHP:** is the server-side, scripting language used for the design with HTML. It provides greater flexibility in the design of websites by enabling the creation of dynamic pages. Page contents are changed based on interaction with the user or data stored in the database.

PHP offers many advantages because it is open source and can be used across different platforms.

- **Cascading Style Sheets (CSS):** used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML
- **JavaScript (JS):** is the scripting language that is used to add interactivity to the Web App; the codes are interpreted and run by the web server.

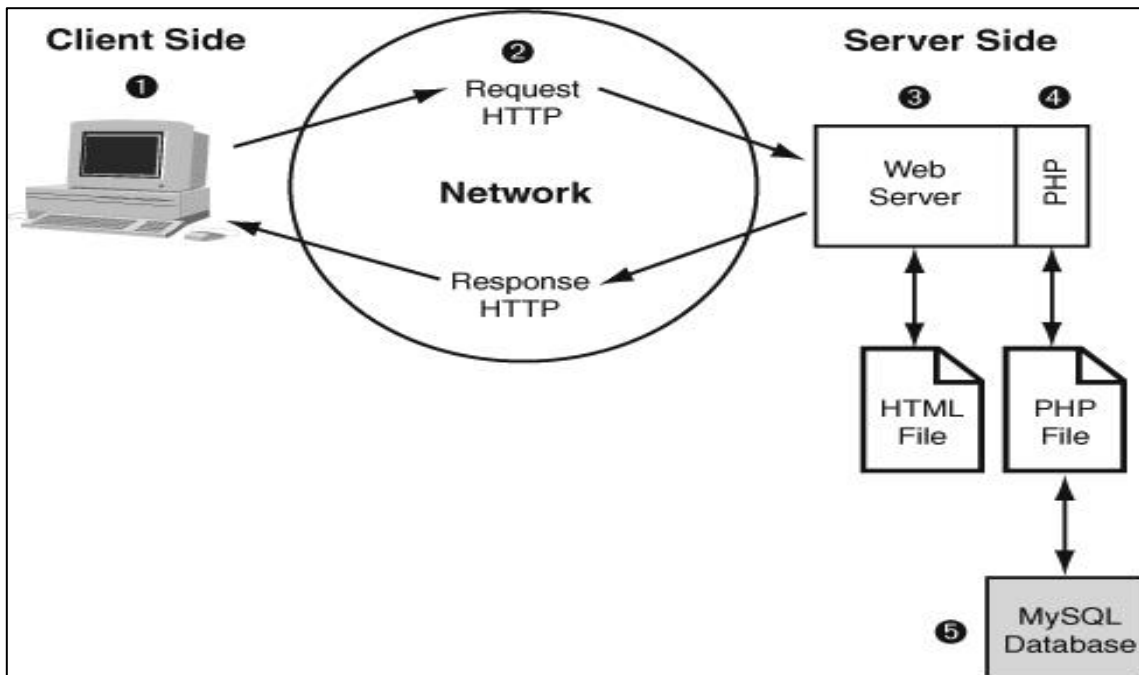


Figure 6-11: The lifecycle of PHP Request Processing Diagram

5. **MYSQL:** uses SQL (structured query language) to create, manage and retrieve information from the database. It is relational database management system in which data is stored in multiple tables by the sharing of keys. The database used to store all the SQL statements is called phpMyAdmin. It is an open source tool written in PHP which proposed to handle the administration of MySQL over the Web. The full SQL answers can be retrieved and exported using the same database as reference for the examiner to be reviewed. It also controls access to the stored data as illustrated in Figure 6-12. The figure illustrates some of the submitted SQL statements that have been divided into clauses (parts) as they are ready to be viewed by the examiner for marking. Each SQL clause is connected with participant ID and question number.

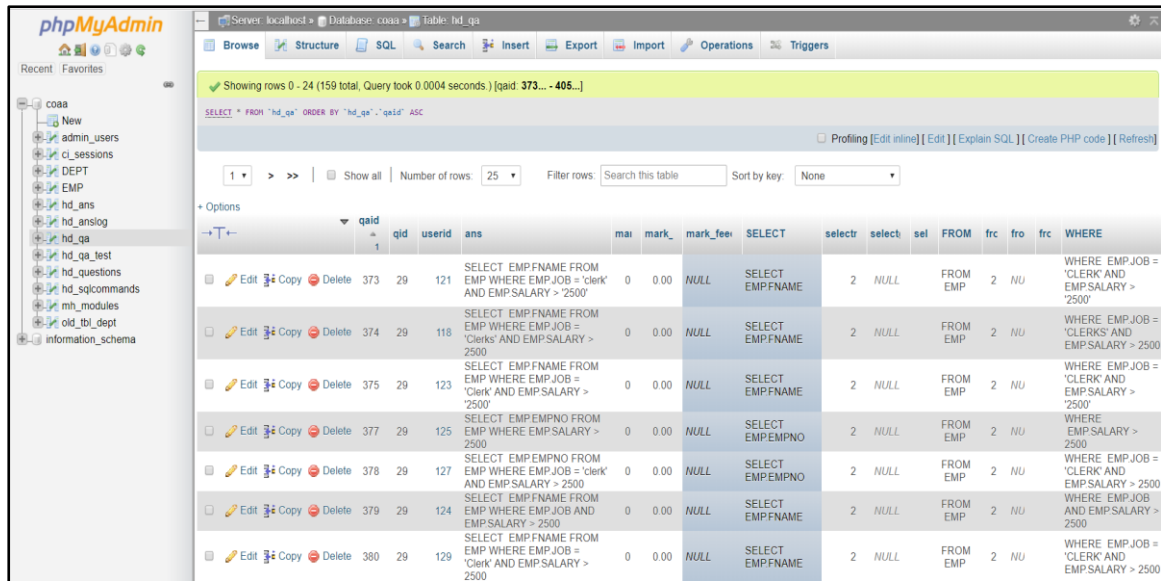


Figure 6-12: Print screen of phpMyAdmin Database

6.2.3.2. Software Source Code

This section describes how this is achieved in code when SQL-FE developed. The example illustrates the functionalities of the components explained above.

1. **Registration Form:** All students should register for first-time access of the SQL-FE tool using the registration form, as shown in Figure 6-13. The registration form allows the lecturers to retrieve students' answers using their email addresses and send them the grades and feedback of their SQL statements. Once a student logs in to the SQL-FE tool, he/she can start solving the queries as shows in Figure 6-14.

The registration form is titled 'REGISTRATION FORM' and contains the following fields:

- Full Name:** A text input field containing 'John Smith'.
- Email:** A text input field containing 'j.smith@lboro.ac.uk'.
- Password:** A password input field with four dots.
- Confirm Password:** A password input field with four dots.

 At the bottom of the form, there are two buttons: a blue 'Login' button and a green 'Register' button.

Figure 6-13: SQL-FE registration form

Source Code: registration-form.php

```
<form role="form" method="post" action="" autocomplete="off" id="formvalidateusers">
  <fieldset>
    <div class="form-group"> <label>Full Name:</label>
      <input class="form-control req" id="fullname" name="fullname"
        autocomplete="off" autofocus style="color:#FFF !important;"> </div>
    <div class="form-group"> <label>Email:</label>
      <input type="email" style="display:none;">
      <input type="text" class="form-control req" id="email" name="email"
        autocomplete="off" style="color:#FFF !important;"> </div>
    <div class="form-group col-md-6" style="padding:0px !important;">
      <label>Password:</label> <input type="text" style="display:none;">
      <input class="form-control req" type="password" id="password" name="password"
        autocomplete="off" style="color:#FFF !important;"> </div>
    <div class="form-group col-md-6" style="padding:0px !important;">
      <label>Confirm Password:</label> <input type="text" style="display:none;">
      <input class="form-control req" type="password" id="cpassword" name="cpassword"
        autocomplete="off" style="color:#FFF !important;"> </div>
    <input class="btn btn-primary" type="button" name="submit"
      onClick="gottopage(this);" data-url="<?PHP echo base_url(); ?>" value="Login" />
    <button type="submit" class="btn btn-success">Register </button>
    <?php $error = $this->session->flashdata('error');
    if(!empty($error)):?>
    <center style="color:#C00;">
    <strong><?php echo $error;?></strong>
    </center>
    <?php endif;?>
  </fieldset>
</form>
<script>
  $('#HIREDATE').datepicker({ dateFormat:'yy-mm-dd'
  });
  $('#checkteacherstudent').click(function(){
  $('#formvalidateusers .req').removeClass('myerror');
  $('#formvalidateusers .req').each(function(index, element) {
    if($(this).val()=="
      { $(this).addClass('myerror'); }
    });
    var errorlen = $('.myerror').length;
    if(errorlen<=0)
    $('#formvalidateusers').submit(); }
  });});
</script>
```

2. *SQL Formulation Editor Example:*

The question scenario mostly contains the field names and table that need to retrieve the data. Some questions contain other SQL commands depending on the question's requirements. Figure 6-14 shows a fully explained example of an SQL question and how it is solved using the SQL-*FE* tool. The SQL question asked to retrieve all female employees' last names with their department name. The left and right navigation bars allow students to enter the SQL statement using the point-and-click technique. The text area enables students to retrieve the employees' gender using the string data-type button, as shown in Figure 6-14. The resulting output of the SQL statement shows the correction of the answer and helps students to check their answers before submitting them to the lecturers for marking.

Figure 6-14 demonstrates the steps involved in solving SQL questions using the SQL-*FE* tool. The figure shows an SQL answer that was attempted using the point-and-click interaction technique. The first step is highlighted using the grey colour on both navigation bars. It shows the commands, tables name, keywords and operators that have been used to complete the SQL answer.

The second step is illustrated with blue colour and involves retrieving only the female employees by clicking on the string data-type, using the keyboard to write the 'Female' keyword and then clicking on the confirm button to insert the keyword into the SQL statement. The last step is to give the student the ability to check the correctness of their SQL statement syntax and query output by clicking on the run query button, where the results of execution are highlighted in the figure using the red rectangle.

COMMAND
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
FUNCTIONS
SUM(
AVG(
MAX(
MIN(
COUNT(
TABLE SCHEMA
EMP
EMPNO int(11)
DEPTNO int(11)
FNAME varchar(255)
LNAME varchar(10)
GENDER text
JOB varchar(255)
SALARY decimal(8,2)
DEPT
DEPTNO int(11)
DEPTNAME varchar(255)
LOC varchar(255)

Question: Marks (3)

2. Retrieve the last names and the department names of all female employees. Sort the result in ascending order of the location.

Previous Next

SQL Statement

```
SELECT EMP.LNAME , DEPT.DEPTNAME
FROM EMP INNER JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO
WHERE EMP.GENDER = 'Female';
```

Undo Redo Reset Run Query Submit

String Numeric

Female

Confirm

Please enter string only

Success#: Your query has been executed successfully and we found 2 records

LNAME	DEPTNAME
Paul	Operation
Louis	Management

KEYWORDS			
ASC	DESC		
ALL	AND		
ANY	IN		
LIKE	NOT		
OR	IS NULL		
IS NOT NULL			
BETWEEN	AS		
ON			
OPERATORS			
,	;	+	-
*	/	%	
=	<>	>	
<	>=	<=	
#	'	(
)			

Figure 6-14: An example of a SQL statement answer

➤ Time Efficiency Evaluation Measurement

To evaluate the new implemented tool, one of the main measurement should be included which is time efficiency. Time efficiency is a measure of amount of time for an algorithm to execute (Adesina, 2016). In this research, time efficiency has been measured in all studies that have been conducted. For example;

1. Pilot study using (Paper-based and SQL-FE) methods as discussed in section 6.3.

This is to measure if the students spend more time on doing the exam using the SQL-FE editor than using the paper-and-pencil mode or the reverse.

2. Experiment using (SQL-FE and SSMS) methods as discussed in section 6.4.

This is to check the time spent after using two different methods and which of these method has taken less time to finish SQL execution.

3. Experiment using (SQL-ME1 and SQL-ME2) editors as discussed in 8.4.

This is measure the time that participants needed to complete the marking and write their feedback on the answers. The objective was to compare the time needed to complete the marking across both editors, such as groups, marks and feedback.

6.3. Pilot Study

A pilot study was conducted to evaluate the time efficiency and usability of the new SQL-FE editor compared to paper-and-pencil formats (Chan & Schmitt, 1997; McDonald, 2002; Koenings *et al.*, 2015). The study observed undergraduate students using the SQL-FE tool and the paper-and-pencil method to formulate SQL statements. The purpose of this experiment was to compare the time efficiency of writing SQL statements using these two different methods.

6.3.1. Participants

The participants were second year undergraduate students aged between 19 and 21. The total numbers of participants were 40 students (23 females and 17 males). The participating students studied two courses, Information System (IS) and Information Technology (IT) at the Modern College of Business and Science, Oman. The study was carried out during the last week of July 2016. The students had some background on database use having studied the Database module in their first-year course. In the second-year, they studied SQL concepts and syntax as the main content of the Management Information System module.

The module included two days of lab practice lasting 100 minutes each, and three days of lectures for a total of 4 hours 30 minutes per week. The purpose of the lectures was to teach and explain the concepts of SQL to students, so they can then apply such knowledge during the lab sessions. For each lab practice, two sessions were run, with approximately 20 students attending each session.

6.3.2. Study Procedure

The students participated in solving an SQL quiz with five questions. A comparative crossover experimental design was implemented to run this experiment. Quinn & Keough (2002) defined the crossover as an experimental design that combines the attributes of Latin Squares and repeated measurement design. It is normally used in experiments that apply multiple tests to individual participants. In this study, it was used by randomly dividing students into two groups. The first group consisted of 20 students (11 females and 9 males) solving SQL questions first on paper-and-pencil and then on *SQL-FE*. The second group also consisted of 20 students (12 females and 8 males) solving similar questions using the *SQL-FE* editor first, before attempting the quiz on paper-and-pencil, as explained in Table 6-2.

Table 6-2: Participating students solving SQL Questions using both modes

Group	Total No. of Participants	Mode 1	Mode 2
Group A	20 Students (11 Female and 9 Male)	Paper-and-Pencil	<i>SQL-FE editor</i>
Group B	20 Students (12 Female and 8 Male)	<i>SQL-FE editor</i>	Paper-and-Pencil

The table explains the groups division used in the experiment. Participants within Group (A) (with 20 students) and Group (B) (with 20 students) were randomly selected to undertake the two different quiz modes.

6.3.3. SQL Questions

The experiment used five SQL questions that required the participating students to write basic SQL queries. The same questions were used in the paper-and-pencil and SQL-*FE* methods.

The research did not focus on the questions used in both modes since it aimed to test the basic SQL commands that student can easily solve directly. A list of the SQL questions used is shown in Table 6-3

Table 6-3: List of SQL quiz questions

Q. No.	Questions
1.	Display only the department name and location for each department
2.	Display the names and salaries of all employees with a salary greater than 2000.
3.	List the names and hire dates of all employees in the order they were hired.
4.	Display the names of all employees with an 'A' as first letter in their name.
5.	Display the hire date, name and job for all salesmen.

The requirements of the five questions covered the basic SQL commands (SELECT, FROM, WHERE and ORDER BY). The paper-and-pencil quiz was supported by a table schema to assist students while solving the SQL queries. The questions were randomly ordered in both SQL-*FE* editor and paper-and-pencil modes. The first and fifth questions required only (SELECT & FROM) commands, while the second and fourth questions required (SELECT, FROM & WHERE) commands. The third question asked students to sort the data using the (ORDER BY) command.

6.3.4. Study Analysis and Discussion

Furthermore, students were asked to anonymously complete an online feedback survey to gather information on their general opinions of the new SQL-*FE* tool. This section presents a detailed data analysis of the experiment using the t-test and the responses of the online student feedback after testing the new SQL-*FE* tool. The initial results show that the time spent on paper-and-pencil mode was more than that spent on the semi-automated editor.

Scenario Question:

Does the student spend more time on doing the experiment using the *SQL-FE* editor than using the paper-and-pencil mode or the reverse by using a one sample paired *t*-test.

The main objectives of the evaluation were to measure the time efficiency, usability and effectiveness of the *SQL-FE* Formulation Editor over the paper-and-pencil SQL assessments that can provide helpful environment for learning and teaching SQL statements. This study focuses on the *SQL-FE* method which uses an online environment to interact between students and lecturers to capture students SQL answers.

Total of 40 students (second year undergraduate) have participated in this experiment. The students were asked to participate in simple SQL quiz which involves five different SQL questions using both media, *SQL-FE* tool and paper-and-pencil assessment. Furthermore, students' have been asked to complete an online feedback survey anonymously to gather information about their acceptance and general opinion about the new SQL assessment tool. The initial results revealed that using *SQL-FE* tool leads to an average of 39% saving time compared to writing quiz using paper-and-pencil. This means that student has used to do lab practice with SQL statements as learn-by-doing approach which gives them the ability to write the statements and check the output easily. Simultaneously, it shows student performance scores are better than manual writing answers from different aspects for example syntax errors, using of reserved words and spelling mistakes.

The main variable of interest is the time needed to complete the quiz across two modes of test administration. This essentially means that average time to complete the quiz must be compared between two modes of test administration with lesser time indicating higher efficiency. Statistically this translates to a comparison of two means across two groups. Since the research design is paired as discussed in Section 4.5.2, where a sample of students take the same test twice across two modes of administration, measurements across two modes are not independent and hence, this becomes a related or paired group design. Therefore, paired *t* test or *t* test for two related samples is used to test the significance of the difference in mean time taken between pen and pencil and *SQL-FE* mode of test administration.

In this test, null hypothesis H_0 : There is no significant difference in mean time taken to complete the test between two modes of test administration ($\mu_1 = \mu_2$) is tested against the alternate hypothesis H_1 : There is a significant difference in mean time taken to complete the test between two modes of test administration ($\mu_1 \neq \mu_2$). That is, null hypothesis assumes no difference in efficiency while the alternate proposes a difference in efficiency of modes of tests. The test is performed at .05 level of significance. This means that upper limit for probability of committing Type I error of rejecting the null hypothesis when it is actually true is kept at an upper limit of 0.05. Actual level of significance for the data collected is indicated by p value of the test. This is a measure of probability that difference in average time between two modes of administration occurs due to chance. Null hypothesis is rejected if the p value of the test is less than .05.

Feedback on several aspects of test administration is collected based on a response measured on a scale of 1 to 5, where 5 represents most positive response and 1 represents the most negative response towards different aspects of SQL-*FE* test as (See Appendix 12 - Section D). Response is taken as an interval scale and is summarized using mean and standard deviation. Also, t test for single mean is used to test whether the response on an average is positive. That is, following statistical hypothesis is tested for response on each item.

Null hypothesis H_0 : Response on an average is not favourable ($\mu \leq 3.0$)

Alternate hypothesis H_1 : Response on an average is favourable ($\mu > 3.0$)

Rejection of the null hypothesis indicates that the response to a particular item is favourable and respondents, in general, report a positive response towards that aspect of semi-automated mode of administration.

6.3.5. Results

In this research, data analysis and statistical results have been measured by using SPSS tool as discussed previously in Section 4.5.2. However, to give an example of how the time spent has been calculated, one example has been discussed in details using a spreadsheet with paired t-test formula to be calculated. Table 6-4 lists the data which have been collected from two methods, SQL-*FE* method and Paper and Pencil method. The sample data has only selected 15 participants for each method.

This means each participant will do two different task, first will solve SQL quires using paper and pencil then solve the same query using SQL-FE. Once the participants finished both tasks then the time spent would be calculated as follows.

Table 6-4: Sample data of time spent between Paper pencil method and SQ-FE method

Std No.	M1/ Paper and Pencil	M2/ SQL-FE	Difference (M1-M2)	(Difference)^2
1	19.05	14.21	4.84	23.4256
2	17.22	14.12	3.10	9.6100
3	18.37	12.15	6.22	38.6884
4	16.30	11.16	5.14	26.4196
5	19.20	10.56	8.64	74.6496
6	11.30	10.27	1.03	1.0609
7	20.00	9.34	10.66	113.6356
8	12.30	9.08	3.22	10.3684
9	19.23	8.28	10.95	119.9025
10	15.20	7.40	7.80	60.8400
11	19.23	7.36	11.87	140.8969
12	16.45	7.27	9.18	84.2724
13	12.30	7.22	5.08	25.8064
14	15.36	10.47	4.89	23.9121
15	16	7.24	8.76	76.7376
Total Sum	247.51	146.13	101.38	830.23
Sample Mean	16.50	9.74	12.67	55.35

Example, Using the above table with n = 15 students, the following results were obtained:

1. Calculate the difference (**di = M1 – M2**) between the two observations on each pair, making sure you distinguish between positive and negative differences.
2. Find the Difference between both methods by calculating = **(M1-M2)**
3. Get the square of the Difference between both methods by calculating = **(Difference)^2**
4. Calculate the sum and mean of **M1, M2, Difference (M1-M2)** and **(Difference)^2**
5. The last thing is to calculate the t test by using following formula:

$$t = \frac{\sum d}{\sqrt{\frac{n(\sum d^2) - (\sum d)^2}{n-1}}} \quad (1) \quad t = \frac{101.38}{\sqrt{\frac{(15 \times 830.23) - (101.38)^2}{15-1}}} \quad (2)$$

$$t = \frac{101.38}{\sqrt{\frac{(12,453.45)-(10,277.90)}{14}}} \quad (3)$$

$$t = \frac{101.38}{\sqrt{\frac{2,175.55}{14}}} \quad (4)$$

$$t = \frac{101.38}{12.47} \quad (5)$$

$$t = 8.13 \quad (6)$$

Table 6-5 reports descriptive statistics of time taken to complete the test using two modes of test. Paper and pencil model reports an average of $M = 15.016$ minutes ($SD = 3.16$) while SQL-*FE* mode reports an average of $M = 8.864$ minutes ($SD = 3.789$). SQL-*FE* mode reports lesser mean time to complete the test.

Table 6-5: Descriptive Statistics of Time

Group	Mean	Std. Deviation	Std. Error Mean
Paper Pencil	15.016	3.1605	.4997
SQL- <i>FE</i>	8.864	3.7892	.6596

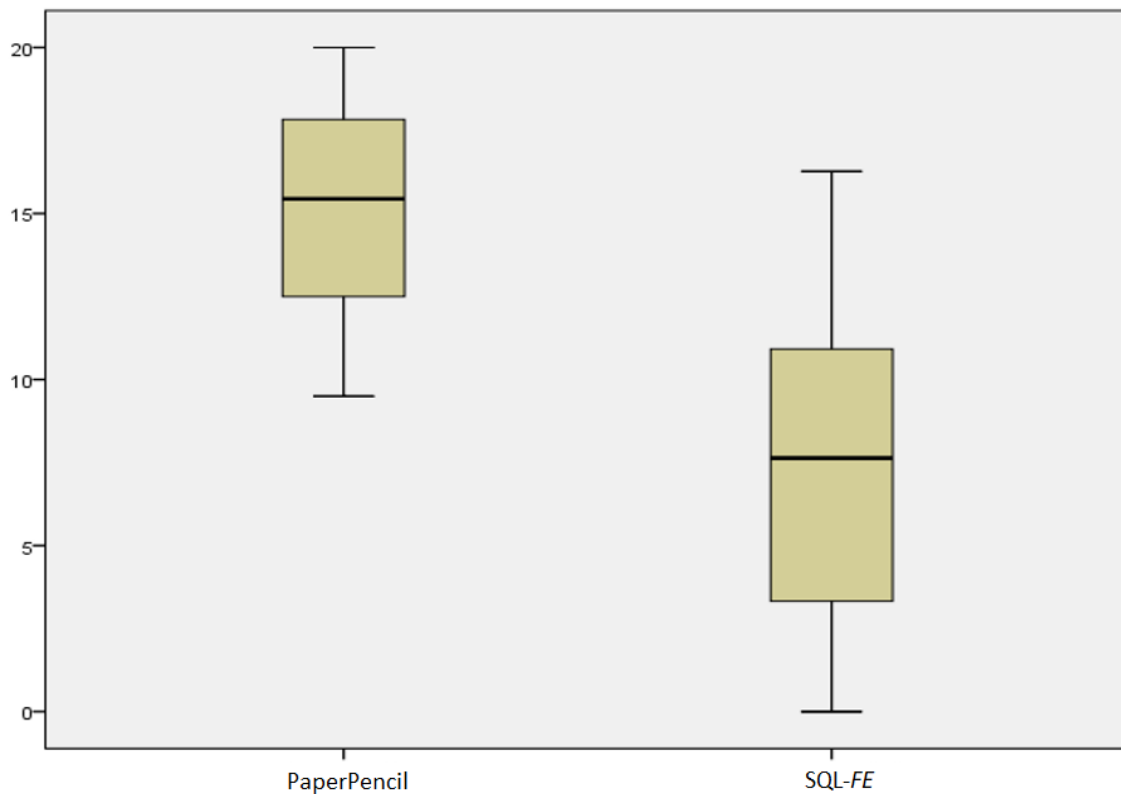


Figure 6-15: Boxplot of time taken to complete the test for two modes

Figure 6-15 reports box plot of distribution of time taken to complete the test across two modes. Box plot reports a difference in the distribution of time taken to complete the test. However, for both the modes, it does not report any abnormal or outlier observation indicating that the distribution does not report large departure from normality, which is an assumption for the validity of results of t test. This is also supported by histogram of distribution of time taken (Figure 6-16 and Figure 6-17) which report fairly symmetric distributions of time taken for pen and pencil and semi-automated methods of test administration.

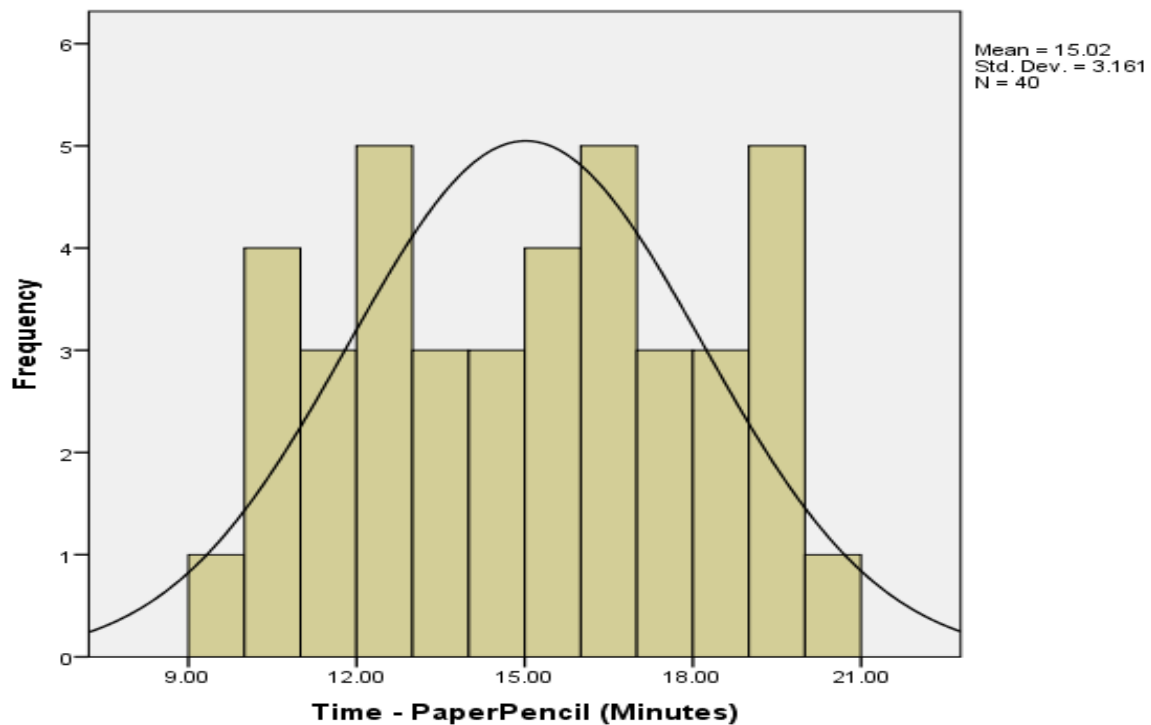


Figure 6-16: Histogram of distribution of time taken to complete the test for pen and pencil mode

Paired t test is used to test the significance of the difference in mean time. *SQL-FE* method reports a lesser mean time of magnitude $\mu_d = 6.538$ minutes compared to pen and pencil mode of administration (42.446% less time on an average). Results of the paired t test indicates that the null hypothesis of no significant difference must be rejected at .05 level of significance ($t(32) = 8.635, p = <.001$). This indicates that there is a significant difference in mean time taken to complete the test or equivalently, there is a significant difference in efficiency. Even for one-sided hypothesis ($H_1: \mu_{SQL-FE} < \mu_{pen \text{ and } pencil}$) results indicate significant difference.

These results clearly provide strong evidence for statistical significance of difference (reduction) in time taken to complete the test between pen and pencil and SQL-*FE* modes of administration. That is SQL-*FE* test reports significantly higher efficiency compared to pen and pencil mode as illustrated in Figure 6-17.

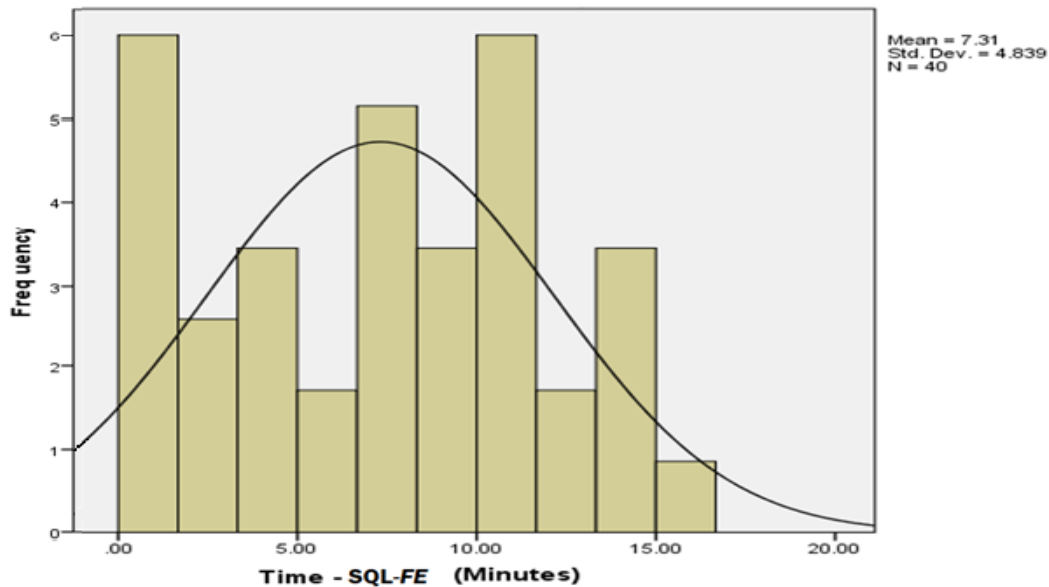


Figure 6-17: Histogram of distribution of time taken to complete the test for SQL-*FE* mode

Table 6-6 reports descriptive statistics of response to different items (questions) related to semi-automated mode of test. All the questions reported mean of the response more than 3.00. Overall satisfaction with the system developed reported the highest mean (M = 4.0476) followed by the overall quality of the system (M = 3.9048).

Table 6-6: Descriptive Statistics for Response to Feedback Questions

Question	N	Mean	Std. Deviation
Overall, how satisfied are you with our SQL- <i>FE</i> editor?	21	4.05	1.1
How well does the SQL- <i>FE</i> editor meet your needs?	21	3.4	1.1
How would you rate the quality of our SQL- <i>FE</i> editor?	21	3.9	.88
How helpful was the help video tutorial?	20	3.5	1.1
How easy was it to find what you were looking for in our SQL- <i>FE</i> editor?	21	3.5	.98

Table 6-6 reports results of t test for single mean testing whether the mean response is significantly more than 3.0. Rejection of the null hypothesis indicates that it is significantly more than 3.0 and provides strong evidence in favour of the item. Two items related to overall satisfaction and overall quality of the system developed, both report p value less than .05 ($p < .05$). This indicates that mean response to these two statements is significantly more than 3.0. That is, the response to these two questions is positive. Students are highly satisfied with the system and quality of the system. Similarly, ease with which students are able to find what they are looking for in the website also reported p value less than .05 ($p = .038$). This indicates that it was easy for students to get whatever information they needed from the website. However, the response to the question, “how well our system meets your needs” reports p value more than .05. This means that mean score for the response to this item is not greater than 3.0. This indicates that there is no evidence to infer that response is positive to the system meeting needs of students. Similarly, the response cannot be termed as decisively positive for the helpfulness of help video tutorial. Analysis of response to questions on feedback related to developed system clearly indicates that overall satisfaction level is high, rating on overall quality of the system is high, ease of finding information is also high but the help video tutorial is not significantly useful and it cannot be inferred that the system meets all the needs of students. Some more features can be incorporated as a part of the system to ensure that it covers all the requirements of students.

Table 6-7: Results of T test for Response to Feedback Questions

Item	t	p	95% CI of Difference	
Overall, how satisfied are you with our SQL-FE editor?	4.481	<.001	(.5600	1.535)
How well does our SQL-FE editor meet your needs?	1.752	.095	(-.0818	.9389)
How would you rate the quality of our SQL-FE editor?	4.663	<.001	(.5001	1.3095)
How helpful was the help video tutorial?	2.032	.056	(-.0149	1.0149)
How easy was it to find what you were looking for in our SQL-FE editor?	2.225	.038	(.0298	.9226)

Table 6-8 reports results of the t-test for single mean testing on whether the mean response is significantly more than 3.0. Rejection of the null hypothesis indicates that it is significantly more than 3.0 and provides strong evidence in favour of the question. Two questions related to overall satisfaction and overall quality of the SQL-FE editor report a p value of less than 0.05 ($p < 0.05$).

This indicates that the mean response to these two statements is significantly more than 3.0, which means that the responses to these two questions are positive and students were highly satisfied with the SQL-*FE* and quality of the editor. Similarly, the ease with which students were able to find what they were looking for in the editor also reported a p value of less than 0.05 ($p = .038$), thus indicating that it was easy for students to get whatever data they needed from the editor.

However, the response to the question “how well does our SQL-*FE* editor meet your needs?” reports a p value higher than (0.05). This means that the mean score for the response to this question is not significantly greater than (3.0), which indicates that there is no evidence to infer that the response is positive with regards to the editor meeting the needs of students. Similarly, the response cannot be termed as decisively positive for the helpfulness of the help video tutorial. In conclusion, analysis of the responses to feedback questions related to the developed SQL-*FE* editor clearly indicates that the overall satisfaction level, rating of the overall quality of the SQL-*FE* editor and ease of finding information are all high. In contrast, the help video tutorial was not significantly useful. In addition, one cannot infer that the SQL-*FE* editor meets all the needs of students. In response to this feedback, more features can be incorporated as part of the SQL-*FE* editor to ensure that it covers all the requirements of students.

Table 6-8: Results of the *t*-test for the response to the feedback questions

Question	t	p
Overall, how satisfied are you with our SQL- <i>FE</i> editor?	4.5	<.001
How well does our SQL- <i>FE</i> editor meet your needs?	1.8	.095
How would you rate the quality of our SQL- <i>FE</i> editor?	4.7	<.001
How helpful was the help video tutorial?	2.0	.056
How easy was it to find what you were looking for in our SQL- <i>FE</i> editor?	2.2	.038

6.4. Experiment

An experimental study was conducted with the objectives of measuring the mean time spent and students’ performance by comparing two query formulation tools, SQL-*FE* and SQL Server Management Studio (SSMS). In order to provide a better understanding of the effect of using SQL-*FE* over the SSMS tool, the research identified two questions for the query formulation experiment, which are:

- **RQ1: Does using SQL-FE during the experiment lead to spending more or less time on solving SQL questions?**

This question aimed to investigate the degree to which students spent more or less time to answer the SQL questions.

- **RQ2: Does using the SQL-FE enhance student grading performance?**

This question aimed to investigate the degree to which students of the SQL-FE tool managed to achieve more marks in solving SQL questions than solving them using the SQL formulation tools.

6.4.1. SQL Formulation Editor (SQL-FE)

In this experiment, two different SQL formulating tools were used, the newly implemented SQL-FE tool and the SSMS tool. The SQL-FE tool is an SQL formulation tool that allows students to solve SQL questions by formulating an SQL SELECT statement posed by the examiners. SQL-FE then collects these SQL solution responses for marking and providing feedback. The SSMS tool is the SQL Server Management Studio, whose user interface is depicted in Figure 6-18.

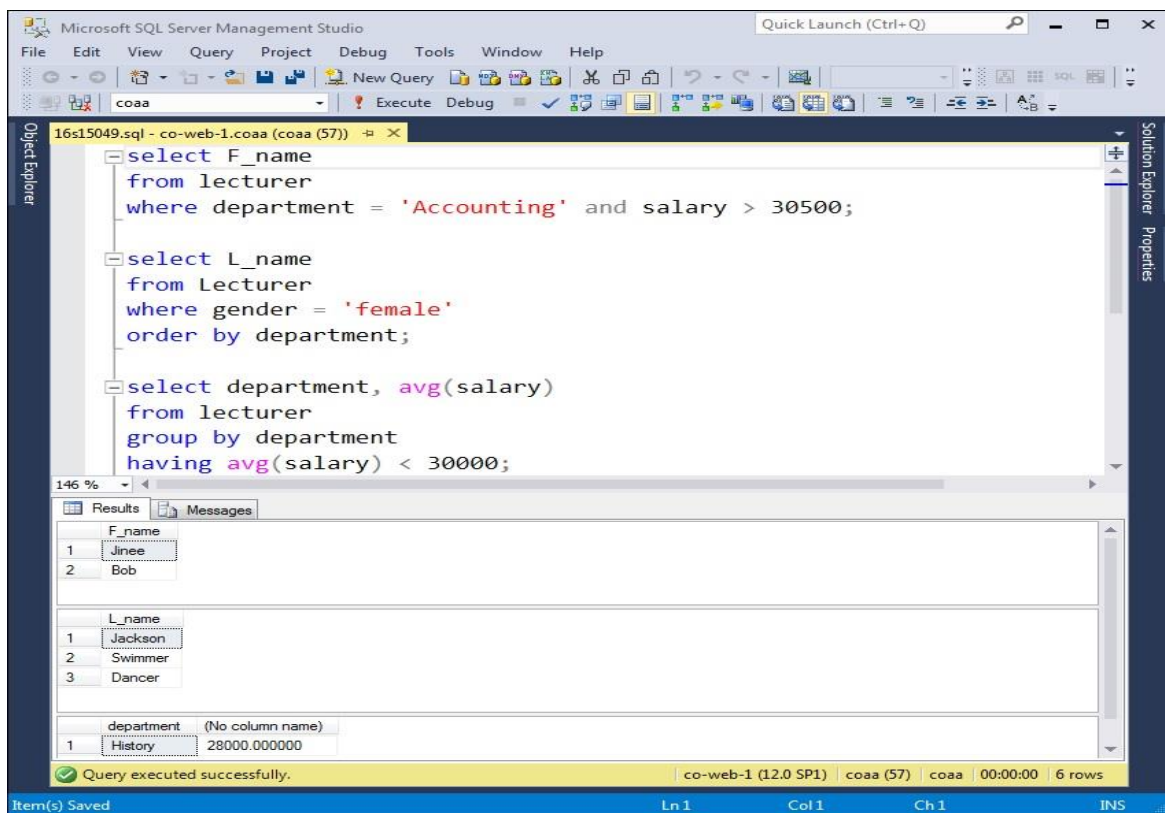


Figure 6-18: Executed SQL statements using the SSMS Tool

Figure 6-18 shows the execution of SQL statements, submitted by participating students, who were able to run one statement at a time or several statements simultaneously. The SSMS tool enables users to enter and execute SQL statements to perform calculations and store and retrieve query results. It was practised by a student in Middle East College, Sultanate of Oman, where this experiment took place. The two tools that were compared during the experiment are based on two different approaches, the keyboard typing approach and the point-and-click approach. The *SQL-FE* tool does not allow students to write or type SQL statements using a keyboard, whereas the SSMS tool only allows user to formulate statements using a keyboard. Restricting the students from using the keyboard aims to minimise the errors of SQL statements like spelling errors, synonyms and adding invalid identifiers.

