Developing data mart using extraction, transformation and loading

Senthil Murugan Lakshmanan

DEVELOPING DATA MART USING EXTRACTION, TRANSFORMATION AND LOADING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Senthil Murugan Lakshmanan

June 2008

DEVELOPING DATA MART USING EXTRACTION, TRANSFORMATION AND LOADING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Senthil Murugan Lakshmanan

June 2008

Approved by:

[redacted]

Dr. Tong Lai Yu, Chair, Computer Science — 5/22/08 Date

[redacted]

Dr. David Turner

[redacted]

Dr. Ernesto Gomez

ABSTRACT

Data warehousing has become one of the top most projects for more than half of IT executives. Data in the warehouse systems is subject oriented, time variant and non-volatile, and supports managerial decision making. In other words, data warehouses are read only integrated applications designed to answer comparative and "what if" questions.

The overall life cycle of this project deals with two forms of processing, one is OLTP and the other is OLAP. Online transaction processing (OLTP) systems are helpful for storing the real time data to the operational systems database, which can help the organizations to run the business. However, they are not suited for addressing decision-oriented queries or business oriented queries for the CEOs or Managers. Such issues needed analytics including slicing/dicing of data, drilling down, aggregation, etc., which are supported by online analytical processing (OLAP). In order to perform analytical processing, data warehouses come into the picture. Data warehouses support online analytical processing applications by gathering the data from various kinds of source information in a multidimensional format. The complete life cycle consists of extracting the data from OLTP systems and loading it to the OLAP warehouse using ETL.

This project is organized to develop the sales data mart for the purpose of reporting data by using ETL. All these database and data transfers have been developed with the help of oracle PL / SQL based stored procedures in order to implement the business requirements. The reports are generated using the Oracle Business Intelligence tool.

ACKNOWLEDGMENTS

First, I wish to thank my advisor Dr Tong Lai Yu, for his strong and persistent support to my project. His guidance enabled me to thoroughly and comprehensively complete the analysis, design and implementation of this project.

In addition, I would like to express my gratitude to Dr. David Turner and Dr. Dr. Ernesto Gomez, my project committee members, who provided insightful comments and suggestions.

Also, I wish to thank my graduate advisor Dr. Josephine Mendoza for advising me throughout my masters program and for her database class which helped me to become as an ETL developer.

Moreover, I would like to thank all of my friends, especially Gopinath Swaminathan, Database developer for providing me some valuable inputs whenever I needed.

Finally I wish to thank my family members starting with my dear parents Lakshmanan and Valliammai for giving me an wonderful opportunity to study my master's program in United States of America, then my ever loving elder brother Narayanan and his family for providing invaluable support and inspirations and last but not the least my sweet younger brother Muthuraman for his continuous encouragement. Without these assistance and support from all of these professors, friends and family members, I most certainly could not have completed this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

As IT professionals, we have been involved in the design, development and maintenance of systems that support various business operations. Depending on the IT industries we have worked in, we might have been involved in applications such as customer information, product information, company accounts, billings, etc. These applications are important systems that run day to day business. As enterprises grew larger, more numbers of software applications were developed to support their business processes. These applications were designed to gather, store and process all the data needed to satisfy the future needs of the businesses. As business processes grew more complex and corporations spread globally, competition between peers became harder, and business executives became desperate for information to improve the bottom line. So they needed different kinds of information that could be used to make strategic decisions. Operational systems did provide information to run the day to day business operations, but what the business executives needed were different forms of information which could help in making strategic decisions. Business, therefore were motivated to look for new ways of getting the strategic information. Hence organizations started to look for various options to overcome the issue, and the successful end result was a data warehouse. With the help of strategic information, organizations began to achieve advantages in the market and compete with their peers as they had more historical data to evaluate their business.

1.1 Background

People who have worked in the Information technology field for more than three decades would have seen massive changes that have taken place. In an enterprise, the name of the computer science department went from Management Information Systems to Information Systems, and is currently Information Technology. Just as the name of the information technology field has changed, the industry has seen improved connectivity and networking, along with newer software providing accessible systems. All these improvements in technology have made computing easier, faster, more reliable, and increasingly web oriented. As a consequence, systems that deal with online transactions have become crucial. Since the early 90's, data warehouses have been at the forefront of information technology applications as a way for enterprises to make business plans and decisions, especially about online transactions. In order to make decisions about their business, all top level executives need to have in depth knowledge about their operations and performance. As an individual with out much information, it is impossible to drive the growth of the company. So to drive the business we need more information like sales and profit figures, customers' shopping habits and demographics, and the quantity of the products sold and services given, etc. The problem for most organizations, though, is that their online transaction systems or operational systems were not designed to support or make strategic decisions. These so called online transaction processing systems (OLTP) were designed to keep track of ever changing transactions, not to generate reports. To meet these needs, there is a new form of systems that has been developed in recent years known as Data Warehouse Systems.

1.2 Significance

A data mart is a logical subset of the contents of the whole data warehouse. A data warehouse, therefore, is a union of all data marts. In general, individual data marts are designed for the specific needs of particular business groups, but the collection of all the data marts form a data warehouse. The most important issue facing every IT manager is whether to develop a data warehouse first or a data mart first. Before deciding whether to build a data warehouse or a data mart, we need to ask the following fundamental and basic questions and address the relevant issues:

1. Should I look at the bigger picture and build a mammoth data warehouse using a top down approach or build a bite sized individual department data mart with a bottom up approach.
2. Should I build a large data warehouse and then let that warehouse repository feed the data into data marts or first built a data mart and combine all other data marts to form a larger data warehouse.
3. Most importantly are we looking for a quick long term result, or to address only immediate subjects?

All these questions can help us decide whether to build a data warehouse or data mart. After considering all these issues, I decided to design a data mart first, as it would be an ideal choice for an individual local department.

1.2.1 Bottom-Up Approach

There are several strategies for schema design, such as Top-Down, Bottom-Up or mixed approaches. Before getting into the Bottom-Up approach, let's quickly review the Top-down approach. Using a Top-down approach, we begin to consider the overall structure of the entire data warehouse, including how individual data marts will fit into the larger scheme. However, when developing a big, enterprise-wide data warehouse, it is impossible to foresee all the business requirements beforehand. So as an alternative, we do have an option of building up data marts first by using the Bottom-Up approach. The Top down approach takes a longer time and it will be a single, central storage of data. But it has a high risk of failure, as it needs a high level of cross functional skills. With the Bottom-Up approach all the local departmental data marts can be constructed one by one. We can set priority to determine which departmental data marts should be constructed first. This approach is faster and easier to implement, has less risk of failure, is inherently incremental, and can schedule important data marts first. Hence the Bottom-Up approach would be an ideal scenario for this situation. The overall life-cycle consists of planning, followed by business requirements definition, dimensional modeling, architecture design, physical design, and deployment.

1.3 Purpose

At this stage, we realize that we do need decision support systems to provide strategic information. Developing this strategic information can be done with the ETL, which means Extract, Transform and Loading. ETL functions reshape the relevant data from source systems into useful information to be stored in the data warehouse or data marts. Without these functions there would not be any strategic information in the data warehouse or data marts. Let us assume that our data has been successfully cleaned and ridden of unwanted data from the source information and loaded into the target. Now what?

1.3.1 Need for Multidimensional Analysis

When we just look the daily sales, we can quickly say the sales are interrelated in many dimensions.

The daily sales are more meaningful only when they convey meaningful statistics, such as what products are sold throughout a given period, through given distribution channels or regions, or through sales promotions, etc. By designing with multidimensional views, planning and making strategic decisions is easy. When we have a multidimensional view, we are able to examine the data by breaking it down into sales by region or sales by products or sales by time period, etc.

After performing all these tasks, we still haven't provided the best mechanism for information delivery to the end users. The user interface for information is what ultimately allows the project to succeed. If the users find the interface easy to use, they will come back again but if the users find it cumbersome we may need to leave the scene. Oracle Business Intelligence provides a good platform for end users to easily query and generate the report using Answers and Dashboard tools respectively.

1.4 Organization of the Project Document

There are six chapters in this document:
1) Introduction, 2) Architecture, Dimensional Modeling and Database Design, 3) ETL Overview, 4) BI in Data warehouse, 5) Deployment and 6) Conclusion. The appendix provides the information about Data Dictionary, Coding, Abbreviations, Acronyms and Definitions.

CHAPTER TWO

ARCHITECTURE, DIMENSIONAL MODELING AND DATABASE DESIGN

This chapter discusses the three important phases in the overall design of the project. The chapter begins with a discussion of architecture, followed by the dimensional modeling and then database design.

2.1 Operational Systems and Data Warehouse Systems

Online transaction processing systems and data warehouse systems have fundamentally different purposes. An Online transaction system supports the execution of business processes, while the warehouse system supports the evaluation of the business process. Apart from this, every data warehouse requires software programs in order to transform the data from operational systems to warehouse systems, as well as software to develop queries and reports. Therefore the architecture of a warehouse system requires the following components:

Table 1. Data Warehouse and Operational Systems

Also Known as	Transaction Systems, Online Transactions Processing System, Source System	Analytic System, Data Mart
Purpose	Execution of business process	Measurement of a business process
Design	3rd Normal form	Dimensional design (star schema)
Data	Current	Historic and current data

2.1.1 Distinguishing Characteristics

Data warehouse architecture is wide, complex and expansive. The overall architecture has to support the requirements for providing the necessary information. In operational systems, the amount of information provided in each session is limited. For example, at a particular time, a user might be interested in getting only a particular product name and all of its order information. Whereas in a data warehouse, a user might be interested in getting a large result set about specific information, such as when the product was sold, to whom it was sold, when it was sold, geographical and demographic statistics, etc. Hence data warehouse design architectures must have components that will provide large data sets in a single session.

Before data is brought from operational systems and stored to the "read-only" data warehouse, many functions must be performed. As we all know, data warehouses cannot represent a snapshot of an operational system; we should have an architecture called a staging area to perform the changes in data such as cleaning and integrating the data into business requirements. Business requirements may be defined as the data produced to facilitate business needs, such as statistics regarding items sold, customer demographics, etc.

Information retrieval from operational systems is more complex when compared to retrieving the information from a data warehouse. A user session might last for a long duration, because the user might query at a high level, review the results and change the next query a little differently to get the desired result and so on. Hence these operations can not be performed in an operational system. By keeping this requirement in consideration, the architecture of aggregate layers is designed at the warehouse level. Overall the data warehouse architecture should make it easy to make strategic decisions quickly.

Architecture diagram of Sales Data Mart

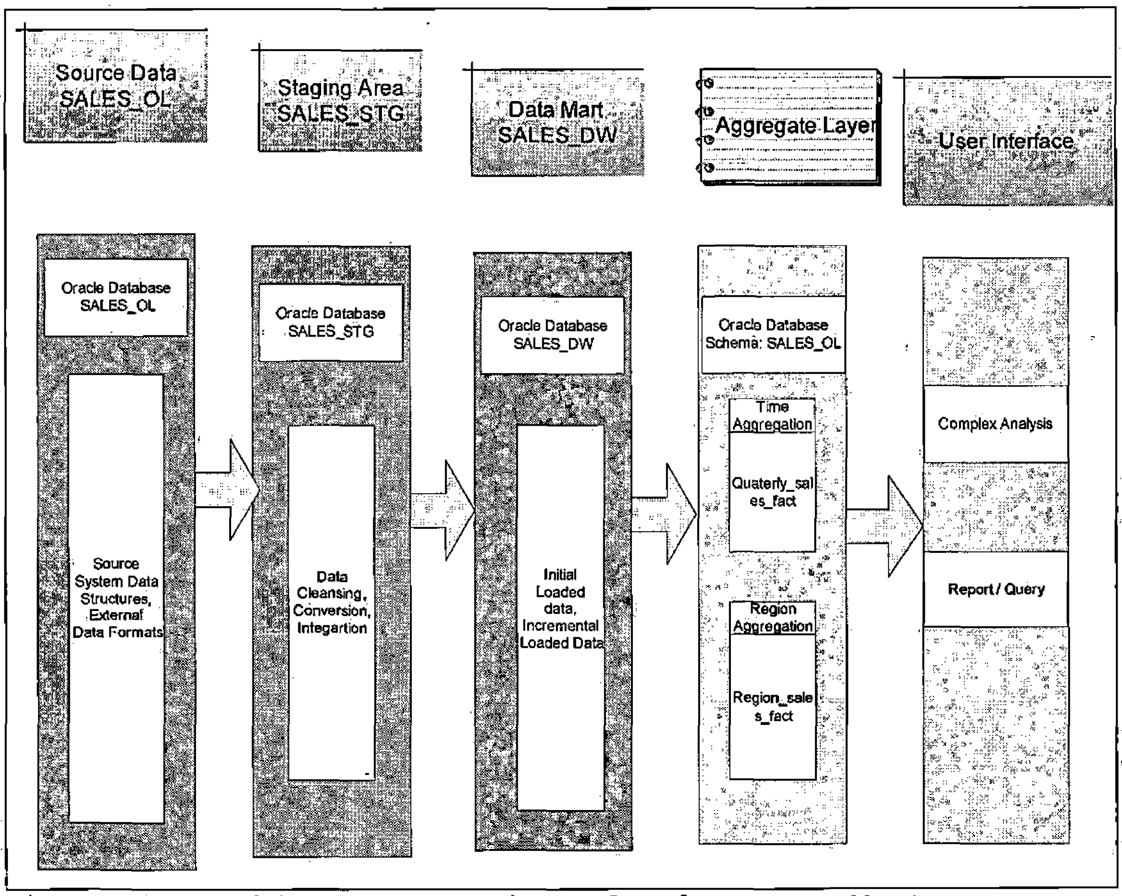


Figure 1. Architecture Design of Sales Data Mart

2.2 Dimensional Model

The name Dimensional Modeling is derived from the business dimensions which we want to incorporate into the logical model. It is a logical design technique used to model the business dimensions and the metrics. Facts and Dimensions are the two major categories associated with the dimensional model. The Dimensional model allows high performance queries and analyses. The basic steps for the design decision should be made before we proceed with the dimensional modeling for the proper result. The basic design decisions are:

- Selecting the facts: Units of measurements or metrics should be predetermined (example: Dollar sales, sales unit etc).
- Identifying the dimensions: In setting up the structure we need to make sure business dimensions are identified (example: Time dimension, Region Dimension).
- Data Grain: Determining the level of detail for the data in the warehouse.

Before getting into Fact tables and different types of schemas used in dimensional modeling, we should see the difference between E-R modeling and Dimensional Modeling.

E R stands for Entity Relationship modeling. This technique is used for operational or OLTP systems. With the ER modeling we can remove the data redundancy, ensure data consistency and express microscopic relationships.

Dimensional modeling is more suitable for the data warehouse. As the data warehouse focuses on answering questions on overall process, business trends, etc., it is better to support it with dimensional modeling, which allows captures of critical measures, views along dimensions and is more intuitive to business users.

Finally, when establishing the relationship between fact tables and dimensional tables, we get the result as a Star Schema.

2.2.1 Inside Dimensional Modeling

Dimensional table Key: Primary Key of the table uniquely identifies the records.

Attributes: We can have more columns inside the table. Therefore we can say the dimensional table is wide.

Not Normalized: The columns or attributes are used again and again in queries. For query performance, it is best if the query picks out the attribute directly from the dimensional table rather than going through other tables. This overcomes performance issues. When we normalize the dimension tables we will be need to pass through other intermediate tables to process the query, which will cause a performance issue.

Total Rows: Total number of rows should be less than the fact tables

2.2.2 Fact Tables

Fact tables are the large tables in our warehouse schema that store business measurements. Fact tables typically contain facts related to the business process and foreign keys to the dimension tables. Fact tables represent data, usually numeric and additive, that can be analyzed and examined. Examples include sales, costs, and profits. Fact tables typically have two types of columns: those that contain numeric facts (often called measurements), and those that are foreign keys to dimension tables. A fact table contains either detail-level facts or facts that have been aggregated. Fact tables that contain aggregated facts are often called summary tables.

2.2.3 Inside a Fact Table

Concatenated Key: Fact tables relate to a combination of rows from all dimension tables. Thus the primary key of the fact should have a concatenation of the primary keys of all the dimension tables.

Table not wide, but deep: The idea that a table should be deep means that it can have more records, but there should be a smaller number of columns for optimized query performance.

2.3 Star Schema

The star schema is the simplest data warehouse schema. In the dimensional model, the facts are placed in the fact tables and each group of dimensions is placed in dimensional tables. Hence the outcome is called a star schema because the diagram resembles a star, with points radiating from a center. The center of the star consists of fact tables and the edges of the star are the dimension tables.

The dimensional tables are wide and have lot of columns to support various forms of reports. All the information about each level is stored in one row. The structure can easily provide us answers for questions such as whom, what, to whom etc. Fact tables in star schema are deep and may contain many numbers of rows. Fact tables have foreign key columns that associate with dimension tables. An example of a star schema with fact and dimensional tables are shown below:

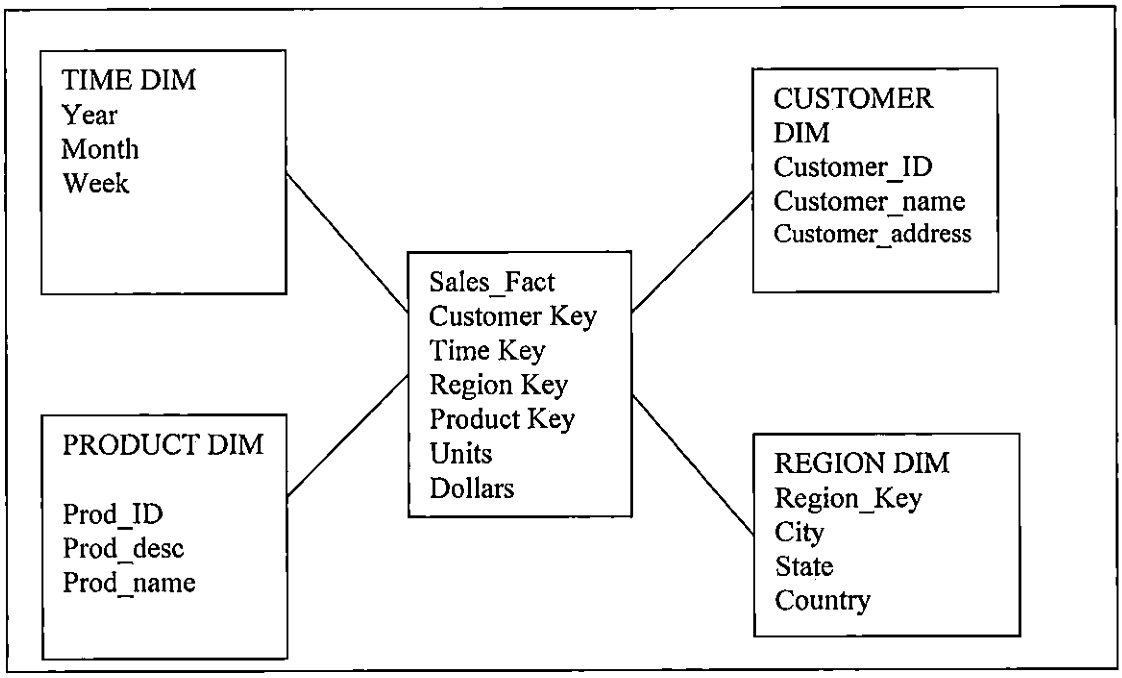


Figure 2. Star Schema for Sales Department

<u>2.3.1 Advantage of a Star Schema</u>

1. Easier to understand and navigate
2. Better Performance due to fewer Joins
3. Supports Multi Dimensional Analysis
4. Extensible design to handle changing Business requirements

2.4 Database Design

After gathering all the business requirements for the database, they must be modeled. There are two types of data modeling: Logical Modeling and Physical modeling

2.4.1 Logical Model

The real purpose of building a logical model was to confirm that the application will satisfy the requirements of both the business entities and their units. In other words, the logical model deals with business requirements and converting the requirements into a model. The logical model is associated with the needs of business, not the database.