There are some cases in which the student may need to use the keyboard in the *SQL-FE* tool, but these have been addressed by adding a text area pane. This pane helps users to add different numeric or string values to limit the data retrieved, which cannot be done using the available navigation described in detail in subsection (5.2.1).

6.4.2. Participants

The participants were 20-to-21-year-old second-year undergraduate students. The total number of participants was 60 students. The participating students were registered under the Computer Science Programme in the Middle East College, Oman. Furthermore, the students had undertaken the Introduction to Database module as a first-year module. In the second year, they studied SQL concepts and syntax in the Fundamentals of Relational Database Management System module. This module is taught twice a week in the college, where the first session is a 2-hour theory lecture and the second involves a 2-hour practical session in a lab. The purpose of the lecture is to teach and explain the concepts of the relational database system and teach students the SQL syntax, so they could apply their knowledge during the lab session. There were two lab course groups that studied SQL, with 30 students in each group. The experiment was implemented during a lab session. The two lab course groups were divided into two days, Sunday and Tuesday.

6.4.3. SQL Questions

The task given to the participants of the experiment was to solve five different SQL questions. The questions were obtained from two SQL practical text books [John, 1992; Bisland, 1989]. The questions contained the basic SQL commands which require participants to write basic SQL queries (such as SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN and SUBQUERY). For each SQL question, the lecturer provided at least one SQL model solution. This allowed for multiple acceptable solutions submitted by the students. These five questions covered most of what students had learned in the SQL module. The SQL questions were selected based on the following specific criteria:

- Each SQL question should have a clear and obvious purpose
- The question should be well asked and be provided with accurate answers with an alternative way of answers if available.
- The question can challenge the participants' SQL skills but should be simple and easy to be formulated.
- All questions should be tested before the experiment takes place.

The experiment design created two question sets attached with two SQL formulating tools such as, set "A" questions for the SSMS tool and set "B" for SQL-FE. For set "A" questions, there were two tables used to retrieve information from: a lecturer table and a course table. The lecturer table contained six columns and seven records, and the course table contained three columns and seven records, as shown in Table 6-9 and Table 6-10.

Table 6-9: The lecturer table

LECT_ID	F_NAME	L_NAME	DEPARTMENT	GENDER	SALARY
D01	Amy	Dancer	Computer Science	Female	34500
J01	Ray	Johnson	Computer Science	Male	40000
S01	Wendy	Swimmer	Computer Science	Female	45000
J02	Bob	Jones	Accounting	Male	35000
N01	Jack	Nelson	History	Male	28000
D02	Jinee	Jackson	Accounting	Female	34500
S02	William	James	Accounting	Male	30500

Table 6-10: The course table

COURSE_ID	COURSE_TITLE	LECT_ID
CSC100	Intro. to Computing	J01
CSC101	Pascal Programming	D01
CSC102	Database Management	J01
ACC200	Principles of Accounting I	J02
ACC201	Principles of Accounting II	D02

The relationship between the lecturer and course tables is a one-to-many relationship since one lecturer teaches many courses, as illustrated in Figure 6-19. The figure shows that the relationship associated with the two tables is linked by the LECTID primary key in the lecturer table and foreign key in the course table.

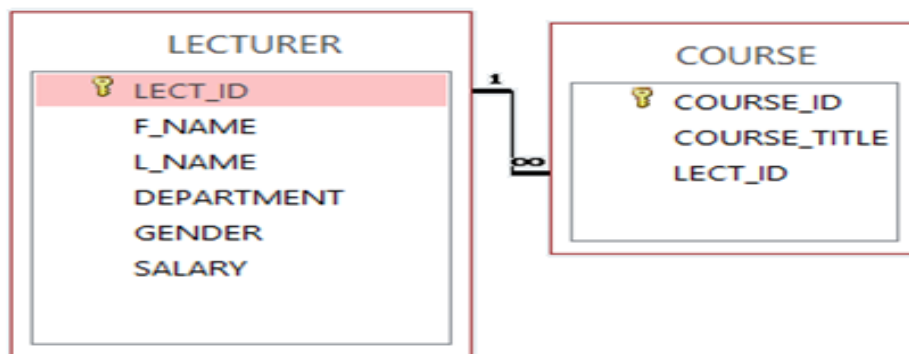


Figure 6-19: The relationship between the lecturer and course tables

The two sets of SQL questions, set “A” and set “B”, are illustrated in Table 6-11 and Table 6-14, respectively. As previously mentioned, each set of questions was run in a different tool, where set “A” questions was run on the SSMS tool and set “B” questions were run on SQL-FE. Both sets contained five similar question requirements, yet each group contained different tables and field names. The similarities between the two question sets were measured using different parameters such as the SQL commands needed for each question, the number of fields used, the required conditions and joining tables, and the gradual complexity of the question. This ensured that the two question sets were closely related, but did not contain identical questions. This was done due to the fact that the aim of the experiment was to evaluate students’ performance using both tools, and as such, if students were to be given the same questions twice, they would get similar grades each time, which would cause the evaluation of the two tools to provide similar statistics and not show the difference between the tools.

Table 6-11: SQL questions and their model answers: SET A

Question 1	Find the first names of all lecturers who work in the accounting department with salaries greater than 30500.						
Model Answer 1	<pre>SELECT F_NAME FROM LECTURER WHERE DEPARTMENT='Accounting' AND SALARY > 30500;</pre>						
Output 1	<table border="1"> <thead> <tr> <th>F_NAME</th> </tr> </thead> <tbody> <tr> <td>Bob</td> </tr> <tr> <td>Jinee</td> </tr> </tbody> </table>	F_NAME	Bob	Jinee			
F_NAME							
Bob							
Jinee							
Question 2	Retrieve the last names and the course titles of all female lecturers. Sort the result in ascending order of the department.						
Model Answer 2.1	<pre>SELECT L.L_NAME, C.COURSE_TITLE FROM LECTURER L INNER JOIN COURSE C ON L.LECT_ID = C.LECT_ID WHERE L.GENDER = 'Female' ORDER BY L.DEPARTMENT;</pre>						
Model Answer 2.2	<pre>SELECT L.L_NAME, C.COURSE_TITLE FROM LECTURER L, COURSE C WHERE L.LECT_ID = C.LECT_ID AND L.GENDER = 'FEMALE' ORDER BY L.DEPARTMENT;</pre>						
Output 2	<table border="1"> <thead> <tr> <th>L_NAME</th> <th>COURSE_TITLE</th> </tr> </thead> <tbody> <tr> <td>Jackson</td> <td>Principles of Accounting II</td> </tr> <tr> <td>Dancer</td> <td>Pascal Programming</td> </tr> </tbody> </table>	L_NAME	COURSE_TITLE	Jackson	Principles of Accounting II	Dancer	Pascal Programming
L_NAME	COURSE_TITLE						
Jackson	Principles of Accounting II						
Dancer	Pascal Programming						
Question 3	Find the department and average salary of lecturers at each department where the average salary is greater than 35000.						
Model Answer 3	<pre>SELECT DEPARTMENT, AVG(SALARY) FROM LECTURER GROUP BY DEPARTMENT HAVING AVG(SALARY) > 35000;</pre>						
Output 3	<table border="1"> <thead> <tr> <th>DEPARTMENT</th> <th>AVG(SALARY)</th> </tr> </thead> <tbody> <tr> <td>Computer Science</td> <td>39833.3333</td> </tr> </tbody> </table>	DEPARTMENT	AVG(SALARY)	Computer Science	39833.3333		
DEPARTMENT	AVG(SALARY)						
Computer Science	39833.3333						
Question 4	Find the title of all courses taught by lecturers in the history department.						
Model Answer 4	<pre>SELECT COURSE_TITLE FROM COURSE WHERE LECT_ID IN (SELECT LECT_ID FROM LECTURER WHERE DEPARTMENT = 'History');</pre>						

	<pre>SELECT C.COURSE_TITLE FROM COURSE C INNER JOIN LECTURER L ON L.LECT_ID = C.LECT_ID WHERE L.DEPARTMENT = 'History';</pre>			
Output 4	<table border="1"> <thead> <tr> <th>COURSE_TITLE</th> </tr> </thead> <tbody> <tr> <td>England History</td> </tr> <tr> <td>Europe History</td> </tr> </tbody> </table>	COURSE_TITLE	England History	Europe History
COURSE_TITLE				
England History				
Europe History				
Question 5	Identify the department with the highest average salary.			
Model Answer 5	<pre>SELECT DEPARTMENT, AVG(SALARY) FROM LECTURER GROUP BY DEPARTMENT HAVING AVG(SALARY) >= ALL (SELECT AVG(SALARY) FROM LECTURER GROUP BY DEPARTMENT);</pre>			
Output 5	<table border="1"> <thead> <tr> <th>DEPARTMENT</th> </tr> </thead> <tbody> <tr> <td>Computer Science</td> </tr> </tbody> </table>	DEPARTMENT	Computer Science	
DEPARTMENT				
Computer Science				

For set “B” questions, there were two tables used to retrieve information from, named Department (as DEPT) and Employee (as EMP). The EMP table contained seven columns and seven records, and the DEPT table contained three columns and five records, as shown in Table 6-12 and Table 6-13.

Table 6-12: EMP Table

EMPNO	FNAME	LNAME	GENDER	JOB	SALARY	DEPTNO
7369	Smith	Jones	Male	Clerk	1500	20
7499	Allen	Louis	Female	Salesman	1600	50
7521	Danny	Dawson	Male	Salesman	1250	30
7566	Jones	William	Male	Clerk	2975	20
7654	Martin	Oliver	Male	Salesman	1250	30
7698	Laura	Paul	Female	Manager	2850	40
7782	Clark	Richard	Male	Manager	2450	10

Table 6-13: DEPT Table

DEPTNO	DEPTNAME	LOC
10	Accounting	New York
20	Research	New Jersey
30	Sales	Chicago
40	Operation	Boston
50	Management	New York

The relationship between the department and employee tables is a one-to-many relationship, as in one department many employees work, as illustrated in Figure 6-20. The figure shows the relationship associated with the two tables is based on the DEPTNO primary key in department table and foreign key in the employee table.

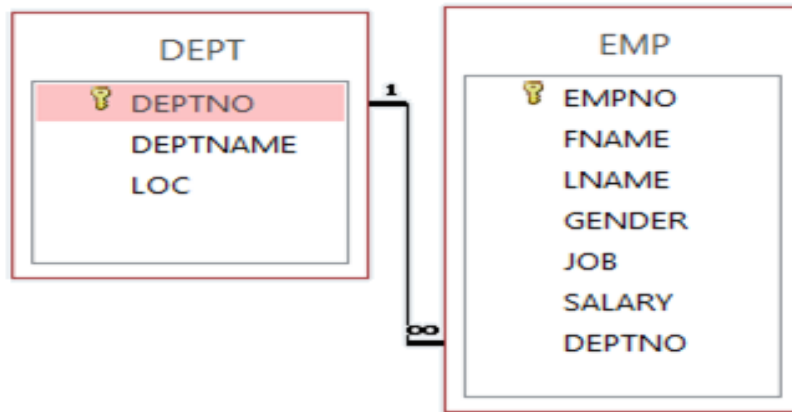


Figure 6-20: The relationship between the department and employee tables

Table 6-14 presents the set “B” list of questions. It contains five SQL questions that ask to retrieve data from the department and employee tables, along with their model answers.

Table 6-14: SQL questions with their model answer: SET B

Question 1	Find the first names of all employees who work as a clerk and earn a salary of more than 2500		
Model Answer 1	<pre>SELECT EMP.FNAME FROM EMP WHERE EMP.JOB= 'CLERK' AND EMP.SALARY > 2500;</pre>		
Output 1	<table border="1"> <thead> <tr> <th>FNAME</th> </tr> </thead> <tbody> <tr> <td>Jones</td> </tr> </tbody> </table>	FNAME	Jones
FNAME			
Jones			
Question 2	Retrieve the last names and the department names of all female employees. Sort the result in ascending order of the location.		
Model Answer 2.1	<pre>SELECT EMP.LNAME, DEPT.DEPTNAME FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO WHERE EMP.GENDER='FEMALE' ORDER BY DEPT.LOC;</pre>		
Model Answer 2.2	<pre>SELECT EMP.LNAME, DEPT.DEPTNAME FROM EMP , DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO AND EMP.GENDER = 'FEMALE' ORDER BY DEPT.LOC;</pre>		

Output 2	<table border="1"> <thead> <tr> <th>LNAME</th> <th>DEPTNAME</th> </tr> </thead> <tbody> <tr> <td>Paul</td> <td>Operation</td> </tr> <tr> <td>Louis</td> <td>Management</td> </tr> </tbody> </table>	LNAME	DEPTNAME	Paul	Operation	Louis	Management
LNAME	DEPTNAME						
Paul	Operation						
Louis	Management						
Question 3	Display the various jobs and the average salary of employees in each job, where the average salary is greater than 2000.						
Model Answer 3	<pre>SELECT EMP.JOB, AVG(EMP.SALARY) FROM EMP GROUP BY EMP.JOB HAVING AVG(EMP.SALARY) > 2000;</pre>						
Output 3	<table border="1"> <thead> <tr> <th>JOB</th> <th>AVG(SALARY)</th> </tr> </thead> <tbody> <tr> <td>Manager</td> <td>2650</td> </tr> </tbody> </table>	JOB	AVG(SALARY)	Manager	2650		
JOB	AVG(SALARY)						
Manager	2650						
Question 4	List all department names of all employees who work as a manager.						
Model Answer 4.1	<pre>SELECT DEPTNAME FROM DEPT WHERE DEPTNO IN (SELECT DEPTNO FROM EMP WHERE JOB = 'MANAGER');</pre>						
Model Answer 4.2	<pre>SELECT DEPTNAME FROM DEPT INNER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO WHERE EMP.JOB = 'MANAGER';</pre>						
Output 4	<table border="1"> <thead> <tr> <th>DEPTNAME</th> </tr> </thead> <tbody> <tr> <td>Accounting</td> </tr> <tr> <td>Operation</td> </tr> </tbody> </table>	DEPTNAME	Accounting	Operation			
DEPTNAME							
Accounting							
Operation							
Question 5	Identify the job with the lowest average salary.						
Model Answer 5	<pre>SELECT JOB, AVG(SALARY) FROM EMP GROUP BY JOB HAVING AVG(SALARY) <= ALL (SELECT AVG(SALARY) FROM EMP GROUP BY JOB);</pre>						
Output 5	<table border="1"> <thead> <tr> <th>JOB</th> </tr> </thead> <tbody> <tr> <td>Salesman</td> </tr> </tbody> </table>	JOB	Salesman				
JOB							
Salesman							

6.4.4. Design of the Experiment

A crossover design (also called “change-over design”) study is a special form of a controlled double randomised trial (Gardiner and Gettinby, 1998). Randomised means that every student has an equal chance of being assigned to the experimental subject on a random basis.

In the context of this experiment, this design is more efficient in establishing the highest possible similarity among SQL questions exposed to different tools (Li, 1964). Therefore, to achieve the purpose of the study, a crossover experimental design was employed. Another reason for adopting a crossover design was to minimise failures from the control group. The study was approved by Loughborough University’s Ethical Committee. Table 6-15 provides a full description of the crossover experimental design implemented over two weeks' time.

In week one, two different sessions took place. The experiment involved a total of 60 students using the two tools (i.e. *SQL-FE* and *SSMS*). They were divided into two different experiment days, where each experiment involved 30 students due to the limited number of available PCs in each computer lab. The students were randomly assigned into two groups, where an equal distribution of 15 students used *SQL-FE* and 15 others used the *SSMS* tool, as shown below. Each tool used in the experiment was attached to a certain set of questions (SET A & B). In addition, a rest period between the two tests was applied so that the effect of one test does not carry over to the next test, as indicated by the period column in Table 6-15. This means that there was one experiment in Session 1.1 involving 30 students, with 15 students using *SQL-FE* and 15 others were using *SSMS*. Subsequently, a week later, Session 1.2 took place, where the two groups of students swapped over the tool used. The same procedure was adopted in Sessions 2.1 and 2.2, where the same process was repeated, involving a total of 30 students using the two tools over two weeks.

Table 6-15: The Crossover Experimental Design Distribution

Group	Tool	Question SET	No. of Participants	Period	Session No.
X	<i>SQL-FE</i>	SET A	15	Week I	Session 1.1
Y	<i>SSMS</i>	SET B	15		
W	<i>SQL-FE</i>	SET A	15		Session 2.1
Z	<i>SSMS</i>	SET B	15		
X	<i>SSMS</i>	SET A	15	Week II	Session 1.2
Y	<i>SQL-FE</i>	SET B	15		
W	<i>SSMS</i>	SET A	15		Session 2.2
Z	<i>SQL-FE</i>	SET B	15		

The experiment preparation went through several activities such as preparing the SQL questions and defining suitable answers for each of them, as well as preparing a set of instructions for students and lecturers. These instructions were used for the lecturer to explain the steps for the students before starting the experiment. The SSMS examiner's instructions are available in Appendix 12 - Section A, while the instructions for *SQL-FE* can be found in Appendix 12 - Section C. In addition, for the students to understand the steps when they are attempting the experiment, they needed to read the instructions that had been prepared for both tools. Before the experiment day, the computer labs were checked to make sure that the required number of students could be accommodated, and where students seating was set randomly.

The examiner copied and pasted the SQL code to create the department and employee tables to be used for the SQL Management Studio. This procedure saved time for students once they started the experiment, as they only needed to write the SQL statement and retrieve the data. Furthermore, in order to ensure the functionality of the student groups, the research provided a brief introduction about how to use the newly implemented *SQL-FE* tool. The participants participated in a simplified version of the experiment, which helped them to clarify the functionality of the new tool and how the experiment would proceed. This was done to account for the fact that the students had familiarity with running SQL statements using the SSMS tool in their lab sessions, but not with the and using the new *SQL-FE* tool.

On the experiment day, the examiner took 20 minutes to set up the lab session, which included the randomisation of the tools and checking the functionality of all PC's SSMS program installation and internet connection. Each participant chose a PC freely upon arrival to the computer lab. However, the examiner and the assistant lecturer made sure that every two participants next to one another conducted the same test. To achieve this, they sat a random tool for the students using number cards in the computer lab containing 30 PC's. Moreover, the examiner and assistant lecturer checked all PCs for the preparation of the tools by giving the participants' time to copy the URL for the *SQL-FE* tool and to start the SSMS application. They also distributed the printed instructions and question lists to the participants to assist them while conducting the exam. Once the setting was ready and participants had taken their place, the experiment started.

Each participant performed one experiment a day involving either *SQL-FE* with five SQL questions or *SSMS* with a list of five SQL questions printed in hardcopy. Each SQL question involved writing an SQL statement with different commands and conditions then saving them using the *SSMS* tool or submitting the answer using *SQL-FE*. The participants were allowed to delete and rewrite the statements as long as that was done within the duration of the experiment. The duration of each task was 45 minutes, which allowed the participants to go through the questions and test them manually before deciding to write the answers. The time spent on each SQL question submitted using the *SQL-FE* was saved automatically by the tool itself. However, the examiner and the assistant made sure to remind the participants to write the start time and submission time in a file, so that analysis can later be conducted by the examiner. This was done to help the examiner to record the time spent by each student, since the *SSMS* tool does not offer a time saving function. The experiment went smoothly and the participants were motivated to perform the tasks and attain experience on the newly implemented tool.

6.4.5. Statistical Analysis

Once the participants finished solving the SQL questions, they were asked to log off if using *SQL-FE* to save all their answers. At the same time, the lecturer and assistants created a shared folder to save all the created files retrieved from the *SSMS* tool. All participants were asked in the instructions to save the file with their college email address to keep it anonymous. The email address allowed the examiner to match between the participants in the first and second day of the experiment.

The data collected from both tools was dated and saved in different folders to be analysed and evaluated. The main objectives of the evaluation were to measure the time efficiency of the *SQL-FE* tool over the *SSMS* tool, and to assess if the former can provide a more helpful environment for learning and teaching SQL statements than the latter. The main variable of interest was the time needed to complete the experiment across the two tools of test administration. This essentially means that the average time to complete the experiment must be compared between the two tools of test administration with shorter time indicating higher efficiency. Statistically, this translates to a comparison of two means across two groups.

Since the research design is paired, where a sample of students take the same test twice across two tools of administration, measurements across two tools are not independent, and as such, this becomes a related or paired group design. Therefore, a paired t-test for two related samples was used to test the significance of the difference in the meantime taken to complete the experiment between *SQL-FE* and the *SSMS* tool.

6.4.6. Mean Time Hypotheses

Null hypothesis H0: There is no significant difference in the meantime taken to complete the experiment between the two tools of test administration ($\mu_1 = \mu_2$).

Alternative hypothesis H1: There is a significant difference in the meantime taken to complete the experiment between the two tools of test administration ($\mu_1 \neq \mu_2$).

That is, the null hypothesis assumes no difference in the meantime while the alternative hypothesis proposes a difference in the mean time between the two tools of the experiment. The test is performed at 0.05 level of significance. This means that the upper limit for a probability of committing Type I error of rejecting the null hypothesis when it is actually true is kept at an upper limit of 0.05. The actual level of significance for the data collected is indicated by the p-value of the test. This is a measure of the probability that a difference in average time between two modes of administration occurs due to chance. The null hypothesis is rejected if the p-value of the test is less than 0.05. The main objective of the evaluation was to measure and compare the participants' performance when using the *SQL-FE* tool over the *SSMS* tool.

6.4.7. Marks/Performance Hypotheses

Null hypothesis H0: There is no difference between the mean *SQL-FE* and *SSMS* marks ($\mu_1 = \mu_2$)

Alternative hypothesis H1: There is a difference between the mean *SQL-FE* and *SSMS* marks ($\mu_1 \neq \mu_2$).

That is, the null hypothesis assumes no difference in the participants' marks while the alternative proposes a difference in participants' marks when using the two tools. The test is performed at 0.05 level of significance.

6.4.8. Results and Discussion

The descriptive statistics of the time taken to complete the test using two the tools of the experiment are as follows. The *SQL-FE* tool reports an average of $M = 20.40$ minutes ($SD = 7.84$) while *SSMS* reports an average of $M = 24.67$ minutes ($SD = 7.31$). In other words, the *SQL-FE* tool reports a lower mean time to complete the test. Figure 6-21 depicts a box plot of the distribution of time taken to complete the test using the two tools. The box plot reports a difference in the distribution of time taken. However, for both tools, the box plot does not report any abnormal outlier observation, which indicates that the distribution does not report a large departure from normality, which is an assumption for the validity of the results of the *t*-test.

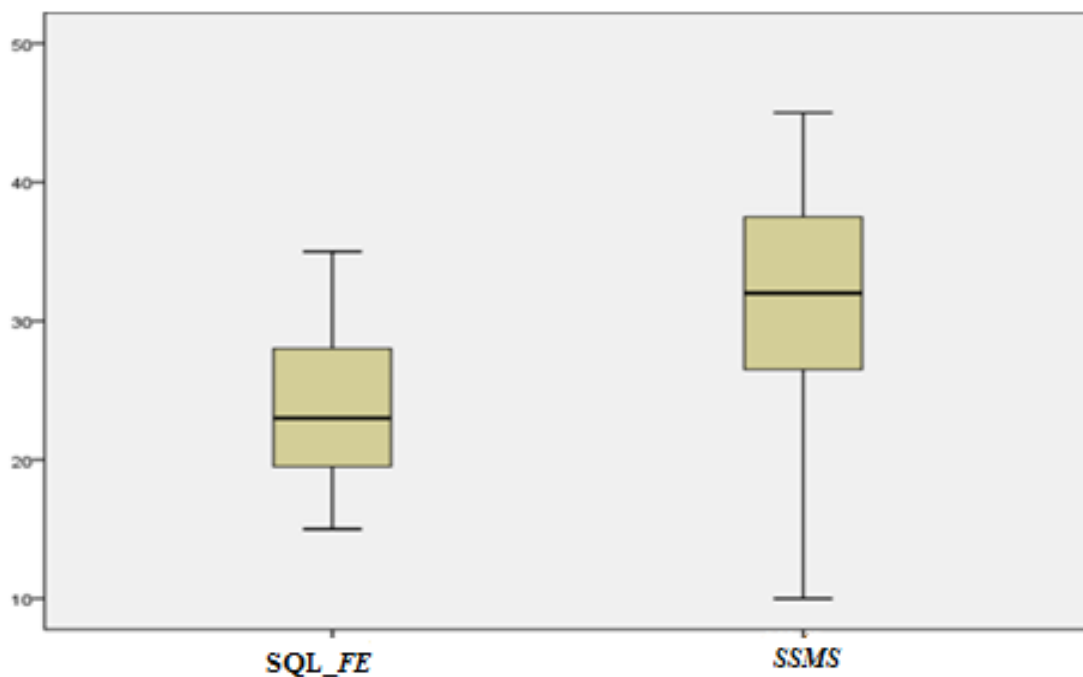


Figure 6-21: Boxplot of the time taken to complete the test using the two tools

This is also supported by a histogram of the distribution of time taken to solve the tests using the *SQL-FE* and *SSMS* tools of test administration (Figure 6-22 and Figure 6-23, respectively).

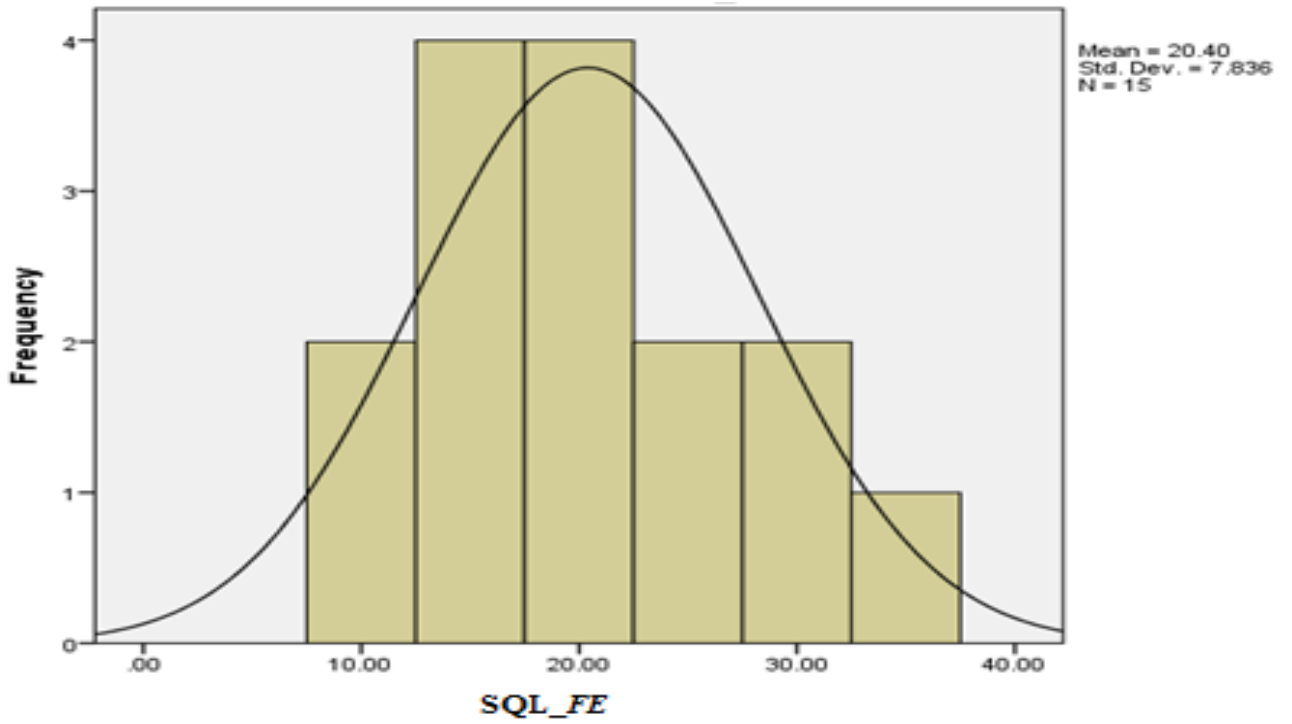


Figure 6-22: Histogram of the distribution of time taken to complete the test using *SQL-FE*

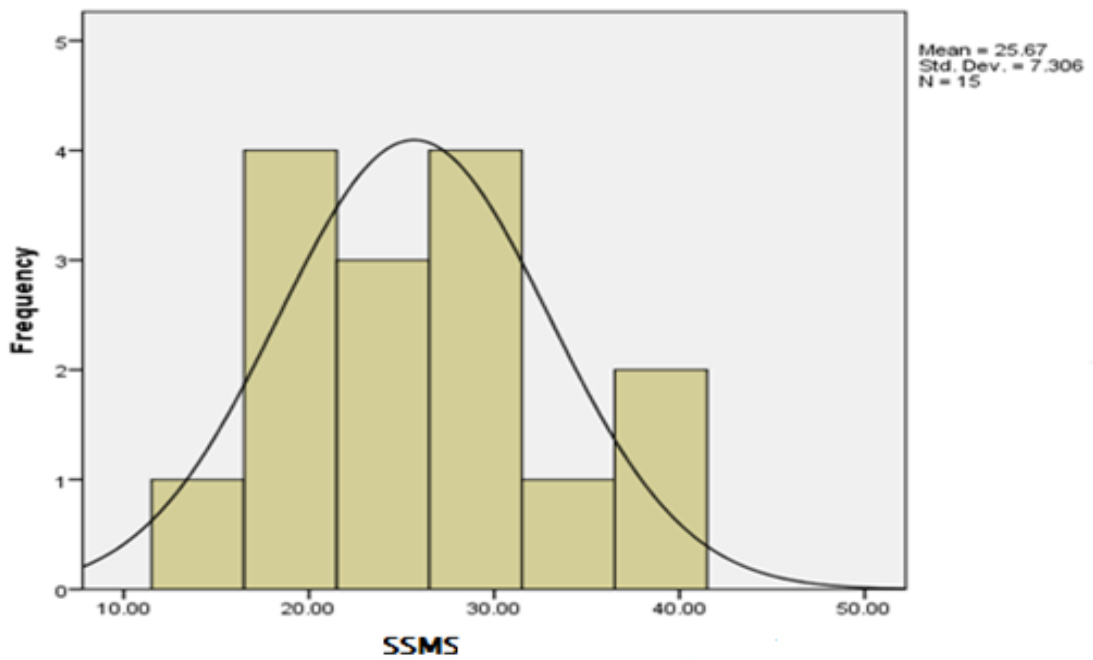


Figure 6-23: Histogram of the distribution of time taken to complete the test using *SSMS*

The paired t-test is used to test the significance of the difference in mean time. The *SQL-FE* tool reports a smaller mean time value compared to the *SSMS* tool of the administration. Results of the paired t-test indicate that the null hypothesis of no significant difference must be rejected at 0.05 level of significance, as shown in Table 6-16. This indicates that there is a significant difference in the meantime taken to complete the test, or equivalently, that there is a significant difference in efficiency. Even for a one-sided hypothesis ($H_1: \mu_{SQL-FE} < \mu_{SSMS}$), the results indicate a significant difference.

Table 6-16: Paired samples test of the two tools

	Paired Differences					t	Df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
Pair 1 SQL_FE - SSMS	-7.533	12.082	3.120	-14.224	-0.842	-2.415	14	0.030

These results clearly provide strong evidence for the statistical significance of difference (reduction) in the time taken to complete the test using the *SQL-FE* tool compared to the *SSMS* tool. That is, the *SQL-FE* test reports significantly higher efficiency compared to the *SSMS* tool. Table 6-17 reports the descriptive statistics of the mean marks obtained by students using the two tools of the experiment. The *SQL-FE* tool reports an average of $M = 10.5$ marks ($SD = 3.1$) while *SSMS* reports an average of $M = 8.8$ marks ($SD = 3.7$). In other words, *SQL-FE* reports higher marks obtained by the participants than the *SSMS* tool.

Table 6-17: Descriptive statistics of the mean marks obtained using the two tools

Tool	Mean	Std. Deviation
<i>SQL-FE</i>	10.5	3.1
<i>SSMS</i>	8.8	3.7

Furthermore, the null hypothesis is rejected, since $p < 0.05$, as illustrated in Table 6-16. In this context, there is strong evidence ($t = 2.41, p = .030$) that formulating SQL questions using *SQL-FE* improves the participants' marks.

In this data set, it improved marks by an average of approximately two marks. If the experiment takes other samples of marks, it could find a 'mean paired difference' in marks that is different from the 1.76 value reported here. This is why it is important to look at the 95% Confidence Interval (95% CI). In this case, the 95% CI ranges from 0.2 to 3.3. This confirms that, although the difference in marks is statistically significant, it is actually relatively small. Figure 6-24 depicts a box plot of the distribution of marks obtained by the participants using the two tools. The box plot reports a difference in the distribution of performance marks that shows an increase in the marks obtained using *SQL-FE*. For the *SSMS* tool, the figure shows lower marks since the participants had to write complete SQL statements, which led them to commit more mistakes.

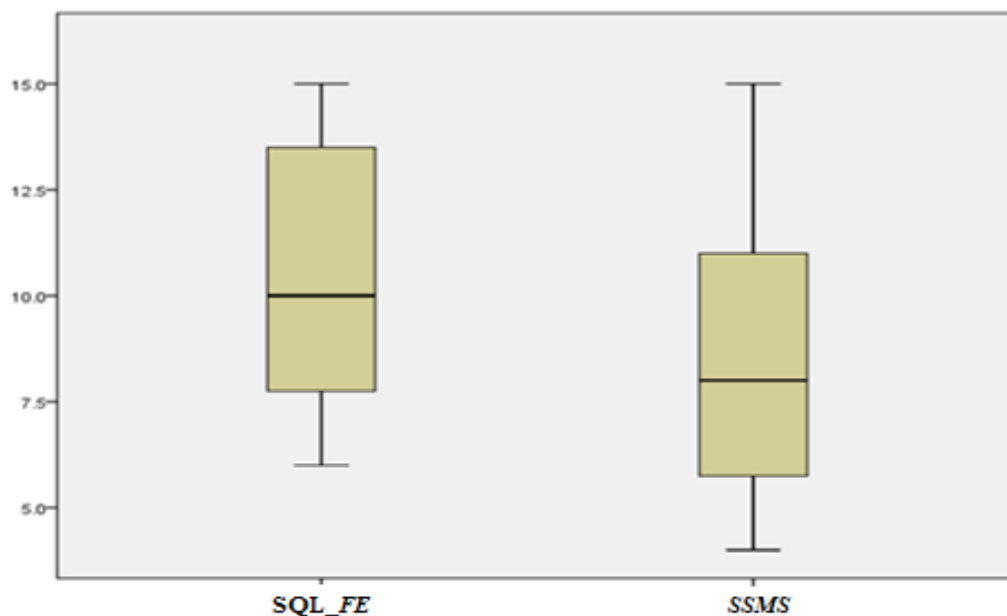


Figure 6-24: Boxplot of Performance Marks of both *SQL-FE* and *SSMS*

This is also supported by a histogram of the marks distribution, illustrated in Figure 6-25 and Figure 6-26. The figures report symmetric distributions of the participants' marks using the *SQL-FE* and *SSMS* tools of test administration.

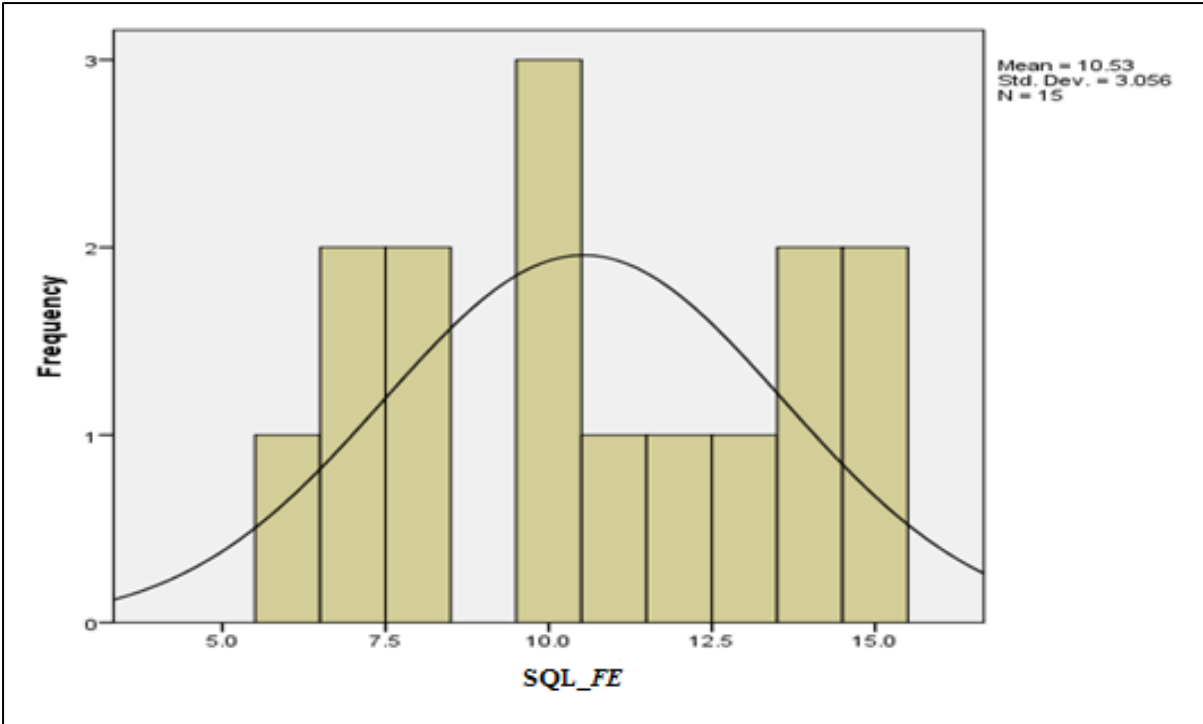


Figure 6-25: Histogram of the distribution of marks obtained using SQL-FE

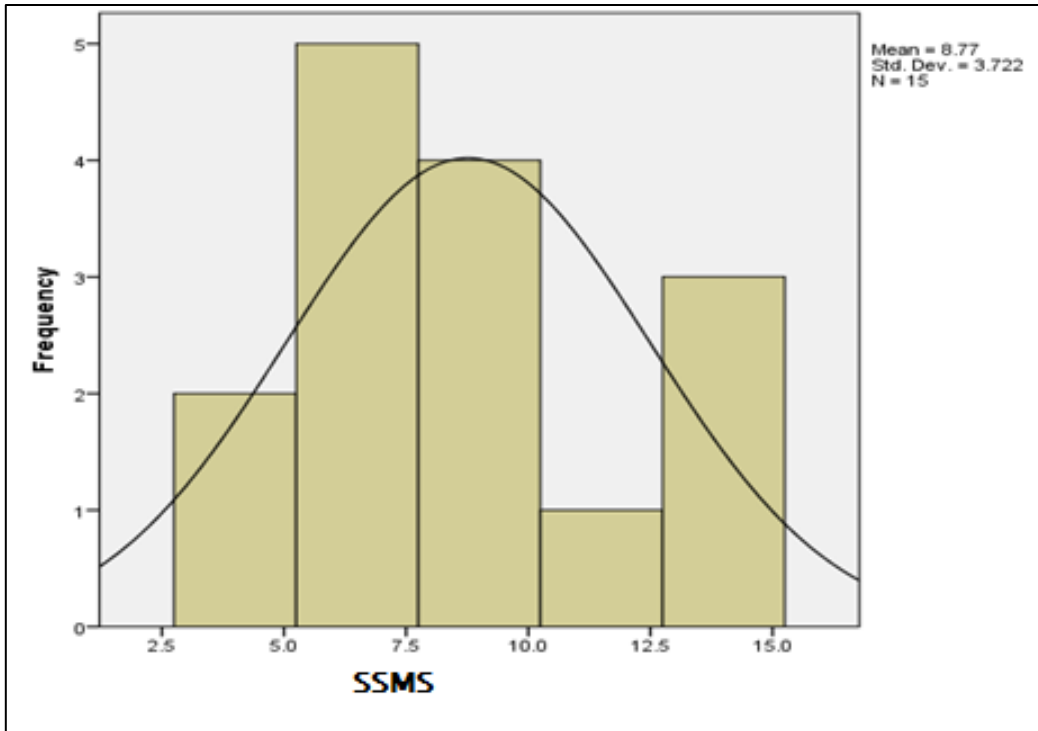


Figure 6-26: Histogram of the distribution of marks obtained using SSMS

This indicates that there is a significant difference in mean marks obtained between the two tests, or equivalently, there is a significant difference (improvement) in the participants' performance after formulating the SQL statements using *SQL-FE*. These results clearly provide strong evidence for a statistically significant difference in the participants' marks after using the *SQL-FE* tools.