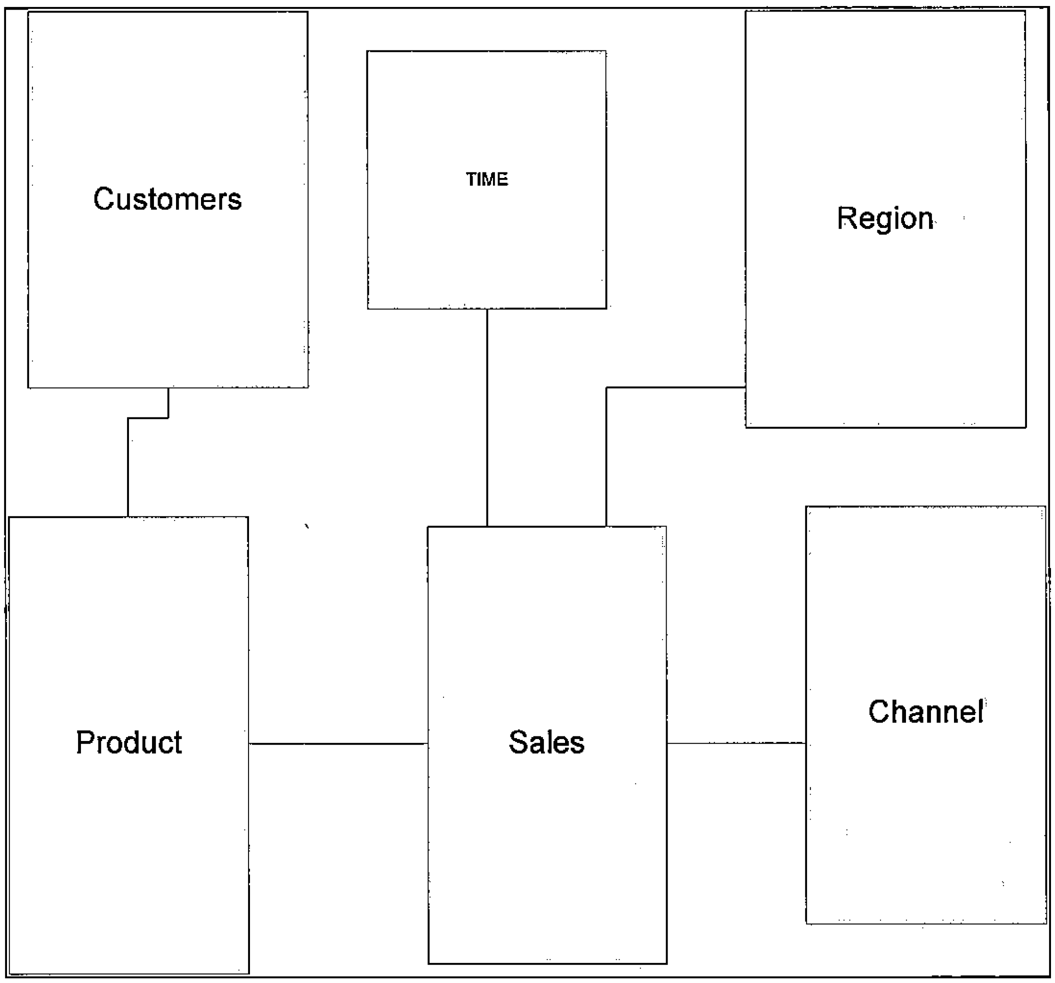


Figure 3. Logical Model of Sales Schema

Logical Data Model for sales Data Mart

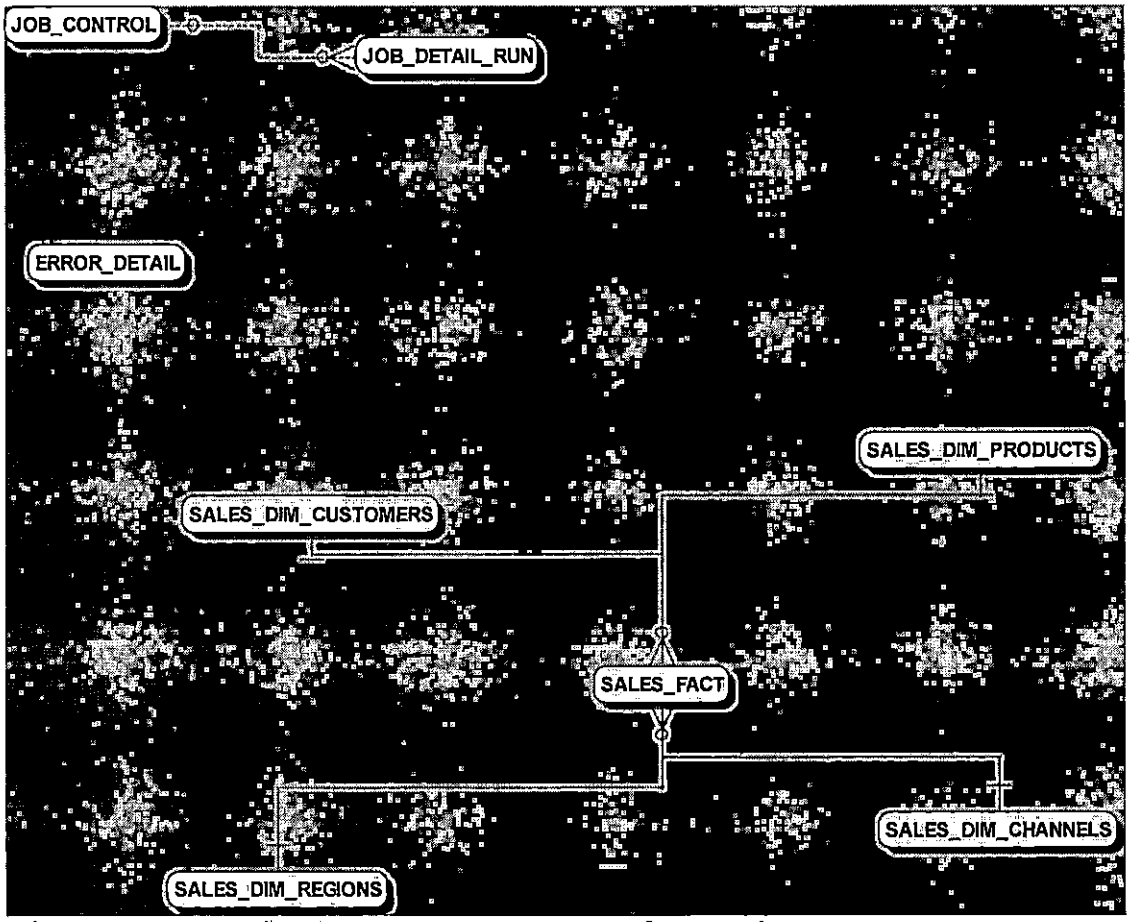


Figure 4. Logical Data Flow of Sales Schema

<u>2.4.2 Physical Model</u>

Physical data modeling is used to design the internal schema of a database, depicting the data tables, the columns of those tables, and the relationships between the tables. Physical Data modeling often proves to be useful on both Agile and traditional projects. Here all the attributes are defined with their data types. Primary key and foreign keys are defined.

Data normalization is performed. It is a process in which data attributes within a data model are organized to increase the cohesion of entity types. In other words, the goal of data normalization is to reduce or eliminate data redundancy. There are three common normalization rules describing how to put entity types into a series of increasing levels of normalization. The achievement of the third degree normal form can increase the performance in OLTP systems, whereas data warehouse systems are not normalized specifically for the purpose of increasing the performance, as the queries used for generating the report can be of complex joins.

Data Normalization rules:

- First Normal Form: An entity type is in 1NF when it contains no repeating groups of data.
- Second Normal Form: An entity type is in 2NF when it is in 1NF and also when all of its non-key attributes are fully dependent on its primary key.
- Third Normal Form: An entity type is in 3NF when it is in 2NF, and also all attributes that are

not dependent on the primary key must be removed.

Physical Data Model for OLTP Systems.

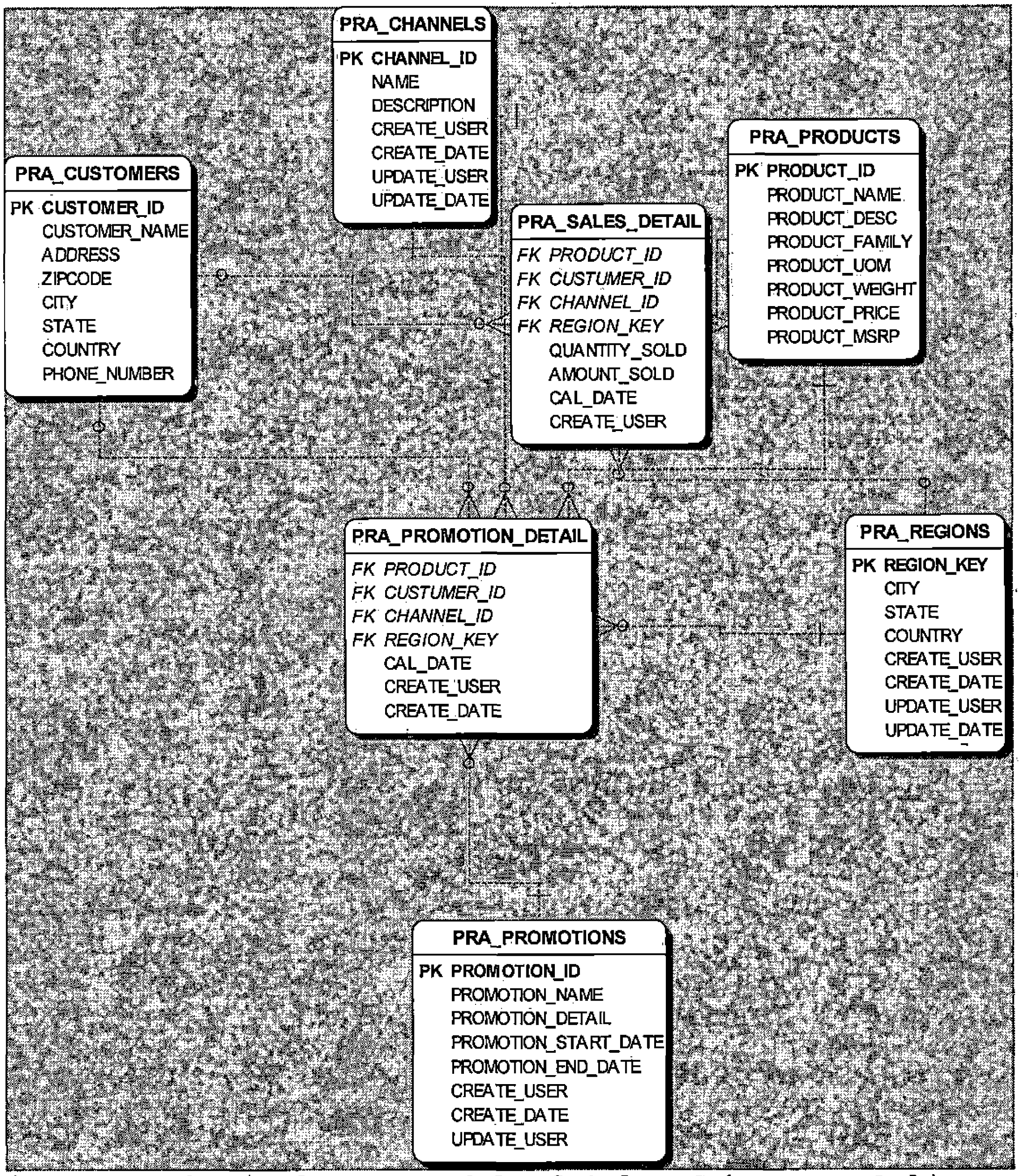


Figure 5. Physical Data Model of Sales Schema on Online Transaction Processing Systems

Physical Model for SALES_DATAMART

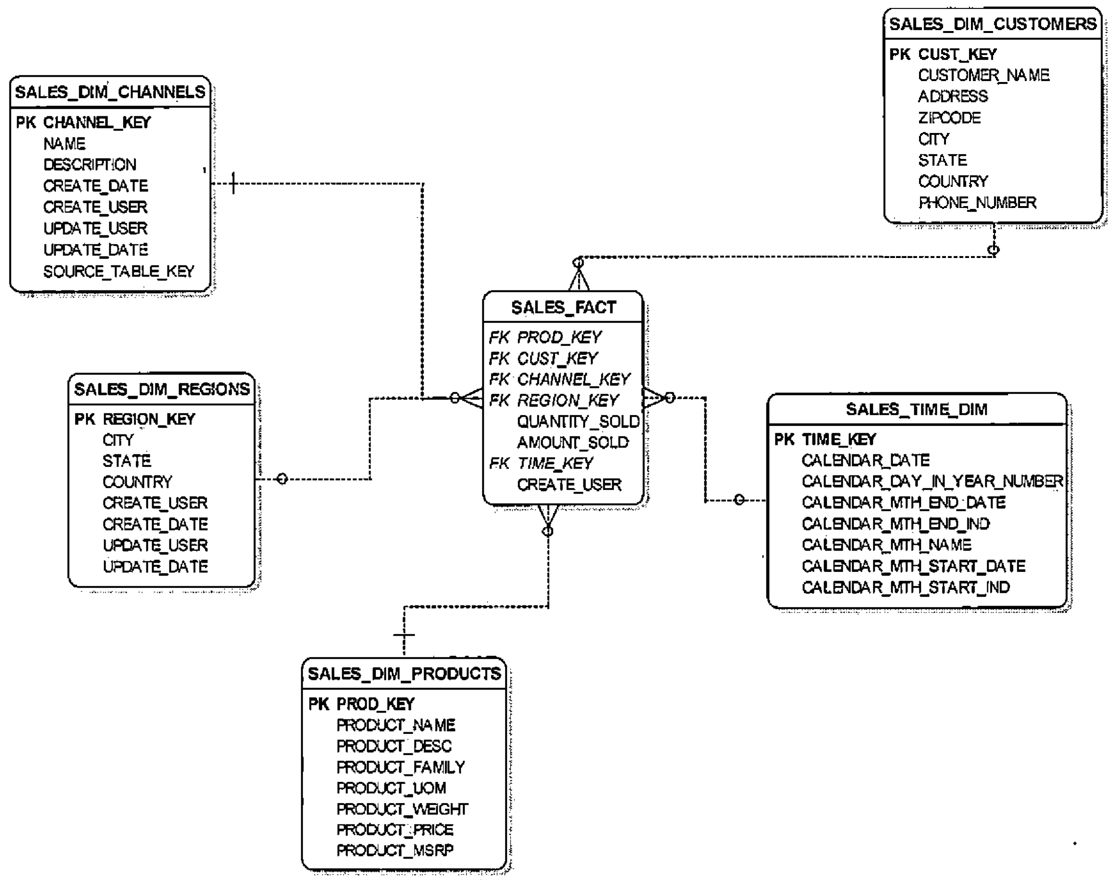


Figure 6. Star Schema with Sales_fact and Dimensional Tables for Sales Data Mart

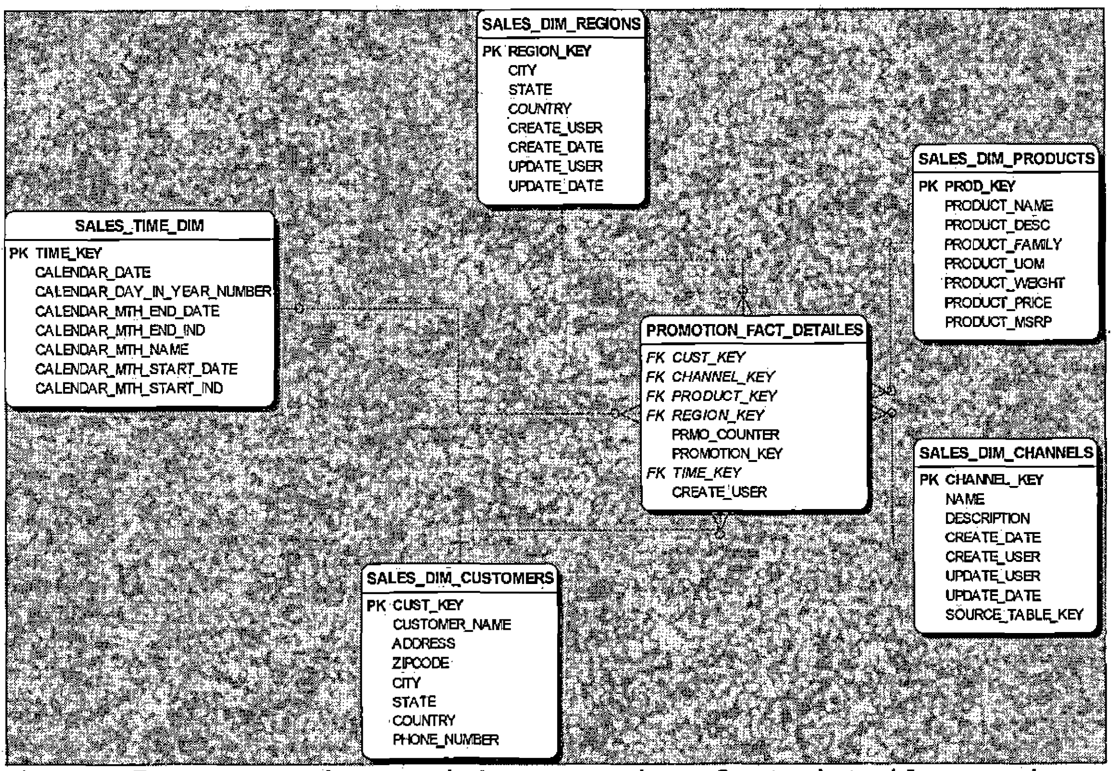


Figure 7. Star Schema with Promotion_fact_detailes and Dimensional Tables on Sales Data Mart

CHAPTER THREE

EXTRACTION, TRANSFORMATION, AND LOADING OVERVIEW

This Chapter discusses the overall view of ETL, ETL mapping specifications and design logic. The term ETL is abbreviated as Extraction, Transformation and Loading.

3.1 Extraction Transformation Loading

3.1.1 Data Source

Source data components can be divided into four categories. They can be Internal Data, Archived Data, External Data or Production Data. In all companies, users have their private profiles, documents, databases etc. Such data is called Internal Data, and parts of such data can be used in a data mart. Another type of data is Archive Data. Because operational systems are primarily intended to run the existing business, in order to have the historical snapshot of data we must construct Archive Data. Also, in order to compete in industry, and compare performance with the market players, we need to have data from external sources, or External Data. And finally, Production Data comes from the various operational systems of different enterprises. In this project, we used an oracle database SALES_OL as our source information.

3.1.2 Staging Area

After extracting the data from operational systems, we need to push it to the data warehouse. The main function is extracting the data from source, transforming it according to the business needs and preparing it for the loading into the data mart or data warehouse that takes place in the staging area. To be precise, we will be doing all the cleaning and converting of data in the staging area. By doing this in the staging area, we make sure that source data are protected from OLTP databases. Our staging area Schema is SALES_STG.

3.1.3 Data Loading

Finally, after cleansing the data according to the business needs, we are ready to load the data to the data mart. There are two types of loading available. One is initial loading and the other is incremental loading. After designing the data warehouse, the system will have no records, so the first time we load data into the system is called initial loading. With the initial loading we move all records of data at one time into a system. If we wished to either add information or modify the information afterwards, we would move data into the already created system by using incremental loading. According to business needs we can determine the refresh cycles. There can be yearly, monthly, weekly, daily, or hourly refreshing made based upon business requirements.

2.2 Data Flow

2.2.1 Data Extracted From Operational System and Loaded into Staging Area

In the Data acquisition area, the data flow begins from source tables and ends at the staging area. Audit tables are created to check the status of the data transfers. After the initial loading the audit tables are populated with the time stamp and user history. In the incremental loading audit tables are also designed to record the time stamp. PL/SQL is used in the ETL process. The Staging Area is a temporary storage area for performing all business transformations.

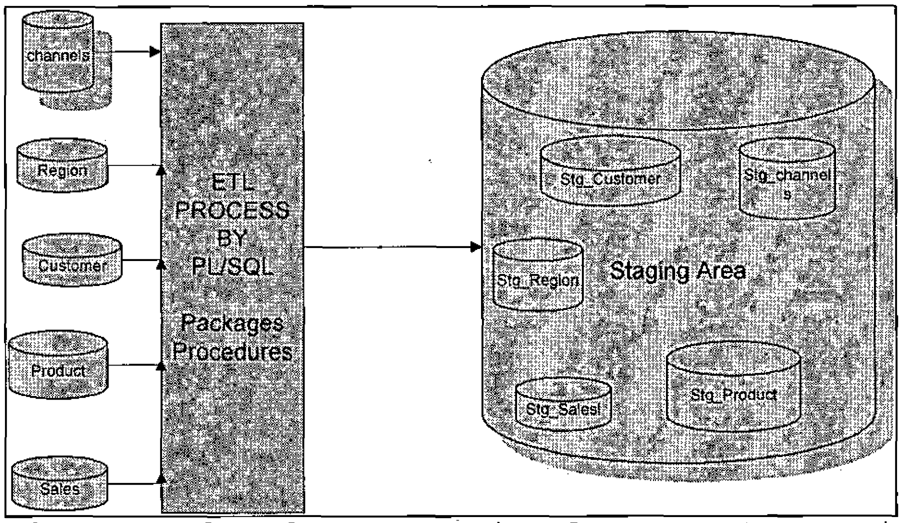


Figure 8. Data Flow from Operational System to Staging Area

2.2.2 Data Transformed from Staging Area to Data Mart

In the staging area, we are preparing to move data into the data mart. The staging area is the place where all the extracted data is not only moved but also transformed according to the business logic. It is like a construction area where we perform all the transformations according to the business logic. We have audit tables to keep track of how many source records have populated and at what time. Like functions and services, initial loading is performed first and followed up with incremental loading into the data mart.

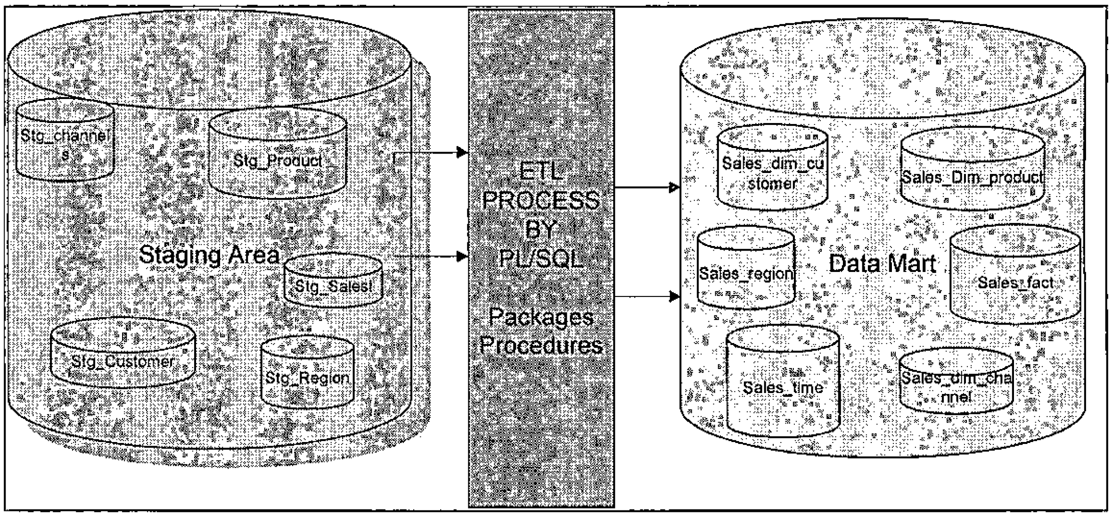


Figure 9. Data Flow from Staging Area to Data Mart

2.2.3 Aggregate Layer Created from Data Mart

The Transformed and integrated data are loaded into data mart. When we are specific about the breakdown of sales by different time periods or geographic areas we populate the data into the aggregation layer.

All these transactions are done with PL/SQL.

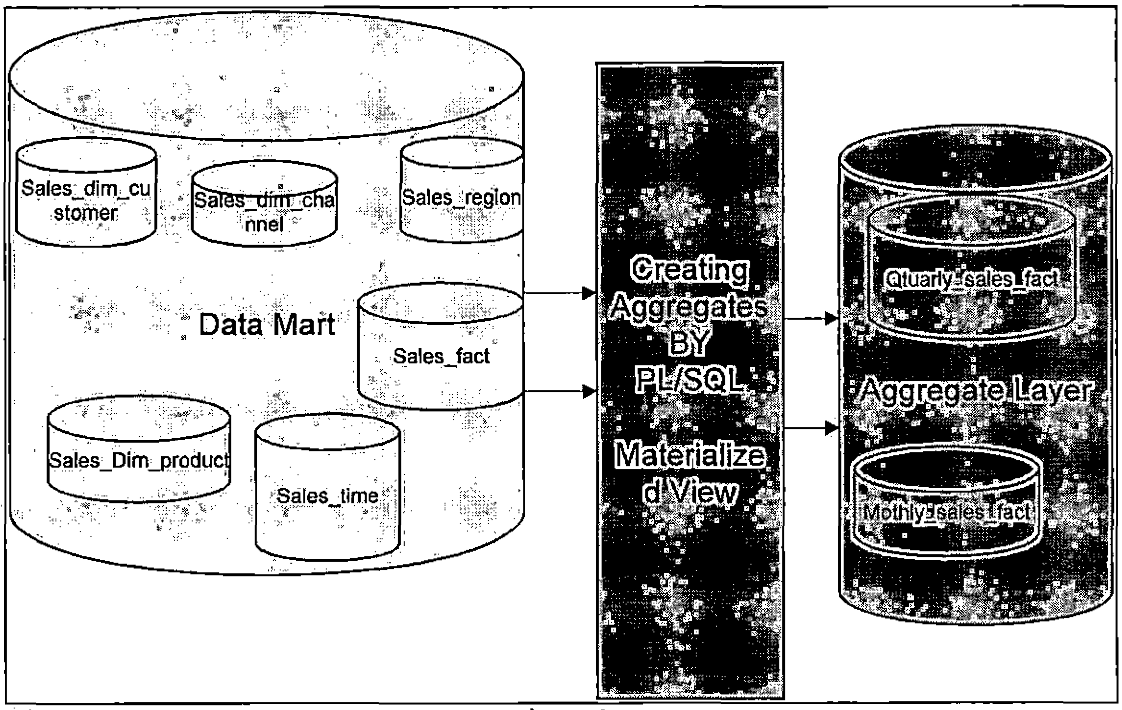


Figure 10. Aggregate Layer in the Data Mart

3.2 Extraction, Transformation, and Loading Mappings

ETL Mappings are performed between a Source table and a Target Table. All the below tables explain the source and target information with transformation details.