6.5. Summary

In this chapter, the design decisions regarding the software tool *SQL-FE* were outlined. The requirements are based on research methodology, and the research approach presented in the Chapter 4. Within the chapter, the implementation of the conceptual design of the new Formulation Editor, as it relates to the solution steps capture part of the framework, was illustrated using a practical example of full *SQL-FE* user interface and their testing and result analysis.

This chapter investigated the use of a point-and-click method to solve basic SQL statements. The experimental study demonstrated that students were able to use the newly implemented *SQL-FE* tool with ease. Furthermore, the tool minimised the unnecessary elements that students often add while formulating SQL statements. This resulted in removing the ambiguity in SQL answers, which should support the examiners in understanding the students' level of SQL learning and enable them to provide accurate feedback. The *SQL-FE* editor answered the two questions of this experiment and confirmed that by using the newly implemented tool, less time is spent formulating SQL statements and students' performance improves, leading to fewer errors and higher grades.

At the same time, this chapter presented different evaluation studies (e.g. pilot study and experiment) carried out to examine the functionalities of the *SQL-FE* in line with design requirements within the semi-automatic CAA framework. The main significant result from those evaluation is that it provided a proof of concept for the point-and-click approach implemented on the *SQL-FE* which satisfied most of the participants (students). It showed that approach to be usable and comparable to conventional test formats. It also demonstrated the extended capability to be enhanced and maintain to except more SQL statements which contains complicated SQL clauses such as (joins and subqueries).

The newly implemented editor provided students with an easy method of solving SQL statements. Further implementations will take place utilising a semi-automated assessment of SQL statements to provide partial marking for the submitted statements from the SQL-*FE* tool. This would be considered as the second stage of the research, which means that the examiners' role will start once students submit their SQL answers by ensuring that the answers are ready for marking and commenting by examiners.

Chapter 7.

A New Semi-Automatic SQL Assessment Framework

7.1. Introduction

Automated assessment of SQL statements could be beneficial for many universities with large numbers of students. For this reason, different approaches have been utilised to attempt to minimise the need for human intervention in marking several programming languages and SQL statements (Batmaz, 2011; Insa and Silva, 2015; Adesina, 2016; Buyrukoglu, 2018). However, almost all existing approaches are based on output comparison. If a student's output matches the model output, the SQL statement is correct. Otherwise, it is reported as wrong, even if there is only one mistake in the statement. In this case, the student cannot achieve even a medium mark, since comparisons offer only two possible outputs: the whole SQL statement is marked either as correct or wrong. Furthermore, during the SQL assessment process, much of the examiners' time is occupied with marking students' SQL statements. They check students' SQL answers against model SQL answers. In such scenarios, computer support can enhance the quality of SQL marking. It can also shorten the assessment time and reduce the assessment cost. Thus, any level of computer support to this process is useful. The intention of this work is to not only provide computer assistance in the marking phase, but also in other phases of the current manual SQL assessment process. As identical tasks are performed less frequently (possibly only once) by examiners, the consistency of marks and feedback on SQL answers can be significantly enhanced.

The main objective of this chapter is to develop a new automated-assessment based framework that aims to reduce the workload of examiners, enhance students' SQL learning experience, and provide them with distinct and detailed feedback. This research presents an approach based on using semi-automatic assessment, which utilises the integration of the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems. In the proposed approach, CBR is used as the main reasoning process, while RBR is used to improve parts of this process.

Furthermore, this approach targets the reduction or removal of as many of the repetitive tasks in any phase of the marking process as possible by applying a normalisation operation and a grouping process. In addition, the proposed approach targets the provision of consistent and effective feedback to students.

This chapter is organised as follows. Section 7.2 discusses the semi-automatic assessment approach, which contains six different processes, and details their functionalities after applying them on several SQL statements. The main topics discussed in this part of the chapter are the normalisation operation, grouping process, marking and feedback. Section 7.2.2 discusses the normalisation operation, which tests different SQL statements answers and checks the decrement number of the statements after each level of normalisation operation, after which the statements are divided into groups, as explained in Section 7.2.3. Next, Section 7.3 explains the SQL marking process that utilises the semi-automatic assessment approach, where firstly, the generic marking rules of SQL statements are explained in Section 7.3.1.1, where a description of the rules' procedure and the flow of written rules by using the RBR system are provided. Secondly, Section 7.3.1.2 describes the partial marking of SQL clauses and operator's parts by applying CBR. Lastly, Section 7.3.1.3 lists the main rules for marking duplicated SQL statements answers by utilising the propagation of marks and feedback by applying RBR. In Section 7.3.2 and 7.3.3, an explanation of each of the rules using conditional sentences and examples is provided, after which the application of the propagation process of marks and feedback on several SQL statements is clarified. Section 7.4 concludes the chapter by providing a summary of its content.

7.2. Approach Description

The semi-automated assessment approach aims to enhance the SQL learning process and provide students with individual and detailed feedback. Besides, it targets the reduction of the lecturers' workload by reducing the amount of the SQL statements they have to mark. The framework develops and justifies the normalisation operations and a set of rules to support the semi-automatic marking of SQL statements. Figure 7-1 illustrates the proposed semi-automatic assessment approach cycle, which proposes a solution that is based on integrating the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems, which need to be adopted in the new marking technique in order to allow the reuse of previous SQL solutions for similar cases, thus contributing towards providing students with consistent marks and feedback.

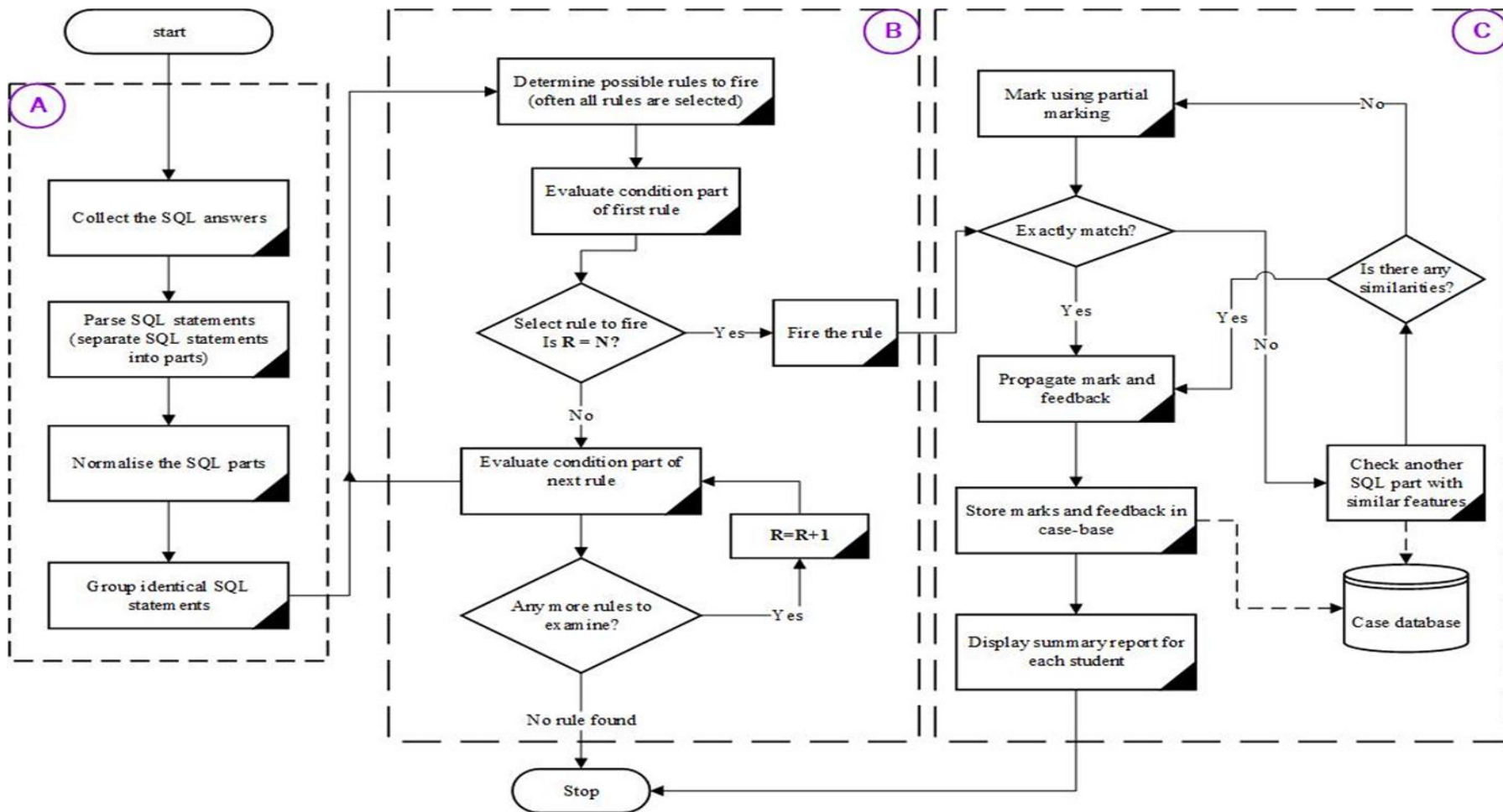


Figure 7-1: The proposed Semi-Automatic Approach

There are three main stages in the proposed semi-automatic SQL assessment approach as demonstrated in Figure 7-1.

- A. Pre-processing stage:** collects the SQL statements from students' answers, before the 'normalisation operation' takes place, which replaces and removes inconsistent data. Subsequently, the 'classification' process is used to classify identical SQL statements clauses into groups.
- B. Generic Marking Rules stage:** The marking rules of the semi-automatic assessment approach are a set of minimum requirements and standards for marking and grading reparative clauses of SQL statements.
- C. Marking process stage:** involves marking of the identical SQL parts and groups, as well as the provision of feedback related to the marked groups.

- *Each of these stages is explained in details as follows.*

7.2.1. Pre-processing

The first stage is the pre-processing stage, where the SQL statements are collected as described in Section 4.4. The statements are retrieved after students submit their SQL answers of existing SQL exam scripts using the SQL-*FE* editor. The SQL-*FE* editor was designed to allow students to formulate SQL statements using the point-and-click method and submit them to the database. The system was implemented with a database that collects all participants' SQL answers for different questions. The data collected is organised such as each clause appears separately to allow the semi-automatic assessment of SQL statements as described in Section 6.2.3.1.

7.2.2. Normalisation Operation

Once the data has been collected, the normalisation operation commences. The data normalisation stage is the phase in which the data is organised and normalised to increase the similarity between SQL statement parts. The primary goal of the SQL data normalisation phase is to organise a variety of clauses and keywords (such as SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN), table and field names, and aliases and brackets, to increase the similarity among the SQL statements. The data normalisation phase does not change the meaning of clauses and keywords.

For instance, it does not change the meaning of a SELECT statement; rather, it generates an equivalent SELECT statement that enables SQL parts to be grouped and converted to an equivalent statement. This increases the similarity of strings between SQL statements. In other words, string matching can be increased after making slight changes to SQL statements, without affecting the final output. However, the original SQL statement can be presented to the examiners in the marking process to compare between the answers and provide accurate feedback to students' statements. Matching does not depend on any SQL question, as it purely depends on SQL statements. The normalisation stage mostly covers the SELECT (fieldnames), FROM (JOIN, INNER JOIN, LEFT, RIGHT, FULL OUTER JOIN), WHERE (single and multiple conditions), GROUP BY, HAVING (single and multiple conditions) and ORDER BY clauses. It consists three different processes, which are the remove, replace and sort processes.

7.2.2.1. The Remove Normalisation Operation

The first step in the remove normalisation process is the elimination of unnecessary elements. This process includes:

- a. **Field name using aliases:** aliases can be used to temporarily assign readable names to columns, which will exist in time of a query output without effecting the original columns (Bisland, 1989). However, to increase the similarities between the SELECT clauses, all aliases should be removed. An example of the application of this is the following:

- **IF**, the "AS" keyword is used with an alias name (AS FIRST_NAME in the example);

```
SELECT EMP.FNAME AS FIRST_NAME
```

- **OR** quotation marks are used as well as the quoted alias name ("FIRST NAME" in the example);

```
SELECT EMP.FNAME "FIRST NAME"
```

- **THEN**, in both cases, the aliases should be removed as;

```
SELECT EMP.FNAME
```

- b. Removing extra spaces:** the SQL-FE editor was designed to generate spaces on either side of all SQL elements, including clauses, field names, tables, mathematical operators, keywords and functions. However, by using the text-area to insert data, participants might add more spaces, which should be removed in this process.

Therefore, all extra spaces between SQL clauses and elements that result in some difference between SQL statements should be removed, and only one space should be kept.

- For example:

```
SELECT EMP.FNAME   , EMP.JOB
FROM   EMP
```

- Should be changed to the following form after normalisation has been applied, where the extra white space is removed:

```
SELECT EMP.FNAME , EMP.JOB
FROM EMP
```

- c. Removing semi-colons:** SQL-FE and other SQL statement formulation tools support executing statements without adding the semi-colon at the end, which saves time when marking.

- For example, the statement:

```
SELECT EMP.FNAME , EMP.JOB
FROM EMP ;
```

- Changes to the following after the normalisation process has been applied:

```
SELECT EMP.FNAME , EMP.JOB
FROM EMP
```

- d. Removing ASC Keyword:** the ascending keyword “ASC” is used to explicitly request ascending order in the ORDER BY clause. However, it is not necessary, since the ascending order is the default option in the ORDER BY clause. For that reason, all instances of the ascending “ASC” keyword used in ORDER BY clauses should be removed to increase the consistency across SQL answers.

- For example, the statement:

```
ORDER BY EMP.SALARY DESC , EMP.LNAME ASC
```

- Has the “ASC” keyword removed after normalisation has been applied:

```
ORDER BY EMP.SALARY DESC , EMP.LNAME
```

7.2.2.2. The Replace Normalisation Operation

The second process in the normalisation phase is the replace process. This process includes:

a. Replacing double quotation with single quotation: if string values were inserted with double quotation marks, all should be replaced with single quotation marks to make the SQL answers consistent. Although both provide the same results, ensuring consistency results in increasing the similarity between SQL statements.

- For example, the statement:

```
WHERE EMP.JOB = "Manager"
```

- Should be changed to:

```
WHERE EMP.JOB = 'Manager'
```

b. Replacing aliases: if all aliases have been removed from the SELECT clauses, and data was sorted using the aliases' names, then the column name used in ORDER BY should be replaced with the original name.

- For example, the statement:

```
SELECT EMP.FNAME AS FIRST_NAME  
FROM EMP  
ORDER BY FIRST_NAME
```

- Should be changed to:

```
SELECT EMP.FNAME  
FROM EMP  
ORDER BY EMP.FNAME
```

7.2.2.3. The Sort Normalisation Operation

The third process in the normalisation phase is the sort process. This process includes:

a. Sorting field names in SELECT clauses

The basic format of the SELECT clause uses the SELECT keyword followed by a list of field names separated by commas. These field names can be normal field names or other style formats such as aggregate functions or mathematical expressions. If the field names in the SELECT clause are not in order, they should be sorted alphabetically. First, the sorting process should start with all simple fields, where the field names should be sorted alphabetically, and aliases should be removed. Secondly, if the SELECT clause contains any aggregate functions, they should be sorted alphabetically too. Finally, if the SELECT clause has any mathematical expressions, they must be sorted in order of operation. The following examples demonstrate the sorting process in SELECT clauses:

- i. **Simple field names:** If a SELECT clause contains multiple field names, they should be sorted in alphabetical order.

- For example, the statement:

```
SELECT EMP.SALARY , EMP.FNAME
```

- Should be changed to the following after sorting field names alphabetically:

```
SELECT EMP.FNAME , EMP.SALARY
```

- ii. **Aggregate functions (field names):** If a SELECT clause contains multiple field names and aggregate functions, the sorting order should start with the simple field names then the aggregate functions, and the sorting should be done alphabetically.

- For example, the statement:

```
SELECT SUM(EMP.SALARY) , EMP.GENDER , EMP.FNAME
```

- Should be changed to:

```
SELECT EMP.FNAME , EMP.GENDER , SUM(EMP.SALARY)
```

iii. **Mathematical expressions:** if a SELECT clause also contains mathematical expressions, the sorting should be done as for normal strings, where the simple field names should be sorted first, followed by field names containing the mathematical expressions.

- For example, the statement:

```
SELECT EMP.SALARY + 100 , EMP.SALARY * 0.1 , EMP.FNAME
```

- Should change to the following form after sorting it alphabetically:

```
SELECT EMP.FNAME , EMP.SALARY * 0.1 , EMP.SALARY + 100
```

Note:

- The sort normalisation operation does not considered cases where the SELECT clause contains string concatenation.

b. Sorting FROM clauses with (JOIN)

The sorting process in FROM clauses is only applicable for two tables that have been joined together using JOIN and INNER JOIN, LEFT, RIGHT and FULL OUTER JOIN. The following examples illustrate the cases where using the alphabetical order does not affect the query output.

i. **Natural join:** when joining two tables using a simple JOIN keyword, the table names should be sorted alphabetically.

- For example, the statement:

```
FROM EMP , DEPT
```

- Changes to the following form after sorting the table names alphabetically;

```
FROM DEPT , EMP
```

ii. **Inner join:** for INNER JOIN, the order of the table names does not matter. That is, the query will return the same results regardless of the order of table names. Therefore, when joining two tables using INNER JOIN, the table names should be sorted alphabetically.

- For example, the statement:

```
FROM EMP INNER JOIN DEPT
```


- Should be changed to:

```
FROM DEPT INNER JOIN EMP
```

- iii. **Full outer join:** in FULL OUTER JOIN, the query returns identical results regardless of the order of table names. Therefore, when joining two tables using FULL OUTER JOIN, the table names should be sorted alphabetically. For example, the statement:

```
FROM EMP FULL OUTER JOIN DEPT
```

- Changes to the following after sorting the table names alphabetically:

```
FROM DEPT FULL OUTER JOIN EMP
```

- iv. **Left and Right join:** For LEFT and RIGHT outer joins, the order of the tables is critical. Therefore, when joining two tables using left or right outer join, the table names should be sorted alphabetically, and the JOIN type should be reversed to make the newly sorted statement equivalent to the unsorted one.

- For example, the statement:

```
FROM EMP LEFT OUTER JOIN DEPT
```

- Is not equivalent to:

```
FROM DEPT LEFT OUTER JOIN EMP
```

- However, sorting the table names alphabetically and changing the join type from LEFT outer join to RIGHT outer join, will result in returning the same results as the unsorted statement:

```
FROM DEPT RIGHT OUTER JOIN EMP
```

c. *Sorting the WHERE clause*

The sorting process in WHERE clauses are divided into two categories; WHERE clauses with a single condition and WHERE clauses with multiple conditions:

- i. **Single Condition:** If the order of the WHERE clause with a single condition is written as:

```
WHERE 2000 > EMP.SALARY
```

- It can be changed (if necessary) to:

```
WHERE EMP.SALARY < 2000
```

ii. **Multiple Conditions:** The SQL SELECT statements allow multiple conditions in WHERE clauses to narrow the data retrieved from the database. This process considers only one type of combining, where either (AND or OR) is used in each WHERE clause. There are no restrictions in the number of conditions, since the same type of operators (AND or OR) are used in multiple conditions.

- For example, if a WHERE clause contains multiple conditions with AND operators:

```
WHERE EMP.LNAME >= "J"  
AND EMP.JOB = "PRESIDENT"  
AND EMP.LNAME <= "S"
```

Then, the alphabetical order should be as follows:

```
WHERE EMP.JOB = "PRESIDENT"  
AND EMP.LNAME >= "J"  
AND EMP.LNAME <= "S"
```

- Another example would be if a WHERE clause contains multiple conditions using OR operators:

```
WHERE EMP.SALARY BETWEEN 1000 AND 2000  
OR DEPT.DEPTNAME = "SALES"  
OR EMP.FNAME = "ALLEN"
```

Then, the alphabetical order should be as follows:

```
WHERE DEPT.DEPTNAME = "SALES"  
OR EMP.FNAME = "ALLEN"  
OR EMP.SALARY BETWEEN 1000 AND 2000
```

Notes:

- This process does not consider the WHERE expressions that contain a calculating or comparison expression.
- This process does not consider the combination of (AND & OR) in a WHERE condition.

d. **GROUP BY clause:** field names of GROUP BY clauses cannot be sorted, as this would change the output.

e. *Sorting HAVING clauses*

The same process used for conditions in WHERE clauses can be applied for conditions in HAVING clauses. The difference between the WHERE clause and HAVING clause, is that the HAVING clause works primarily on aggregate function columns, whereas the WHERE clause works on columns and other expressions without an aggregation operation.

- i. **Single Condition:** if a query containing a HAVING clause is written in non-alphanumeric order, it should be sorted alphanumerically.

- For example, the query:

```
HAVING 2000 <= SUM(EMP.SALARY)
```

- Should be reorder as:

```
HAVING SUM(EMP.SALARY) >= 2000
```

- ii. **Multiple Conditions:** SQL SELECT statements allow multiple conditions in the HAVING clause to narrow the data retrieved from the database.

This sorting process considers only one type of combining, where either (AND or OR) are be used in each HAVING clause. There are no restrictions in the number of conditions, since the same type of operators (AND or OR) are used in multiple conditions.

- For example, if a HAVING clause contains multiple conditions with AND operators:

```
HAVING SUM(EMP.SALARY) < 100000  
AND COUNT(EMP.EMPNO) >= 1  
AND EMP.DEPTNO BETWEEN 20 AND 40
```

Then, they should be sorted alphabetically as follows:

```
HAVING COUNT(EMP.EMPNO) >= 1  
AND EMP.DEPTNO BETWEEN 20 AND 40  
AND SUM(EMP.SALARY) <100000
```

- Another example would be if a HAVING clause contains multiple conditions with OR operators:

```
HAVING SUM(EMP.SALARY) < 100000  
OR COUNT(EMP.EMPNO) >= 1  
OR EMP.DEPTNO BETWEEN 20 AND 40
```

In this case, the alphabetical order should be as follows:

```
HAVING COUNT (EMP.EMPNO) >= 1  
OR EMP.DEPTNO BETWEEN 20 AND 40  
OR SUM (EMP.SALARY) < 100000
```

Notes:

- This process does not consider the HAVING expressions that contain a calculating or a comparison expression.
- This process does not consider the combination of (AND & OR) in a HAVING condition.

f. Field names in ORDER BY Clauses: field names of ORDER BY clauses cannot be sorted, as this would change the output.

7.2.2.4. Normalisation operation applied in real data of SQL statements

D. SQL Data Collection

Two different set of SQL statements were used as data collection sources to test the normalisation operation processes. One is the SQL statement answers retrieved from existing exam scripts (described in detail in Chapter 5). There were five different questions retrieved from the exam scripts with their answers, however, only three questions were selected along with their answers since this research focuses only on SELECT clauses, whereas the other two questions covered the CREATE table and VIEW clauses, which represent the Data Definition Language (DDL).

The second data collection source is the SQL statement answers that were retrieved from the SQL-FE experiment (described in detail in Chapter 6). The total numbers of questions were five, all of which focused on the basic SQL SELECT statements. The questions were designed to assess the basic SQL SELECT statements, which cover SELECT, FROM, WHERE, JOIN, GROUP BY, HAVING and ORDER BY.

E. SQL Data Normalisation

The normalisation operation increases the similarities between SQL statement clauses and allows the CBR system to find the similarities between the previous and current answers for certain queries. This can be explained in detailed steps by using one example from the data collected, since all other questions go through the same steps. This means that each question will go through different steps depending on the number of clauses required in the statement answer.

The example used is Q1 along with 30 students' SQL answers retrieved from the SQL-FE editor, and can be described as follows.

"Find the first names of all employees who work as clerks and earn a salary of more than 2500".

The question requires three main clauses, which are SELECT, FROM and WHERE. The WHERE clause is divided into two parts, which contain the WHERE clause plus the AND operator. This is done to increase the matching between the parts and enhance the consistency between the statements. As the question is basic and direct, the numbers of clauses are mostly similar, especially in the SELECT and FROM clauses. However, all clauses should be checked and normalised as described in the following steps as illustrated in the following diagram.

Step 1: Original SQL Statement

Initially, once the 30 SQL statement answers have been split into clause and operator parts, the normalisation operation will manually start to function. However, in this step, only the original data will be displayed without any normalisation operation. The reason for doing so is to carry out a simple comparison of the total number of the matched SQL statements before and after applying the normalisation operation, as displayed in Table 7-1. The table shows only the division of the clauses and operators of students' SQL answers. In addition, the count represents how many duplicates of the same SQL statements have occurred. In this stage, a total of 18 duplicate answers were found, which shows that many students wrote the same SQL answer.

Table 7-1: Original SQL statements

NO	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
6	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
7	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
8	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
9	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
10	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERK '	AND EMP.SALARY > 2500	1
11	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY < 2500	1
12	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
13	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
14	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERKS '	AND EMP.SALARY > 2500	1
15	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
16	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
17	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
18	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
19	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
20	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		1
21	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
22	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
23	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
24	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
25	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'		1
26	SELECT EMP.FNAME	FROM EMP.EMPNO	WHERE EMP.SALARY > 2500		1
27	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
28	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		0
29	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
30	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
Total Number of answers					18

Step 2: SELECT clause

Table 7-2 shows the steps of the normalisation process taking place for the SELECT clause. In this case, only the remove and sort normalisation process have been applied, which removing aliases and white spaces and sort field names alphabetically.

Table 7-2: Normalisation Operation applied on the SELECT clauses

NO	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
6	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
7	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
8	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
9	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
10	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERK '	AND EMP.SALARY > 2500	1
11	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY < 2500	1
12	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
13	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
14	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERKS '	AND EMP.SALARY > 2500	1
15	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
16	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
17	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
18	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
19	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
20	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		1
21	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
22	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
23	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
24	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
25	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'		1
26	SELECT EMP.FNAME	FROM EMP.EMPNO	WHERE EMP.SALARY > 2500		1
27	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
28	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		0
29	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
30	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
Total Number of answers					17

After applying the remove and sort normalisation process, the number of the SQL statements decreased from 30 students' answers to 17 answers. This is because identical SQL statement clauses have been gathered together, which count as one answer. As such, the similarities between SQL statements can be increased even after only applying the normalisation on the SELECT clause.

Step 3: FROM clause

Using the same processes from Step 2, the number of SQL statements in Table 7-3 remained the same as that of Table 7-3, totalling 17 SQL answers. Since Q1 requires one table to be retrieved from “FROM EMP”, most of the students got the same answer, therefore no changes were required. Only one student added a different table name, which results in a different group.

Table 7-3: Normalisation operation applied on the FROM clauses

NO	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
6	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
7	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
8	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
9	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
10	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERK '	AND EMP.SALARY > 2500	1
11	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY < 2500	1
12	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
13	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
14	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = ' CLERKS '	AND EMP.SALARY > 2500	1
15	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
16	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
17	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
18	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
19	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
20	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		1
21	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
22	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
23	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
24	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
25	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'		1
26	SELECT EMP.FNAME	FROM EMP.EMPNO	WHERE EMP.SALARY > 2500		1
27	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
28	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		0
29	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
30	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
Total Number of answers					17

Step 4: WHERE clause

Table 7-4 shows a decrease in the number of statements to 12 after applying the normalisation process on the WHERE clause. The normalisation process steps of the WHERE clause are as follows. First, the remove normalisation process involves removing white spaces and semi-colons. Secondly, the replace normalisation process is applied by replacing double quotation marks with single quotation marks. Finally, the sort normalisation process is applied on the WHERE clause.

Table 7-4: Normalisation operation applied on the WHERE clauses

NO	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
6	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
7	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
8	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
9	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
10	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
11	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY < 2500	1
12	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
13	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
14	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
15	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
16	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
17	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
18	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
19	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
20	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		1
21	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
22	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
23	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
24	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
25	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'		1
26	SELECT EMP.FNAME	FROM EMP.EMPNO	WHERE EMP.SALARY > 2500		1
27	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
28	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		0
29	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
30	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
Total Number of answers					12

Step 5: AND operator

In this stage, the last part of the SQL statements is normalised, and the final number of unique statements is determined. Table 7-5 shows the results of the normalisation process which has been applied on the last part of the SQL statement. The process starts by removing the extra spaces between the single quote and the value. Then, it proceeds by removing the single quote from the numerical data retrieved in the AND operator.

Table 7-5: Normalisation Operation applied on the AND operator

NO	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1
6	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
7	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
8	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
9	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
10	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
11	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY < 2500	1
12	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
13	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
14	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
15	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
16	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
17	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
18	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
19	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
20	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		1
21	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
22	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
23	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
24	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
25	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'		1
26	SELECT EMP.FNAME	FROM EMP.EMPNO	WHERE EMP.SALARY > 2500		1
27	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	0
28	SELECT EMP.FNAME	FROM EMP	WHERE EMP.SALARY > 2500		0
29	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
30	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	0
Total Number of answers					8

7.2.2.5. Analysis and Discussion

After applying the normalisation operation in each clause, the SQL statements can be compared and calculations can be made within a range of data using a formula based on the “IF (SUMPRODUCT)” function in a spreadsheet. This function counts how many times a specific SQL statement appears inside a range of cells, as shown in Table 7-6. By using the spreadsheet, the SQL answer statements are divided into clause parts.

Table 7-6: Divisions of SQL clause parts using a spreadsheet

No.	SELECT	FROM	WHERE	AND	Count
1	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	1
2	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERKS'	AND EMP.SALARY > 2500	1
3	SELECT EMP.FNAME	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > '2500'	0
4	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.SALARY > 2500		1
5	SELECT EMP.EMPNO	FROM EMP	WHERE EMP.JOB = 'CLERK'	AND EMP.SALARY > 2500	1

This table has been divided into 4 columns, which represent each clause part of the full SQL statement. This means that if the question requires 4 different clauses to be added to the answer, then 4 columns will be generated. This division assists the normalisation operation of the semi-automatic approach to be applied on all clause parts and be adaptable to different normalisation processes. The counting of the SQL statements is conducted using two numbers, which are Zero (0) and One (1). Number (1) represents the first instance of the SQL statement (for example; No. 1) and (0) represents an identical statement (for example; No. 3) from the above table. String matching between the SQL statements clause parts increased after applying the normalisation operation as discussed in Chapter 2. These parts were later manually grouped before the marking process (which will be described in detail in Section 7.2.6). The two different sets of SQL statements are analysed and discussed below.

A. SQL statements retrieved from existing exam scripts (2014)

Figure 7-2 depicts the number of identical SQL statements provided as answers for three SQL questions. The statements were randomly selected from the answers of the 30 students that participated in solving the 2014 exam script. It can be clearly seen that the number of unique statements declined after applying the normalisation operation.

Furthermore, only few statements remained unchanged after going through the normalisation stages of the different clauses. The variation of the SQL statements answers affected the process of the normalisation, but still made significant changes that helped to keep the similarities between the statements high. The figure shows a reduction from 17 SQL statements to 13 SQL statements for Q1, whereas the number remained mostly unchanged in Q2, with a reduction from 30 statements to 28 statements in most stages, and a final total of 26 statements after applying the normalisation operation on the HAVING clause.

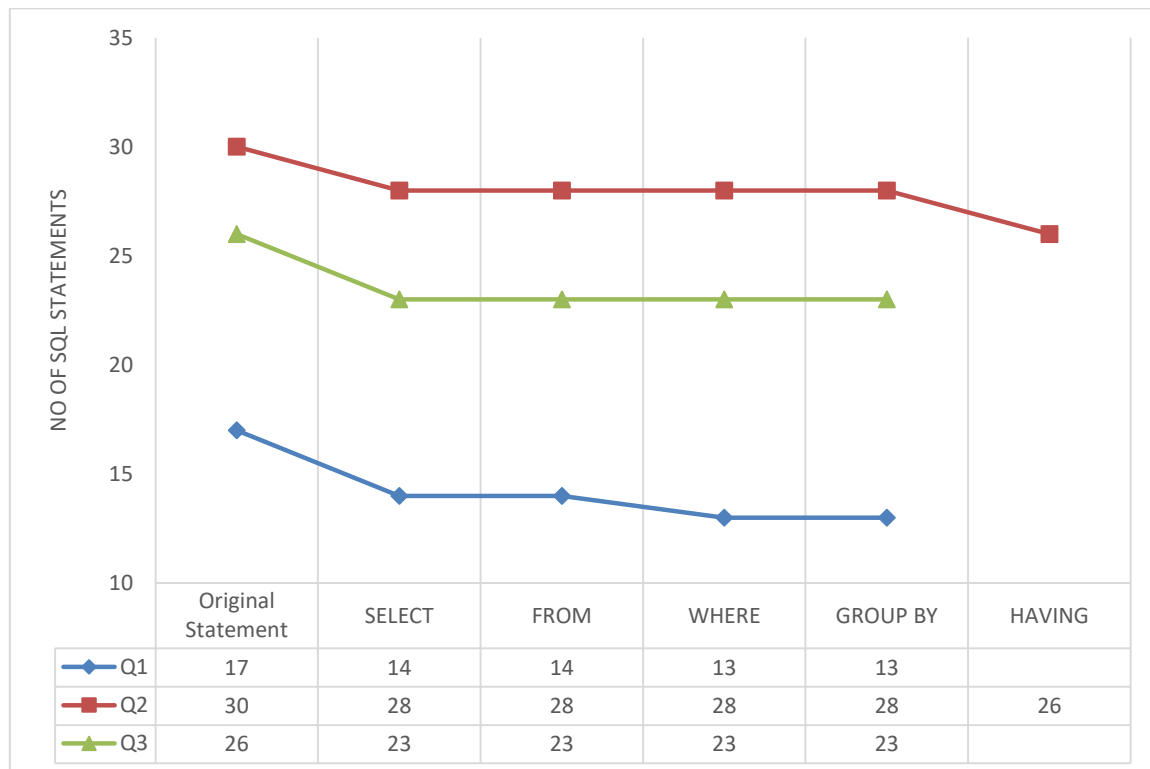


Figure 7-2: Normalisation process applied on the SQL statements of the exam scripts

B. SQL Statements Retrieved from SQL-FE (2016)

Figure 7-3 indicates the number of the identical SQL statements provided as answers for five SQL questions. A total of 30 students participated in solving the questions using SQL-FE in 2016. It can be clearly seen that the overall number of unique statements declined after applying the normalisation operation. In addition, only few statements remained unchanged after going through the normalisation stages of the different clauses. The number of SQL statements decreased by around 10 statements in Q1 and Q4, while the number of statements in Q2 declined from 28 statements to 23 statements after applying the full normalisation stages.

In both Q3 and Q5, the number of SQL statements decreased by 2-3 statements over the different stages of the normalisation processes. Overall, it can be seen from both figures (Figure 7-2 and Figure 7-3) that before the normalisation process, there were multiple identical SQL answers for each question. However, after applying the normalisation process to the different SQL clauses, the numbers of unique statements decreased and the similarities among the statements slightly increased. This decrease in the number of SQL statements, however small, still demonstrates that human marking time can be saved compared to manual marking methods.