3.2.1 Extraction, Transformation, and Loading Mapping for Staging Channels

Table 2. Extraction, Transformation, and Loading Mapping for Staging Channels

SOURCE_TABLE_NAME	SOURCE_COLUMN_NAME	TARGET_TABLE_NAME	TRAGET_COLUMN_NAME	TRANSFORMATION
PRA_CHANNELS		STG_CHANNELS	CHANNEL_ID	Sequence Number
PRA_CHANNELS	NAME	STG_CHANNELS	NAME	
PRA_CHANNELS	DESCRIPTION	STG_CHANNELS	DESCRIPTION	
PRA_CHANNELS		STG_CHANNELS	CREATE_DATE	SYSDATE
PRA_CHANNELS		STG_CHANNELS	CREATE_USER	SALES_STG
PRA_CHANNELS		STG_CHANNELS	UPDATE_USER	SALES_STG
PRA_CHANNELS		STG_CHANNELS	UPDATE_DATE	SYSDATE
PRA_CHANNELS	SOURCE_TABLE_KEY	STG_CHANNELS	SOURCE_TABLE_KEY	
PRA_CHANNELS	CHANNEL_ID	STG_CHANNELS	SOURCE_RECORD_ID	

3.2.2 Extraction, Transformation, and Loading Mapping for Staging Customers

Table 3. Extraction, Transformation, and Loading Mapping for Staging Customers

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
PRA_CUSTOMERS		STG_CUSTOMERS	CUSTOMER_ID	Sequence Number
PRA_CUSTOMERS	CUSTOMER_NAME	STG_CUSTOMERS	CUSTOMER_NAME	
PRA_CUSTOMERS	ADDRESS	STG_CUSTOMERS	ADDRESS	
PRA_CUSTOMERS	ZIPCODE	STG_CUSTOMERS	ZIPCODE	
PRA_CUSTOMERS	CITY	STG_CUSTOMERS	CITY	
PRA_CUSTOMERS	STATE	STG_CUSTOMERS	STATE	
PRA_CUSTOMERS	COUNTRY	STG_CUSTOMERS	COUNTRY	
PRA_CUSTOMERS	PHONE_NUMBER	STG_CUSTOMERS	PHONE_NUMBER	
PRA_CUSTOMERS	EMAIL	STG_CUSTOMERS	EMAIL	
PRA_CUSTOMERS		STG_CUSTOMERS	CREATE_DATE	SYSDATE
PRA_CUSTOMERS		STG_CUSTOMERS	CREATE_USER	SALES_STG
PRA_CUSTOMERS		STG_CUSTOMERS	UPDATE_USER	SALES_STG
PRA_CUSTOMERS		STG_CUSTOMERS	UPDATE_DATE	SYSDATE
PRA_CUSTOMERS	SOURCE_TABLE_KEY	STG_CUSTOMERS	SOURCE_TABLE_KEY	
PRA_CUSTOMERS	CUSTOMER_ID	STG_CUSTOMERS	SOURCE_RECORD_ID	

3.2.3 Extraction, Transformation, and Loading Mapping for Staging Promotions

Table 4. Extraction, Transformation, and Loading Mapping for Staging Promotions

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
PRA_PROMOTIONS		STG_PROMOTIONS	PROMOTION_ID	Sequence Number
PRA_PROMOTIONS	PROMOTION_ NAME	STG_PROMOTIONS	PROMOTION_ NAME	
PRA_PROMOTIONS	PROMOTION_ DETAIL	STG_PROMOTIONS	PROMOTION_ DETAIL	
PRA_PROMOTIONS	PROMOTION_ START_DATE	STG_PROMOTIONS	PROMOTION_ START_DATE	
PRA_PROMOTIONS	PROMOTION_ END_DATE	STG_PROMOTIONS	PROMOTION_ END_DATE	
PRA_PROMOTIONS		STG_PROMOTIONS	CREATE_DATE	SYSDATE
PRA_PROMOTIONS		STG_PROMOTIONS	CREATE_USER	SALES_STG
PRA_PROMOTIONS		STG_PROMOTIONS	UPDATE_USER	SALES_STG
PRA_PROMOTIONS		STG_PROMOTIONS	UPDATE_DATE	SYSDATE
PRA_PROMOTIONS	SOURCE_ TABLE_KEY	STG_PROMOTIONS	SOURCE_ TABLE_KEY	
PRA_PROMOTIONS	PROMOTION_ ID	STG_PROMOTIONS	SOURCE_ RECORD_ID	

3.2.4 Extraction, Transformation, and Loading Mapping for Staging Products

Table 5. Extraction, Transformation, and Loading Mapping for Staging Products

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET _COLUMN_NAME	TRANSFORMATION
PRA_PRODUCTS	PROD_ID	STG_PRODUCTS	PROD_ID	
PRA_PRODUCTS	PRODUCT_NAME	STG_PRODUCTS	PRODUCT_NAME	
PRA_PRODUCTS	PRODUCT_DESC	STG_PRODUCTS	PRODUCT_DESC	
PRA_PRODUCTS	PRODUCT_FAMILY	STG_PRODUCTS	PRODUCT_FAMILY	
PRA_PRODUCTS	PRODUCT_UOM	STG_PRODUCTS	PRODUCT_UOM	
PRA_PRODUCTS	PRODUCT_WEIGHT	STG_PRODUCTS	PRODUCT_WEIGHT	
PRA_PRODUCTS	PRODUCT_PRICE	STG_PRODUCTS	PRODUCT_PRICE	
PRA_PRODUCTS	PRODUCT_MSRP	STG_PRODUCTS	PRODUCT_MSRP	
PRA_PRODUCTS		STG_PRODUCTS	CREATE_DATE	SYSDATE
PRA_PRODUCTS		STG_PRODUCTS	CREATE_USER	SALES_STG
PRA_PRODUCTS		STG_PRODUCTS	UPDATE_USER	SALES_STG
PRA_PRODUCTS		STG_PRODUCTS	UPDATE_DATE	SYSDATE
PRA_PRODUCTS	SOURCE_ TABLE_KEY	STG_PRODUCTS	SOURCE_ TABLE_KEY	
PRA_PRODUCTS	PRODUCT_ID	STG_PRODUCTS	SOURCE_ RECORD_ID	

3.2.5 Extraction, Transformation, and Loading Mapping for Staging Region

Table 6. Extraction, Transformation, and Loading Mapping for Staging Region

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
PRA_REGIONS		STG_REGIONS	REGION_KEY	Sequence Number
PRA_REGIONS	CITY	STG_REGIONS	CITY	
PRA_REGIONS	STATE	STG_REGIONS	STATE	
PRA_REGIONS	COUNTRY	STG_REGIONS	COUNTRY	
PRA_REGIONS		STG_REGIONS	CREATE_USER	SALES_STG
PRA_REGIONS		STG_REGIONS	CREATE_DATE	SYSDATE
PRA_REGIONS		STG_REGIONS	UPDATE_USER	SALES_STG
PRA_REGIONS		STG_REGIONS	UPDATE_DATE	SYSDATE
PRA_REGIONS	SOURCE_ SYSTEM_TABLE_KEY	STG_REGIONS	SOURCE_ SYSTEM_TABLE_KEY	
PRA_REGIONS	SOURCE_ SYSTEM_RECORD_KEY	STG_REGIONS	SOURCE_ SYSTEM_RECORD_KEY	

3.2.6 Extraction, Transformation, and Loading Mapping for Staging Sales Transactions

Table 7. Extraction, Transformation, and Loading Mapping for Staging Sales Transactions

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
PRA_SALES_DE TAIL	CAL_DATE	STG_SALES_ TRANSACTION	CAL_DATE	
		STG_SALES_ TRANSACTION	CREATE_USER	SALES_STG
		STG_SALES_ TRANSACTION	CREATE_DATE	SYSDATE
		STG_SALES_ TRANSACTION	UPDATE_USER	SALES_STG
		STG_SALES_ TRANSACTION	UPDATE_DATE	SYSDATE
PRA_ PRODUCTS	PRODUCT_ NAME	STG_SALES_ TRANSACTION	PRODUCT_NAME	
PRA_ CUSTOMERS	CUSTOMER_ NAME	STG_SALES_ TRANSACTION	CUSTOMER_ NAME	
PRA_CHANNELS	CHANNEL_ NAME	STG_SALES_ TRANSACTION	CHANNEL_NAME	
PRA_REGIONS	CITY	STG_SALES_ TRANSACTION	CITY	
PRA_ SALES_DETAIL	QUANTITY_ SOLD	STG_SALES_ TRANSACTION	QUANTITY_ SOLD	
PRA_ SALES_DETAIL	AMOUNT_ SOLD	STG_SALES_ TRANSACTION	AMOUNT_SOLD	
PRA_CUSTOMER S	CUSTOMER_ NAME	STG_PROMOTION_ TRANSACTION	CUSTOMER_ NAME	
PRA_PRODUCTS	PRODUCT_ NAME	STG_PROMOTION_ TRANSACTION	PRODUCT_ NAME	
PRA_CHANNEL	CHANNEL_ NAME	STG_PROMOTION_ TRANSACTION	CHANNEL_ NAME	
PRA_REGIONS	CITY	STG_PROMOTION_ TRANSACTION	CITY	
PRA_ PROMOTIONS	PROMOTION_ NAME	STG_PROMOTION_ TRANSACTION	PROMOTION_ NAME	
PRA_ PROMOTION_DETAIL	CAL_DATE	STG_PROMOTION_ TRANSACTION	CAL_DATE	
PRA_ PROMOTION_DETAIL		STG_PROMOTION_ TRANSACTION	CREATE_ USER	SALES_STG
PRA_ PROMOTION_DETAIL		STG_PROMOTION_ TRANSACTION	CREATE_ DATE	SYSDATE
PRA_ PROMOTION_DETAIL		STG_PROMOTION_ TRANSACTION	UPDATE_ USER	SALES_STG
PRA_ PROMOTION_DETAIL		STG_PROMOTION_ TRANSACTION	UPDATE_ DATE	SYSDATE

3.2.7 Extraction, Transformation, and Loading Mapping for Sales Dim Channel

Table 8. Extraction, Transformation, and Loading Mapping for Staging Dim Channels

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
STG_CHANNELS		SALES_DIM_ CHANNELS	CHANNEL_KEY	Sequence Number
STG_CHANNELS	NAME	SALES_DIM_ CHANNELS	NAME	
STG_CHANNELS	DESCRIPTION	SALES_DIM_ CHANNELS	DESCRIPTION	
STG_CHANNELS		SALES_DIM_ CHANNELS	CREATE_DATE	SYSDATE
STG_CHANNELS		SALES_DIM_ CHANNELS	CREATE_USER	SALES_STG
STG_CHANNELS		SALES_DIM_ CHANNELS	UPDATE_USER	SALES_STG
STG_CHANNELS		SALES_DIM_ CHANNELS	UPDATE_DATE	SYSDATE
STG_CHANNELS	SOURCE_ TABLE_KEY	SALES_DIM_ CHANNELS	SOURCE_ TABLE_KEY	
STG_CHANNELS	CHANNEL_ KEY	SALES_DIM_ CHANNELS	SOURCE_ RECORD_ID	

3.2.8 Extraction, Transformation, and Loading Mapping for Sales Dim Products

Table 9. Extraction, Transformation, and Loading Dim Products

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
STG_PRODUCTS		SALES_DIM_ PRODUCTS	PROD_KEY	Sequence Number
STG_PRODUCTS	PRODUCT_NAME	SALES_DIM_ PRODUCTS	PRODUCT_NAME	
STG_PRODUCTS	PRODUCT_DESC	SALES_DIM_ PRODUCTS	PRODUCT_DESC	
STG_PRODUCTS	PRODUCT_FAMILY	SALES_DIM_ PRODUCTS	PRODUCT_FAMILY	
STG_PRODUCTS	PRODUCT_UOM	SALES_DIM_ PRODUCTS	PRODUCT_UOM	
STG_PRODUCTS	PRODUCT_WEIGHT	SALES_DIM_ PRODUCTS	PRODUCT_WEIGHT	
STG_PRODUCTS	PRODUCT_PRICE	SALES_DIM_ PRODUCTS	PRODUCT_PRICE	
STG_PRODUCTS	PRODUCT_MSRP	SALES_DIM_ PRODUCTS	PRODUCT_MSRP	
STG_PRODUCTS		SALES_DIM_ PRODUCTS	CREATE_DATE	SYSDATE
STG_PRODUCTS		SALES_DIM_ PRODUCTS	CREATE_USER	SALES_STG
STG_PRODUCTS		SALES_DIM_ PRODUCTS	UPDATE_USER	SALES_STG
STG_PRODUCTS		SALES_DIM_ PRODUCTS	UPDATE_DATE	SYSDATE
STG_PRODUCTS	SOURCE_ TABLE_KEY	SALES_DIM_ PRODUCTS	SOURCE_ TABLE_KEY	
STG_PRODUCTS	PRODUCT_KEY	SALES_DIM_ PRODUCTS	SOURCE_ RECORD_ID	

3.2.9 Extraction, Transformation, and Loading Mapping for Sales Dim Promotions

Table 10. Extraction, Transformation, and Loading Sales Dim Promotions

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
STG_PROMOTIONS		SALES_DIM_ PROMOTIONS	CREATE_DATE	SYSDATE
STG_PROMOTIONS		SALES_DIM_ PROMOTIONS	CREATE_USER	SALES_STG
STG_PROMOTIONS		SALES_DIM_ PROMOTIONS	UPDATE_USER	SALES_STG
STG_PROMOTIONS		SALES_DIM_ PROMOTIONS	UPDATE_DATE	SYSDATE
STG_PROMOTIONS	SOURCE_ TABLE_KEY	SALES_DIM_ PROMOTIONS	SOURCE_ TABLE_KEY	
STG_PROMOTIONS	PROMO_KEY	SALES_DIM_ PROMOTIONS	SOURCE_ RECORD_ID	
STG_PROMOTIONS		SALES_DIM_ PROMOTIONS	PROMO_KEY	Sequence Number
STG_PROMOTIONS	PROMOTION_ NAME	SALES_DIM_ PROMOTIONS	PROMOTION_ NAME	
STG_PROMOTIONS	PROMOTION_ DETAIL	SALES_DIM_ PROMOTIONS	PROMOTION_ DETAIL	
STG_PROMOTIONS	PROMOTION_ START_DATE	SALES_DIM_P ROMOTIONS	PROMOTION_ START_DATE	
STG_PROMOTIONS	PROMOTION_ END_DATE	SALES_DIM_ PROMOTIONS	PROMOTION_ END_DATE	

3.2.10 Extraction, Transformation, and Loading Mapping for Sales Dim Customers

Table 11. Extraction, Transformation, and Loading Sales Dim Customers

SOURCE_ TABLE_NAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TRAGET_ COLUMN_NAME	TRANSFORMATION
STG_CUSTOMERS		SALES_DIM_ CUSTTIONS	CUST_KEY	Sequence Number
STG_CUSTOMERS	CUSTOMER_NAME	SALES_DIM_ CUSTTIONS	CUSTOMER_NAME	
STG_CUSTOMERS	ADDRESS	SALES_DIM_ CUSTTIONS	ADDRESS	
STG_CUSTOMERS	ZIPCODE	SALES_DIM_ CUSTTIONS	ZIPCODE	
STG_CUSTOMERS	CITY	SALES_DIM_ CUSTTIONS	CITY	
STG_CUSTOMERS	STATE	SALES_DIM_ CUSTTIONS	STATE	
STG_CUSTOMERS	COUNTRY	SALES_DIM_ CUSTTIONS	COUNTRY	
STG_CUSTOMERS	PHONE_NUMBER	SALES_DIM_ CUSTTIONS	PHONE_NUMBER	
STG_CUSTOMERS	EMAIL	SALES_DIM_ CUSTTIONS	EMAIL	
STG_CUSTOMERS		SALES_DIM_ CUSTTIONS	CREATE_DATE	SYSDATE
STG_CUSTOMERS		SALES_DIM_ CUSTTIONS	CREATE_USER	SALES_STG
STG_CUSTOMERS		SALES_DIM_ CUSTTIONS	UPDATE_USER	SALES_STG
STG_CUSTOMERS		SALES_DIM_ CUSTTIONS	UPDATE_DATE	SYSDATE
STG_CUSTOMERS	SOURCE_ TABLE_KEY	SALES_DIM_ CUSTTIONS	SOURCE_TABLE_KEY	
STG_CUSTOMERS	CUST_KEY	SALES_DIM_ CUSTTIONS	SOURCE_RECORD_ID	

3.2.11 Extraction, Transformation, and Loading Mapping for Sales Dim Regions

Table 12. Extraction, Transformation, and Loading Sales Dim Regions

SOURCE_ TABLENAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TARGET_ COLUMN_NAME	TRANSFORMATION
		SALES_DIM_ REGIONS	REGION_KEY	SEQUENCE GENERATOR
STG_REGION	CITY	SALES_DIM_ REGIONS	CITY	
STG_REGION	STATE	SALES_DIM_ REGIONS	STATE	
STG_REGION	COUNTRY	SALES_DIM_ REGIONS	COUNTRY	
		SALES_DIM_ REGIONS	CREATE_USER	SALES_STG
		SALES_DIM_ REGIONS	CREATE_DATE	SYSDATE
		SALES_DIM_ REGIONS	UPDATE_USER	SALES_STG
		SALES_DIM_ REGIONS	UPDATE_DATE	SYSDATE
STG_REGION	SOURCE_SYSTEM_ TABLE_KEY	SALES_DIM_ REGIONS	SOURCE_SYSTEM_ TABLE_KEY	
STG_REGION	SOURCE_SYSTEM_ RECORD_KEY	SALES_DIM_ REGIONS	SOURCE_SYSTEM_ RECORD_KEY	

3.2.12 Extraction, Transformation, and Loading Mapping for Sales Fact

Table 13. Extraction, Transformation, and Loading Sales Fact

SOURCE_ TABLENAME	SOURCE_ COLUMN_NAME	TARGET_ TABLE_NAME	TARGET_ COLUMN_NAME	TRANSFORMATION
STG_SALES_ TRANSACTION	PROD_KEY	SALES_FACT	PROD_KEY	
STG_SALES_ TRANSACTION	CUST_KEY	SALES_FACT	CUST_KEY	
STG_SALES_ TRANSACTION	CHANNEL_KEY	SALES_FACT	CHANNEL_KEY	
	REGION_KEY	SALES_FACT	REGION_KEY	
STG_SALES_ TRANSACTION	QUANTITY_SOLD	SALES_FACT	QUANTITY_SOLD	
STG_SALES_ TRANSACTION	AMOUNT_SOLD	SALES_FACT	AMOUNT_SOLD	
STG_SALES_ TRANSACTION	TIME_KEY	SALES_FACT	TIME_KEY	
		SALES_FACT	CREATE_USER	SALES_STG
		SALES_FACT	CREATE_DATE	SYSDATE
		SALES_FACT	UPDATE_USER	SALES_STG
		SALES_FACT	UPDATE_DATE	SYSDATE

CHAPTER FOUR

BUSINESS INTELLIGENCE

After the successful extraction and transformation of data from the source systems, data is loaded into the data mart. Now What? After performing all these tasks efficiently, if we could not provide the best mechanism for information delivery to end users, we have not established anything from the end users point of view. In practice, data marts, which based on the dimensional model, can be adequate for basic analysis. However, for today's business needs we might find the user going beyond the basic analysis.

Need for Multidimensional Analysis

One of the key concepts in decision support systems is multidimensional analysis: examining the enterprise from all necessary combinations of dimensions. We use the term dimensions to mean any category used in specifying questions. Among the most commonly specified dimensions are time, geography, product, department, and distribution channel, but the potential dimensions are as endless as the varieties of enterprise activity. Decision makers must be able to analyze data in a multiple dimension. They must be able to slice and dice, as well as drill up and down along the hierarchies of all dimensions. To perform a multidimensional analysis we certainly need the online analytical processing power in the data mart. The traditional use of online analytical processing applications is that of ad-hoc queries often made by people who are not highly technical. It can be mainly used for

- Aggregations
- Drill Down
- Rolling Up
- Slicing
- Dicing

A query which produces the total sales over a time period for each specific product in the region can be grouped through aggregation. There may be a hierarchy also in dimensional tables. For examples in our Region table we have a hierarchy of City, State and Country. When we execute a sequence of queries that move down a hierarchy from the general to a specific, such as moving from an aggregation over the country element to aggregation over the state element, it is said to be drilled down. When we move up the hierarchy, for example, from aggregation over the city element to aggregation over the state element, we are rolling up.

The important thing to be noted is not all hierarchy is linear. For example, in our time hierarchy, we don't have a linear progression, but instead we have a lattice. Weeks are not fully contained in months, as the same week can fall on the two different months. Thus we can't roll up days into months or weeks, but only to quarters. Oracle Business Intelligence (OBI) is used as a BI tool to do the analysis from Sales Data Mart.

4.1 Oracle Business Intelligence

Oracle Business Intelligence is one of the BI tools which can be used in generating reports. ODBC connections are used to extract the data from the data mart, which is residing on the server. Various components of Oracle Business Intelligence are

1. OBI Server
2. Answers
3. Dashboards
4. OBI Administration Tool

4.1.1 Oracle Business Intelligence Server

The oracle business intelligence Server provides the power behind Dashboards for access and analysis of structured data that is distributed across an organization's supply chain. It provides efficient processing in order to intelligently access the data sources. Connections can be made natively or via ODBC to the RDBMS. The various server components are repositories which store the metadata information in a cache that can contain the result of queries such as NQSconfig.ini, configuration files used for defining the repository such as DBFeatures.ini files. These files define the features supported by each database. Commands such as NQServer.log records the server messages, and NQQery.log records the result of query informations.

4.1.2 Dashboards and Answers

Oracle Business Intelligence Answers are what is used to generate the queries. Dashboards are user-friendly, providing pre-built access to information with interactivity through drilldown. No knowledge of the underlying schemas is required. It can be used by everyone and it is highly intuitive. Below are the few examples of reports generated

Customer_Name	Amount_Sold
Hesper Ness	30,625
Herbert Pakerman	30,125
Henrietta Snodgrass	29,625
Bille Wipple	29,550
Hatty Daily	29,125
Rollin Lu	28,625
Ralph Kenyon	28,125
Radley Barker	27,625
Ozelle Rowley	27,125
Oriel Ziegler	26,625