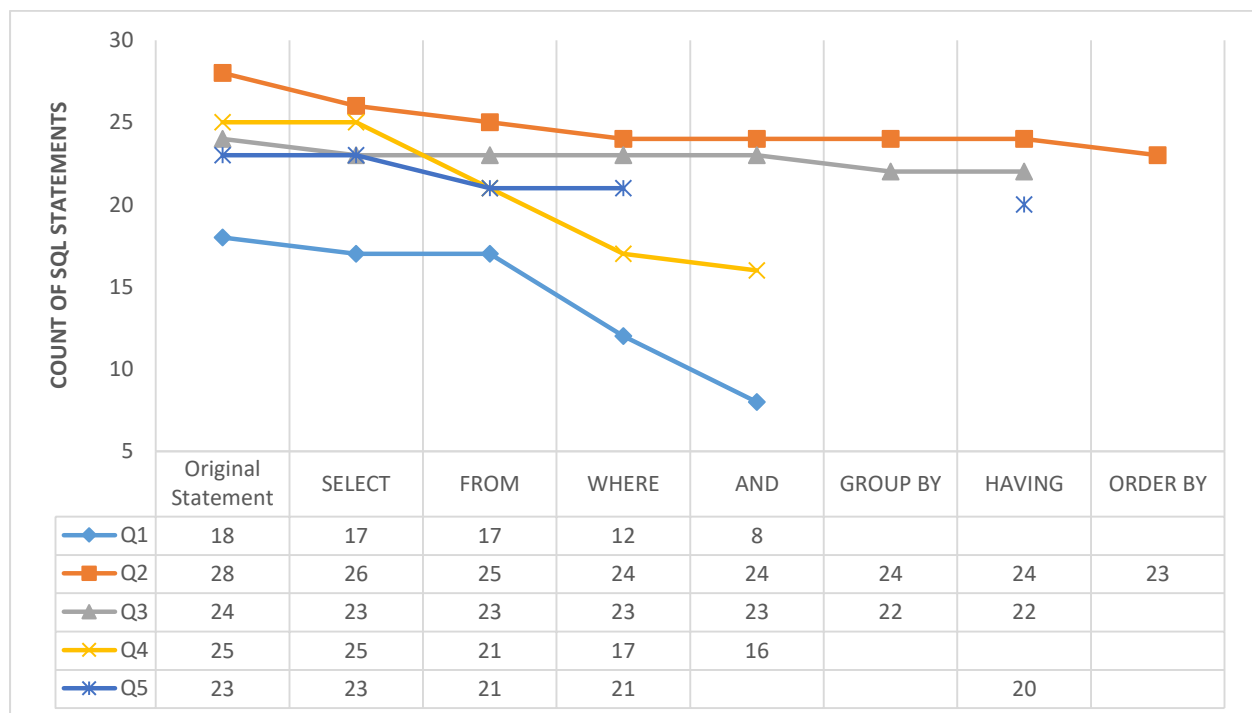


Figure 7-3: Normalisation process applied to SQL statements of SQL-FE

Figure 7-2 and Figure 7-3 show that Normalisation operation has increased the similarity between SQL statement parts. Where the primary goal of the SQL data normalisation phase is to organise a variety of clauses and keywords (such as SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN), table and field names, and aliases and brackets, to increase the similarity among the SQL statements. In the normalisation stages the SQL clauses are filtered in sequence where every time one clause is selected to be normalised, then count the final number of participants for each change happened. However, the data normalisation phase does not change the meaning of clauses and keywords, only remove the unnecessarily elements which makes the string unmatched.

7.2.3. Grouping Process

The next stage of the semi-automated assessment approach is to check for and group similar SQL statements. In this stage, all SQL statements are organised and ordered to identify their similarities among all of their clauses. This assists in defining the identical parts in the statements after the normalisation operation. Table 7-7 shows the grouping process, where the similarities among SQL statements' parts are defined and categorised into groups. This means that the identical SQL parts are clustered as one group. As can be seen from Table 7-7, there are eight groups of SQL statements created from the 30 students' answers.

Each group shows a different way of formulating a SQL statement, as collected from the students' full answers. Furthermore, the table shows that each SQL statement clause or part belongs in a different group of SQL statements, which makes it easier to categorise them while marking. It is also clear from tale that there are some groups that contain a larger number of identical SQL statements than others. For example, in Group 1, there are 16 identical answers, while Group 2 contains only two identical statements that exactly match. This allows the examiners to mark just one unique statement; with the rest of the identical groups being marked automatically, and as a result, the students can be provided with the same marks and feedback. Although there are some groups with fewer instances of repetition, they cannot be ignored while marking, since marks and feedback should be given even for specific answers.

Table 7-7: Number of SQL statement occurrences in each group

GNO	SQL Statements	Count
G1	SELECT EMP.FNAME FROM EMP WHERE EMP.JOB = 'CLERK' AND EMP.SALARY > 2500	16
G2	SELECT EMP.FNAME FROM EMP WHERE EMP.JOB = 'CLERKS' AND EMP.SALARY > 2500	7
G3	SELECT EMP.FNAME FROM EMP WHERE EMP.SALARY > 2500	2
G4	SELECT EMP.EMPNO FROM EMP WHERE EMP.SALARY > 2500	1
G5	SELECT EMP.EMPNO FROM EMP WHERE EMP.JOB = 'CLERK' AND EMP.SALARY > 2500	1
G6	SELECT EMP.FNAME FROM EMP WHERE EMP.JOB = 'CLERK' AND EMP.SALARY < 2500	1
G7	SELECT EMP.FNAME FROM EMP WHERE EMP.JOB = 'CLERK'	1
G8	SELECT EMP.FNAME FROM EMP.EMPNO WHERE EMP.SALARY > 2500	1
	Total Number of SQL Answers	30

7.3. Marking Process of SQL Statements

The main novel contribution of this research is the development of a novel framework that provides a platform to support the assessment process of SQL statements, which supports the integration of both the Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems that use application of the Artificial Intelligence (AI) methodology. Such a framework advantages enables human and computer association during the assessment process.

Reduce the overall SQL statement clauses marked by examiners. This means to reduce the human intervention on marking and reuse the comments given for similar SQL parts and the most important merit is enhances the accuracy of marking and provides students with immediate feedback. To achieve this, once the normalisation and grouping processes are done, a set of rules can be executed and checked (the SQL generic rules are listed in Section 7.3.2). If a rule's condition is met, then the rule will be enforced and applied, whereas if there is any rule's condition does not receive any action, then other rules will be enforced and the process starts checking again for more rules, as illustrated in stage 'B' of Figure 7-1. This section explains in detail the following three main topics:

1. The semi-automatic generic marking rules of SQL statements, which set the minimum requirements and standards for marking and grading clauses of SQL statements. They also identify the common repetitive tasks in the assessment process. This is discussed in detail in Section 7.3.1.1.
2. The partial marking process of SQL parts, whereby finding the similarities and matching between SQL statement parts takes place. Section 7.3.1.2 provides a discussion of this topic, as well as a clarification of how SQL statement parts should be marked using CBR.
3. The process of propagating marks and feedback, which involves propagating the same marks and feedback from the ideal solution that has been previously marked and using them again for other students' solution using the Case-based Reasoning (CBR) cycle (introduced in Chapter 3). This means that the lecturers' feedback can be copied to assess the rest of the repetitive SQL statements parts based on the CBR cycle. This is discussed in detail in Section 7.3.1.3.

7.3.1.1. SQL Generic Marking Rules

The Generic Marking Rules of basic SQL SELECT statements (GMR-SQLS) are proposed to assess statements written in Structured Query Language (SQL). They aim to reduce the number of SQL SELECT statements marked by the examiners by utilising the previous cases with matched problems and adopt the same solution. GMR-SQLS have been identified using the basic SQL SELECT statements. GMR-SQLS are used to explain the generic marking rules of SQL statements. GMR-SQLS indicates several SQL SELECT statements with different SQL question components. However, this analysis uses only the first five statements that contain SELECT, FROM, WHERE and ORDER BY clauses, as a first study. In addition, if the basic SQL testing works successfully, then GROUP BY and HAVING clauses will be tested rapidly as its complexity does. In this research, GMR-SQLS were only tested on a number of student SELECT queries, with the results obtained being analysed and compared against manual marking. The semi-automatic SQL marking process can be considered as a collaborative process between human and computer marking. The examiners must be aware of this while marking to enhance the marking consistency, grading and feedback. The principal purposes of semi-automatic SQL marking are to reduce examiners assessment workload and to provide students with appropriate feedback on their performance as part of a formative assessment process.

This research focuses on marking basic SQL clauses by applying different SQL marking rules. The basic SQL clauses contain SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clauses. The SQL marking rules support the semi-automatic marking approach of SQL statements, whereby this approach involves commenting on the repetitive clauses of students' SQL statements by using certain marking rules. These rules are applied to the SQL statements that have been normalised and classified into groups.

7.3.1.2. SQL Partial Marking

As described in Chapter 5, each student has a different way of writing SQL statements, making it more difficult to group identical answers. In this context, splitting the statement's clauses into parts facilitates the partial marking process. Partial marking is a marking process whereby students' SQL statements are marked once they have been divided into parts. The SQL parts specify the main SQL clauses, such as SELECT, FROM, and WHERE clauses.

The partial marking approach eliminates the repetitive task of marking since similarities among the SQL statements are identified, and the same previous marking may be applied for new statements. This allows lecturers to assign marks and feedback for each part of the statement. At the same time, students get partial marks when they write SQL answers that are close to being the correct answer, since the string matching method is used, which does not match between a student's answer and a model answer.

Instead, it groups matching parts of the students' SQL commands and then asks the examiners to approve the correctness of the SQL part from each of the different groups. In other words, once the examiner marks one student's SQL answers, the same mark might be applied to other students' answers using the same criteria. This would help examiners to define similar mistakes in students' answers and provide consistent marks and feedback to all students who make the same mistake. For example, using the groups of SQL statements of Table 7-7, the SQL statements would be divided into parts as shown in Table 7-8. As we can see from Table 7-8, the division of the SQL statements depends on the students' answers, since each statement has different number of parts. In G1, there are four parts, whereas in G3 there are three parts. There are several reasons for dividing full SQL statements into parts and not marking them as full SQL statements. Using partial marking can help markers to identify the same mistakes in different student answers. This means that once the marker marks SQL statements clauses, they can find the correct parts as well as the incorrect parts of students' answers, and, depending on the similar clauses parts in the different students' SQL statements, and give them the same marks. However, the examiner must separately mark the dissimilar parts and provide detailed feedback for the students SQL answers. Insa and Silva (2015) found that partial marking could not automatically conduct assessment by only testing the code's final output. Rather, it does so by checking the requirements needed by the lecturer and whether they are fulfilled or not. Once identical SQL statements parts have been found, the propagation of the marks and feedback can take place using the CBR method.

Table 7-8: SQL statements parts

G1	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERK'	Part 3
	AND EMP.SALARY > 2500	Part 4
G3	SELECT EMP.EMPNO	Part 6
	FROM EMP	Part 2
	WHERE EMP.SALARY > 2500	Part 7

7.3.1.3. Propagation of Marks and Feedback

The marks and feedback propagation process serves to broadcast the same marks and feedback from the ideal solution that has been previously marked and use them again for another student's solution, as illustrated in Figure 7-4. The marking and grading of repetitive clauses of SQL statements can identify the common repetitive tasks in the assessment process, where similarities between matching SQL parts are found. The process then proceeds to propagate the same marks and feedback from the ideal solution that has been previously marked and use them again for other students' solutions using the Case-based Reasoning (CBR) system. This means that the lecturer's feedback can be copied to assess the rest of the repetitive SQL parts based on the CBR cycle.

G1	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERK'	Part 3
	AND EMP.SALARY > 2500	Part 4
G2	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERKS'	Part 5
	AND EMP.SALARY > 2500	Part 4
G3	SELECT EMP.EMPNO	Part 6
	FROM EMP	Part 2
	WHERE EMP.SALARY > 2500	Part 7
G4	SELECT EMP.EMPNO	Part 6
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERK'	Part 3
	AND EMP.SALARY > 2500	Part 4
G5	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERK'	Part 3
	AND EMP.SALARY < 2500	Part 8
G6	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.SALARY > 2500	Part 7
G7	SELECT EMP.FNAME	Part 1
	FROM EMP	Part 2
	WHERE EMP.JOB = 'CLERK'	Part 3
G8	SELECT EMP.FNAME	Part 1
	FROM EMP.EMPNO	Part 9
	WHERE EMP.SALARY > 2500	Part 7

SQL Part Color Key		
Part	SQL Part	Color
Part 1	SELECT EMP.FNAME	Yellow
Part 2	FROM EMP	Light Green
Part 3	WHERE EMP.JOB = 'CLERK'	Light Blue
Part 4	AND EMP.SALARY > 2500	Orange
Part 5	WHERE EMP.JOB = 'CLERKS'	Light Purple
Part 6	SELECT EMP.EMPNO	Red
Part 7	WHERE EMP.SALARY > 2500	Pink
Part 8	AND EMP.SALARY < 2500	Cyan
Part 9	FROM EMP.EMPNO	Grey

Figure 7-4: Propagation of SQL statement parts

7.3.2. Generic Marking Rules of Semi-Automatic SQL Assessment

The marking rules of the semi-automatic assessment approach are a set of minimum requirements and standards for marking and grading reparative clauses of SQL statements. The aim of setting SQL marking rules is to assess the duplicated SQL statements clauses and provide equivalent grading for all students. The key advantage of these marking rules is their flexibility. This means that it is possible to add new rules or modify existing ones without any side effects. In addition, they are expressed in an easy-to-understand language that is logical and not complicated. As such, these rules should not cause any errors for the original SQL statements or lead to any syntax errors.

7.3.2.1. SQL Marking Rules Classifications

In this phase, generic rules are applied for grouping SQL elements. The rules are created by analysing data for frequent **If/Then** patterns to identify the most important relationships between the SQL statements. Clauses are therefore grouped together for the semi-automatic marking process. The literature shows that one of the most important advantages of grouping SQL clauses is marking them without the need for using any SQL model answers.

The classification of marking rules is used to decide which clauses will be marked together and which clauses will be marked separately. The terms “together” and “separately” are used in this context to classify (a) the clauses that should be marked as one group called (together), and (b) a separate clause which can be marked separately without joining it with other clauses is called (separately). These two terms are used in the action part of the **If/Then** SQL marking rules. Table 7-9 illustrates an example of a SQL statement answer which contains three clauses (SELECT, FROM and WHERE). The table shows that the SELECT and FROM clauses (in some cases) can be marked together as one group after checking the field names and the table used. Simultaneously, the WHERE clause can be marked separately from the full statement depending on the marking process of the examiner.

Table 7-9: Marking rules classification (Sample)

No	SQL Statement Clauses	Marking Type
1	SELECT EMP.FNAME	Marking SELECT and FROM clauses in conjunction (Together)
2	FROM EMP	
3	WHERE EMP.JOB = 'CLERK'	Marking WHERE clause (Separately)

7.3.2.2. Marking Rules Procedure

The application of marking rules follows a particular procedure once the statements have been checked by the examiner. The procedure has a series of actions that do not need to have specific order. However, these actions should be interpreted and applicable without affecting the execution of the SQL statements. The following are some questions and their answers to help explain the procedure of the SQL marking rules.

- **How are the marking rules processed?**

This research used the RBR cycle to examine and analyse certain forms of the all-marking rules, which can be activated and executed at the same time. This means that if a rule's condition is met, then the rule will be enforced and applied. On the other hand, if there is any rule condition that is not met then other rules will take place and the process of checking for more rules will commence. The reason of choosing this type of process is that by using semi-automatic SQL marking approach, multiple SQL statement clauses can be marked simultaneously. The marking rules cannot account for all possible SQL answers, nor can they predict how SQL statements will be formulated by the students. However, as long as the marking rules are formulated to accept new entries, any SQL statements that have not been marked due to not being any possible rules that can be applied, can still be marked by making new marking rules and adding them to the list.

- **How are marking rules formulated?**

The marking rules were formulated using a declarative language that is clear and easy to understand to anyone who understands English. They do not follow any specific order and it is possible to add new or modify existing rules without causing any side effects.

Moreover, to make the marking rules more reasonable and manageable, the SQL statement clauses were formulated along with the rules to support the functionality of each rule. The following is sample of one of the marking rules, which was written by using the **If/Then** statement.

IF	SELECT clause is having only one fieldname
AND	Fieldname match with table used in FROM clause
THEN	Mark the SELECT and FROM clauses together

The rule is written in a clear and concise English language. Each of the marking rules can have more or fewer conditions and actions depending on the different SQL statements cases. The rules were written and formulated following the SQL statements' style in order to enhance clarity

7.3.3. SQL Marking Process and Marking Rules

Identical SQL statement clauses can be categorised into groups. Each group might contain either single or multiple SQL statements clauses. However, the more clauses the SQL statement has, the less likely it is to be identical. Therefore, partial marking is important at this stage to increase the similarity between identical clauses. This approach limits the markers' involvement in the assessment process to only a number of SQL statements groups rather than the total number of students' statements. As such, it reduces the number of the SQL statements assessed by the marker. The string matching technique does not match between students' answers and the model answers. Rather, it groups the matching parts of the students' SQL statements and then asks the marker to approve the correctness of SQL parts from each of the different groups. In other words, once the marker marks one student's SQL answers, the same marking can be applied to other similar students' answers. The marking rules are formulated to dictate how students' SQL answers can be split into several parts that can be marked individually by the marker. Those parts can then be propagated to other mentioned parts in other statements. As such, the marking rules serve to remove the repetitive parts, thus reducing the number of statements that the marker have to mark, which contributes towards providing students with consistent feedback.

This research focuses on the basic SQL SELECT statement clauses, because a typical SQL statement can be made up of two or more of (**SELECT, FROM**) WHERE, GROUP BY, HAVING and ORDER BY clauses, where the SELECT and FROM clauses are the only two mandatory clauses in SQL statements. That is, an SQL statement can be minimally composed of SELECT and FROM clauses (Bobak, 1996). Yet, the statement can be extended to more clauses depending on the requirements of each query. To follow, the SQL marking rules have been categorised into three main sections; (1) SELECT, FROM, (WHERE or/and ON) and ORDER BY (2) SELECT, FROM, WHERE and GROUP BY (Aggregate Functions), (3) SELECT, FROM, GROUP BY and HAVING.

This means that each section considers the SELECT and FROM as the main clauses, while the rest of clauses are based on the SQL questions given in the first SQL-*FE* experiment. However, these sections are not restricted to limited number of SQL clauses. Each of these sections lists one or more rules and explains them in detail utilising SQL statements examples. These examples are coloured to illustrate the various marking status, where the white colour indicates an unmarked clause, a green colour indicates a fully corrected clause, a yellow colour indicates a partially correct clause, while the red colour symbolises a fully incorrect clause.

7.3.3.1. SELECT, FROM, (WHERE or/and ON) and ORDER BY Clauses

The following rules cover four SQL statement clauses; namely SELECT, FROM, WHERE and ORDER BY clauses. These rules differ from one other and depending on the context of a given SQL statements. For instance, some statements might only have a WHERE clause, others might only have an ON clause, while some may have both. In addition, the ORDER BY clause can be applied in all statements and always appears at the end of the statement. For this reason, the rules were applied to the SQL statements answers that had been collected by the SQL-*FE* editor. This section explains the different rules by presenting samples of SQL statements from Questions 1 and 2 as an explanation of the marking process.

Rule I. Fieldname/s match table with ≥ 1 condition/s

This rule is applicable for SQL statements that contain one or more fieldname, that are retrieved from one table, and in which the WHERE clause contains one or more conditions. In such a case, the rule is formulated as follows:

If	SELECT has one or more Fieldnames
AND	Fieldname/s match the table used
AND	WHERE have condition
THEN	mark the SELECT and FROM clauses together as a group
AND	mark the WHERE clause as a separate part

In this case, the marker checks the table used to retrieve the fieldname from with the fieldname in SELECT clause. Once the marker makes sure they match, the SELECT and FROM clauses are marked as one group (together).

Subsequently, the WHERE clause is split into two parts; one is the WHERE clause and second is the AND operator, where each part is marked separately, and the marks are propagated (i.e. applied) to other identical clauses. Figure 7-5 illustrates the description of the rules by using G1 and G2 of SQL statements answers.

Group No.	SQL Statement
G1	SELECT EMP.FNAME
	FROM EMP
	WHERE EMP.JOB = 'CLERK'
	AND EMP.SALARY > 2500
G2	SELECT EMP.FNAME
	FROM EMP
	WHERE EMP.JOB = 'CLERKS'
	AND EMP.SALARY > 2500

Figure 7-5: An illustration of the marking process after applying the rules

As one can see, Figure 7-5 illustrates two groups of SQL statements, where in both groups; the students selected the EMP.FNAME by using the EMP table. The marker in this case will match the table name “EMP” of the SELECT clause with the “EMP” of the FROM clause.

- A.** Once the table and fieldnames show that they match, the marker will mark the SELECT and FROM clauses together as one group.
- B.** The marking is then propagated (i.e. applied) to other groups that have identical SELECT and FROM clauses.

Subsequently, the marker can mark the WHERE conditions clause by clause. This means that partial marking can be applied in this stage, where a single clause can be matched with another single clause from another statement that is identical.

- C.** WHERE clause in G1 will be marked by the examiner as a separate clause.
- D.** The marks cannot be propagated to G2 (as illustrated with the **X** sign) since G1 and G2 are not identical in the value part of the WHERE clause. However, if there are any other groups which have identical WHERE clauses, the marks may be propagated.

This means that mark propagation does not only work with correct statements, as it can be applied on statements that contain identical parts, as is the case in G9 shown in Figure 7-6. In this case, the parts highlighted with yellow colour indicate that they are not identical with other groups from previous marked groups. In this case, the marker can predict the correct answer after marking the first groups, and makes sure that the cases that do not match should be marked separately and be provided with detailed feedback.

Group No.	SQL Statement
G2	WHERE EMP.JOB = 'CLERKS'
	AND EMP.SALARY > 2500
G9	WHERE EMP.JOB = 'CLERKS'
	AND EMP.SALARY > 2500

Figure 7-6: The mark propagation process with other groups

- E. The AND part in G1 will be marked by the examiner as a separate clause.
- F. Propagate the marking with other groups that have identical AND parts.

Rule II. Fieldname/s un-match table with >= 1 condition/s

This rule is applicable for SQL statements, which contain one, or more columns are retrieved from a table, but the fieldname does not match the table used. Furthermore, the WHERE clause contains one or more conditions. The rule is formulated as follows:

if	SELECT has a fieldname
AND	Fieldname does not match the table used
AND	WHERE contains a condition
THEN	mark the SELECT clause as a separate part
AND	mark the FROM clause as a separate part
AND	mark the WHERE clause as a separate part

Once the examiner makes sure that the fieldname and table used do not match, he/she will start marking the SELECT and FROM clauses separately, clause by clause. Subsequently, the WHERE and (AND or OR) operators will be marked separately and the marks will be propagated to other identical clauses. Figure 7-7 illustrates the description of this rules using G11 and G12 of the SQL statements answers.

Group No.	SQL Statement
G11	SELECT EMP.FNAME
	FROM EMP
	WHERE EMP.SALARY > 2500
G12	SELECT EMP.FNAME
	FROM EMP.EMPNO
	WHERE EMP.SALARY > 2500

Figure 7-7: E.g. the FROM clause in G11 is not identical to that in G12

- A. By using Rule I, the SELECT and FROM clauses of G11 will have already been marked using the same propagated marking used in G1.
- B. However, in G12, the marking of the SELECT and FROM clauses cannot be propagated with other groups, since the fieldname EMP.FNAME uses EMP.EMPNO as a table name. The action which should be taken at this point is to mark the FROM clause separately and provide feedback to the student.

Group No.	SQL Statement
G11	SELECT EMP.FNAME
	FROM EMP
	WHERE EMP.SALARY > 2500
G12	SELECT EMP.FNAME
	FROM EMP.EMPNO
	WHERE EMP.SALARY > 2500

Figure 7-8: The FROM clause should be marked as a separate part

- C. As illustrated in Figure 7-8, the marking process for G12 will be performed on a clause-by-clause basis. This means that the SELECT, FROM and WHERE clauses are marked separately.
- D. The marking is then propagated to other clauses that consist of identical clauses.

Rule III. Inner Join from multiple tables using JOIN...ON Clause

This rule is applicable for SQL statements that contain more than one column and are joining two or more tables with fieldnames that match the multiple tables that they are retrieved from by using JOIN...ON clause. The statement also contains ORDER BY clause, which is sorted either in ascending or in descending order. In such cases, the applicable rule is formulated as follows:

If	SELECT has two or more fieldnames
AND	fieldnames match the tables used
AND	ON contains an INNER JOIN condition
AND	data is sorted in ASC or DESC order
THEN	mark the SELECT, FROM and ON clauses together as a group
AND	mark the ORDER BY clause as a separate part

Since they match, the marker will mark the SELECT, FROM and ON clauses together as a group. The ORDER BY clause will be marked separately and the marking will be propagated to other identical clauses.

Group No	SQL Statement
G4	SELECT DEPT.DEPTNAME , EMP.LNAME
	FROM DEPT INNER JOIN EMP
	ON DEPT.DEPTNO = EMP.DEPTNO
	WHERE EMP.GENDER = 'FEMALE'
	ORDER BY DEPT.LOC




Figure 7-9: An SQL answer containing ON as a JOIN statement

As can be seen from Figure 7-9 and 7-10, both statements give the same output, as one is an alternative solution for the same query. Therefore, the marking will be carried out as follows:

- A.** If the answer contains SELECT, FROM and ON clauses, they will be marked together as a group, as shown in G4.
- B.** The WHERE clause will be marked as a separate clause as it follows Rule I.
- C.** Finally, the ORDER BY clause will be marked separately, and the marking will be propagated to other identical ORDER BY clauses.

If a WHERE clause was used instead of ON in INNER JOIN, then in this case, Rule III will have a different concept to that of Rule I in the context of the WHERE clause. This is because the WHERE clause in Rule III represents the join syntax which checks the primary key and foreign key between two tables, whereas in Rule I, it is used to find the condition of the statement. For example, if an answer contains SELECT, FROM and WHERE clauses in a JOIN statement (as shown in G7 in Figure 7-10), these clauses should be marked in exactly the same manner as that dictated by Rule III, since they have same output.

Group No	SQL Statement
G7	SELECT DEPT.DEPTNAME , EMP.LNAME
	FROM DEPT, EMP
	WHERE DEPT.DEPTNO = EMP.DEPTNO
	AND EMP.GENDER = 'FEMALE'
	ORDER BY DEPT.LOC

Figure 7-10: An SQL answer using WHERE as JOIN

As can be seen from Figure 7-9 and Figure 7-10, both statements give the same output, as one is an alternative solution for the same query. Therefore, the marking for the SQL statement in Figure 7-10 should be performed as follows:

- A.** If the answer contains SELECT, FROM and WHERE clauses, they should be marked together as one group.
- B.** The AND keyword should be marked separately since the WHERE clause is used in the JOIN syntax.
- C.** Finally, the ORDER BY clause should be marked separately and the marking propagated to other identical ORDER BY clauses.

If a WHERE clause was used instead of ON in FULL, RIGHT and LEFT OUTER JOIN, then in this case, using INNER JOIN, WHERE and ON can be acceptable, as both produce the same output. However, for LEFT, RIGHT and FULL OUTER JOIN, the WHERE clause cannot be used in the SQL statement's JOIN syntax. In such a case, the statement should be represented with ON and the rule should restrict the marking group to SELECT, FROM and ON for the JOIN syntax.

7.3.3.2. SELECT, FROM, WHERE and GROUP BY (Aggregate Functions)

The following rules cover four SQL statement clauses, namely SELECT, FROM, WHERE and GROUP BY (Aggregate Functions) clauses. The rules were formulated regarding SQL statements answers that were collected using *SQL-FE* editor. This section explains the different rules using sample SQL statement answers of Question 3 and 5 as an explanation of the marking process.

Rule IV. SELECT includes Aggregate Functions and GROUP BY Clause

This rule is applicable for SQL statements that contain fieldnames and aggregated functions that match the tables that they are retrieved from and the result-set is grouped by one or more fieldnames. The rule is formulated as follows:

If SELECT fieldnames with aggregate functions
AND fieldnames match the tables used
AND the result-set is grouped by one fieldname
THEN mark the SELECT and FROM clauses together as a group
AND mark the GROUP BY clause as a separate part.

When a query asks to add a GROUP BY clause in the statement, this means putting all those with the same value for certain field in one group. The example in Figure 7-11 shows that GROUP BY EMP.JOB indicates putting all those with the same value for EMP.JOB in the one group. As one can see from this SQL statement, the student selected EMP.JOB as a fieldname and AVG(EMP.SALARY) as an aggregate function.

Group No	SQL Statement
G9	SELECT EMP.JOB , AVG(EMP.SALARY)
	FROM EMP
	GROUP BY EMP.JOB

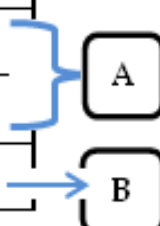


Figure 7-11: An SQL answer using a GROUP BY clause

Since they match, the marker will mark the SELECT and FROM clauses together as group. Once the matching is ensured, the availability of the fieldname of the GROUP BY clause is checked and the clauses are then marked them as follows:

- A. Mark the SELECT and FROM clauses together as one group.
- B. Mark the GROUP BY clause separately and propagated the marking to other identical GROUP BY clauses.

If more than one fieldname exists in a GROUP BY clause, the marking should start with the GROUP BY clause with the first fieldname, then the GROUP BY clause with second fieldname and so on, until all fieldnames are marked and the marking is propagated to other groups. Furthermore, when a GROUP BY clause lists multiple fieldnames, the fieldnames will be executed one by one, and then all aggregate functions (COUNT, SUM, AVG, MIN and MAX) are calculated. For example, if a statement was grouped by two fieldnames such as GENDER and JOB as shown in Figure 7-12, the marking process will be then performed such as each fieldname is marked separately.

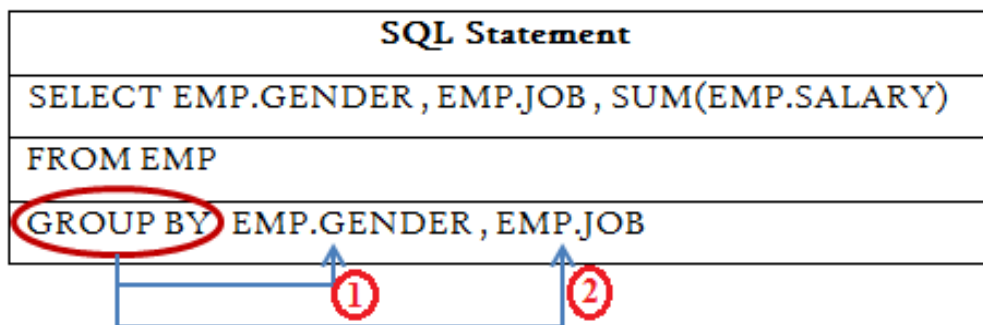


Figure 7-12: The marking process of a GROUP BY clause with multiple fieldnames

In other words, the marking process of this statement is done as follows:

1. The GROUP BY EMP.GENDER will be marked first.
2. Then GROUP BY EMP.JOB is marked after.

Rule V. GROUP BY Clause with WHERE Condition

This rule shares a similar marking process as Rule IV, with the only difference being the addition of a WHERE clause to the statement. This means that the marking process for SELECT, FROM and GROUP BY will follow Rule IV. Subsequently, the WHERE clause can be marked separately as an individual clause and marking can then be propagated to identical answers.

If	SELECT fieldnames with aggregate functions
AND	fieldnames match with table used
AND	the result-set is grouped by one or more fieldnames
AND	WHERE contains a condition
THEN	mark the SELECT and FROM clauses together as a group
AND	mark the GROUP BY clause as a separate part
AND	mark the WHERE clause as a separate part

The marking process of this rule is further described by Figure 7-13.

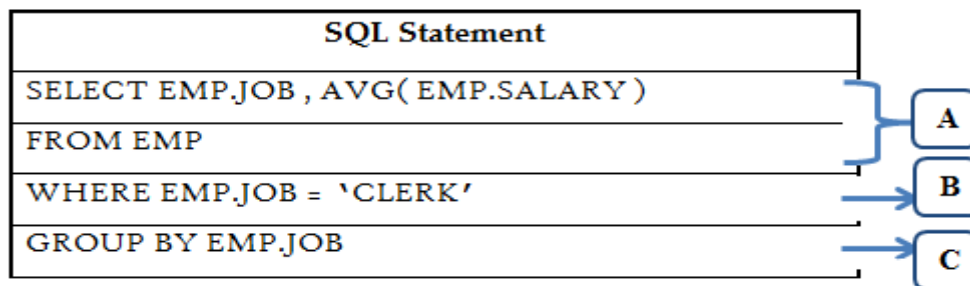


Figure 7-13: The WHERE clause marking process with a GROUP BY clause

As can be seen from this SQL statement, the student selected EMP.JOB as a fieldname and AVG(EMP.SALARY) as an aggregate function. The marking process will check the matching between the SELECT clause and the FROM clause's table names. Once matching is ensured, the availability of the fieldname of the GROUP BY clause is checked.

- A. Mark the SELECT and FROM as a group.
- B. Mark the GROUP BY clause as a separate clause.
- C. Mark the WHERE clause as separate clause and check the similarity with other statements following Rule I and Rule II for the WHERE clause.

In this case, marking the SELECT, FROM and GROUP BY clauses together is unwanted since it will increase the diversity of the SQL statements and reduce the similarities between them.

7.3.3.3. SELECT, FROM, GROUP BY and HAVING

The following rules cover four SQL statement clauses; namely the SELECT, FROM, GROUP BY and HAVING clauses. This section explains the rules by using sample of SQL statements answers of Question 3 as an explanation of the marking process.

Rule VI.GROUP BY and HAVING Clauses

This rule is applicable for SQL statements that contain fieldnames and aggregated functions that match the table that they are retrieved from, and the result-set is grouped by one or more fieldnames. In addition, the statements in question contain a single or multiple HAVING conditions.

If	SELECT fieldnames are used with aggregate functions
AND	fieldnames match the table used
AND	the result-set is grouped by fieldname
AND	having contains a condition
THEN	mark the SELECT and FROM clauses together as a group
AND	mark the GROUP BY and HAVING clauses together as a group

In the context of the HAVING clause, it cannot be separated from the group by a clause since the HAVING clause requires GROUP BY to be present. This is because the HAVING clause filters records that work on summarised GROUP BY clause results. As such, GROUP BY and HAVING clauses should be marked as a group, as shown in Figure 7-14.

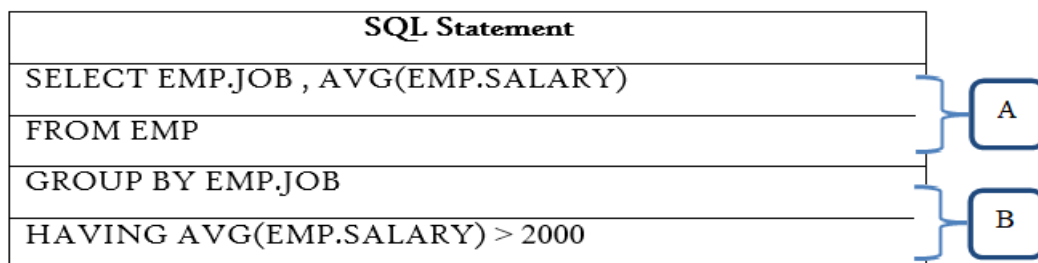


Figure 7-14: Marking GROUP BY and HAVING clauses as a group

As one can see from Figure 7-14, the marking process will check the matching between the SELECT clause and FROM clause's table names. Once matching is ensured, the availability of the fieldname of the GROUP BY clause is checked to be marked.

- A. The SELECT and FROM clauses should be marked as a group.
- B. The GROUP BY and HAVING clauses should be marked as a group.

There is one case in which the HAVING clause can be marked separately and it is addressed by the following rule.

Rule VII. HAVING Clause without GROUP BY Clause

This rule is applicable to SQL statements that contain aggregate functions that match the table that they are retrieved from without grouping the result-set by one or more fieldnames. In addition, the statements in question contain a single or multiple HAVING conditions.

If	SELECT has aggregate functions
AND	SELECT matches the table used
AND	HAVIG contains a condition
THEN	mark the SELECT and FROM clauses together as a group
AND	mark the HAVING clause as a separate part

In this case, the GROUP BY clause is omitted, which makes the aggregate function calculate a value for the entire table. The HAVING clause excludes the non-matching rows from the result group as shown in Figure 7-15.

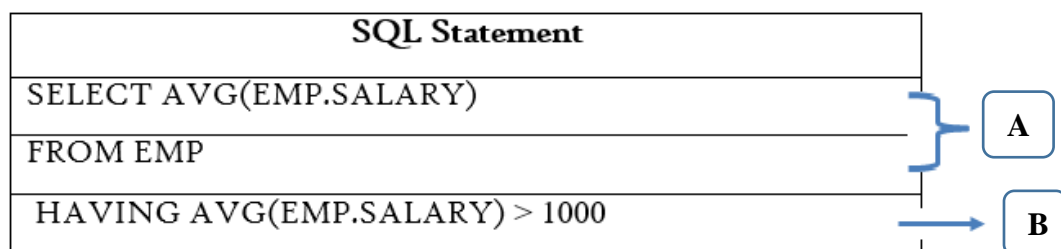


Figure 7-15: An SQL statement with a HAVING clause marked separately

As we can see from the SQL statement in Figure 7-15, the student selected AVG(EMP.SALARY) as an aggregate function. Moreover, the HAVING clause was executed without the GROUP BY clause, which makes it separate from the GROUP BY clause. The marking process for this statement is conducted as follows:

- A. Mark the SELECT and FROM clauses as a group.
- B. Mark the HAVING clause as a separate part.

7.4. Summary

This chapter has maintained the novelty of this research by integrating both Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) systems that use application of the Artificial Intelligence (AI) methodology. It provides a platform to support the assessment process of SQL statements, which supports the integration of both reasoning systems to enable human and computer association during the assessment process. This has increased the the accuracy of marking and provides students with immediate feedback. in addition, it reduce the overall SQL statement clauses marked by examiners. This means to reduce the human intervention on marking and reuse the comments given for similar SQL parts and the most important merit is enhances

As summary, this chapter discussed three main topics; the generic marking rules of SQL statements, the partial marking process of the SQL parts and propagation of marked SQL statements parts to identical parts. The semi-automatic marking rules of SQL statements are a set of minimum requirements and standards for marking and grading reparative clauses of SQL statements. Furthermore, the semi-automatic marking process involves the identification of common repetitive tasks in the assessment process.

The main objective of this chapter was to develop techniques to reduce the repetitive tasks or eliminate them from the marking process where possible by applying a normalisation operation. This objective has been achieved by proposing the semi-automatic assessment approach. The approach produces many outcomes, which help examiners to increase the similarities between SQL statements parts through removing any unnecessary elements from the SQL parts. In addition, it replaces the parts with an original format and sorts the data to be matched with other students' answers. This process was explained and tested with two different SQL statement data collection process.

Furthermore, the process of grouping the identical SQL statements was demonstrated as a means of saving marking time and providing consistent marks to students. This is because the similarities between SQL statements parts increases as a result of the normalisation operation and the number of statements that need to be marked reduces, which translates to less human intervention in the marking of SQL statements.

The semi-automatic assessment approach was implemented by a specialised tool based on the proposed approach. Using this tool, the process of finding the similarities between matching SQL parts acts to increase the marking process propagation between SQL parts, where the same marks and feedback from a solution that was previously marked can be used again for other identical students' solution. This assists the examiners in understanding the full process of marking using the semi-automatic marking approach.

The marking process using the developed SQL Marking Editor (*SQL-ME*) will be explained and tested in Chapter 8.

Chapter 8. Design, Implementation and Evaluation

SQL Marking Editor (SQL~ME)

8.1. Introduction

The proposed semi-automated marking approach aims to reduce the workload associated with the assessment task, and, more importantly, provide timely feedback for students. The proposed semi-automated approach utilises a specialised tool that uses the new partial marking techniques and propagation of marks and feedback. This chapter discusses the design and implementation details of semi-automated SQL assessments using the newly implemented SQL Marking Editor named as (SQL-*ME*), which follows the CBR and RBR approaches.

The rest of this chapter is organised as follows. Section 8.2 describes the different requirements and components for designing and implementing the SQL-*ME* editor. It also illustrates a simple example to illustrate the process of formulating SQL statements using the editor. Section 8.3 describes the process of marking SQL statements using the implemented SQL-*ME* editor. The full experiment on the marking process and a description of the study on the SQL generic marking rules are detailed in Section 8.4. Section 8.5 outlines the findings of the new system and the overall evaluation. Section 8.6 concludes the chapter by presenting a summary of its findings.