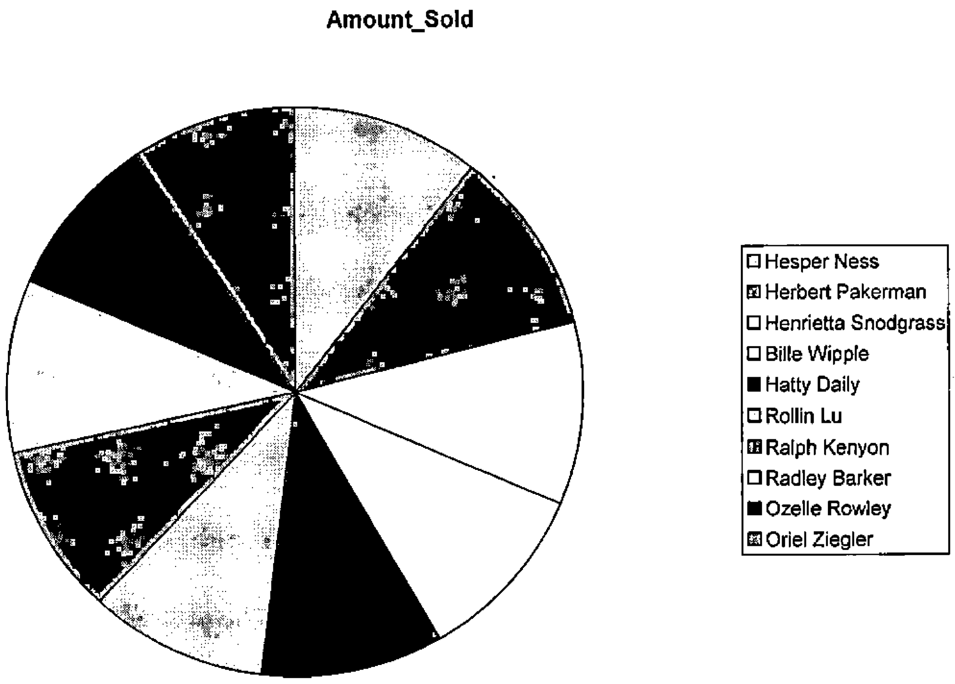


Figure 11. Top 10 Customer Report

State	Amount_Sold
NJ	136,775
MA	110,250
MI	87,375
CA	72,525
NY	65,800

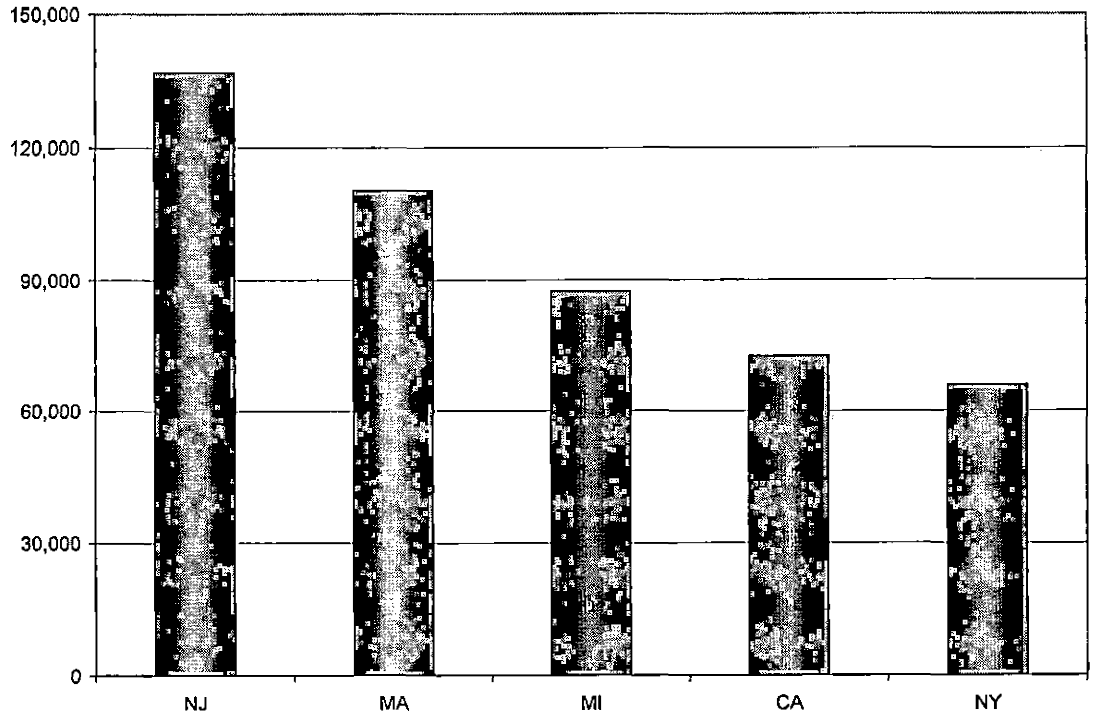
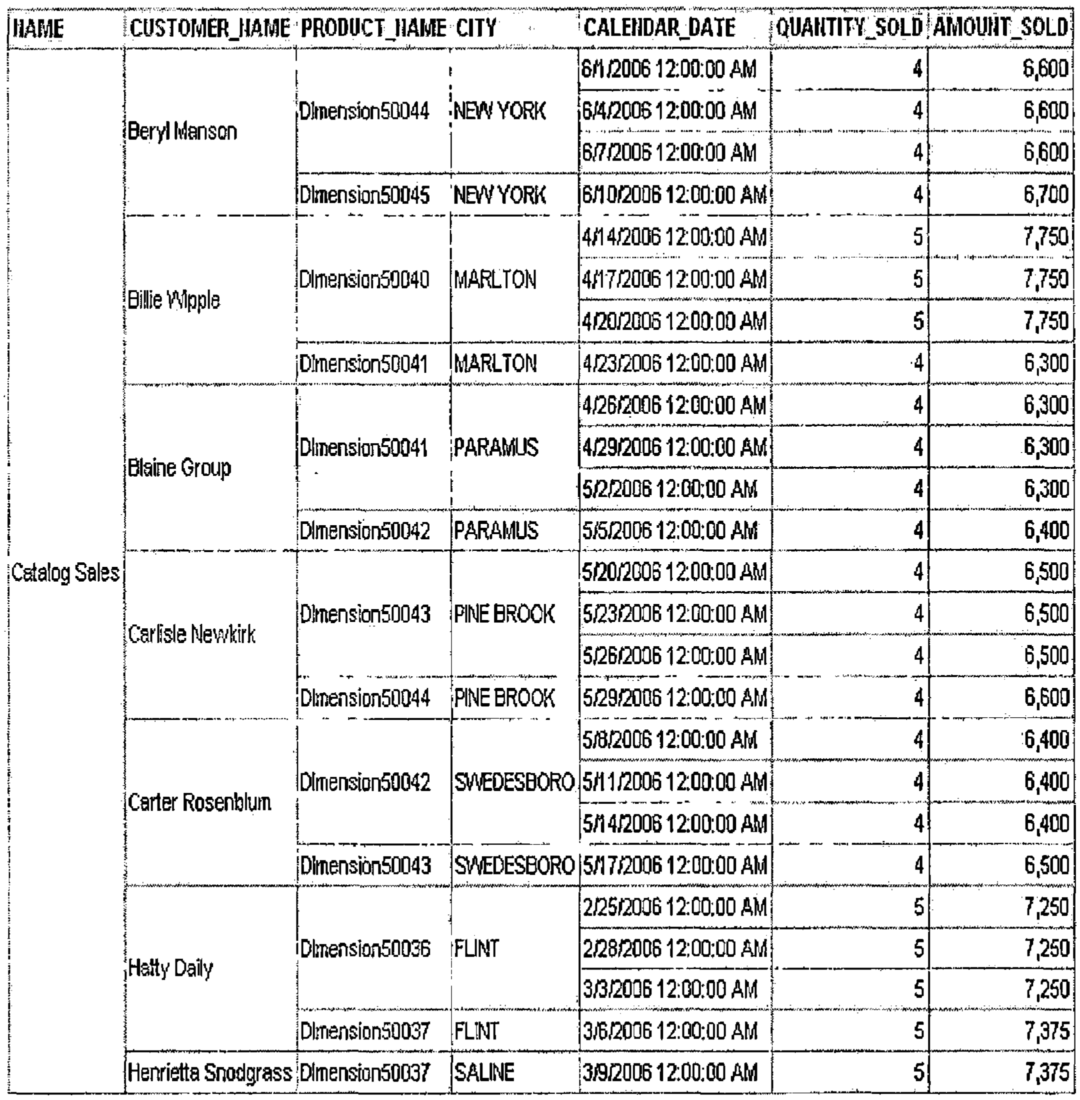


Figure 12. Top 5 Selling States



NAME	CUSTOMER_NAME	PRODUCT_NAME	CITY	CALENDAR_DATE	QUANTITY_SOLD	AMOUNT_SOLD
Catalog Sales	Beryl Manson	Dimension50044	NEW YORK	6/1/2006 12:00:00 AM	4	6,600
				6/4/2006 12:00:00 AM	4	6,600
				6/7/2006 12:00:00 AM	4	6,600
		Dimension50045	NEW YORK	6/10/2006 12:00:00 AM	4	6,700
	Billie Wipple	Dimension50040	MARLTON	4/14/2006 12:00:00 AM	5	7,750
				4/17/2006 12:00:00 AM	5	7,750
				4/20/2006 12:00:00 AM	5	7,750
		Dimension50041	MARLTON	4/23/2006 12:00:00 AM	4	6,300
	Blaine Group	Dimension50041	PARAMUS	4/26/2006 12:00:00 AM	4	6,300
				4/29/2006 12:00:00 AM	4	6,300
				5/2/2006 12:00:00 AM	4	6,300
		Dimension50042	PARAMUS	5/5/2006 12:00:00 AM	4	6,400
	Carlisle Newkirk	Dimension50043	PINE BROOK	5/20/2006 12:00:00 AM	4	6,500
				5/23/2006 12:00:00 AM	4	6,500
				5/26/2006 12:00:00 AM	4	6,500
		Dimension50044	PINE BROOK	5/29/2006 12:00:00 AM	4	6,600
	Carter Rosenblum	Dimension50042	SWEDESBORO	5/8/2006 12:00:00 AM	4	6,400
				5/11/2006 12:00:00 AM	4	6,400
				5/14/2006 12:00:00 AM	4	6,400
		Dimension50043	SWEDESBORO	5/17/2006 12:00:00 AM	4	6,500
	Hatty Daily	Dimension50036	FLINT	2/25/2006 12:00:00 AM	5	7,250
				2/28/2006 12:00:00 AM	5	7,250
				3/3/2006 12:00:00 AM	5	7,250
		Dimension50037	FLINT	3/6/2006 12:00:00 AM	5	7,375
	Henrietta Snodgrass	Dimension50037	SALINE	3/9/2006 12:00:00 AM	5	7,375

Figure 13. Sales_detail Report

CHAPTER FIVE

DEPLOYMENT

Step 1

Oracle Database server is installed

Step 2

Log-in to SYSTEM user

Execute Following Commands

```
CREATE USER SALES_OL IDENTIFIED BY SALES_OL;
CREATE USER SALES_STG IDENTIFIED BY SALES_STG;
CREATE USER SALES_DW IDENTIFIED BY SALES_DW;
GRANT CONNECT, RESOURCE TO SALES_OL;
GRANT CONNECT, RESOURCE TO SALES_STG;
GRANT CONNECT, RESOURCE TO SALES_DW;
```

Step 3

Log into SALE_OL

Run create Table command (see Appendix A for the code)

STEP 4

Log into Sales_STg

Run create table command; (see Appendix B for the code)

STEP 5

Run Util.pkg

Run Util.pkb

RUN SALES_STG.Load_lookup_table.pkg

RUN SALES_STG.Load_lookup_table.pkb

Run LOAD_TRANSACTION_TABLE.pkg

Run LOAD_TRANSACTION_TABLE.pkb

STEP 6

Log into SALES_DW

RUN Create table commands

CREATE TABLE ERROR_DETAIL (see Appendix C for the code)

STEP 7

Run Util.pkg

Run Util.pkb

RUN LOOKUP.pKg

RUN LOOKUP.pkb

RUN Load_Dim_table.pkg

RUN Load_Dim_table.pkb

RUN Load_fact.pkg

Run LOAD_Fact.pkb

CHAPTER SIX

CONCLUSIONS

6.1 Accomplishments

The successful deployment of our sales data mart has allowed us to create a user interface that will permit users to make strategic decisions. This sales data mart provides analysts and end users the ability to improve their bottom line and their market share on the market, since it can give clear information about their business. For example, it helps in evaluating the customer's portfolio as their customer analysis report can summarize the costumer's business value. Due to the fact that we can complete these reports, we are now able to see various helpful facts, such as the top ten customers and products, sales details, etc. The reports can help in analyzing how the company's products are being sold. With the help of techniques such as slicing and dicing, and drilling up and down, we narrow our scope of query in order to see the product performance in more specific categories such as narrowing the results to a state or city level. Time dimensions help in the overall company's evaluation of the profit margin. It helps the end user to evaluate their business on specific time zones like specific quarters or months to compare their product movement and their profit margins for those periods. Overall, on the presentation level, Data Mart improves navigation capabilities and generally more accessible to end users.