8.2. The SQL Marking Editor (SQL-*ME*)

The SQL Marking Editor (SQL-*ME*) is an online environment used to mark and evaluate students' SQL answers. The aim of the SQL-*ME* is to reduce the number of elements of SQL statements marked by the examiner and to ensure the consistency of marking of SQL statements the lecturers. In addition, it provides support for the submission of SQL statements. SQL-*ME* was implemented to support the partial marking approach. Researchers have generally used the partial marking approach for different tasks (Batmaz, 2011; Wong *et al.*, 2012; Adesina *et al.*, 2015).

This approach eliminates the repetitive marking task by exploiting the traits of human behaviour during the marking process and finding the similarities between old and new problems, then adopting the same marking. This is processed by identifying the identical elements across SQL statements and classifying them into groups and parts in which each clause is separated and marked. Subsequently, the identical properties of each student's answers are identified and marked automatically. This section describes the *SQL-ME* requirements and the *SQL-ME* user interface.

8.2.1. SQL-ME Requirements

There are three main requirements associated with designing the *SQL-ME* editor. The user interface requirements are as follows:

- 1) The *SQL-ME* should contain two main user interfaces:
 - A. One dedicated for the marking process using partial marking.
 - B. One dedicated for the marking process using generic marking rules (grouped statements).
- 2) The *SQL-ME* user interface must support the delivery of the commented SQL parts using the partial marking technique.
- 3) The *SQL-ME* should allow the propagation of marks to identical answers by supporting the reuse of comments for repetitive SQL parts based on the CBR cycle.

8.2.2. SQL-ME User Interface

The main functionality of the new marking environment (*SQL-ME*) is matching SQL parts and reusing comments for the repetitive SQL parts based on the CBR cycle. All students' SQL statements are represented partially. This means the lecturer will mark the SQL parts of the full SQL statement by matching each part together. In this process, the similarities between the parts are automatically marked by the editor, where the CBR system takes the SQL part and compares and matches it (as a new case) with other solutions that has been marked, resulting in adopting the same marking by reusing the existing marks and feedback. The idea behind finding the similarities between the parts is to provide the same marks and feedback to students, thus ensuring consistency and a reduced workload. For that, the new marking environment needs to have textual marks and textual feedback generators to evaluate the students' submitted SQL statements.

Once the SQL statement answers have been graded and feedback generated by the lecturer, the marking environment will need to demonstrate the marking report of all answers submitted by the students, or alternatively, the lecturer may want to check the output of the executed query or delete any query. This will potentially save marking time and improve the consistency of the marking process. The SQL marking editor was also designed to help the evaluation experiment and to show which parts should be evaluated and why they are important to evaluate. The architecture of the SQL marking editor is shown in Figure 8-1. The SQL marking process architecture consists of four major engines: (1) the examination process, which is explained in detail in Chapter 6, (2) the SQL marking engine, which consists of using partial marking and the propagation of marks and feedback, (3) the SQL marking process, which utilises the marking rules, and (4) feedback presentation. In each of these engines, there are different processes whereby the system goes through different steps until the marking of all SQL statements is concluded.

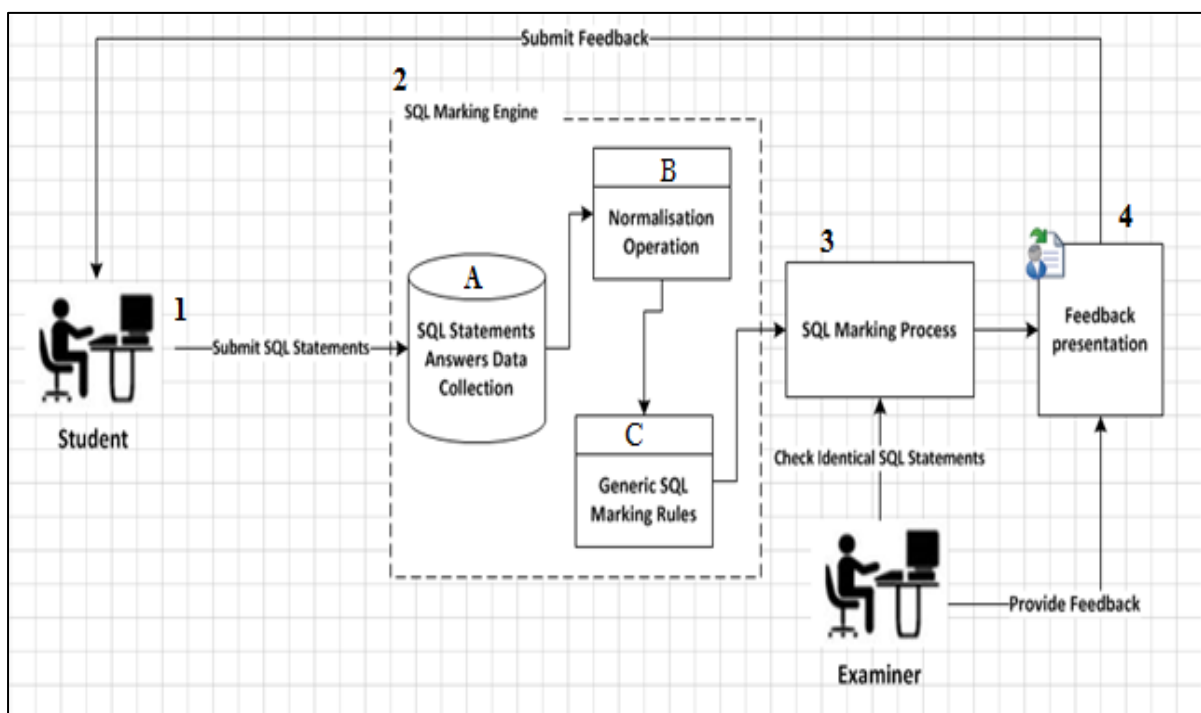


Figure 8-1: The SQL Marking process architecture

The SQL-ME is designed and implemented to clarify the requirements of the partial marking approach. The editor marks several answers simultaneously and depends on the similarities between the SQL parts. This section illustrates and explains how the SQL marking editor (SQL-ME) uses the partial marking approach.

Figure 8-2 illustrates the new *SQL-ME* environment that supports the partial marking approach (before starting the marking of statements). The circled letters represent the functions of each component in the new marking editor.

- a) Represents the selection list of the SQL questions, which the examiner will use to retrieve all students' SQL answers related to the same question.
- b) The marks have been categorised into three different colour categories: green, yellow and red. Each of these colours represents the status of each SQL statement part, where correct is represented by green, partially correct by yellow and incorrect by red. This helps to provide students with detailed feedback, which in turn aids them in understanding their mistakes and appreciating what they need to change in their statements.
- c) Shows the SQL statements answers that have been submitted by students using the *SQL-FE*, listed after the model answer, so they can be matched and marked together.
- d) There are three more components that the examiner will need to update the students' grading reports: saving marks and feedback for all identical SQL parts, executing the SQL query, and deleting data that is not needed.

The *SQL-ME* displays each question with a set of multiple student SQL answers. The examiner will start marking the SQL answer by matching the SQL parts and giving the same grades and feedback for matching answers. In other words, the same feedback for a specific part is used for other identical parts in multiple student SQL statement answers. The lecturer will view all the SQL answers for each question using the marking editor and start marking each part separately. Once the first part has been marked and graded, then another group with different answers will be assessed, where the similarities between the answers is noted. The editor divides partial marking into three categorisations: where green colour denotes a fully correct SQL part, yellow denotes a partially correct one, and red denotes a fully incorrect one. The examiner will be able to mark part by part as illustrated in the following figure. In order to identify the similarities in the divided-up parts of students' SQL answers and give consistent marks, the matching can be done either by individual division or by merging more than one parts together. This reduces the number of SQL statements marked and results in equal and fair marks to all students.

Select Question:
 1. Find the first names of all employees who are work as clerks and earn a salary of more than 2500.

Question : 1. Find the first names of all employees who are work as clerks and earn a salary of more than 2500.

Email: 14f12616@mec.edu.om Time Spent: 00:16:55

SQL Answer	Mark	Feedback
SELECT EMP.FNAME	Select Mark	Feedback
FROM EMP	Select Mark	Feedback
WHERE EMP.JOB = 'CLERKS'	Correct	Feedback
AND EMP.SALARY > 2500	Partially Correct	Feedback
	Incorrect	Feedback
	Select Mark	Feedback

Email: 13f10461@mec.edu.om Time Spent: 00:22:00

SQL Answer	Mark	Feedback
SELECT EMP.FNAME	Correct	correct
FROM EMP	Correct	correct
WHERE EMP.JOB = 'CLERK'	Partially Correct	the Clerk should add 's'
AND EMP.SALARY > '2500'	Incorrect	Numric value should not contains single quote

Figure 8-2: The user interface of the SQL-ME (partial marking interface)

8.3. SQL Marking Process (Generic Rules)

There are many reasons for which this research considers implementing a new system for learning and assessing SQL statements. First, by exploiting the similarities among SQL statements, the new system could solve the problem of manually marking the same sets of SQL statements submitted by hundreds of students' time and again. The matching process involves grouping statements into parts and groups and defining the similarities between them. Secondly, the avoidance of trivial mistakes (i.e. spelling mistakes, unnecessary words and synonyms). The new system would attempt to ignore irrelevant information by skipping those words that do not match certain keywords, or even not add them at all, since the system could prohibit writing anything inside the answer bar; and instead allow clicking on the navigation bar and selecting what is needed. Thirdly, reducing the need for human intervention in the marking process by reducing the number of SQL statements marked by lecturers. Finally, the last reason is to provide students with effective and encouraging feedback. Lecturers can provide personalised or generic feedback to their students depending on the student numbers. If student numbers are high, the lecturers may give generic feedback, in which feedback can be provided to a group of students who have made the same common mistakes. Alternatively, lecturers may prefer to specify individual, personalised feedback for each exam paper.

The semi-automatic assessment approach provides detailed and consistent feedback for SQL statements based on formative assessment. It focuses on commenting the repetitive clauses of students' SQL statements by using certain marking rules. This research focuses on the basic SQL clauses to apply the rules on, such as the `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `HAVING` and `ORDER BY` clauses. These rules are implemented on the new SQL Marking Editor (*SQL-ME*) as a back-end. This is because *SQL-ME* is a dynamic site that constantly changes and updates in real-time. In addition, all marks and feedback should be stored in the database to be viewed by the students. These rules are applied on SQL statements that have been normalised and classified into groups, where each group may have either a single SQL statement or repetitive (identical) SQL statements as illustrated in Figure 8-3. The figure shows the two main components in the *SQL-ME* generic rules user interface;

- a) The SQL student groups were combined in Section 7.2.3. In this case, all identical SQL clauses of the students' answers were checked and matched using the CBR approach.

- b) The number of identical SQL statements were matched and counted to show the number of SQL statements that can be marked and commented at one time.

The semi-automatic marking rules use some conditional sentences containing a conditional clause referred as (If-then statements). For example, “If a certain condition is true, then a particular result happens”. It is represented as: **If <condition> then <conclusion>**. The key advantage of semi-automatic marking rules is their flexibility. This means that it is possible to add new rules or modify existing ones without any side effects. However, every rule should be well written to attempt most of the SQL statements students’ answers. Following these rules can generally support the marker during the marking of SQL statements. Each of these rules is were explained with examples in Chapter 7 of SQL statements answers.

8.4. The Marking Process Experiment

Initially, the SQL Formulation Editor (SQL-*FE*) was tested to evaluate the amount of time spent to solve several SQL questions and the performance of students after using it. The second experiment, presented in this section, was conducted by the examiners to test the usefulness and usability of the semi-automatic SQL assessment approach by using the newly implemented SQL Marking Editor (SQL-*ME*). The study was conducted with six (6) Ph.D. research students in Loughborough University in January 2018. Each session involved one participant, who performed two tasks during a one hour session. This experiment used only the SQL-*ME* editor for evaluation and testing. The reason of not involving manual marking was due to the fact that the focus was to enable the examiners to test the new semi-automatic marking approach and how the marking process can be done using the partial marking technique. The study was approved by the ethics committee of Loughborough University. Furthermore, the participants were given simple introduction about the functionalities of the generic marking rules and were instructed on what they needed to do after testing the rules. They were also asked to comment on the proposed marking technique marking after using these rules. The main objectives are:

1. *To evaluate the feasibility of the semi-automatic approach, focusing on the assessment aspects.*
2. *To investigate the effects of SQL-*ME* on examiners and to know whether they consider it to be a useful marking editor.*
3. *To examine the standard of feedback generated and whether SQL-*ME* provides better feedback quality than other tools.*

Select Question:
 1. Find the first names of all employees who are work as clerks and earn a salary of more than 2500.

Question : 1. Find the first names of all employees who are work as clerks and earn a salary of more than 2500.

Group: 1 Number of Identical SQL statements: 5

SQL Answer	Mark	Feedback
SELECT EMP.FNAME	Select Mark	Feedback
FROM EMP	Select Mark	Feedback
WHERE EMP.JOB = 'CLERK'	Select Mark	Feedback
AND EMP.SALARY > '2500'	Select Mark	Feedback

Group: 2 Number of Identical SQL statements: 1

SQL Answer	Mark	Feedback
SELECT EMP.EMPNO	Select Mark	Feedback
FROM EMP	Select Mark	Feedback
WHERE EMP.SALARY > 2500	Select Mark	Feedback

Figure 8-3: SQL statements in groups (generic marking rules)

8.4.1. Participants

There were six participants who agreed to participate in this study. The study was presented to Ph.D. students from the Computer Science Department in Loughborough University, who had taught the Database module and had background on SQL formulation and marking techniques. The participants were required to be qualified in teaching and assessing different modules of the Database Program, with at least have 3-5 years of experience in teaching database modules, so they could be able to provide an objective evaluation based on their experiences. They were also required to be qualified in formulating and assessing SQL statements. The participants were invited to participate in the experiment through an official email (a sample of the email is presented in Appendix 10 – Section A). These requirements aimed to ensure the provision of consistent results and feedback from the different participants using the questionnaire.

The experiment took place on mid-January 2018 at Loughborough University. A sample of the printed list of instructions of the experiment was given to the examiners (see Appendix 10 – Section B). At the same time, a printed list of reference answers to the SQL questions and alternative ways to solve them was distributed to the participants.

8.4.2. Questions

The study used three SQL questions as shown in Table 8-1. Each question had a total of 30 students' answers. This means that each answer showed up as group of identical answers. In addition, each question represented one or multiple of the five generic marking rules. As there were five different marking rules, the three chosen questions were able to fit these rules and define the purposes of applying them. For these reasons, the questions used were not randomly selected from those used in the SQL Formulation Editor (*SQL-FE*) study, but were specifically chosen for this study. The SQL questions were categorised into three different requirements:

- **SELECT, FROM and WHERE (Question 1)**
- **SELECT, FROM, JOIN and ORDER BY (Question 2)**
- **SELECT, FROM, GROUP BY and HAVING (Question 3)**

Table 8-1: The SQL questions used in the experiment with their model answers

Question 1	Find the first names of all employees who work as a clerk and earn a salary of more than 2500							
Model Answer 1	<pre>SELECT EMP.FNAME FROM EMP WHERE EMP.JOB= 'CLERK' AND EMP.SALARY > 2500</pre>							
Output 1	<table border="1"> <thead> <tr> <th>FNAME</th> </tr> </thead> <tbody> <tr> <td>Jones</td> </tr> </tbody> </table>		FNAME	Jones				
FNAME								
Jones								
Question 2	Retrieve the last names and the department names of all female employees. Sort the result in ascending order of the location.							
Model Answer 2.1	<pre>SELECT DEPT.DEPTNAME , EMP.LNAME FROM DEPT INNER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO WHERE EMP.GENDER='FEMALE' ORDER BY DEPT.LOC</pre>							
Model Answer 2.2	<pre>SELECT DEPT.DEPTNAME , EMP.LNAME FROM DEPT , EMP WHERE DEPT.DEPTNO = EMP.DEPTNO AND EMP.GENDER = 'FEMALE' ORDER BY DEPT.LOC</pre>							
Output 2	<table border="1"> <thead> <tr> <th>LNAME</th> <th>DEPTNAME</th> </tr> </thead> <tbody> <tr> <td>Paul</td> <td>Operation</td> </tr> <tr> <td>Louis</td> <td>Management</td> </tr> </tbody> </table>		LNAME	DEPTNAME	Paul	Operation	Louis	Management
LNAME	DEPTNAME							
Paul	Operation							
Louis	Management							
Question 3	Display the various jobs and the average salary of employees in each job, where the average salary is greater than 2000.							
Model Answer 3 (Group by & Having Commands)	<pre>SELECT EMP.JOB, AVG(EMP.SALARY) FROM EMP GROUP BY EMP.JOB HAVING AVG(EMP.SALARY) > 2000;</pre>							
Output 3	<table border="1"> <thead> <tr> <th>JOB</th> <th>AVG(SALARY)</th> </tr> </thead> <tbody> <tr> <td>Manager</td> <td>2650</td> </tr> </tbody> </table>		JOB	AVG(SALARY)	Manager	2650		
JOB	AVG(SALARY)							
Manager	2650							

8.4.3. Measurements

Each of the following measurements was represented with a list of questions that were evaluated and answered by the participants after the experiment (see the Appendix 11). The following characteristics were measured during the experiment:

a) The SQL-*ME* environment: the study measured the feasibility of the semi-automatic approach, focusing on the assessment aspects by using both marking system pages. The objective was to gain insight into the quality of the two different environments by measuring the number of students appearing on the list and the groups available, and to gain insight into the quality of the semi-automated marking process.

b) The time spend on the marking process: the time that participants needed to complete the marking and write their feedback on the answers was measured. The objective was to compare the time needed to complete the marking across different environments, such as groups, marks and feedback.

c) Satisfaction: a questionnaire was filled by each participant after finishing the marking to assess satisfaction. The questionnaire's questions intended to measure the overall satisfaction on the use of the SQL-*ME* environments. The objective was to collect additional qualitative feedback from participants about the quality of the newly implemented editor, as well as the perceived usefulness, main difficulties and drawbacks.

8.4.4. Data Collection

The data collection and analysis of the marking rules were only used on the first SQL question of the SQL-*FE* experiment. The data went through the normalisation and grouping processes, and was subsequently retrieved to be tested with the new generic marking rules. The first question contained a total of eight groups, as listed previously in Table 7-7. It was designed to assess the basic SQL SELECT statements which cover the SELECT, FROM and WHERE clauses, as described in detail in Section 8.2.4. According to the list of the questions, Q1 was used since it is a direct question and has fewer clauses requirements (SELECT, FROM and WHERE). This made the explanation of the making rules more efficient and simpler to understand by the markers. The SQL marking rules classifications were categorised into three main categories:

- (1) SELECT, FROM, (WHERE or/and ON) and ORDER BY
- (2) SELECT, FROM and GROUP BY,
- (3) SELECT, FROM, GROUP BY and HAVING.

All questions were tested with the generic marking rules. The main aim of the testing was to check the reduction of the number of SQL statements after applying the new marking rules technique and to observe the marking process before the implementation of a specialised editor that is specifically developed to mark the SQL parts and groups.

8.4.5. Experimental Results and Discussion

As described in Chapter 7, the CBR and RBR were applied on several SQL statements to evaluate the marking process. The experiment was divided into three parts.

The first part involved testing the SQL marking using a comparison between single and grouped SQL statements. The second part involved testing the propagation of marks and feedback using the CBR cycle, and lastly, the third part tested the usefulness of *SQL-ME* using all features.

Testing the marking process involved both the single and grouped statements using both user interfaces, *SQL-ME 1* and *SQL-ME 2*. In this experiment, the participants were given three SQL questions (as mentioned in Section 8.4.2) along with the model answer. The task they were asked to complete was to start marking these three questions using *SQL-ME 1* first (Figure 8-2), where they needed to mark the questions using the partial marking for single students. Once they finished this process, the participants needed to go through the same process but using *SQL-ME 2* (Figure 8-3), in which they had to mark in a group. The objective of this task was to compare the time spent and effort taken to perform both tasks, and which one the participants had a preference the SQL marking tool to use. By using the questionnaire (Appendix 11 – Section A), the participants had to answer according to what they experienced during testing of both user interfaces.

In Questions 1, 2 and 3, the participants found that *SQL-ME 2* was more effective and less marking was involved for all participants. The participants' responses on Question 2 differed from one to another, however, most of them showed their satisfaction of the newly implemented tool. The responses are shown in Table 8-2.

Table 8-2: The participants' responses on Q1, 2 and 3

PNo	Q1: Is there any difference between the two pages of the SQL-ME (marking system 1 and making system 2)?	Q2: If you answered 'yes' to the question above, please provide details	Q3: Which of these two pages do you prefer?
1	Yes	Provides identical marking	SQL-ME 2
2	Yes	Marking is identical	SQL-ME 2
3	Yes	Less time spent in marking system	SQL-ME 2
4	Yes	Students are grouped by the same answers in marking system 2	SQL-ME 2
5	No	Both of them provide identical marks and feedback	Marking System 1 (partial marking)
6	Yes	Saves time and workload	Marking System 2

Question 4 measured the time spent on marking the SQL statements using the desired user interface. As most participants chose SQL-ME 2, they needed to predict how much time they had spent regarding on the task given to them.

Table 8-3: The participants' responses on Q4

Questions	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6
a. Do you think you saved time?	Yes	Yes	Yes	Yes	Yes	Yes
b. If yes, what is the proportion of time do you think was saved?	5%	20%	20%	5%	20%	20%
c. Do you think this proportion would increase as you get more used to SQL-ME?	Yes	Yes	Yes	Yes	Yes	Yes
d. If yes, what is the proportion of time do you think you will save?	20%	50%	5%	20%	20%	50%

In Q4, the participants had different thoughts about the time spent, but 100% of the participants agreed in sections (a) and (c) that using the SQL-ME saved their time and that the amount of time saved would increase if they were to keep practicing the use of the new tool. Furthermore, according to the participants, the time saving was either 5% or 20%, however, most of the participants (4) answered 20%, which shows that SQL-ME saves a non-negligible amount of their marking time.

The last part of Q4 (d) showed that with practice, the participants believed that the proportion of marking time that could potentially be saved ranged 5% to 50%, as illustrated in Figure 8-4.

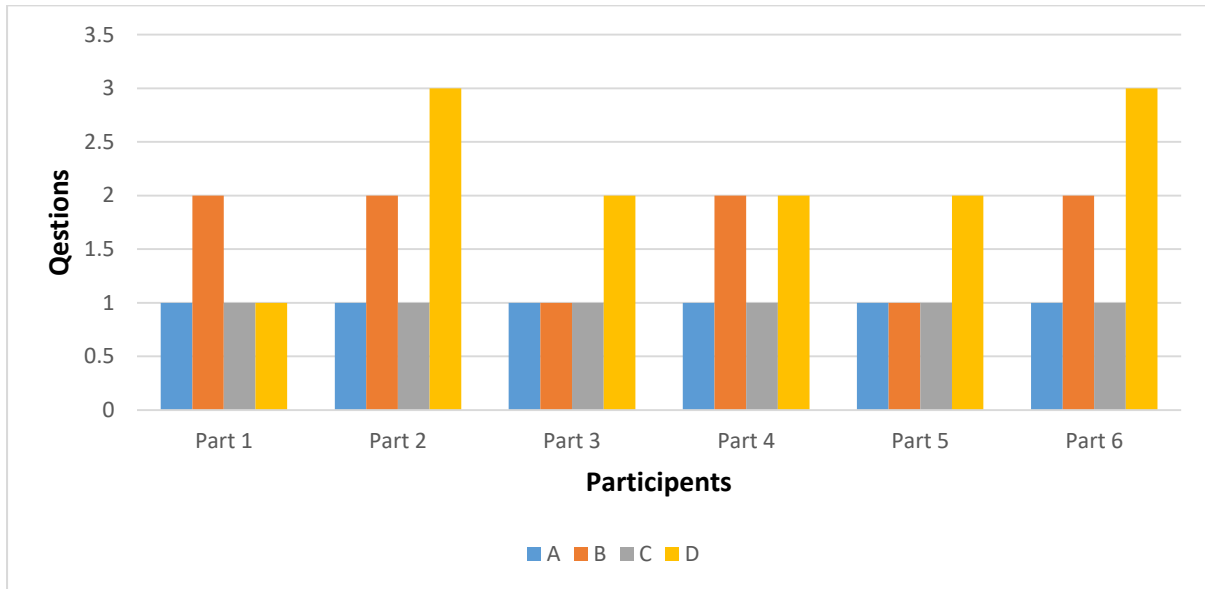


Figure 8-4: The participants' responses' on Q4

The second part of the experiment evaluated the feedback quality and how students can obtain similar feedback as a result of marking propagation. While doing the first test, the participants had the opportunity to understand the use of CBR as an approach, which can assist the marker in assessing and marking the similarities between the existing SQL part and the current SQL part. Furthermore, the test allowed measuring the effect of propagation of both marks and feedback. For Questions 1, 2 and 3, the responses of the participants is displayed in Table 8-2. The overall answers of the participants show that they were well satisfied with the new SQL-*ME* marking process and that the feedback produced using the editor is likely encourage the examiners as well as the students once they use it. In Question 4 on the feedback quality, multiple choices on some statements were presented to the participants about how they related to the new system. The answers differed from one participants to another depending on their understanding of the statement after they tested SQL-*ME*. For Question 4.a, three participants selected '3', whereas two participants selected '2' and only one participant selected '1' as an answer.

Table 8-4: The participants’ responses on feedback quality (Q1, 2 & 3)

	Part 1	Part 2	Part 3	Part 4	Part 5	Part 6
1. Do you feel you gave better feedback with SQL-ME?	Yes	Yes	Yes	Yes	Yes	Yes
2. Can you estimate whether you gave more or less feedback with the SQL-ME?	More feedback	More feedback	More using of the system will lead to more feedback	With the system more can improve the feedback	More feedback	Consistent feedback for all students and more feedback
3. Do you think you gave better quality feedback with SQL-ME? Explain.	Better quality because you don’t waste time on writing the same feedback for all students with the same answer, so you can spend time on writing better and more feedback.	Because it give specific feedback	Yes, student performance will improve by time with using the system	Yes, using the partial marking will improve the students’ performance	-	It can improve feedback and marks of the students

In Q4.b, four participants selected answer number ‘1’ and two selected answer number ‘2’, while In Q4.c, five participants chose number ‘3’ as their answer, and only one participant selected answer number ‘1’, as illustrated in Figure 8-5.

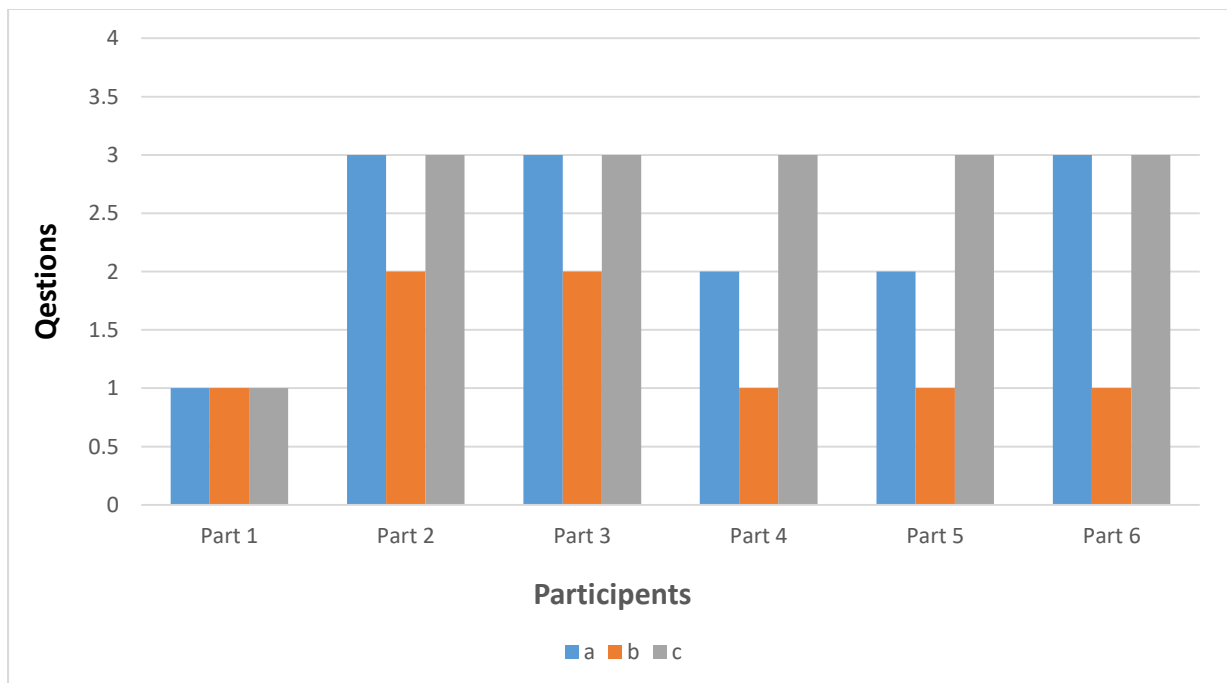


Figure 8-5: The participants’ responses on feedback quality (Q4)

The last part of this experiment tested the usefulness of *SQL-ME*. The participants were asked to test the rules on the SQL statements available on *SQL-ME* and give their predictions of number of reductions of the SQL statements after applying the rules, as well as write their feedback about these rules and how they can provide the examiners with an enhanced approach of reducing the number of the SQL statements marked by the examiners. Figure 8-6 illustrates the opinions of participants about the new marking system. The figure shows that all participants were very satisfied with *SQL-ME* once they used it and experimented with its features.

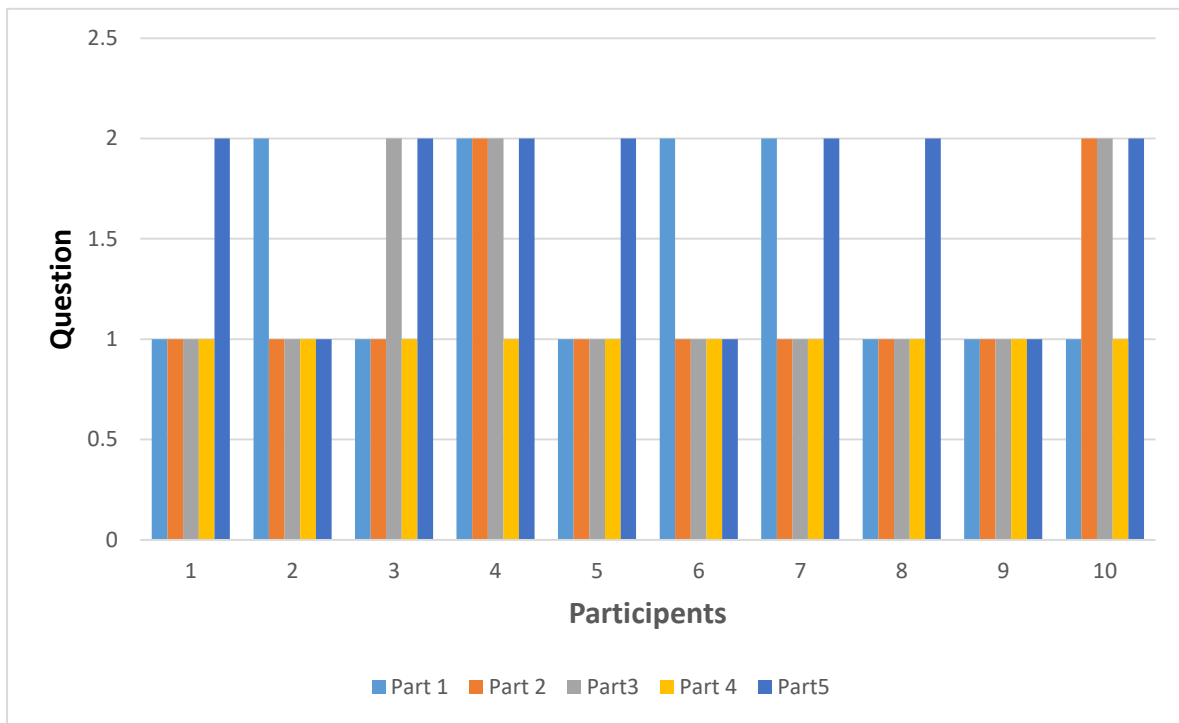


Figure 8-6: The participants' responses on usefulness of *SQL-ME*

8.5. Findings

The research presented in this chapter evaluated the semi-automatic assessment approach, which is based on the integration of CBR and RBR systems. These systems need to be adopted in the new marking technique for reusing previous SQL solutions for similar cases, which may contribute towards providing students with consistent marks and feedback. Also, providing timely feedback to students either individually or in groups would help students to improve their SQL skills.

The same feedback can be used for different students if they make the same mistakes in the same scenarios. Furthermore, using groups of SQL statements should help lecturers to send either individualised feedback or detailed feedback to different students. This has encouraged the standard of feedback generated and that *SQL-ME* has provided better feedback quality than other tools since each clause will contain different feedback and it can be propagated. In addition, the adoption of old solutions assists the marking process in identifying the unique values of the statements that have not taken more time, as most of the statements would have already been marked and highlighted. The CBR approach assists in saving markers valuable marking time, and once the marking is accomplished, different students receive similar feedback. The student may get a score for the partial answers, while the examiner needs all students' achievement info of the experiment for each question. This process is faster for the examiner in terms of providing the same consistent feedback for all students. In addition, the feedback part can have more specific features, where each student can receive more specific detailed feedback about what exactly their mistakes were, which can help them to improve and address their shortcomings.

8.6. Summary

This chapter discussed three main topics; the generic marking rules of SQL statements, the partial marking process of the SQL parts, and the propagation of marked SQL statements parts using the *SQL-ME* tool. The partial marking and grouped SQL statements were tested and evaluated. Most of the participants preferred the grouped SQL statements approach since it saved more of their time and provided consistent feedback for all students. The integration of the CBR and RBR systems have allowed the adoption of a new marking technique based on reusing previous SQL solutions for similar cases, which has resulted in enhancing the marking process of SQL statements and provided ideas on how to enhance the user interface to make it more efficient. Furthermore, the human intervention can be further reduced by adding more features through the integration of both systems. Overall, this research focused on improving the learning and assessment of SQL statements by providing students with consistent and high quality feedback. Furthermore, this research contributed toward saving marking time for all examiners in the marking process using the CBR cycle. In addition, the use of RBR served to enhance the marking of grouped statements and the evaluation of all answers equally.

Chapter 9.

Conclusion and Future Work

9.1 Introduction

Manual grading of SQL exams is time consuming for most of the lecturers. This research proposing a semi-automated assessment approach as a solution to ensure the consistency of the SQL grades and feedback generated during the marking. It aims to minimise the required human effort for assessing and evaluating SQL statements. Besides, it provides timely feedback to the students, which can include individual and detailed feedback.

It summaries the work described in this thesis in section 9.2. The chapter list the main contributions of the project in section 9.3. Section 9.4 discusses some of the limitations of the work and how they might be overcome as a future work. Section 9.5 is the chapter summary.

9.2 Summary of each chapter

Chapter 1: an overview of computer assisted assessment and motivation. It discusses the aims and objectives of the research. It discusses the research approach and contributions and lastly it concludes by outlining the structure of this thesis.

Chapter 2: This chapter illustrates assessment in education and shows the process of computer-assisted assessment and points out three different techniques of CAA and their features by comparing them. In addition, it shows the definitions and specifies the difference between manual and automated assessments.

Chapter 3: This chapter gives an overview of the Structured Query Language (SQL) in and demonstrates the process of acceptable SQL assessment marking and SQL grading techniques. In addition, the review of existing SQL learning and assessment tools and their features explained in detail. It represents the two types of systems in AI used in education such as CBR and RBR.

Chapter 4: This chapter discusses the three approaches to research and gives the types of research designs and which design has been selected for this research. The chapter then discusses the research data collection and data analysis methods.

Chapter 5: describes the analysis of existing SQL statements exam scripts. It analysed the two prospective areas of students and examiners. One is the common errors and different ways of solving the queries of SQL statements done by students. Second, the time spent and consistency difficulties to mark manual exams by examiners. It proposed specialised editor to enhance the student query formulation and examiners marking process.

Chapter 6: It explains the design, implementation and evaluation of the specialised SQL Formulation Editor (SQL-FE). It aimed to allow students to formulate the SQL statements without adding any unnecessarily elements and make it more effective in case of spelling mistakes done by students. Three different studies have taken place to test and evaluate the new SQL-FE editor.

Chapter 7: explains the semi-automatic assessment approach stages. The approach stages in this chapter are as follows pre-processing, normalisation and grouping, generic marking rules and feedback propagation. The CBR and RBR systems are applied and tested in different applications of the new marking techniques. The partial marking and generic rules marking have given a consistent feedback and marks after using the marking process in the existing SQL statements.

Chapter 8 explains the design, implementation and evaluation of the specialised SQL Marking Editor (SQL-ME). It aimed to mark the students SQL answers of the grouped identical statements. It shows the big contribution after adding the CBR approach on the system and test the RBR to set up all the rules of the SQL statements.

Chapter 9 is the conclusion and future work of the semi- automatic assessment project. The project has ensured the work of the semi-automatic approach.

9.3 Contributions

The novel contribution of this research is the development of a novel framework that provides a platform to support the assessment process of SQL statements. Such a framework enables human and computer association during assessment. Furthermore, this framework helps to analyse beginner students' SQL statements in terms of SQL clauses to provide consistent feedback. The framework also reduces the overall SQL statement clauses marked by examiners, enhances the accuracy of marking and provides students with immediate feedback.

It utilises a semi-automated assessment approach, which supports the integration of both case-based reasoning system and rule-based reasoning system to allow human markers. In addition, it aims to reduce or remove as many of the repetitive tasks in all phases of the marking process of SQL statements as possible. This contribution has led to several contributions to add more effective SQL semi-automatic assessment project.

1. To identify the common mistakes committed by students and find the alternative ways of solving the same SQL query, the researcher has collected and analysed previous SQL exam papers. The analysis has gone through different phases to identify them.
2. To formulate SQL statements that eliminate adding unnecessary elements to SQL statements and prevent students from making minor and avoidable mistakes, the researcher has designed and implemented a new SQL Formulation Editor named as *SQL-FE*.
3. To obtain the students feedback of the new implemented editor and to test the editor performance that reduce the errors while solving SQL statements, the researcher has evaluated the *SQL-FE* from several college students and collect their opinions of how to enhance it.
4. To reduce the repetitive marking in duplicated SQL answers or remove them completely where possible, the researcher has applied the normalisation operation, which is based on the proposed semi-automatic SQL assessment framework. This lead to develop a new technique for marking process using the SQL generic marking rules. The SQL marking process is an integration of both Rule-based Reasoning (RBR) and Case-based Reasoning (CBR) systems. This method shows how efficiency and savings in marking time may be obtained by reducing repetitive activities.
5. The marking process of the SQL statement has proposed a new semi-automatic assessment framework to mark the identical SQL statements using a new SQL Marking Editor named as *SQL-ME*.
6. To obtain the lecturers feedback of the new implemented editor and to evaluate the feasibility of the editor performance, the researcher has performed an appropriate experimental study to evaluate the feasibility of the semi-automatic assessment approach using the new implemented *SQL-ME* through several SQL experienced lecturers and collect their opinions of how to enhance it.

9.4 Limitation and Future Work

This research has successfully achieved the main goals and objective. However, there are some drawbacks, which are listed. In addition, each limitation listed can be considered as future work and they way to solve it. For that, there are some limitations and their solutions, which can be listed as follows.

1. Student is not allowed to use the Keyboard. This have made students confused as first time use the editor, however, the main objective is not to add unnecessarily elements to the SQL code.

Future work: This can be enhanced by converting the tool to be abdicable to work in touch screens where some of them allow using touch pen. This would make the SQL-*FE* more attractive and they will not need to use any typing since they need just to click-and-point and get the results. In addition, Add the restricted programming language to restrict the number of clauses appear for each SQL scenario. This would enhance the students SQL practice skills and restricted the dissimilarities between the student answers.

2. The SQL-*FE* editor list of rows in each table that student can use to retrieve data from. This is because of the design of the site has kept the table schema very limited with list of columns and data types only.

Future work: This can be solved by adding a link to the other webpage that allow maximising the table for students to retrieve all data they required and at the same time.

3. The SQL-*FE* is implemented to do simple work for first year of higher education.

Future work: Enhance the formulation editor to be more efficient for higher education assessments. This means, make it more effective where it should contain more features to be used by higher levels in education.

4. The generic rules have been formulated regarding to the existed SQL questions and used limited number of SQL clauses to be tested. This has caused some drawback when testing the rules on other statements.

Future work: Enhance the formulation of the generic marking rules to address the different cases of SQL statements.

5. As the SQL generic marking rules has taken long time to be formulated to make it generic and can work with several types of syntax, the implementation has not been completed in this section. On the other hand, the testing of rules in normal statements has given beneficial results as mentioned in Chapter 8.

Future work: Implement the generic marking rules to enhance the grouping of each answer and allow examiners to mark and give proper feedback. Simultaneously, this can be solved by enhancing the usage of CBR and RBR for the marking purpose where more attributes can be added and evaluated first. Then add the features on the user interface to make more effective.

9.5 Summary

The novel contribution of this project was to develop the semi-automatic assessment of the SQL statements. A novel framework that provides a platform to support the assessment process of SQL statements has been implemented. It implement a new marking technique by integrating the CBR and RBR systems by checking similarities between the old and new cases or problems and find the matching parts to be adopted as solution. As well as, an SQL Formulation Editor has been developed to enhance the learning of the SQL statements for students.

Overall, participants (students and examiners) were satisfied with the performance of the new *SQL-FE* and *SQL-ME* tools. This is because, it provides students with better environments to formulate the SQL statements using *SQL-FE*. Also, it provides examiners with new marking approach by using either partial marking by using *SQL-ME1* or group identical SQL answer by using *SQL-ME2*. Furthermore, they were also satisfied with the tools since they allowed them to provide more consistent and personalised feedback in a short time period compared to their traditional way of marking.

References

- Aamodt, A. and Plaza, E. (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*. AI Communications. IOS Press. Available at: <https://ibug.doc.ic.ac.uk/media/uploads/documents/courses/CBR-AamodtPlaza.pdf>.
- Abelló, A. *et al.* (2008) 'LEARN-SQL: Automatic Assessment of SQL Based on IMS QTI Specification.', in *ICALT*. Citeseer, pp. 592–593.
- Adams, A. and Cox, A. (2008) *Questionnaires, in-depth interviews and focus groups*.
- Adesina, A. *et al.* (2013) 'Use of multi-touch gestures for capturing solution steps in arithmetic word problems', pp. 6–8.
- Adesina, A. *et al.* (2015) 'A Semi-Automatic Computer-Aided Assessment Approach for Marking and Providing Feedback Comments', pp. 93–100.
- Adesina, A. O. (2016) 'A semi-automatic computer-aided assessment framework for primary mathematics', *PQDT - UK & Ireland*. Available at: <https://search.proquest.com/docview/1917320364?accountid=12063%0Ahttp://fg2fy8yh7d.search.serialssolutions.com/directLink?&atitle=A+semi-automatic+computer-aided+assessment+framework+for+primary+mathematics&author=Adesina%2C+Adewale+O.&issn=&title=A+semi->
- Ahadi, A., Prior, J., *et al.* (2016) 'Students' Semantic Mistakes in Writing Seven Different Types of SQL Queries', in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '16*. New York, New York, USA: ACM Press, pp. 272–277. doi: 10.1145/2899415.2899464.
- Ahadi, A., Behbood, V., *et al.* (2016) 'Students' Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students' Success'. doi: 10.1145/2839509.2844640.
- Al-Salmi, A. (2018) *A Web-based Semi-Automatic Assessment Tool for Formulating Basic SQL Statements: Point-and-Click Interaction Method*. Available at: <https://pdfs.semanticscholar.org/7afc/59a765b6641720566e261c464d91786bb09f.pdf>.
- Ala-Mutka, K. M. (2005) 'A survey of automated assessment approaches for programming assignments', *Computer Science Education*, 15(2), pp. 83–102.
- Batmaz, F. (2011) *Semi-Automatic assessment of students' graph-based diagrams*. Available at: <https://dspace.lboro.ac.uk/dspace-jspui/handle/2134/8431>.
- Batmaz, F. and Hinde, C. J. (2007) 'A web-based semi-automatic assessment tool for conceptual database diagram', in *Proceedings of the sixth Web-Based Education conference*, pp. 427–432.

- Bedford, S. and Price, G. (2007) 'A study into the use of computer aided assessment to enhance formative assessment during the early stages of undergraduate chemistry courses'.
- Benford, S. D., Burke, E. K. and Foxley, E. (1993) *Ceilidh: A course administration and marking system*. na.
- Bennett, S. *et al.* (2017) 'How technology shapes assessment design: Findings from a study of university teachers', *British Journal of Educational Technology*, 48(2), pp. 672–682. doi: 10.1111/bjet.12439.
- Berka, P. (2011) 'NEST: A Compositional Approach to Rule-Based and Case-Based Reasoning', *Advances in Artificial Intelligence*. Hindawi, 2011, pp. 1–15. doi: 10.1155/2011/374250.
- Bichindaritz, I., Kansu, E. and Sullivan, K. M. (1998) *Integrating Case-Based Reasoning, Rule-Based Reasoning and Intelligent Information Retrieval for Medical Problem-Solving*. Available at: www.aaai.org.
- Bisland, R. B. (1989) *Database Management: Developing Application Systems Using ORACLE*. Prentice Hall (Works of Jonathan Edwards; 9).
- Bloom, B. S. (1956) 'Taxonomy of educational objectives. Vol. 1: Cognitive domain', *New York: McKay*.
- Bluman, A. G. (2012) *Elementary statistics - A step by step approach 8th edition (international student edition)*. 8th edn. New York, USA: McGraw Hill Companies.
- Bobak, A. R. (1996) *Distributed and Multi-Database Systems*. 2nd edn. Artech House, INC (Artech House computer science library).
- Bonastre, O., Benavent, A. and Belmonte, F. (2006) 'Pedagogical Use of Tablet PC for Active and Collaborative Learning', in *2006 IEEE International Professional Communication Conference*. IEEE, pp. 214–218. doi: 10.1109/IPCC.2006.320350.
- Boritz, J., Booth, K. S. and Cowan, W. B. (1991) 'Fitts's Law Studies of Directional Mouse Movement', pp. 216–223.
- Bruno, N. (2003) *Statistics on Query Expressions in Relational Database Management Systems*.
- Brusilovsky, P. *et al.* (2008) 'An open integrated exploratorium for database courses', *ACM SIGCSE Bulletin*, 40(3), pp. 22–26.
- Brusilovsky, P. *et al.* (2010) 'Learning SQL programming with interactive tools: From integration to personalization', *ACM Transactions on Computing Education (TOCE)*, 9(4), p. 19.
- Bryman, A. *et al.* (2011) 'Business Research Methods', *Oxford University Press*, p. 220. doi: 0195430298.

- Bull, B. J. and Mckenna, C. (2004) 'Blueprint for Computer-Assisted Assessment', 2(November 2003).
- Bull, J. and Danson, M. (2004) 'Computer-assisted Assessment (CAA)', 14(14), p. 26. Available at: https://www.heacademy.ac.uk/system/files/id350_computer_assisted_assessment_caa_.pdf.
- Buyrukoglu, S. (2018) 'Semi-automated assessment of programming languages for novice programmers'.
- Buyrukoglu, S., Batmaz, F. and Lock, R. (2016) *Semi-automatic assessment approach to programming code for novice students*. Available at: <https://dspace.lboro.ac.uk/2134/20477> (Accessed: 16 August 2018).
- Cabrera, M. and Edye, E. (2010) 'Integration of rule based expert systems and case based reasoning in an acute bacterial meningitis clinical decision support system', *International Journal of Computer Science and Information Security*, 7(2), pp. 112–118.
- Cabrera, M. M. M. and Edye, E. O. E. (2010) 'Integration of Rule Based Expert Systems and Case Based Reasoning in an Acute Bacterial Meningitis Clinical Decision Support System', *International Journal of Computer Science and Information Security*, 7(2), pp. 112–118. Available at: <http://arxiv.org/abs/1003.1493>.
- Carter, J. *et al.* (2003) 'How shall we assess this?', in *ACM SIGCSE Bulletin*. ACM, pp. 107–123.
- Chalmers, D. and McAusland, W. D. M. (2002) 'Computer-assisted assessment', *The Handbook for Economics Lecturers: Assessment*. Edited by J. Houston and D. Whigham. *Economics LTSN*, online at www.economics.ltsn.ac.uk/handbook.
- Chan, D. and Schmitt, N. (1997) 'Video-based versus paper-and-pencil method of assessment in situational judgment tests: subgroup differences in test performance and face validity perceptions.', *The Journal of applied psychology*, 82(1), pp. 143–59. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/9119795> (Accessed: 3 September 2016).
- Clark, I. (2011) 'Formative Assessment: Policy, Perspectives and Practice', *Florida Journal of Educational Administration & Policy*, 4(2), pp. 158–180.
- Codd, E. F. (1970) 'A relational model of data for large shared data banks', *Communications of the ACM*, 13(6), pp. 377–387.
- Conole, G. and Warburton, B. (2005) 'A review of computer-assisted assessment', *Research in learning technology*, 13(1).
- Cresswell, J. W. (2014) 'The selection of a research approach', in *Research design: qualitative, quantitative and mixed methods approaches*, pp. 3–23. doi: 10.3917/rsi.100.0107.
- Cumming, A. and Russell, G. (2005) 'Automatic checking of SQL: Computerised grading.', *International Journal of Learning*, 12(3), pp. 127–134.

- Dalziel, J. (2001) 'ENHANCING WEB-BASED LEARNING WITH COMPUTER ASSISTED ASSESSMENT: PEDAGOGICAL AND TECHNICAL CONSIDERATIONS considerations'. Available at: <https://dspace.lboro.ac.uk/dspace-jspui/bitstream/2134/1795/1/dalziel01.pdf>.
- Dekeyser, S., de Raadt, M. and Lee, T. Y. (2007) 'Computer assisted assessment of SQL query skills', in *Proceedings of the eighteenth conference on Australasian database-Volume 63*. Australian Computer Society, Inc., pp. 53–62.
- Donahoo, M. J. and Speegle, G. D. (2010) *SQL: Practical Guide for Developers*. Elsevier Science (The Practical Guides). Available at: <http://books.google.co.uk/books?id=19mS1g45fUEC>.
- Douce, C., Livingstone, D. and Orwell, J. (2005) 'Automatic test-based assessment of programming: A review', *Journal on Educational Resources in Computing (JERIC)*, 5(3), p. 4.
- Dutta, S. and Bonissone, P. P. (2013) 'Integrating Case-Based and Rule-Based Reasoning: the Possibilistic Connection'. Available at: <http://arxiv.org/abs/1304.1116> (Accessed: 19 August 2018).
- Entwistle, N. (2000) 'Promoting deep learning through teaching and assessment: conceptual frameworks and educational contexts', in *TLRP conference, Leicester*.
- Fehily, C. (2010) *SQL: Visual QuickStart Guide*. Pearson Education. Available at: <http://books.google.co.uk/books?id=k9SE25v12I4C>.
- Gardiner, W. P. and Gettinby, G. (1998) *Experimental Design Techniques in Statistical Practice: A Practical Software-Based Approach*. Elsevier Science (Horwood Series in Mathematics & Applications).
- Gillan, D. J. *et al.* (1990) 'How does Fitts' law fit pointing and dragging?', in *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. New York, New York, USA: ACM Press, pp. 227–234. doi: 10.1145/97243.97278.
- Grange, J. (2011) *T-test in Microsoft Excel*. Youtube. Available at: https://www.youtube.com/watch?v=BIS11D2VL_U. (Accessed: 19 August 2016).
- Greener, S. (2008) *Business Research Methods*. Ventus Publishing ApS.
- Grosan, C. and Abraham, A. (2011) 'Rule-Based Expert Systems', in Springer, Berlin, Heidelberg, pp. 149–185. doi: 10.1007/978-3-642-21004-4_7.
- Harlen, W. *et al.* (1992) 'Assessment and the improvement of education*', *The Curriculum Journal*, 3(3), pp. 215–230.
- Harlen, W. and James, M. (1997) 'Assessment and learning: differences and relationships between formative and summative assessment', *Assessment in Education*, 4(3), pp. 365–379.
- Higgins, C. *et al.* (2003) 'The coursemarker cba system: Improvements over ceilidh', *Education and Information Technologies*. Springer, 8(3), pp. 287–304.

- Higgins, C. A. *et al.* (2009) 'Authoring diagram-based CBA with CourseMarker', *Computers & Education*, 52(4), pp. 749–761.
- Higgins, C., Symeonidis, P. and Tsintsifas, A. (2002) 'Diagram-based CBA using DATsys and CourseMaster', in *Computers in Education, 2002. Proceedings. International Conference on. IEEE*, pp. 167–172.
- Hopgood, A. a. (2012) *Intelligent Systems for Engineers and Scientists*. Third, Library. Third. CRC Press.
- Ihantola, P. *et al.* (2010) 'Review of recent systems for automatic assessment of programming assignments', in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. ACM, pp. 86–93.
- Inkpen, K. M. (2001) 'Drag-and-Drop versus Point-and-Click Mouse Interaction Styles for Children', *ACM Transactions on Computer-Human Interaction*, 8(1), pp. 1–33.
- Insa, D. and Silva, J. (2015) 'Semi-Automatic Assessment of Unrestrained Java Code', in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '15*. New York, New York, USA: ACM Press, pp. 39–44. doi: 10.1145/2729094.2742615.
- Jackson, D. (1996) 'A software system for grading student computer programs', *Computers & Education*, 27(3–4), pp. 171–180. doi: 10.1016/S0360-1315(96)00025-5.
- Jackson, D. and Usher, M. (1997) 'Grading student programs using ASSYST', in *ACM SIGCSE Bulletin*. ACM, pp. 335–339.
- James, R., McInnis, C. and Devlin, M. (2002) 'Assessing learning in Australian universities', *Melbourne: The University of Melbourne Centre for the Study of Higher Education*.
- John, C. (1992) *Programming in SQL with ORACLE, Ingres and dBase IV*. 1e edn. McGraw-Hill Education - Europe.
- Joy, M. S., Chan, P.-S. and Luck, M. (2000) 'Networked submission and assessment'.
- Kakkonen, T., Myller, N. and Sutinen, E. (2004) 'Semi-Automatic Evaluation Features in Computer-assisted Essay Assessment.', in *Cate*, pp. 456–461.
- Karavirta, V., Korhonen, A. and Malmi, L. (2007) 'On the use of resubmissions in automatic assessment systems', *Computer Science Education*. Routledge, 16(3), pp. 229–240. doi: 10.1080/08993400600912426.
- Ke, H., Zhang, G. and Yan, H. (2009) 'Automatic grading system on sql programming', in *Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009. SCALCOM-EMBEDDED COM'09. International Conference on. IEEE*, pp. 537–540. doi: 10.1109/EmbeddedCom-ScalCom.2009.105.
- Kearns, R., Shead, S. and Fekete, A. (1997) 'A teaching system for SQL', in *Proceedings of the 2nd Australasian conference on Computer science education*. ACM, pp. 224–231.

- Kenny, C. and Pahl, C. (2005) 'Automated tutoring for a database skills training environment', in *ACM SIGCSE Bulletin*. ACM, pp. 58–62.
- Kleerekoper, A. and Schofield, A. (2018) 'SQL tester: an online SQL assessment tool and its impact', in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. New York, New York, USA: ACM Press, pp. 87–92. doi: 10.1145/3197091.3197124.
- Kleiner, C., Tebbe, C. and Heine, F. (2013) 'Automated grading and tutoring of SQL statements to improve student learning', in *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. ACM, pp. 161–168.
- Koenings, M. *et al.* (2015) 'From Paper and Pencil to Tablet and Stylus: Automating the Sneakers and Spokes Environmental Audit', *The FASEB Journal*. Federation of American Societies for Experimental Biology, 29(1 Supplement), p. 740.7.
- Kolodner, J. (2014) *Case-Based Reasoning*. Elsevier Science.
- Lans, R. F. (2007) *Introduction to SQL: Mastering the Relational Database Language*. 4th edn. Addison-Wesley.
- Li, C. C. (1964) *Introduction to experimental statistics*. McGraw-Hill (Probability and Statistics Series).
- Litoriya, R. and Ranjan, A. (2010) 'Implementation of Relational Algebra Interpreter Using Another Query Language', in *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, pp. 24–28.
- López-Pastor, V. and Sicilia-Camacho, A. (2017) 'Formative and shared assessment in higher education. Lessons learned and challenges for the future', *Assessment and Evaluation in Higher Education*, 42(1), pp. 77–97. doi: 10.1080/02602938.2015.1083535.
- Luck, M. and Joy, M. (1999) 'A secure on-line submission system', *Software-Practice and Experience*. London, New York, Wiley Interscience [etc.], 29(8), pp. 721–740.
- Luger, G. and Stubblefield, W. A. (1998) 'Artificial Intelligence: Strategies and Structures for Complex Problem Solving'. Reading, MA: Addison-Wesley.
- MacKenzie, I. S. (1992) 'Fitts' law as a research and design tool in human-computer interaction', 7, pp. 91–139.
- McDonald, A. S. (2002) 'The impact of individual differences on the equivalence of computer-based and paper-and-pencil educational assessments', *Computers & Education*, 39(3), pp. 299–312. doi: 10.1016/S0360-1315(02)00032-5.
- McQuain, W. (2003) 'Curator: An electronic submission management environment', *Web page last accessed July*, 24, p. 2003.
- Melton, J. (1993) *Understanding the new SQL: a complete guide*. Morgan Kaufmann.

- Mitrovic, A. (1998) 'Learning SQL with a computerized tutor', in *ACM SIGCSE Bulletin*. ACM, pp. 307–311.
- Mitrovic, A. (2003) 'An intelligent SQL tutor on the web', *International Journal of Artificial Intelligence in Education*. IOS Press, 13(2), pp. 173–197.
- Moran, M., Hawkes, M. and El Gayar, O. (2010) 'Tablet Personal Computer Integration in Higher Education: Applying the Unified Theory of Acceptance and Use Technology Model to Understand Supporting Factors', *Journal of Educational Computing Research*. SAGE Publications, 42(1), pp. 79–101. doi: 10.2190/EC.42.1.d.
- Murray, O. T. and Olcese, N. R. (2011) 'Teaching and Learning with iPads, Ready or Not?', *TechTrends*. Springer US, 55(6), pp. 42–48. doi: 10.1007/s11528-011-0540-6.
- National Institute of Education (1997) *ERIC Resources in Education*. D.H.E.W., National Institute of Education. Available at: <http://books.google.co.uk/books?id=BEJwNJdHIs0C>.
- Noonan, R. E. (2006) 'The back end of a grading system', *ACM SIGCSE Bulletin*. ACM, 38(1), p. 56. doi: 10.1145/1124706.1121360.
- O'Reilly, M. and Morgan, C. (1999) 'Online assessment: creating communities and opportunities'. Kogan Page, SEDA.
- Pardo, A. (2002) 'A multi-agent platform for automatic assignment management', in *Proceedings of the 7th annual conference on Innovation and technology in computer science education - ITiCSE'02*. New York, New York, USA: ACM Press, p. 60. doi: 10.1145/544414.544434.
- Patel, J. (2012) *SQL PL/SQL Programming*. eBookIt. com.
- Peat, M. and Franklin, S. (2002) 'Use of online and offline formative and summative assessment opportunities: have they had any impact on student learning?', in *ASCILITE*, pp. 505–513.
- Pieterse, V. (2013) 'Automated Assessment of Programming Assignments', *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*. Open Universiteit, Heerlen, pp. 45–56. Available at: <http://dl.acm.org/citation.cfm?id=2541917.2541921> (Accessed: 10 May 2015).
- Pinckok, N. and Brandt, W. C. (2009) 'Connecting Formative Assessment RESEARCH to PRACTICE An Introductory Guide for Educators', p. 22. Available at: <http://files.eric.ed.gov/fulltext/ED509943.pdf>.
- Pribela, I. *et al.* (2014) 'Tool for Testing Bad Student Programs'.
- Prior, J. C. (2003) *Online Assessment of SQL Query Formulation Skills, Proceedings of the fifth Australasian conference on Computing education-Volume 20*. Australian Computer Society, Inc. Available at: <http://dl.acm.org/citation.cfm?id=858403.858433> (Accessed: 10 March 2015).

- Prior, J. C. and Lister, R. (2004) 'The backwash effect on SQL skills grading', *ACM SIGCSE Bulletin*, 36(3), pp. 32–36.
- Quinn, G. P. and Keough, M. J. (2002) *Experimental Design and Data Analysis for Biologists*. Cambridge University Press.
- Raadt, M. De, Dekeyser, S. and Lee, T. Y. (2006) 'Do students SQLify? improving learning outcomes with peer review and enhanced computer assisted assessment of querying skills', in *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*. ACM, pp. 101–108.
- Raadt, M. De, Dekeyser, S. and Lee, T. Y. (2007) 'A system employing peer review and enhanced computer assisted assessment of querying skills', *Informatics in education*, 6(1), pp. 163–178.
- Rawles, S., Joy, M. S. and Evans, M. (2002) 'Computer-assisted assessment in computer science: issues and software'. Department of Computer Science.
- Renaud, K. and van Biljon, J. (2004) 'Teaching SQL—Which Pedagogical Horse for This Course?', in Springer (Key Technologies for Data Management), pp. 244–256.
- Riesbeck, C. K. and Schank, R. C. (1989) *Inside case-based reasoning*. Lawrence Erlbaum Associates, Pubs. Available at: <https://dl.acm.org/citation.cfm?id=575859>.
- Rob, P., Coronel, C. and Crockett, K. (2008) *Database systems: design, implementation & management*. Cengage Learning (Course technology). Available at: <https://books.google.co.uk/books?id=628ArHnYOR8C>.
- Romli, R., Sulaiman, S. and Zamli, K. Z. (2010) 'Automatic programming assessment and test data generation a review on its approaches', in *2010 International Symposium on Information Technology*. IEEE, pp. 1186–1192. doi: 10.1109/ITSIM.2010.5561488.
- Rowley, J. (2014) 'Designing and using research questionnaires', *Management Research Review*, 37(3), pp. 308–330. doi: 10.1108/MRR-02-2013-0027.
- Rowntree, D. (1987) *Assessing students: How shall we know them?* Taylor & Francis.
- Russell, G. and Cumming, A. (2004) 'Improving the student learning experience for SQL using automatic marking.', in *CELDA*, pp. 281–288.
- Sadiq, S. *et al.* (2004) 'SQLator: an online SQL learning workbench', in *ACM SIGCSE Bulletin*. Leeds, United Kingdom: ACM, pp. 223–227.
- Sadler, D. R. (1998) 'Formative assessment: Revisiting the territory', *Assessment in education*. Taylor & Francis, 5(1), pp. 77–84.
- Saikkonen, R., Malmi, L. and Korhonen, A. (2001) 'Fully automatic assessment of programming exercises', *ACM SIGCSE Bulletin*, 33(3), pp. 133–136. doi: 10.1145/507758.377666.

Sasikumar, M. *et al.* (2007) 'A Practical Introduction to Rule Based Expert Systems', (October), pp. 1–294.

Sclater, N. and Howie, K. (2003) 'User requirements of the "ultimate" online assessment engine', *Computers & Education*, 40(3), pp. 285–306. doi: 10.1016/S0360-1315(02)00132-X.

Shah, P. and Oza, R. (2018) 'Improved parallel Rabin-Karp algorithm using compute unified device architecture', in *Smart Innovation, Systems and Technologies*, pp. 236–244. doi: 10.1007/978-3-319-63645-0_26.

Slevitch, L. (2011) 'Qualitative and Quantitative Methodologies Compared: Ontological and Epistemological Perspectives', *Journal of Quality Assurance in Hospitality & Tourism*, 12(1), pp. 73–81. doi: 10.1080/1528008X.2011.541810.

Stephens, D., Bull, J. and Wade, W. (1998) 'Computer-assisted Assessment: suggested guidelines for an institutional strategy', *Assessment & Evaluation in Higher Education*. Routledge, 23(3), pp. 283–294. doi: 10.1080/0260293980230305.

Taras, M. (2001) 'The Use of Tutor Feedback and Student Self-assessment in Summative Assessment Tasks: Towards transparency for students and for tutors', *Assessment & Evaluation in Higher Education*. Routledge, 26(6), pp. 605–614. doi: 10.1080/02602930120093922.

Taras, M. (2005) 'Assessment–summative and formative–some theoretical reflections', *British Journal of Educational Studies*, 53(4), pp. 466–478.

The Council of Chief State School Officers (2008) 'A TRIBUTES OF E FFECTIVE F ORMATIVE A SSESSMENT A WORK PRODUCT COORDINATED I BY S ARAH M C M ANUS', pp. 1–6.

Thompson, M. K. and Ahn, B. (2012) 'The Development of an Online Grading System for Distributed Grading in a Large First Year Project-Based Design Course', in *119th ASEE Annual Conference & Exposition*.

Tremblay, G. and Labonté, É. (2003) 'Semi-automatic marking of java programs using junit', in *International Conference on Education and Information Systems: Technologies and Applications (EISTA'03)*, pp. 42–47.

Tropashko, V. and Burleson, D. (2007) *SQL Design Patterns: Expert Guide to SQL Programming*. Rampant Techpress.

Tselonis, C., Sargeant, J. and Wood, M. M. (2005) 'Diagram matching for human-computer collaborative assessment'.

Tshibalo, A. E. (2007) 'The potential impact of computer-aided assessment technology in higher education', *South African Journal of Higher Education: NADEOSA 2006: Special Edition 6*. Sabinet Online, 21, pp. 684–693.

University Northern Illinois (2004) *Formative and Summative Assessment*. Available at: https://www.azwestern.edu/academic_services/instruction/assessment/resources/downloads/formative_and_summative_assessment.pdf.

Walker, K. (2011) 'Immediate Feedback to Students and Student Learning.', *Education Partnerships, Inc.*

Weinberger, A. (2011) *Semi-Automatic Essay Assessment based on a flexible Rubric*
Computer-unterstützte Aufsatz Beurteilung basierend auf flexiblen Rubriken.

Woit, D. and Mason, D. (2003) 'Effectiveness of online assessment', *ACM SIGCSE Bulletin*.
ACM, 35(1), p. 137. doi: 10.1145/792548.611952.

Wong, K. Y. *et al.* (2012) 'Linking IT-based semi-automatic marking of student mathematics responses and meaningful feedback to pedagogical objectives', *Teaching Mathematics and its Applications*, 31(1), pp. 57–63. doi: 10.1093/teamat/hrr023.

Appendix 1: Conference Paper

A Web-based Semi-Automatic Assessment Tool for Formulating Basic SQL Statements: Point-and-Click Interaction Method

Aisha AL-Salmi

Department of Computer Science, Loughborough University, Loughborough, U.K.

Keywords: Semi-Automated, Learning, Assessment, Online, SQL.

Abstract: Learning the Structured Query Language (SQL) is an important step towards developing students' database skills. As such, the number of higher education students learning SQL is constantly increasing. In this context, most researches focus on marking and providing feedback on the final query output rather than on the formulation of the SQL statement clauses. Focusing on statements formulation can assist the examiners in diagnosing the strengths and weaknesses of students' answers and provide detailed feedback on SQL statements that have been submitted for marking. This paper proposes a new semi-automatic assessment tool called SQL Formulation Editor (SQL-FE) for higher levels of education. The tool allows students to formulate SQL statements using point-and-click interaction method. To ensure the effectiveness of the method; the research has conducted an experiment which compares SQL-FE with the SQL Management Studio (SSMS) tool. The results have provided reasonable evidence that using SQL-FE can have a beneficial effect on formulating SQL query on-time and demonstrated a significant improvement in students' performance.

1 INTRODUCTION

Paper-based assessment has shown a number of problems due its manual nature, especially when greater number of students are enrolled in one class (Carter et al., 2003). Manual assessment might affect examiners' time management as the marking workload is increased, therefore forcing the examiners to either set their students less assessment tasks (e.g. mid-terms, quizzes and assignments) or add additional marking time to their schedules (Carter et al., 2003). In addition, large class sizes, limited time for marking assessments and non-effective feedback have led educators to consider computerised assessments. Automated assessment is becoming more useful for both students and staff since computer networking technology can support teaching and learning in higher education. Peat and Franklin (2002) stated that online assessment has become more popular and supporting the improvement of teaching and learning. A study conducted by Woit and Mason (2003) shows that automated assessment may improve students' motivation and programming efficiencies when it is implemented securely and efficiently. In addition, online assessment provides students with appropriate

feedback that can help them enhance their learning progress (Thantola et al., 2010).

1.1 SQL Manual Assessment

The Structured Query Language (SQL) is a database language for querying and manipulating relational databases (Bobak, 1996). It is one of the essential topics in database modules taught in higher education. Formulating and executing SQL queries is an essential part of relational database courses. However, manual SQL formulation poses a great challenge for both examiners and students. A research by Renaud and van Biljon (2004) states that the difficulties of solving SQL questions are "...due to the nature of SQL, and the fact that it is fundamentally different from the other skills students master during their course of study".

1.2 Case Study

To confirm the challenges of manual SQL assessments, several SQL statements were retrieved from 150 exam scripts of two years (2013 and 2014). This data was collected from the Database module taught to undergraduate students at Loughborough University. Each question on the exam script was

191

analysed individually to find common errors as well as the number of students who made the same error. After analysing all the SQL script answers, there were multiple common errors in SQL statements attempted in manual SQL assessments. This research initially categorised the students' common errors as synonyms, syntax errors, incorrect keywords/functions and incomplete SQL statements. Table 1 illustrates several common errors made by students and their descriptions.

Table 1: Examples of common errors made by students in the Database exam of June 2013.

Question	Model Answer	Students' Answer	Common Error Description
Display the department number and total salary of employees in each department that employ five or more people.	<pre>SELECT DEPTNO, SUM(SALARY) FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO) >= 5;</pre>	<pre>SELECT DEPTNO, SUM(SALARY) FROM EMP GROUP DE DEPTNO HAVING COUNT (DEPTNO) >= 5;</pre>	a) Use where instead of having clause
		<pre>SELECT DEPTNO, * SALARY FROM EMP GROUP DE DEPTNO HAVING COUNT (DEPTNO) >= 5;</pre>	b) Wrong SQL function (SUM)
		<pre>SELECT DEPTNO, TOTAL(SALARY) FROM EMP GROUP DE DEPTNO HAVING COUNT (DEPTNO) >= 5;</pre>	c) Use Total instead of SUM
		<pre>SELECT DEPTNO, COUNT(SALARY) FROM EMP GROUP DE DEPTNO HAVING COUNT(SALARY) >= 5;</pre>	d) Use COUNT instead of SUM

The number of common errors made by the students in both years suggests that students might have found understanding the queries a challenge, because most of them made the same errors. In common error "a", many students tried to solve the first question using the "WHERE" clause instead of the "HAVING" clause, when there cannot be an aggregate function in a WHERE clause.

In common error "b", students attempted the query; however, they failed to add an important component of an SQL query into their solution – namely the "SUM" function. Common error "c" shows that some students could understand the requirement of the query, that is, that they needed to use a function. However, they used "TOTAL" instead of "SUM", which causes errors in the query. The last common error, "d", shows another way of changing the keyword, where students attempted the query using "COUNT" instead of the SUM function. As is clear from Table 1, the last three common errors are based on functions, which indicate that students might have had some confusion or lack of awareness of functions and their usage.

Therefore, one can conclude that manual assessment leads to a less efficient learning process and creates difficulty in assessing students' work;

whereas automated assessment can achieve an improvement in learning and teaching processes, since it can encourage interactions between examiners and students and enhance the marking after submission.

This paper addresses the problems of manual formulation of SQL statements. It discusses the point-and-click method that aims to minimise or remove trivial errors of SQL statements. Furthermore, it describes an experiment that was conducted using the new implemented SQL formulation editor (SQL-FE) with an existing SQL tool and highlights its impact on time efficiency and students' performance.

2 METHOD

The point-and-click interaction method can be used with different input devices; for instance, computer mouse devices, touch pads, and touch screens. However, there are two questions to identify the selection of the point-and-click method, which are:

- Why has this tool been chosen to use the point-and-click interaction method rather than the drag-and-drop interaction method or typing using the keyboard?
- Does using this method lead to enhancing the performance of students in SQL assessment exercises?

Several researchers have examined the differences in speed and accuracy between the two methods — point-and-click and drag-and-drop — in various tasks (Boritz et al. 1991; Gillan et al. 1990; MacKenzie, 1992). However, the decision to select either drag-and-drop or point-and-click depends mostly on the task to be completed. For example, Adesina et al. (2013) used multi-touch drag-and-drop method to solve basic arithmetic problems. Such a method allows the student to drag numbers from the problem and drop them in the solution pad; then by using multiple gestures, the mathematical operation can be computed using the arithmetic operators. The study demonstrated improvements in the students' performance when solving mathematical problems and gave more functionality to the learning process. This means that the editor restricts students from writing SQL statements using the keyboard to avoid any trivial errors such as spelling errors, unnecessary words and synonyms. Furthermore, as it works based on the point-and-click interaction method, it is compatible with different touch screen technology devices (e.g. tablets). These technologies have improved the effectiveness of students' performance in various educational aspects (Bonastre et al., 2006;

Murray & Olcese, 2011; Moran et al., 2010; Adesina et al., 2015). As such, this means that students might find it easier to touch the screen and complete the syntax using tablet devices.

3 DESIGN OF SQL-FE

The new SQL formulation editor (SQL-FE) is critical to supporting students and improving their performance. It was designed to provide an effective avenue for testing students' SQL statements, as well as to provide quick feedback responses after marking students' SQL statements using the automated system. Figure 1 shows the use case diagram which displays the core functionalities of the SQL-FE tool. The use case identifies the primary actors (users) of the SQL-FE tool along with the key use cases. Two types of actors use the tool: examiners and students.

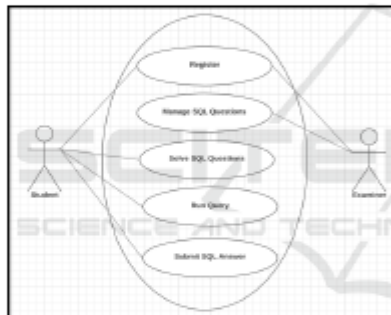


Figure 1: Use case diagram of the SQL-FE tool.

In order to enforce proper security, each actor must first register into the editor before he/she can use any of the other functionalities. Registration ensures that a proper email address and password are created for each new user. The two actors —examiners and students — will have different functionalities using the editor. The first step for the examiner is to handle a given SQL assignment by creating and managing the SQL questions.

The examiner will then assign SQL answers for each question, providing alternative ways of solving

the same question when applicable. Once the student logs in to the editor, the time count will start automatically for each submitted SQL answer. The student will then solve the SQL questions and try to run them before submitting them for marking.

4 IMPLEMENTATION

SQL-FE has been developed to enable students to formulate SQL statements, execute or run the queries and submit the statements for marking.

4.1 Components

The SQL-FE user interface contains eight main components illustrated in Figure 2. Component (A) represents the question pane that shows the SQL question scenario and identifies the query requirements needed to solve the SQL statements. The SQL question is placed in the same SQL-FE web page making it more convenient for students to solve the SQL statements. In addition, it saves on printed paper normally used for listing the SQL questions manually. Component (B) of the interface consists of the left navigation bar which is composed of two main parts, basic "SELECT" clauses and functions. The clauses list assists students while solving the SQL statements. In addition, functions have been added for performing calculations on data. These clauses and functions are placed on the left side of the interface where students can easily find and access them to solve the queries. Element (C) of the interface represents the right navigation bar which consists of reserved SQL keywords used for defining, manipulating and accessing a database. Furthermore, it contains a set of operators used in the "WHERE" clause to perform operations such as comparisons and arithmetic functions. Separating the navigation bars into left and right panels provides more vertical space for the main contents such as the SQL questions and the SQL statement answer bars. Component (D) of the interface represents the table schema that displays the table name, field names and their datatype to be used while solving the SQL questions. This means that there is no need for printed paper to display the table schema to the student, as it is ready to view on the web page.



Figure 2: Description of SQL-FE user interface.

Component (E) is the SQL answer pane used to enter the SQL answer using the left and right navigation bars. Component (F) represents the text-area pane that helps students to add different numeric or string values to limit the data retrieved, which cannot be done by using the available navigation bars. Component (G) consists of the control buttons which are divided into two categories; one is used to make any amendment in the SQL statements, for example, to redo, undo or reset the SQL statements. The second control buttons are used to deal with the functionality of the SQL statements and include the "Run Query" button which shows the SQL result output (indicated by letter (H)) and the "Submit" button which saves students' SQL answers for marking.

4.2 SQL-FE User Interface

Figure 2 depicts the SQL-FE user interface where an SQL answer has been attempted using a point-and-click interaction technique. The figure shows there are four steps to complete an SQL answer using the SQL-FE tool. Firstly, the student reads the SQL question and understands the requirements needed to write the SQL statements. Secondly, the student starts pointing and clicking on the SQL clauses and navigation bars to formulate the SQL statement (as illustrated in B, C and D). Thirdly, the student clicks on the "String" button to retrieve string "Female" value as the question requested and then clicks on the "Confirm" button to insert the string into the SQL statement answer (as illustrated in F and G). The last step is to provide the student with the ability to check the correctness of their SQL

statement syntax and query output by clicking on the "Run Query" button (as illustrated in G and H).

5 EXPERIMENT

The main objectives of the experiment were to measure the mean time spent and students' performance (grades) by comparing two query formulation tools, SQL-FE and SQL Management Studio tool (SSMS) illustrated in Figure 3.

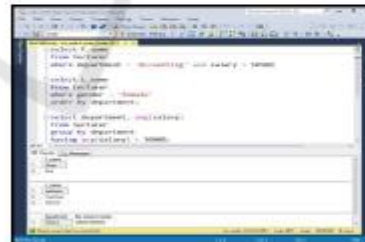


Figure 3: SQL Management Studio (SSMS) user interface.

The SSMS enables users to enter and execute SQL statements to perform calculations, store and retrieve query results.

It shows the execution of SQL statements submitted by participating students, which can run one statement at a time or several statements simultaneously. In order to provide a better understanding of the effect of using SQL-FE over

SSMS, the research has identified two questions for the query formulation experiment, which are:

RQ1: Does using SQL-FE during the experiment lead to spending more or less time on solving SQL questions? To investigate if students were spending more or less time to complete the SQL questions.

RQ2: Does using SQL-FE enhance students' performance (i.e. grades)? To investigate if students using SQL-FE were achieving higher marks in solving SQL questions than solving them using SQL formulation tools.

5.1 The Experimental Design

A crossover design (also called "change-over design") study is a special form of a controlled double randomised trial (Gardiner and Gettinby, 1998). The randomised nature of the study means that every student has an equal chance of being assigned to the experimental subject on a random basis. This design is more efficient in establishing the highest possible similarity among SQL questions exposed to different tools (Li, 1964). To attain the purpose of the study, a cross-over experimental design has been employed. Table 2 provides a full description of the cross-over experimental design implemented over a two-week time period. In one week, two different sessions took place.

Table 2: Cross-over experimental design.

Group	Tool	Quizzes SET	No. of Participants	Period	Session No.
X	SQL-FE	SET A	15	Week I	Session 1.1
Y	SSMS	SET B	15		
W	SQL-FE	SET A	15	Week I	Session 2.1
Z	SSMS	SET B	15		
X	SSMS	SET A	15	Week II	Session 1.2
Y	SQL-FE	SET B	15		
W	SSMS	SET A	15	Week II	Session 2.2
Z	SQL-FE	SET B	15		

5.1.1 Participants

The experiment involved a total of 60 college undergraduate students in the 20 and 21 years age group.

They were divided into two different experiment days, as each experiment involved 30 students and the number of available PCs in each computer lab was limited as illustrated in Table 2. The students were randomly assigned into two groups. An equal distribution of 15 students used SQL-FE and another 15 students used the SSMS tool. This means that there was one experiment in session 1.1 involving 30 students, with 15 students using SQL-FE and 15 students using SSMS. Then, a week after, session 1.2 was held and the participants swapped order. The same process was repeated in session 2.1 and 2.2, with a total of 30 students taking part over a two-week time period.

5.1.2 SQL Questions and Model Answers

Each tool used in the experiment was attached to a certain set of questions and alternative methods of solving the queries, as illustrated in Table 3 and Table 4. Basic SQL SELECT clauses were included in the experiment such as SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY.

Table 3: SQL questions with model answer: SET A.

Question 1	Find the first names of all lecturers who work in the secondary department with salaries greater than 50000.
Model Answer 1	<pre>SELECT F_NAME FROM EMPLOYEES WHERE DEPARTMENTID = 'Secondary' AND SALARY > 50000;</pre>
Output 1	
Question 2	Retrieve the first names and the salary title of all faculty members, sort the results in ascending order of the department.
Model Answer 2.1	<pre>SELECT F_NAME, S_TITLE, D_NAME FROM EMPLOYEES ORDER BY DEPARTMENTID;</pre>
Model Answer 2.2	<pre>SELECT F_NAME, S_TITLE, D_NAME FROM EMPLOYEES ORDER BY DEPARTMENTID;</pre>
Output 2	
Question 3	Find the department and average salary of lecturers at each department where the average salary is greater than 50000.
Model Answer 3	<pre>SELECT DEPARTMENTID, AVG(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENTID HAVING AVG(SALARY) > 50000;</pre>
Output 3	
Question 4	Find the title of all courses taught by lecturers in the tertiary department.
Model Answer 4	<pre>SELECT COURSE_TITLE FROM COURSES WHERE INSTRUCTOR_ID IN (SELECT EMPLOYEE_ID FROM EMPLOYEES WHERE DEPARTMENTID = 'Tertiary');</pre>
Output 4	
Question 5	Identify the department with the highest average salary.
Model Answer 5	<pre>SELECT DEPARTMENTID, AVG(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENTID ORDER BY AVERAGE(SALARY) DESC BY DEPARTMENTID;</pre>
Output 5	

Table 4: SQL questions with model answer: SET B.

Question	SQL	Model Answer
Question 1	List the first names of all employees who work in a department with a salary of more than 2000.	SELECT EMP_FIRST FROM EMP WHERE EMP_SAL > 2000 ORDER BY EMP_FIRST;
Question 2	Compare the first names and the department names of all female employees. Sort the result in ascending order of the last name.	SELECT EMP_LAST, DEPT_DEPARTMENT FROM EMP JOIN DEPT ON EMP_DEPTNO = DEPT_DEPTNO WHERE EMP_LAST <> EMP_FIRST ORDER BY EMP_LAST;
Question 3	Compare the names jobs and the average salary of employees in each job when the average salary is greater than 2000.	SELECT JOB_ID, AVG(EMP_SALARY) FROM EMP GROUP BY JOB_ID HAVING AVG(EMP_SALARY) > 2000;
Question 4	List all department names of all employees who work in a manager.	SELECT DEPT_DEPARTMENT FROM DEPT WHERE DEPT_DEPTNO IN (SELECT EMP_DEPTNO FROM EMP WHERE JOB_ID = 'MANAGER');
Question 5	Search the job with the lowest average salary.	SELECT EMP_JOB_ID, AVG(SALARY) FROM EMP GROUP BY EMP_JOB_ID ORDER BY AVG(SALARY) ASC LIMIT 1;

The descriptive statistics of the time taken to complete the test using the two tools of the experiment can be summarised as follows: the SQL-FE tool reported an average of $M = 20.4$ minutes ($SD = 7.8$) while SSMS reported an average of $M = 24.7$ minutes ($SD = 7.3$). As such, the SQL-FE tool reported less mean times to complete the test. Figure 4 depicts a box plot of the distribution of time taken to complete the test using each of the two tools. The box plot reports a difference in the distribution of the time taken. However, for both tools, it does not report any abnormal outlier observation indicating that the distribution does not report a large departure from normality, which is an assumption for the validity of the results of the *t*-test.

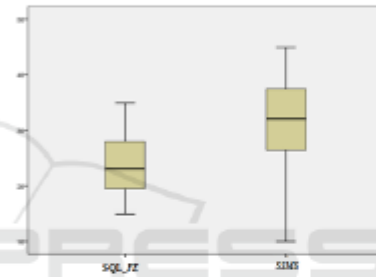


Figure 4: Box plot of the time taken to complete the test using each tool.

6 DATA COLLECTION

The data collected from both tools was saved and dated in different folders to be analysed and evaluated. Once the participants finished solving the SQL questions and made sure they were satisfied by their answers, they were asked to log off (if using SQL-FE) to save all their answers. In addition, the examiner and assistants created a shared folder to save all the created files retrieved from the SSMS tool. All participants were asked in the instructions to save the file with their college email address to keep it anonymous.

This is also supported by histograms of the distribution of time taken to complete the test (Figure 5 and Figure 6) using the SQL-FE tool and SSMS tool of test administration.

7 RESULTS AND DISCUSSIONS

The main objective of the evaluation was to measure the participants' performance when using the SQL-FE tool over the SSMS tool.

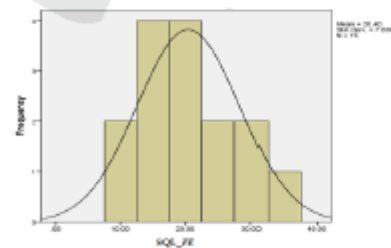


Figure 5: Histogram of the distribution of time taken to complete the test using the SQL-FE tool.

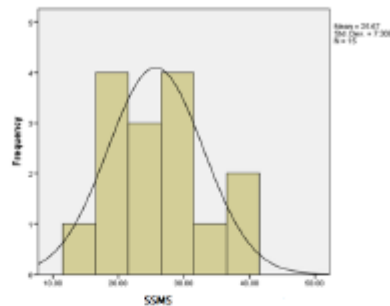


Figure 6: Histogram of the distribution of time taken to complete the test using the SSMS tool.

The SQL-FE tool reported less mean time compared to the SSMS tool of test administration. Results of the paired *t*-test indicate that the null hypothesis of no significant difference must be rejected at (0.05) level of significance. This indicates that there is a significant difference in the mean-time taken to complete the test or equivalently, that there is a significant difference in efficiency. Even for one-sided hypothesis ($H_1: \mu_{SQL-FE} < \mu_{SSMS}$), the results indicate a significant difference. These results clearly provide strong evidence for the statistical significance of difference (reduction) in the time taken to complete the test between the SQL-FE tool and SSMS tool of test administration. That is, the SQL-FE test reported significantly higher efficiency compared to the SSMS tool. The descriptive statistics of the mean performance marks using the two tools of the experiment can be summarised as follows: the SQL-FE tool reported an average of $M = 10.5$ marks ($SD = 3.1$) while SSMS reported an average of $M = 8.8$ marks ($SD = 3.7$). As such, SQL-FE reported higher marks to complete the test. The null hypothesis is rejected, since $p < 0.05$. In light of this, there is strong evidence ($t = 2.41$, $p = .030$) that formulating the SQL statements using SQL-FE improves the participants' marks. In this data set, it improved marks by an average of approximately 2 marks. If the experiment takes other samples of marks, it could get a 'mean paired difference' in marks different from 1.76. This is why it is important to look at the 95% Confidence Interval (95% CI). In this case, the 95% CI ranges from 0.2 to 3.3. This confirms that, although the difference in marks is statistically significant, it is actually relatively small.

Figure 7 presents a box plot of the distribution of performance marks obtained from completing the test using the two tools. The box plot reports a difference

in the distribution of performance marks which shows an increase in marks achieved using SQL-FE. For the SSMS tool, the figure depicts low marks since the participants had to write all SQL statements, which often led to making more errors.

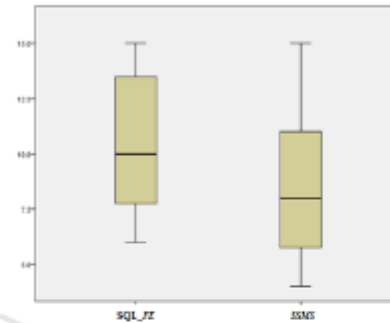


Figure 7: Boxplot of the performance marks obtained using the SQL-FE and SSMS tools.

These results clearly indicate that there is a significant difference in the mean performance marks obtained by completing the test using the two tools or equivalently, that there is a significant difference in the participants' performance after formulating the SQL statements using SQL-FE. That is the SQL-FE test reported significantly higher marks compared to the SSMS tool.

8 LIMITATIONS

There are two main limitations related to the newly implemented SQL-FE editor.

Firstly, the student is not allowed to use a keyboard. This made some students confused during their first use of the editor. However, restricting the use of the keyboard fulfils the main objective of avoiding unnecessary elements to the SQL statement syntax. The second drawback is the list of rows of table schema is small, restrictive and too difficult to view. This is because the design of the site has kept the table schema very limited, with lists of columns and data types only.

9 CONCLUSIONS

This paper has investigated the use of a point-and-click method to solve basic SQL statements. The experimental study has demonstrated that students were able to use the newly implemented SQL-FE tool.

Furthermore, the tool has minimised the unnecessary elements that students often add while formulating SQL statements. This resulted in removing the ambiguity in the SQL answers which should support the examiners in understanding the students' level of SQL learning and enable them to provide accurate feedback. The SQL-FE editor has answered the two questions of this experiment and confirmed that by using the newly implemented tool, less time is spent formulating SQL statements and students' performance improves, leading to fewer errors and higher grades.

The newly implemented editor has provided students with an easy method of solving SQL statements. However, it should be noted that the experimental study was conducted under two limitations, which can be solved to accommodate the students' requirements.

10 FUTURE WORK

Further implementations will take place utilising a semi-automated assessment of SQL statements to provide partial marking for the submitted statements from the SQL-FE tool. This would be considered as second stage of the research, which means the examiners' role will start once students submit their SQL answers, thus ensuring that the answers are ready for marking and commenting by examiners.

ACKNOWLEDGEMENTS


I would like to thank my sponsor the Ministry of Manpower, Sultanate of Oman for their continuing support and motivation.

REFERENCES

Adesina, A. et al., 2013. Use of multi-touch gestures for capturing solution steps in arithmetic word problems. , pp.6-8.
 Bobák, A.R., 1996. Distributed and Multi-Database Systems 2nd ed., Artech House,INC.

Bonastre, O., Beauvant, A. & Belmonte, F., 2006. Pedagogical Use of Tablet PC for Active and Collaborative Learning. In *2006 IEEE International Professional Communication Conference*. IEEE, pp. 214-218.
 Boritz, J., Booth, K.S. & Cowan, W.B., 1991. Fitts' Law Studies of Directional Mouse Movement. , pp.216-223.
 Carter, J. et al., 2003. How shall we assess this? In *ACM SIGCSE Bulletin*. ACM, pp. 107-123.
 Gardner, W.P. & Getzby, G., 1998. *Experimental Design Techniques in Statistical Practice: A Practical Software-Based Approach*, Elsevier Science.
 Gillan, D.J. et al., 1990. How does Fitts' law fit pointing and dragging? In *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. New York, New York, USA: ACM Press, pp. 227-234.
 Iisanto, P. et al., 2010. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. ACM, pp. 86-93.
 Li, C.C., 1964. *Introduction to experimental statistics*, McGraw-Hill.
 MacKenzie, I.S., 1992. Fitts' law as a research and design tool in human-computer interaction. , 7, pp.91-139.
 Moran, M., Hawkes, M. & El Gayar, O., 2010. Tablet Personal Computer Integration in Higher Education: Applying the Unified Theory of Acceptance and Use Technology Model to Understand Supporting Factors. *Journal of Educational Computing Research*, 42(1), pp.79-101.
 Murray, O.T. & Olcese, N.R., 2011. Teaching and Learning with iPads, Ready or Not? *TechTrends*, 55(6), pp.42-48.
 Peat, M. & Franklin, S., 2002. Use of online and offline formative and summative assessment opportunities: have they had any impact on student learning? In *ASCLITE*, pp. 505-513.
 Ransud, K. & van Biljon, J., 2004. Teaching SQL—Which Pedagogical Horse for This Course? In *Key Technologies for Data Management*. Springer, pp. 244-256.
 Woit, D. & Mason, D., 2003. Effectiveness of online assessment. *ACM SIGCSE Bulletin*, 35(1), p.137.

Appendix 2: Ethical Clearance Checklist (for student involving Human Participants)

<p>Ethics Approvals (Human Participants) Sub-Committee</p>	 <p>Loughborough University</p>		
<p>Ethical Clearance Checklist for studies involving Human Participants</p>			
<p>INVESTIGATOR 1 (APPLICANT)</p>			
<p><input type="checkbox"/> <input type="checkbox"/></p>			
Name	School/Organisation	Position	Email
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
<p>RESPONSIBLE INVESTIGATOR (IF DIFFERENT TO ABOVE)</p>		<p>NOTE: For undergraduate and postgraduate students this will be your project supervisor/tutor</p>	
Name	School/Organisation	Position	Email
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
<p>ADDITIONAL INVESTIGATORS</p>			
Name	School/Organisation	Position	Email
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
Click or tap here to enter text.	Click or tap here to enter text.	Choose an item. or Click or tap here to enter text.	Click or tap here to enter text.
List any further investigators: Click or tap here to enter text.			
<p>PROJECT DETAILS</p>			
Project Title: Click or tap here to enter text.			
Location(s) of Project: Click or tap here to enter text.			
		YES	NO
Does your research involve recruitment of NHS patients or staff or the use of NHS data, premises or equipment or recruitment of patients from the National Centre for Sport and Exercise Medicine (NCSEM)?		<input type="checkbox"/>	<input type="checkbox"/>
If YES , HRA/NHS approval may be required please contact the Secretary at J.A.Green@lboro.ac.uk to clarify. If approval has already been obtained please send details to the Secretary at J.A.Green@lboro.ac.uk .			
Ethical Clearance Checklist v1		1	

Please complete both Section A and Section B.

SECTION A

If you answer **YES** to any of the questions in Section A a full [Research Proposal](#) submission is required, (unless the study is covered by an existing Generic Protocol – see below*). Please attach this checklist to the completed Research Proposal Application.

		Please select	
		YES	NO
A1	Does your research involve participants who are knowingly recruited from vulnerable groups? For example, but not limited to, children under 18 years of age, pregnant women, prisoners/detained persons, persons lacking mental capacity to making an informed decision for themselves, adults in care homes, adults who are vulnerable because of their social, psychological or medical circumstances, other vulnerable group.	<input type="checkbox"/>	<input type="checkbox"/>
A2	Is your study being carried out overseas by investigators travelling to (or being present in) a country or area deemed to be high or very high risk by the insurers (UMAL) or the Foreign and Commonwealth Office?	<input type="checkbox"/>	<input type="checkbox"/>
A3	Does your research involve participants who are outside of the UK that will be exposed to increased physical, emotional or cultural risk because of taking part in your study?	<input type="checkbox"/>	<input type="checkbox"/>
A4	Does your research involve participants taking part without their written informed consent (or without parental consent for under 18 year olds)?	<input type="checkbox"/>	<input type="checkbox"/>
A5	Does your research involve intentional deception of participants?	<input type="checkbox"/>	<input type="checkbox"/>
A6	Does your research include the observation or recording of participants without their knowledge?	<input type="checkbox"/>	<input type="checkbox"/>
A7	Will it be necessary for participants to take part without their knowledge and consent at the time or without being informed of objectives of the study or the use of the data collected?	<input type="checkbox"/>	<input type="checkbox"/>
A8	Does the proposed study involve the administration of over-the-counter or prescription medicines or drugs, placebos or other substances (e.g. food substances, vitamins,) to the research participants? (Please refer to Guidance Note on Use of Pharmaceutical Drugs)	<input type="checkbox"/>	<input type="checkbox"/>
A9	Does the proposed study involve the testing of non-CE marked medical equipment/devices or a medical device which has been modified or is being used outside of its CE marked intended purpose?	<input type="checkbox"/>	<input type="checkbox"/>
A10	Does the proposed study involve intake of compounds additional to daily diet, or other dietary manipulation/supplementation or application of cosmetic products?	<input type="checkbox"/>	<input type="checkbox"/>
A11	Does the proposed study involve the collection of bodily samples from participants?	<input type="checkbox"/>	<input type="checkbox"/>
A12	Does the proposed study involve procedures which are physically invasive, e.g. the collection of bodily secretions by physically invasive methods?	<input type="checkbox"/>	<input type="checkbox"/>
A13	Is your research designed to be challenging physically or psychologically in any way (includes any study involving physical exercise/activity)?	<input type="checkbox"/>	<input type="checkbox"/>

A14	Does your research expose investigators to risks or distress greater than those encountered in their normal lifestyle?	<input type="checkbox"/>	<input type="checkbox"/>
A15	Does your research expose participants to risks or distress greater than those encountered in their normal lifestyle? For example, does it involve discussion of sensitive topics (e.g. sexual activity, drug use, illegal activities) or procedures which could cause physical, psychological, social or emotional distress to participants?	<input type="checkbox"/>	<input type="checkbox"/>
A16	Does the proposed study involve any process that would: <ul style="list-style-type: none"> - involve an MRI scan - affect contraception or assist/alter the process of conception? - involve the use of radiation? (Please refer to published guidelines and contact the University's Radiological Protection Officer before beginning any study which exposes participants to ionising radiation) - involve the use of hazardous materials? (Please refer to published guidelines on using hazardous materials) - involve genetic engineering? - involve analysis of DNA from bodily material or acellular material without consent. 	<input type="checkbox"/>	<input type="checkbox"/>
A17	Will your research involve the sharing or use of data or personal information, including transcripts and video/audio recording of participants, beyond the initial consent given?	<input type="checkbox"/>	<input type="checkbox"/>
A18	Will your research involve sharing participant's personal information with third parties?	<input type="checkbox"/>	<input type="checkbox"/>
A19	Will your research involve participants being identifiable in the resulting outcomes e.g. name included in published material or identifiable features recognisable in videos or pictures?	<input type="checkbox"/>	<input type="checkbox"/>

If you answer YES to any of the questions in Section A a full [Research Proposal](#) submission is required, (unless the study is covered by an existing generic protocol – see below). Please attach the completed checklist to the Research Proposal Application.

If you, or the relevant investigator, are listed as an investigator on an [existing Generic Protocol](#) which covers the study please give details and quote the generic protocol number below.

Click or tap here to enter text.

SECTION B

If you answer **YES** to any question in Section B (but none in Section A) please provide further details in the space provided below and explain how this will be addressed. Once signed by the School/Department the checklist should be submitted to the Secretary of the Sub-Committee for approval.

		Please select	
		YES	NO
B1	Does your research involve participants who are under the direct authority of investigators (e.g. academic staff using student participants, sports coaches using his/her athletes in training, teachers using their students)?	<input type="checkbox"/>	<input type="checkbox"/>

B2	Does your research involve any incentives, reimbursements or payments being offered to the participants ?	<input type="checkbox"/>	<input type="checkbox"/>
B3	Does your research involve any incentives, reimbursements or payments (additional to salary) being offered to the investigator(s) to conduct the study? Do investigators stand to gain from particular conclusions of the study?	<input type="checkbox"/>	<input type="checkbox"/>
B4	Does your proposed study involve testing of new non-medical equipment/products? <i>(excluding non-mechanical/non-electrical prototypes made from paper, cardboard, modelling clay or blue foam).</i>	<input type="checkbox"/>	<input type="checkbox"/>
B5	Is your research being conducted <u>without</u> a risk assessment being carried out, and approved by the School, to ensure the physical, emotional and cultural safety of the investigator and participants involved in the study?	<input type="checkbox"/>	<input type="checkbox"/>
B6	If your research involves working alone with participants or visiting them at home, will any of your procedures conflict with the guidance and recommendations given in the Guidance Note on Conducting interviews off campus and working alone .	<input type="checkbox"/>	<input type="checkbox"/>
B7	Will your research involve administrative or secure data that requires permission from the appropriate authorities before use?	<input type="checkbox"/>	<input type="checkbox"/>
B8	Will your research involve collecting personal data or sensitive personal data using assumed or opt-out consent (e.g. not using explicit written informed consent or parental consent for under 18 year olds)? See ICO guidance on personal data .	<input type="checkbox"/>	<input type="checkbox"/>
B9	Will storage of data and personal information conflict with Data Protection legislation or the Guidance Note on Data Collection and Storage ? See ICO guidance on Data Protection .	<input type="checkbox"/>	<input type="checkbox"/>
B10	Does your research involve the use of bodily samples previously collected with consent for further research?	<input type="checkbox"/>	<input type="checkbox"/>

If you answer **YES** to any question in Section B (but none in Section A) please provide further details in the space provided below and explain how this will be addressed. Once signed by the School/Department the checklist should be submitted to the Secretary of the Sub-Committee for approval at J.A.Green@lboro.ac.uk.

Click or tap here to enter text.

IF YOU ANSWER YES TO ANY OF THE QUESTIONS IN SECTION A: PLEASE ATTACH THE COMPLETED CHECKLIST TO YOUR FULL RESEARCH PROPOSAL SUBMISSION.

IF YOU ANSWER YES TO ANY QUESTION IN SECTION B (BUT NONE IN SECTION A): ONCE SIGNED BY THE SCHOOL/DEPARTMENT THE COMPLETED CHECKLIST, INCLUDING THE ADDITIONAL INFORMATION REQUESTED AND A COPY OF THE RISK ASSESSMENT, SHOULD BE SUBMITTED TO THE SECRETARY: J.A.GREEN@LBORO.AC.UK

IF YOU HAVE ANSWERED NO TO ALL QUESTIONS IN SECTION A AND SECTION B (*or your study is covered by a Generic Protocol); YOU SHOULD SUBMIT THE CHECKLIST FOR SCHOOL APPROVAL, PLEASE ATTACH A COPY OF THE RISK ASSESSMENT. ONCE SIGNED ON BEHALF OF THE SCHOOL/DEAN THE STUDY HAS ETHICAL APPROVAL.

INSURANCE

Cover is automatic if the research is within the UK & limited to the following activities:

- i. Questionnaires, interviews, focus groups, physical activity/exercise, psychological activity including CBT;
- ii. Venepuncture (withdrawal of blood);
- iii. Muscle biopsy;
- iv. Measurements or monitoring of physiological processes including scanning;
- v. Collections of body secretions by non invasive methods;
- vi. Intake of foods or nutrients or variation of diet (other than administration of drugs).

All other Research involving human participants, including studies outside of the UK, should be referred to the [Insurance Officer](#) along with the completed [Insurance Questionnaire](#) to arrange cover - which may incur a charge. Early submission is recommended. If you require further guidance please contact Insurance Support: insurance.support@lboro.ac.uk / 222026

DECLARATION

I confirm that I have read the [Code of Practice on Investigations Involving Human Participants](#) and have accurately completed this application. I confirm that the above investigation complies with published codes of conduct, ethical principles and guidelines of professional bodies associated with my research discipline.

Signature of Applicant: Click or tap here to enter text.

Signature of Supervisor (if applicable):Click or tap here to enter text.

Signature of Dean of School/Head of Department or his/her nominee: Click or tap here to enter text.

Date: Click or tap here to enter text.

Appendix 3: Permission to Conduct Research Study (SQL-ME Experiment)

Dear Sir/Madam

Director of Research Degree Programmes;

I am writing to request permission to conduct a research study at Loughborough University at Computer Science Department. I am currently enrolled as a researcher in the Computer Science Department at Loughborough University, UK. The study is entitled as Semi-Automatic Assessments of basic SQL Statements.

I hope that the department administration will allow me to recruit at least 6 individuals from the Computer Science Department to anonymously complete 4 pages questionnaire (Pdf file attached). Due to the nature of the study, I hope to recruit examiners qualified in teaching and assessing different modules of Database Program. They should at least have 3-5 years in teaching database so they can be able to provide an objective evaluation based on their experiences.

If approval is granted, examiner participants will complete the questionnaire after conducting the marking experiment using the SQL Marking Editor (SQL-ME) in their offices in their preferable timing. The questionnaire process should take no longer than 10 minutes from the experiment time. The questionnaire results will be combined for the thesis project and individual results of this study will remain absolutely confidential and anonymous. Should this study be published, only pooled results will be documented. No costs will be incurred by either your department or the individual participants.

Your approval to conduct this study will be greatly appreciated. I would be happy to answer any questions or concerns that you may have at that time. If you agree, kindly contact me at my email address: a.al-salmi@lboro.ac.uk

Sincerely,

Aisha AL Salmi

Computer Science Department, Loughborough University

Supervisors:

Professor. Eran Edirisinghe

Email: E.A.Edirisinghe@lboro.ac.uk

Dr. Shaheen Fatima

Email: S.S.Fatima@lboro.ac.uk

Appendix 4: Instructions and Rolls

A. Examiner Instructions

1. Open the following URL: <https://co-project.lboro.ac.uk/coaa/student/>
2. Login as: lct@gmail.com and password as: 1234
3. Click on the marking system on the left navigation bar.
4. From the list of three SQL question, select any question and start marking.
5. Write feedback for each clause for the student answer
6. Sign-out from the marking editor.
7. Fill up your questionnaire and submit them to the researcher.

B. Researcher Roll

1. Send email to the head of research programme, Loughborough University to get permission to conduct study with Database experts.
2. Set date and time with advisor for the experiment setup.
3. Prepare instructions for the participants to be read and understood before the experiment starts.
4. Distribute a list of three SQL questions along with SQL reference/model answers and alternative answer for each question.
5. Briefly explain the objectives of the research experiment and what they need to do while marking the SQL statements.
6. Give the proper time for the examiner to do the experiment and time to fill up the questionnaire.
7. Combine the data retrieved from the experiment and start the analysis and evaluation of the new implemented system.

Appendix 5: Common errors made by the students in June 2013 of Database exam scripts

QID	Question Description	Model Answer	Common Errors	Examples of Students' Errors	Common Error/78
1.	Display the department number and total salary of employees in each department that employs five or more people.	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO) >=5;	Use WHERE instead of HAVING.	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO WHERE COUNT (DEPTNO)>=5;	29
			Missing SQL function SUM()	SELECT DEPTNO, (SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO)>5;	10
			Use TOTAL instead of SUM	SELECT DEPTNO, TOTAL (SAL) FROM EMP GROUP BY DEPTNO HAVING (COUNT (DEPTNO)>=5);	4
			Use COUNT instead of SUM	SELECT DEPTNO, COUNT (SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(SAL)>=5;	3
2.	Display the name of each employee with his/her department name.	SELECT DEPTNAME, EMPNAME FROM DEPT INNER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO;	Use OUTER JOIN instead of INNER JOIN	SELECT EMPNAME, DEPTNAME FROM DEPT OUTER JOIN EMP WHERE EMP.DEPTNO =DEPT.DEPTNO;	13
			Didn't complete all SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMP, DEPT;	11
			Add GROUP BY on the statement	SELECT EMPNAME, DEPTNAME FROM EMP, DEPT GROUP BY DEPTNO;	7
3.	Display the names of all employees who work in a department that employs an analyst.	SELECT EMPNAME FROM EMP WHERE DEPTNO IN (SELECT DISTINCT DEPTNO FROM EMP WHERE JOB ='ANALYST');	Add GROUP BY on the statement	SELECT EMPNAME FROM EMP WHERE DEPNO= (SELECT DEPTNO FROM EMP WHERE JOB='ANALYST' GROUP BY DEPTNO);	20
			Didn't complete all SQL Syntax	SELECT EMPNAME FROM EMP WHERE JOB= 'ANALYST';	35

4.	Create a new empty table called EMP1. This table should have the same field names and types as the EMP table.	CREATE TABLE "EMP1" ("EMPNO" INTEGER, "EMPNAME" VARCHAR(15), "JOB" VARCHAR(15), "MGR" INTEGER, "HIREDATE" DATE, "SAL" INTEGER, "COMM" INTEGER, "DEPTNO" INTEGER, "JOBNO" INTEGER);	Create table without adding data type	CREATE TABLE EMP1 (EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE);	30
5.	Fill your new EMP1 table with the data from the EMP table.	INSERT INTO EMP1 SELECT EMP.* FROM EMP;	Use COPY rather than INSERT	COPY EMP INSERT EMP1 (EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE);	2
			Use UPDATE rather than INSERT	UPDATE TABLE EMP1 VALUES (SELECT * FROM EMP);	5
			Didn't complete all SQL Syntax	SELECT INTO EMP1 FROM EMP;	7
			Use ALTER rather than INSERT	ALTER TABLE EMP1 (UPDATE EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE WITH EMP);	1
			Didn't complete all SQL Syntax	INSERT EMP1 VALUE (SELECT * FROM EMP);	25
6.	Change the DEPT table so that the DEPTNO field is specified as the primary key.	ALTER TABLE DEPT ADD CONSTRAINT PKEY PRIMARY KEY (DEPTNO);	Using MODIFY instead of ALTER	MODIFY DEPTNO FROM DEPT;	6
			Using UPDATE instead of ALTER	UPDATE TABLE DEPT SET DEPTNO PRIMARY KEY;	22
7.	Configure the EMP1 table such that if a department is deleted from the DEPT table any associated employees are automatically deleted from the EMP1 table.	ALTER TABLE EMP1 ADD CONSTRAINT FKEY FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO) ON DELETE CASCADE;	Missing Constraint References for the Foreign Key	ALTER TABLE EMP ADD (DEPTNO) DEPT(DEPTNO) ON DELETE CASCADE;	24
			Using UPDATE instead of ALTER	UPDATE EMP1 (DEPTNO INTEGER REFERENCES DEPARTMENT(DEPTNO));	7
			Using DELETE instead of ALTER	DELETE CASCADE EMP1 INNER JOIN DEPT;	4
			Using DROP instead of ALTER	DROP EMP1 ON SELECT DEPT.DEPTNO FROM DEPT LEFT OUTER JOIN EMP ON DEPT.DEPTNO= EMP.DEPTNO IS NULL;	5

Appendix 6: Common errors made by the students in June 2014 of Database exam scripts

QID	Question Description	Model Answer	Common Errors Description	Examples of Students' Errors	Common Error/72
1.	List the names and hire dates of all employees in the order they were hired.	SELECT EMPNAME, HIREDATE FROM EMP ORDER BY HIREDATE;	Use Group by instead of ORER BY	SELECT EMPNAME, HIREDATE FROM EMP GROUP BY HIREDATE;	10
			Use sort by instead of ORDER BY	SELECT EMPNAME, HIREDATE FROM EMP SORT BY HIREDATEASC;	6
			Use list by instead of ORDER BY	SELECT EMPNAME, HIREDATE FROM EMP LIST BY HIREDATEASC;	2
			Didn't complete all SQL Syntax	SELECT EMPNAME FROM EMP;	8
2.	List the department number and total salary of employees in each department that employs four or more people.	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO) >= 4;	Use WHERE instead of GROUP BY and HAVING	SELECT DEPTNO, SUM(SAL) FROM EMP WHERE COUNT (EMPNO) >= 4;	27
			Missing SQL function Sum()	SELECT DEPTNO, SAL FROM EMP GROUP BY DEPTNO HAVING COUNT (DEPTNO) >3;	10
			Use Total instead of SUM	SELECT DEPTNO, TOTAL (SAL) FROM EMPGROUP BY DEPTNO HAVING COUNT (EMPNO)>=4;	4
			Didn't complete all SQL Syntax	SELECT DEPTNO,EMPNO, SUM(SAL) FROM EMP GROUP BY DEPTNO >=4;	19
3.	Give a list of ALL department names with the employees in each department.	SELECT DEPTNAME, EMPNAME FROM DEPT LEFT OUTER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO;	Add Group by on the statement	SELECT EMPNAME, DEPTNAME FROM EMP, DEPT GROUP BY DEPTNO;	15
			Forgot to add JOIN	SELECT DEPTNAME, EMPNO FROM DEPT, EMP WHERE EMP.DEPTNO= DEPT.DEPTNO;	22
			Didn't complete all SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMP WHERE DEPT.DEPTNO=EMP.DEPTNO;	30

4.	Produce a list the names of salesmen together with their department names. List only those salesmen that work in an existing department.	SELECT DEPTNAME, EMPNAME FROM EMPINNER JOIN DEPT ON EMP.EMPNO =DEPT.DEPTNO WHERE JOB="SALESMAN" AND DEPTNO IS NOT NULL;	Didn't complete all SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMP WHERE JOB='SALESMAN' AND;	28
				SELECT EMPNAME, DEPTNAME FROM EMP, DEPT WHERE EMP. DEPTNO = DEPT. DEPTNO;	18
5.	Create an empty new table called JOBS with two fields, an integer field called JOBNO and a 15-character text field called JOB.	CREATE TABLE JOBS (JOBNO INTEGER, JOB VARCHAR(15));	Use String or Text instead of varchar or char	CREATE TABLE JOBS (JOBNO INT, JOB TEXT (15));	7
			Use Update or Create instead of Create	UPDATE JOBS (JOBNO INTEGER (10), JOB VARCHAR (15));	2
			Didn't complete all SQL Syntax	CREATE TABLE (JOBNO, JOB)	12
6.	Fill your new JOBS table with null values for the JOBNO and the job values from the EMP table.	INSERT INTO JOBS(JOB) SELECT DISTINCT JOB FROM EMP;	Use Update or Create instead of Insert	UPDATE JOBS AS (SELECT JOB FROM EMP) WHERE JOBNO=NULL.	10
			Didn't complete all SQL Syntax	INSERT INTO JOBS (JOB) SELECT EMP(JOB)	40
7.	Create a view called BOSS which has the name and number of each employee with the name and number of his or her manager (with blanks alongside any employee that has no manager).	CREATE VIEW BOSS AS SELECT A.EMPNAME AS EMPLOYEEENAME, A.EMPNOAS EMPLOYEEENO, B.EMPNAME AS BNAME,B.EMPNO AS BOSSNO FROM EMPA LEFT OUTER JOIN EMPB ON A.MGR = B.EMPNO;	Use Insert instead of SELECT	CREATE VIEW BOSS INSERT EMPNAME, MGR FROM EMP, DEPTDEPTNO. DEPT=DEPTNO= EMP;	3
			Add GROUP BY on the statement	CREATE VIEW BOSS(SELECT EMPNAME, MGR, EMPNO FROM EMP GROUP BY EMPNAME);	4
			Didn't complete all SQL Syntax	CREATE VIEW BOSS AS A.EMPNAME, A.EMPNO, B.ENAME, B.EMPNO LEFT OUTER JOIN ON A.MGR=B.EMPNO;	30

Appendix 7: Different Model Answers for SQL Questions of 2013 Exam Scripts

QID	Question Description	Model Answers i. Model Answer ii. Instructor Model Answer iii. Student Model Answer	No. of Student Answered Correct	No. of Student Attempted of /78	% of Student Answered Correct	No. of Student Answered Incorrect	% of Student Answered Incorrect
1.	Display the department number and total salary of employees in each department that employs five or more people.	i. SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO) >=5;	41	78	$(41/78)*100 = 53\%$	$78-(41) = 37$	$(37/78)*100 = 47\%$
2.	Display the name of each employee with his/her department name.	ii. SELECT DEPTNAME, EMPNAME FROM DEPT INNER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO;	15	78	$(15+38/78)*100 = 68\%$	$78-(53) = 25$	$(25/78)*100 = 32\%$
		iii. SELECT EMPNAME, DEPTNAME FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;	38				
3.	Display the names of all employees who work in a department that employs an analyst.	i. SELECT EMPNAME FROM EMP WHERE DEPTNO IN (SELECT DISTINCT DEPTNO FROM EMP WHERE JOB ='ANALYST');	24	77	$(24+2/77)*100 = 34\%$	$77-(26) = 51$	$(51/77)*100 = 66\%$
		ii. SELECT DISTINCT E1.EMPNAME FROM EMPE1, EMPE2 WHERE E1.DEPTNO= E2.DEPTNO AND E2.JOB= 'ANALYST';	1				
		iii. SELECT EMPNAME FROM EMPA, EMPB WHERE A.DEPTNO=B.DEPTNO AND B.JOB LIKE 'ANALYST';	1				

4.	Create a new empty table called EMP1. This table should have the same field names and types as the EMP table.	i. CREATE TABLE "EMP1" ("EMPNO" INTEGER, "EMPNAME" VARCHAR(15), "JOB" VARCHAR(15), "MGR" INTEGER, "HIREDATE" DATE, "SAL" INTEGER, "COMM" INTEGER, "DEPTNO" INTEGER, "JOBNO" INTEGER);	48	75	$(48/75)*100 = 64\%$	75-48 =27	$(27/75)*100 = 36\%$
		ii. CREATE TABLE EMP1 AS SELECT EMPNO, EMPNAME, MGR, HIREDATE, SAL, COMM, DEPTNO, JOBNO FROM EMPLOYEE;	0				
5.	Fill your new EMP1 table with the data from the EMP table.	i. INSERT INTO EMP1 SELECT EMP.* FROM EMP;	22	70	$(22+8/70)*100 = 43\%$	70-(30) =40	$(40/70)*100 = 57\%$
		ii. INSERT INTO EMP1 AS (SELECT EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE FROM EMP);	8				
6.	Change the DEPT table so that the DEPTNO field is specified as the primary key.	i. ALTER TABLE DEPT ADD CONSTRAINT PKEY PRIMARY KEY (DEPTNO);	2	53	$(2/53)*100 = 4\%$	53-2 =51	$(51/53)*100 = 96\%$
7.	Configure the EMP1 table such that if a department is deleted from the DEPT table any associated employees are automatically deleted from the EMP1 table.	i. ALTER TABLE EMP1 ADD CONSTRAINT FKEY FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO) ON DELETE CASCADE;	1	33	$(1/33)*100 = 3\%$	33-1 =32	$(32/33)*100 = 97\%$

Appendix 8: Different Model Answers for SQL Questions of 2014 Exam Scripts

QID	Question Description	Model Answers i. Instructor Model Answer ii. Instructor Model Answer iii. Student Model Answer	No. of Student Answered Correct	No. of Student Attempted Out of/ 72	% of Student Answered Correct	No. of Student Answered Incorrect	% of Student Answered Incorrect
1.	List the names and hire dates of all employees in the order they were hired.	i. SELECT EMPNAME, HIREDATE FROM EMP ORDER BY HIREDATE;	48	72	$(48/72)*100 = 53\%$	72-48 = 37	$(37/72)*100 = 47\%$
2.	List the department number and total salary of employees in each department that employs four or more people.	i. SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO) >= 4;	31	72	$(31/72)*100 = 43\%$	72-31 = 41	$(41/72)*100 = 57\%$
3.	Give a list of ALL department names with the employees in each department.	i. SELECT DEPTNAME, EMPNAME FROM DEPT LEFT OUTER JOIN EMP ON DEPT.DEPT = EMP.DEPTNO;	8	71	$(8+3/71)*100 = 15\%$	71-11 = 60	$(60/71)*100 = 85\%$
		ii. SELECT DEPTNAME, EMPNAME FROM DEPT LEFT OUTER JOIN EMP WHERE DEPT.DEPT = EMP.DEPTNO;	3				
4.	Produce a list the names of salesmen together with their department names. List only those salesmen that work in an existing department.	i. SELECT DEPTNAME, EMPNAME FROM EMP INNER JOIN DEPT ON EMP.DEPTNO =DEPT.DEPTNO WHERE JOB="SALESMAN" AND DEPTNO IS NOT NULL;	5	70	$(5+23/70)*100 = 40\%$	70-28 = 42	$(42/70)*100 = 60\%$
		ii. SELECT EMPNAME, DEPTNAME FROM EMP, DEPT WHERE JOB='SALESMEN' AND EMP.DEPTNO=DEPT.DEPTNO;	23				
5.	Create an empty new table called JOBS with two fields, an integer field called JOBNO and a 15 character text field called JOB.	i. CREATE TABLE JOBS (JOBNO INTEGER, JOB VARCHAR(15));	53	70	$(53/70)*100 = 76\%$	70-53 = 17	$(17/70)*100 = 24\%$

6.	Fill your new JOBS table with null values for the JOBNO and the job values from the EMP table.	i. INSERT INTO JOBS(JOB) SELECT DISTINCT JOB FROM EMP;	11	67	$(11/67)*100$ = 16%	67-11 = 56	$(56/67)*100$ = 84%
7.	Create a view called BOSS which has the name and number of each employee with the name and number of his or her manager (with blanks alongside any employee that has no manager).	i. CREATE VIEW BOSS AS SELECT A.EMPNAME AS EMPNAME, A.EMPNO AS EMPNO, B.EMPNAME AS BNAME, B.EMPNO AS BOSSNO FROM EMPA LEFT OUTER JOIN EMPB ON A.MGR = B.EMPNO;	4	52	$(4+4/52)*100$ = 15%	52-8 = 44	$(44/52)*100$ = 85%
		ii. CREATE VIEW BOSS AS SELECT EMPNO, EMPNAME, JOB, MGR, HIREDATE, DEPTNAME FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;	4				

Appendix 9: Grouping the student error in 2013 under each error category

QID	No. of Student Answered Incorrect	Errors Category	Examples of Students' Common Errors	No. of Students Committed Errors	Total No. of Students Committed Errors
1.	37	Syntax Error	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO WHERE COUNT (DEPTNO)>=5;	28	46
		Incomplete SQL Syntax	SELECT DEPTNO, ? SAL FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO)>5;	11	
		Synonyms Error	SELECT DEPTNO, TOTAL (SAL) FROM EMP GROUP BY DEPTNO HAVING (COUNT (DEPTNO)>=5);	4	
		Incorrect Keyword/Function	SELECT DEPTNO, COUNT (SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(SAL)>=5;	3	
2.	25	Incomplete SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMPLOYEE, DEPARTMENT?	11	18
		Incorrect Keyword/Function	SELECT EMPNAME, DEPTNAME FROM EMP, DEPT GROUP BY DEPTNO;	7	
3.	51	Incorrect Keyword/Function	SELECT EMPNAME FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM EMP WHERE JOB='ANALYST' GROUP BY DEPTNO);	20	55
		Incomplete SQL Syntax	SELECT EMPNAME FROM EMP WHERE JOB= 'ANALYST';	35	
4.	27	Incomplete SQL Syntax	CREATE TABLE EMP1 (DEPTNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE)?;	30	30
5.	40	Synonyms Error	COPY EMPINSERT EMP1 (EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE);	2	40
		Incorrect Keyword/Function	UPDATE TABLE EMP1(EMP.EMPNO, EMP.EMPNAME, EMP.EMPNAME, EMP.JOB, EMP.SAL, EMP.DEPTNO, EMP.MGR, EMP.HIREDATE)	5	

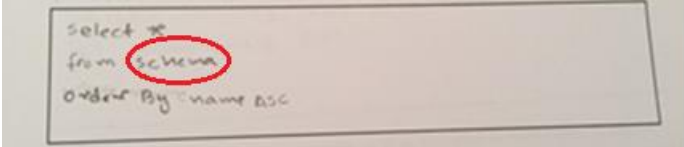
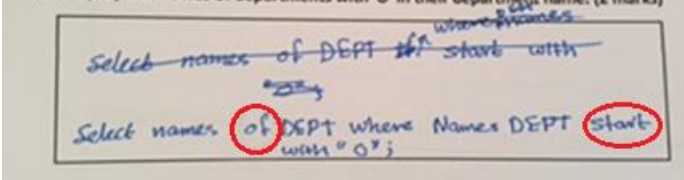
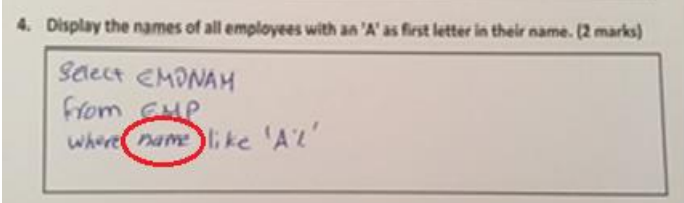
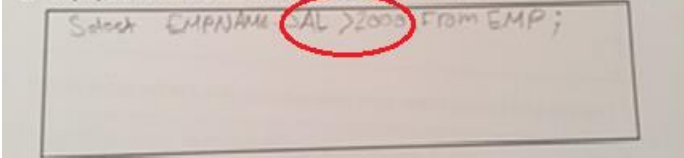
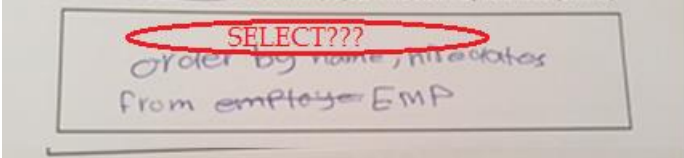
		Incomplete SQL Syntax	SELECT INTO EMP 1 FROM EMP;	7	
		Synonyms Error	ALTER TABLE EMP1(UPDATE EMPNO, EMPNAME, JOB, SAL, DEPTNO, MGR, HIREDATE WITH EMP);	1	
		Incomplete SQL Syntax	INSERT EMP1 VALUE (SELECT * FROM EMP)?;	25	
6.	51	Incorrect Keyword/Function	UPDATE TABLEDEPT SET DEPTNO PRIMARY KEY;	22	28
		Synonyms Error	MODIFY DEPTNO FROM DEPT;	6	
7.	32	Incorrect Keyword/Function	UPDATE EMP1(DEPTNO INTEGER REFERNES DEPT(DEPTNO)); UPDATE DEPT(DEPTNO DELETE CASCADE);	7	16
			DELETE CASCADE EMPLOYEE1 INNER JOIN DEPARTMENT;	4	
		Syntax Error	DROP EMP1 ON SELECT DEPT.DEPTNOFROM DEPT LEFT OUTER JOIN EMPON DEPT.DEPTNO =EMP.DEPTNOWHERE DEPT.DEPTNO IS NUIL;	5	

Appendix 10: Grouping the student error in 2014 under each error category

QID	No. of Student Answered Incorrect	Errors Category	Examples of Students' Common Errors	No. of Students Committed Errors	Total No. of Students Committed Errors
1.	37	Incorrect Keyword/Function	SELECT EMPNAME, HIREDATE FROM EMP GROUP BY HIREDATE;	10	26
		Synonyms Error	SELECT EMPNAME. HIREDATE FROM EMP SORT BY HIREDATE;	6	
			SELECT EMPNAME. HIREDATE FROM EMP LIST BY HIREDATE;	2	
		Incomplete SQL Syntax	SELECT EMPNAME FROM EMP?;	8	
2.	41	Syntax Error	SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO WHERE COUNT(EMPNO)>=4;	27	60
		Incomplete SQL Syntax	SELECT DEPTNO, SAL FROM EMP GROUP BY DEPTNO HAVING COUNT(DEPTNO)>3;	10	
		Synonyms Error	SELECT DEPTNO, TOTAL (SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO)>4;	4	
		Incomplete SQL Syntax	SELECT DEPTNO, EMPNO, SUM(SAL) FROM EMP GROUP BYDEPTNO =>4;	19	
3.	60	Incorrect Keyword/Function	SELECT EMPNAME, DEPTNAME FROM EMP, DEPT GROUP BY DEPTNO;	15	67
		Incomplete SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMP,DEPT?	22	
		Incorrect Keyword/Function	SELECT EMPNAME, DEPTNAME FROM EMP WHERE DEPT.DEPTNO= EMP. DEPTNO;	30	
4.	42	Incomplete SQL Syntax	SELECT EMPNAME, DEPTNAME FROM EMP WHERE JOB='SALESMAN' AND ;	28	46
5.	17	Incorrect Keyword/Function	UPDATE JOBS (JOBNO CHAR(10), JOB VARCHAR(15));	2	21

		Syntax Error	CREATE TABLE (JOBNO INTEGER, JOB TEXT (15));	7	
			CREATE TABLE (JOBNO, JOB)?	12	
6.	56	Incorrect Keyword/Function	UPDATE TABLE JOBSWHERE JOBNO=NULL AND JOBS.JOB=EMP.JOB FROM EMP;	10	50
		Incomplete SQL Syntax	INSERT INTO JOBS (JOB) SELECT EMP(JOB)?;	40	
7.	44	Syntax Error	CREATE VIEW BOSS INSERT EMPNAME, MGR FROM EMP, DEPTDEPTNO. DEPT=DEPTNO= EMP;	3	37
		Incorrect Keyword/Function	CREATE VIEW BOSS (SELECT EMPNAME, MGR, EMPNO FROM EMP GROUP BY EMPNAME);	4	
		Incomplete SQL Syntax	CREATE VIEW BOSS AS A.EMPNAME, A.EMPNO, B.ENAME, B.EMPNO LEFT OUTER JOIN ON A.MGR=B.EMPNO;	30	

Appendix 11: Student SQL errors in manual method

ENo	Error Description	Student Answer Samples
1.	Unwanted words e.g. Schema	<p>3. List the names and hire dates of all employees in the order they were hired. (2 marks)</p> 
2.	Reserved SQL Keywords e.g. OF, START and NAME	<p>1. Display the names of departments with 'O' in their department name. (2 marks)</p>  <p>4. Display the names of all employees with an 'A' as first letter in their name. (2 marks)</p> 
3.	Incorrect SQL syntax	<p>2. Display the names and salaries of all employees with a salary greater than 2000. (2 marks)</p> 
4.	Missing SQL command	<p>3. List the names and hire dates of all employees in the order they were hired. (2 marks)</p> 

Appendix 12: SQL-FE Experiment and Feedback details

A. Instruction for Examiner: Microsoft SQL Server Tool

1. Run the following SQL:

1.1. Create Faculty Table:

```
CREATE TABLE FACULTY (FAC_ID VARCHAR (15), FAC_NAME VARCHAR (20), BIRTH_DATE DATE, DEPARTMENT VARCHAR(20), GENDER VARCHAR(15), SALARY NUMBER(8,2), CONSTRAINT PK_FAC_ID PRIMARY KEY (FAC_ID));
```

Faculty Table:

FAC_ID	FAC_NAME	BIRTH_DATE	DEPARTMENT	GENDER	SALARY
D01	Amy Dancer	25-JUN-71	Computer Science	Female	34500
J01	Ray Johnson	05-OCT-70	Computer Science	Male	40000
S01	Wendy Swimmer	22-AUG-70	Computer Science	Female	45000
J02	Bob Jones		Accounting	Male	35000
N01	Jack Nelson	10-JAN-71	History	Male	28000
D02	Jinee Jackson		Accounting	Female	34500
S02	William James	11-NOV-67	Accounting	Male	30500

1.2. Insert rows into Faculty table:

```
INSERT INTO FACULTY VALUES('D01', 'Amy Dancer', '25-JUN-71', 'Computer Science', 'Female', 34500);
INSERT INTO FACULTY VALUES('J01', 'Ray Johnson', '05-OCT-70', 'Computer Science', 'Male', 40000);
INSERT INTO FACULTY VALUES('S01', 'Wendy Swimmer', '22-AUG-70', 'Computer Science', 'Female', 45000);
INSERT INTO FACULTY VALUES('J02', 'Bob Jones', NULL, 'Accounting', 'Male', 35000);
INSERT INTO FACULTY VALUES('N01', 'Jack Nelson', '10-JAN-71', 'History', 'Male', 28000);
INSERT INTO FACULTY VALUES('D02', 'Jinee Jackson', NULL, 'Accounting', 'Female', 34500);
INSERT INTO FACULTY VALUES('S02', 'William James', '11-NOV-67', 'Accounting', 'Male', 30500);
```

1.3. Create Course Table:

```
CREATE TABLE COURSE (COURSE_ID VARCHAR(15), COURSE_TITLE VARCHAR(30), SECTION NUMBER, FAC_ID VARCHAR(15), PRIMARY KEY(COURSE_ID), FOREIGN KEY (FAC_ID) REFERENCES FACULTY(FAC_ID));
```

Course Table:

COURSE_ID	COURSE_TITLE	SECTION	FAC_ID
CSC100	Intro. to Computing	1	J01
CSC101	Pascal Programming	1	D01
CSC102	Database Management	2	J01
ACC200	Principles Of Accounting I	2	J02
ACC201	Principles Of Accounting II		D02
HIS200	England History	1	N01
HIS201	Europe History		N01

1.4. Insert rows into course table:

```
INSERT INTO COURSE VALUES('CSC100','Intro. To Computing', 1, 'J01');
INSERT INTO COURSE VALUES('CSC101','Pascal Programming', 1, 'D01');
INSERT INTO COURSE VALUES('CSC102','Database Management', 2, 'J01');
INSERT INTO COURSE VALUES('ACC200','Principles Of Accounting I', 2, 'J02');
INSERT INTO COURSE VALUES('ACC201','Principles Of Accounting II',NULL, 'D02');
INSERT INTO COURSE VALUES('HIS200', 'England History', 1, 'N01');
INSERT INTO COURSE VALUES('HIS201', 'Europe History', NULL, 'N01');
```

2. Make sure student save the answer by creating new SQL file and name it as:

StudentEmailAddress_SETA

3. Check student Microsoft SQL Server Tool tables by run:

- SELECT * FROM FACULTY;

FAC_ID	FAC_NAME	BIRTH_DATE	DEPARTMENT	GENDER	SALARY
D01	Amy Dancer	25-JUN-71	Computer Science	Female	34500
J01	Ray Johnson	05-OCT-70	Computer Science	Male	40000
S01	Wendy Swimmer	22-AUG-70	Computer Science	Female	45000
J02	Bob Jones		Accounting	Male	35000
N01	Jack Nelson	10-JAN-71	History	Male	28000
D02	Jinee Jackson		Accounting	Female	34500
S02	William James	11-NOV-67	Accounting	Male	30500

- SELECT * FROM COURSE;

COURSE_ID	COURSE_TITLE	SECTION	FAC_ID
CSC100	Intro. to Computing	1	J01
CSC101	Pascal Programming	1	D01
CSC102	Database Management	2	J01
ACC200	Principles Of Accounting I	2	J02
ACC201	Principles Of Accounting II		D02
HIS200	England History	1	N01
HIS201	Europe History		N01

4. Give students copy of exam questions which contains 5 questions as showing in Table 1:

Table 1: SQL Questions (SET A)

QNo.	Question
1.	Find the names of all faculties who work at the accounting department with salary amounts greater than 34500.
2.	List the faculty names and departments of all faculties which their birth date is recorded and have at least one section. Sort the result in ascending order of the departments.
3.	Write a SQL statement that retrieves only those departments having average salary more than 30500.
4.	Find out all the course titles of faculties work in accounting department.
5.	Find those departments for which the average salary is greater than or equal to all average salaries.

4. Check student saving file.
5. Collect all files in one folder.

B. Instruction for Student: Microsoft SQL Server Tool

1. Create file as: `StduentEmailAddress_SETA`

- Check Microsoft SQL Server Tool tables by run: `SELECT * FROM FACULTY;`

FAC_ID	FAC_NAME	BIRTH_DATE	DEPARTMENT	GENDER	SALARY
D01	Amy Dancer	25-JUN-71	Computer Science	Female	34500
J01	Ray Johnson	05-OCT-70	Computer Science	Male	40000
S01	Wendy Swimmer	22-AUG-70	Computer Science	Female	45000
J02	Bob Jones		Accounting	Male	35000
N01	Jack Nelson	10-JAN-71	History	Male	28000
D02	Jinee Jackson		Accounting	Female	34500
S02	William James	11-NOV-67	Accounting	Male	30500

- `SELECT * FROM COURSE;`

COURSE_ID	COURSE_TITLE	SECTION	FAC_ID
CSC100	Intro. to Computing	1	J01
CSC101	Pascal Programming	1	D01
CSC102	Database Management	2	J01
ACC200	Principles Of Accounting I	2	J02
ACC201	Principles Of Accounting II		D02
HIS200	England History	1	N01
HIS201	Europe History		N01

2. Copy of exam questions which contains 5 questions as showing in Table 2:

Table 2: SQL Questions (GA)

QNo.	Question
1.	Find the names of all faculties who work at the accounting department with salary amounts greater than 34500.
2.	List the faculty names and departments of all faculties which their birth date is recorded and have at least one section. Sort the result in ascending order of the departments.
3.	Write a SQL statement that retrieves only those departments having average salary more than 30500.
4.	Find out all the course titles of faculties work in accounting department.
5.	Find those departments for which the average salary is greater than or equal to all average salaries.

3. You need to save all commands and log off.

C. Instruction for Examiner and Students: SQL-FE Tool

1. Ask student to follow the URL: <https://co-project.lboro.ac.uk/coaa/student/>
2. All students should register with any active email and write their full name.
3. Check SQL-FE tool tables as following:

- `SELECT * FROM EMP;`

EMPNO	EMPNAME	HIREDATE	GENDER	JOB	SALARY	DEPTNO
7369	Smith	1980-12-17	Male	Clerk	1500	20
7499	Allen	1981-02-20	Female	Salesman	1600	
7521	Jennifer		Female	Salesman	1250	30
7566	Jones	1984-04-02	Male	Clerk	2975	
7654	Martin	1981-09-28	Male	Salesman	1250	30
7698	Laura	1981-05-01	Female	Manager	2850	20
7782	Clark		Male	Manager	2450	10

- `SELECT * FROM DEPT;`

DEPTNO	DEPTNAME	NO_OF_EMP	LOC
10	Accounting	12	New York
20	Research	10	
30	Sales	5	Chicago
40	Operation	3	Boston
50	Management		New York

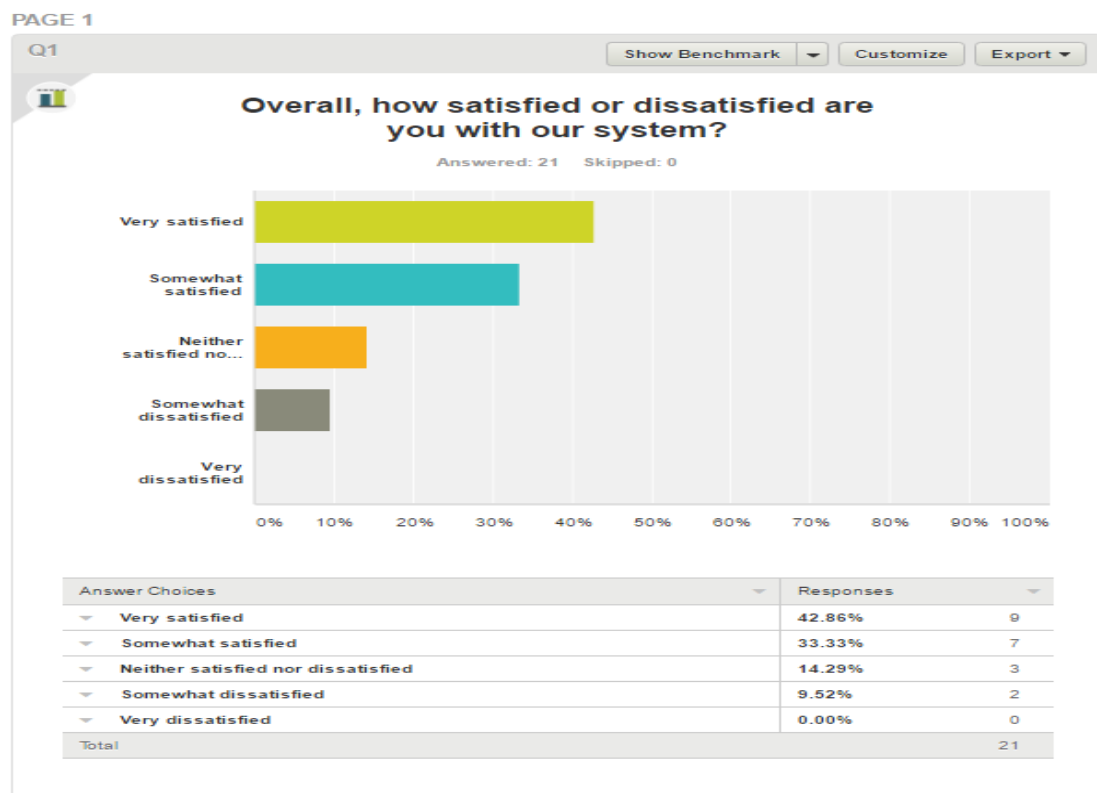
4. The test will start directly once the page is load and the time would be counted for each question.
5. There are 5 questions in the test as showing in table 3.

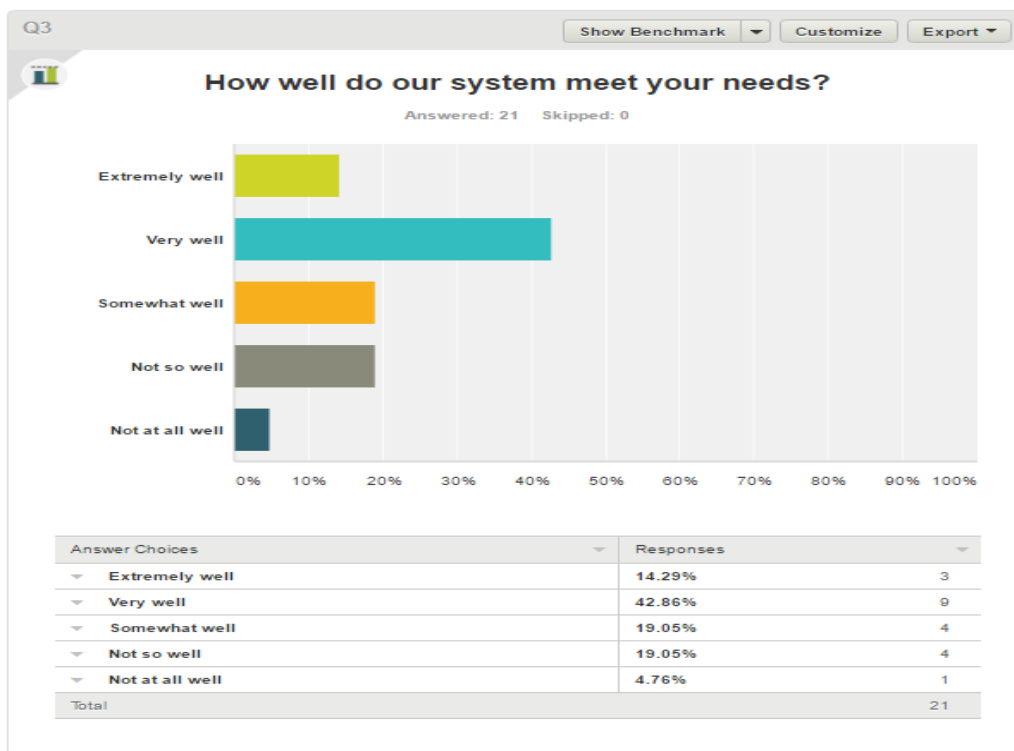
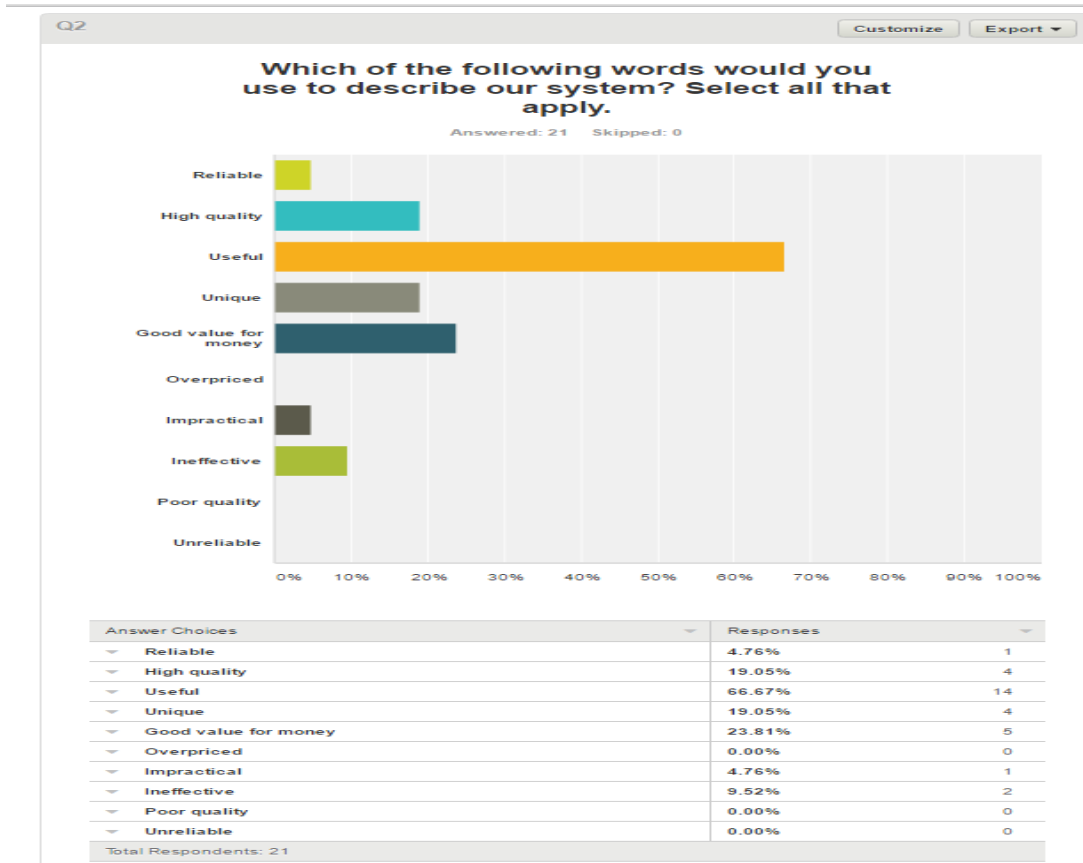
Table 3: SQL Questions (SET B)

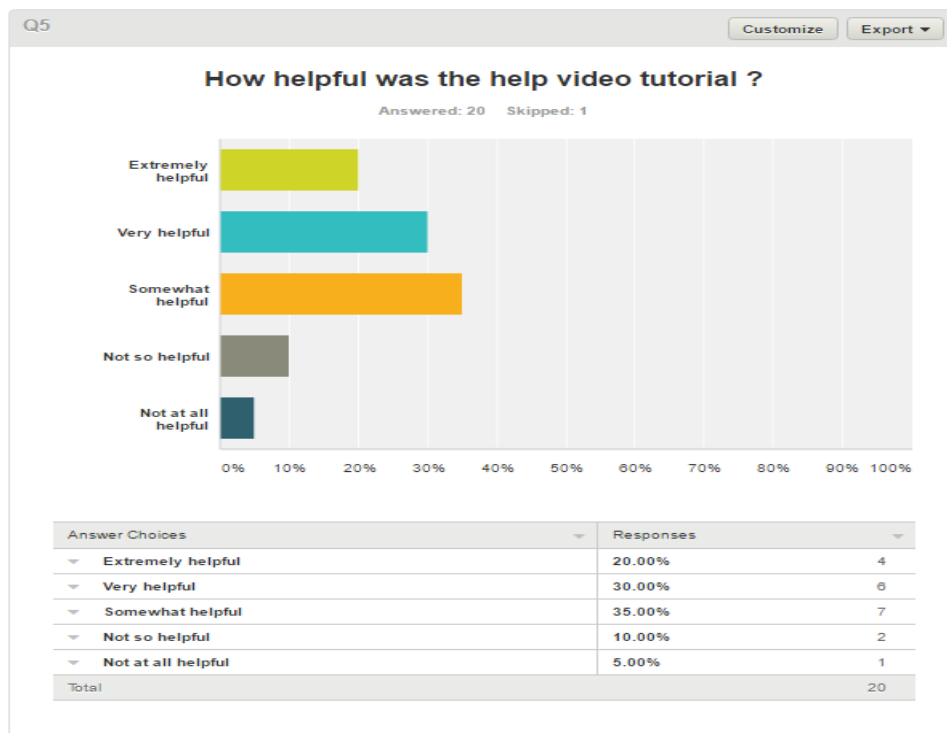
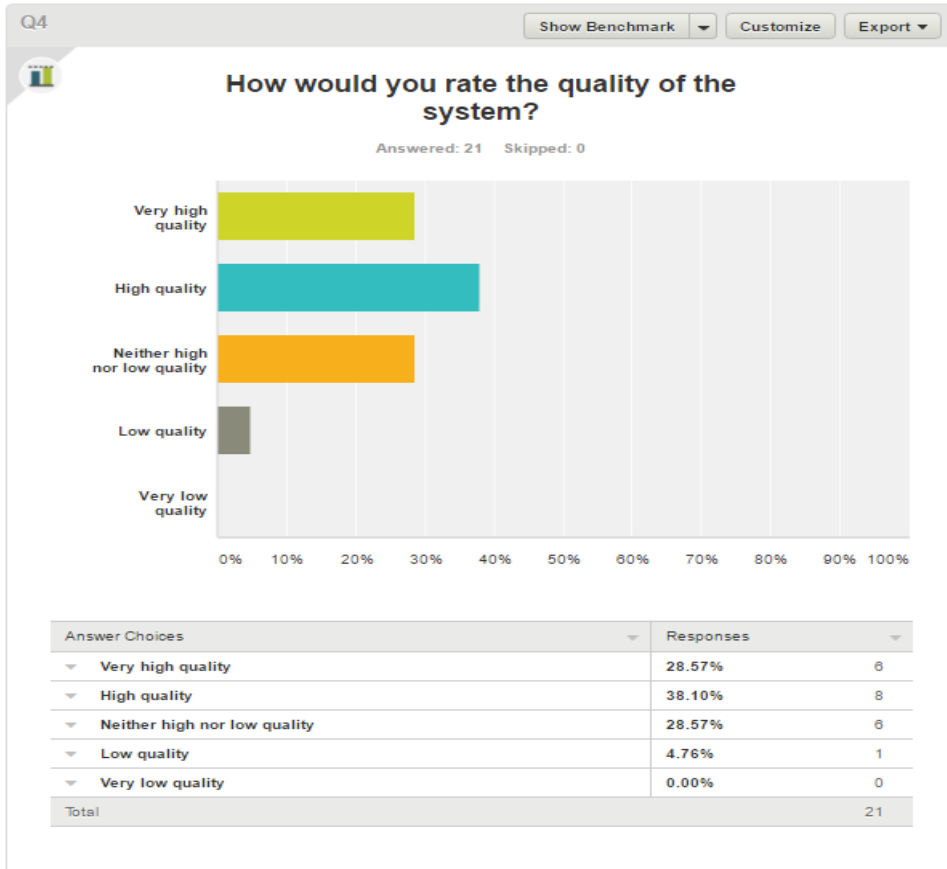
QNo.	Question
1.	Display the names of all the employees who are working as clerks and earning a salary more than 2500.
2.	List the employee names and job of all employees who are having hire date records and have 6 and more employees working in same department. Sort the result in the order of the job.
3.	Display the various jobs for each of the jobs where average salary is greater than 1500.
4.	List all department names of all employees who work as a manager.
5.	List all jobs which the average salary is less than or equal to all average salaries.

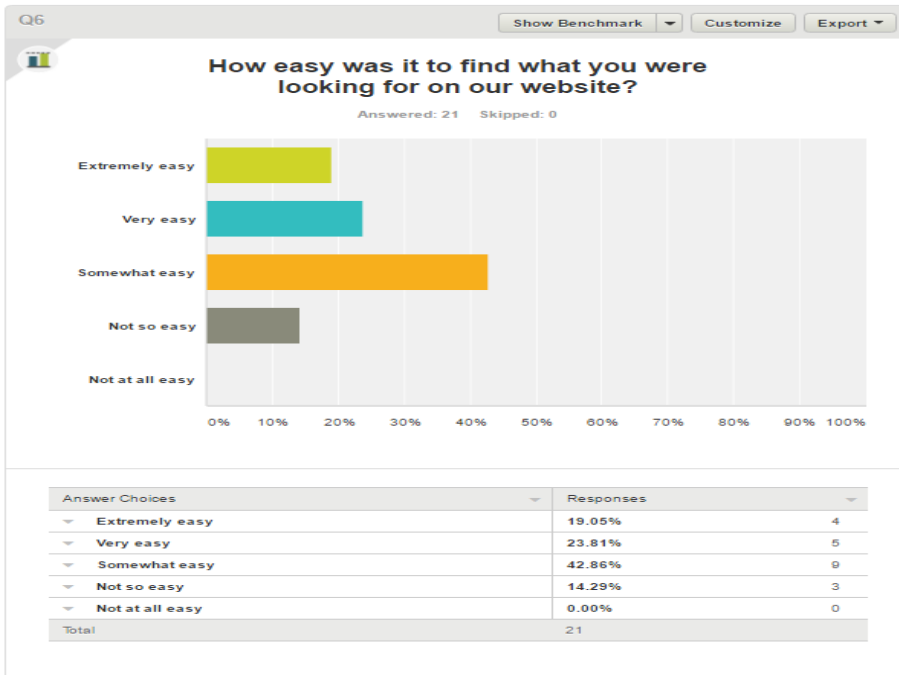
6. Use the text area to retrieve specific data by using the keyboard.
7. Once the question has been solved, the student should click on submit for each answer so it can be saved for marking.
8. If student used previous or next button in the question bar, the previous answer will be overwritten which they can't see their previous answer.
9. Run button is available for checking the answer before moving to next question.
10. If the student face any difficulties on entering values they can use the help link on the top right of the navigation bar.
11. Once the student finishes their exam ask them to log off.

D. Student feedback of SQL-FE









Appendix 13: SQL-ME Experiment Questionnaire

Dear participants,

Thank you for giving me the chance to ask the following questions. I sincerely request you to fill in the important information in this questionnaire. A high standard of truth while answering the questions will provide the semi-automatic assessment of SQL statements research a very clear picture about what kind of solutions could be implemented. Guarantee that all responses will be held strictly confidential. All participants should complete the questionnaire after testing the SQL Marking Editor (SQL-ME) with two web-pages such as marking system and marking system 2.

The questionnaire contains three measurement categories; SQL-ME user interface and time spent of marking statements, SQL-ME feedback quality and usefulness of SQL-ME. The questions for each category as follows:

A. SQL-ME user interface and time spent

1. Is there any difference between the two pages of the SQL-ME (marking system and marking system 2)?

Yes No

2. If you answered 'yes' to the question above, please provide details

3. Which of these two pages you preferred more?

Marking System Marking System 2

4. Specify the reasons by answering the following questions:

a. Do you think you saved time?

Yes No

b. If yes what proportion of time did you think was saved?

5% 20% 50% 100%

c. Do you think this proportion would increase as you get more used to the SQL-ME?

Yes No

d. If yes what proportion of time do you think you will save?

5% 20% 50% 100%

B. Feedback Quality

1. Do you feel you gave better feedback with the SQL-ME?

- Yes No

2. Can you estimate whether you gave more or less feedback with the SQL-ME?

3. Do you think you gave better quality feedback with SQL-ME? Explain.

4. Which of the following statements seems closest to describe?

a. *Partial marking process in SQL-ME*

- Has been designed to enhance student marks.
- Has been designed to enhance the marking process by increasing the consistency of marks between the students and provide students with detailed feedback of each clause in SQL statement
- Has been designed to enhance the students' future SQL formulation performance.

b. *Feedback process in SQL-ME*

- It provides students with detailed feedback during the course of Database module, so they are able to use it to improve the way they learn and enhance their future SQL formulation performance.
- It has been designed to enhance student marks.
- It is a continuous process of conversation between students and examiners.

c. *Propagation technique (the same marking is applied to all identical clauses for same question answers) in SQL-ME*

- It might enhance consistency of marking.
- It might minimise the human intervention (examiners workload).
- It might minimise the workload of the examiners (marking and feedback of identical clauses), identify same SQL errors and commenting then at the same time might enhance student performance.

C. Usefulness of SQL-ME

Using the SQL Marking Editor would

		Very Satisfied	Satisfied	Neither Satisfied nor Dissatisfied	Dissatisfied	Very Dissatisfied	No Opinion
1	Enable examiners to accomplish marking tasks more quickly	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2	Makes the marking process to accomplish easier to get done	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3	Saves the examiners time when mark and give feedback	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4	Improve examiners performance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5	Increase examiners productivity	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6	Enhance examiners effectiveness on teaching	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7	Make it easier to do marking	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8	Find it useful in marking.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9	Provide consistent marks for all students	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10	Provide timely and detailed feedback about students' answers	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>