6.2 Future Directions

We can further extend this project by increasing the sales data mart with new dimensions like Store, Sales Person, Sales Target and Item. Shipping and Tariff for customers can also be brought as a new dimension to increase the flexibility of the data mart on the reporting side. With an ongoing sales data mart there is more room for analysis on how data marts perform for the purpose of fine tuning such systems. There is always more room for performance tuning, such as dealing with bottlenecking when handling large amounts of data. Platform Upgrades can include changes to the infrastructure, data transport, storage management, database and OLAP system components can be enhanced for the long term growth of the data mart.

APPENDIX A

CODE FOR DEPLOYMENT STEP 3

```
CREATE TABLE SALES_OL.PRA_CHANNELS
(
 CHANNEL_ID   NUMBER,
 NAME         VARCHAR2(20 BYTE)            NOT NULL,
 DESCRIPTION  VARCHAR2(200 BYTE),
 CREATE_USER  VARCHAR2(50 BYTE),
 CREATE_DATE  DATE,
 UPDATE_USER  VARCHAR2(50 BYTE),
 UPDATE_DATE  DATE
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


CREATE TABLE SALES_OL.PRA_CUSTOMERS
(
 CUSTOMER_ID    NUMBER,
 CUSTOMER_NAME  VARCHAR2(200 BYTE)          NOT NULL,
 ADDRESS        VARCHAR2(40 BYTE)           NOT NULL,
 ZIPCODE        VARCHAR2(10 BYTE)           NOT NULL,
 CITY           VARCHAR2(30 BYTE)           NOT NULL,
 STATE          VARCHAR2(40 BYTE),
 COUNTRY        CHAR(2 BYTE)                NOT NULL,
 PHONE_NUMBER   VARCHAR2(25 BYTE),
 EMAIL          VARCHAR2(30 BYTE),
 CREATE_USER    VARCHAR2(50 BYTE),
 CREATE_DATE    DATE,
 UPDATE_USER    VARCHAR2(50 BYTE),
 UPDATE_DATE    DATE
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


CREATE TABLE SALES_OL.PRA_PRODUCTS
(
 PRODUCT_ID      NUMBER,
 PRODUCT_NAME    VARCHAR2(50 BYTE)           NOT NULL,
 PRODUCT_DESC    VARCHAR2(4000 BYTE)         NOT NULL,
 PRODUCT_FAMILY  VARCHAR2(50 BYTE)           NOT NULL,
 PRODUCT_UOM     VARCHAR2(20 BYTE),
 PRODUCT_WEIGHT  VARCHAR2(30 BYTE),
 PRODUCT_PRICE   NUMBER(8,2)             NOT NULL,
 PRODUCT_MSRP    NUMBER(8,2)             NOT NULL,
 CREATE_USER     VARCHAR2(50 BYTE),
 CREATE_DATE     DATE,
 UPDATE_USER     VARCHAR2(50 BYTE),
 UPDATE_DATE     DATE,
 SUPPLIER_ID     NUMBER
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


CREATE TABLE SALES_OL.PRA_PROMOTIONS
(
 PROMOTION_ID          NUMBER,
 PROMOTION_NAME        VARCHAR2(60 BYTE),
 PROMOTION_DETAIL      VARCHAR2(60 BYTE),
 PROMOTION_START_DATE  DATE,
 PROMOTION_END_DATE    DATE,
 CREATE_USER           VARCHAR2(50 BYTE),
 CREATE_DATE           DATE,
 UPDATE_USER           VARCHAR2(50 BYTE),
 UPDATE_DATE           DATE
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;

CREATE TABLE SALES_OL.PRA_PROMOTION_DETAIL
```

```
(
 PRODUCT_ID    NUMBER                    NOT NULL,
 CUSTUMER_ID  NUMBER,
 CHANNEL_ID    NUMBER                    NOT NULL,
 REGION_KEY    NUMBER,
 PROMOTION_ID NUMBER,
 CAL_DATE      DATE,
 CREATE_USER  VARCHAR2(50 BYTE),
 CREATE_DATE  DATE,
 UPDATE_USER  VARCHAR2(50 BYTE),
 UPDATE_DATE  DATE
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


CREATE TABLE SALES_OL.PRA_REGIONS
(
 REGION_KEY   NUMBER,
 CITY         VARCHAR2(50 BYTE),
 STATE        VARCHAR2(50 BYTE),
 COUNTRY      VARCHAR2(50 BYTE),
 CREATE_USER  VARCHAR2(50 BYTE),
 CREATE_DATE  DATE,
 UPDATE_USER  VARCHAR2(50 BYTE),
 UPDATE_DATE  DATE
)
TABLESPACE USERS
PCTUSED    0
```

```
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        BUFFER_POOL    DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


CREATE TABLE SALES_OL.PRA_SALES_DETAIL
(
 PRODUCT_ID    NUMBER                    NOT NULL,
 CUSTUMER_ID   NUMBER,
 CHANNEL_ID    NUMBER                    NOT NULL,
 REGION_KEY    NUMBER,
 QUANTITY_SOLD NUMBER,
 AMOUNT_SOLD   NUMBER(10,2),
 CAL_DATE      DATE,
 CREATE_USER   VARCHAR2(50 BYTE),
 CREATE_DATE   DATE,
 UPDATE_USER   VARCHAR2(50 BYTE),
 UPDATE_DATE   DATE
)
TABLESPACE USERS
PCTUSED    0
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        BUFFER_POOL    DEFAULT
       )
LOGGING
```

```
NOCACHE
NOPARALLEL
MONITORING;


CREATE UNIQUE INDEX SALES_OL.CHANNEL_ID_PK ON
SALES_OL.PRA_CHANNELS
(CHANNEL_ID)
LOGGING
TABLESPACE SYSTEM
PCTFREE    10
INITRANS   2
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
NOPARALLEL;


CREATE UNIQUE INDEX SALES_OL.CUSTOMER_ID ON
SALES_OL.PRA_CUSTOMERS
(CUSTOMER_ID)
LOGGING
TABLESPACE SYSTEM
PCTFREE    10
INITRANS   2
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
NOPARALLEL;
```

```
CREATE UNIQUE INDEX SALES_OL.PRODUCT_ID_PK ON
SALES_OL.PRA_PRODUCTS
(PRODUCT_ID)
LOGGING
TABLESPACE SYSTEM
PCTFREE    10
INITRANS   2
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
NOPARALLEL;

CREATE UNIQUE INDEX SALES_OL.PROMO_ID_PK ON
SALES_OL.PRA_PROMOTIONS
(PROMOTION_ID)
LOGGING
TABLESPACE SYSTEM
PCTFREE    10
INITRANS   2
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
NOPARALLEL;
```

```
CREATE UNIQUE INDEX SALES_OL.REGION_KEY_PK ON
SALES_OL.PRA_REGIONS
(REGION_KEY)
LOGGING
TABLESPACE SYSTEM
PCTFREE    10
INITRANS   2
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
NOPARALLEL;


ALTER TABLE SALES_OL.PRA_CHANNELS ADD (
 CONSTRAINT CHANNEL_ID_PK PRIMARY KEY (CHANNEL_ID)
  USING INDEX
  TABLESPACE SYSTEM
  PCTFREE    10
  INITRANS   2
  MAXTRANS   255
  STORAGE    (
          INITIAL         64K
          MINEXTENTS       1
          MAXEXTENTS       2147483645
          PCTINCREASE      0
          FREELISTS        1
          FREELIST GROUPS  1
         ));


ALTER TABLE SALES_OL.PRA_CUSTOMERS ADD (
 CONSTRAINT CUSTOMER_ID PRIMARY KEY (CUSTOMER_ID)
  USING INDEX
  TABLESPACE SYSTEM
  PCTFREE    10
  INITRANS   2
  MAXTRANS  255
  STORAGE  (
        INITIAL       64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        FREELISTS      1
        FREELIST GROUPS 1
        ));

ALTER TABLE SALES_OL.PRA_PRODUCTS ADD (
 CONSTRAINT PRODUCT_ID_PK PRIMARY KEY (PRODUCT_ID)
  USING INDEX
  TABLESPACE SYSTEM
  PCTFREE   10
  INITRANS  2
  MAXTRANS  255
  STORAGE  (
        INITIAL       64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        FREELISTS      1
        FREELIST GROUPS 1
        ));

ALTER TABLE SALES_OL.PRA_PROMOTIONS ADD (
 CONSTRAINT PROMO_ID_PK PRIMARY KEY (PROMOTION_ID)
  USING INDEX
  TABLESPACE SYSTEM
  PCTFREE   10
  INITRANS  2
  MAXTRANS  255
  STORAGE  (
        INITIAL       64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        FREELISTS      1
        FREELIST GROUPS 1
        ));
```

```
ALTER TABLE SALES_OL.PRA_REGIONS ADD (
 CONSTRAINT REGION_KEY_PK PRIMARY KEY (REGION_KEY)
  USING INDEX
  TABLESPACE SYSTEM
  PCTFREE    10
  INITRANS   2
  MAXTRANS   255
  STORAGE    (
          INITIAL        64K
          MINEXTENTS       1
          MAXEXTENTS      2147483645
          PCTINCREASE      0
          FREELISTS        1
          FREELIST GROUPS  1
         ));


ALTER TABLE SALES_OL.PRA_PROMOTION_DETAIL ADD (
 CONSTRAINT CHANNEL_ID_PROM_FK FOREIGN KEY (CHANNEL_ID)
  REFERENCES SALES_OL.PRA_CHANNELS (CHANNEL_ID));

ALTER TABLE SALES_OL.PRA_PROMOTION_DETAIL ADD (
 CONSTRAINT CUDTOMER_ID_PROM_FK FOREIGN KEY (CUSTUMER_ID)
  REFERENCES SALES_OL.PRA_CUSTOMERS (CUSTOMER_ID));

ALTER TABLE SALES_OL.PRA_PROMOTION_DETAIL ADD (
 CONSTRAINT PRODUCT_PROMO_FK FOREIGN KEY (PRODUCT_ID)
  REFERENCES SALES_OL.PRA_PROMOTIONS (PROMOTION_ID));

ALTER TABLE SALES_OL.PRA_PROMOTION_DETAIL ADD (
 CONSTRAINT PROMO_FK FOREIGN KEY (PROMOTION_ID)
  REFERENCES SALES_OL.PRA_PROMOTIONS (PROMOTION_ID));


ALTER TABLE SALES_OL.PRA_SALES_DETAIL ADD (
 CONSTRAINT CHANNEL_ID_FK FOREIGN KEY (CHANNEL_ID)
  REFERENCES SALES_OL.PRA_CHANNELS (CHANNEL_ID));

ALTER TABLE SALES_OL.PRA_SALES_DETAIL ADD (
 CONSTRAINT CUSTOMER_ID_FK FOREIGN KEY (CUSTUMER_ID)
  REFERENCES SALES_OL.PRA_CUSTOMERS (CUSTOMER_ID));
```

```
ALTER TABLE SALES_OL.PRA_SALES_DETAIL ADD (
 CONSTRAINT PRODUCT_FK_KEY FOREIGN KEY (PRODUCT_ID)
  REFERENCES SALES_OL.PRA_PRODUCTS (PRODUCT_ID));

ALTER TABLE SALES_OL.PRA_SALES_DETAIL ADD (
 CONSTRAINT REGION_SALES_FK FOREIGN KEY (REGION_KEY)
  REFERENCES SALES_OL.PRA_REGIONS (REGION_KEY));


GRANT SELECT ON  SALES_OL.PRA_CHANNELS TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_CUSTOMERS TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_PRODUCTS TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_SALES_DETAIL TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_PROMOTION_DETAIL TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_REGIONS TO SALES_STG;

GRANT SELECT ON  SALES_OL.PRA_PROMOTIONS TO SALES_STG;
```

APPENDIX B

CODE FOR DEPLOYMENT STEP 4

```
CREATE TABLE SALES_STG.ERROR_DETAIL
(
 JOB_NAME          VARCHAR2(50 BYTE),
 TABLE_NAME         VARCHAR2(50 BYTE),
 TABLE_PK_KEY_VALUE  VARCHAR2(50 BYTE),
 ERROR_CODE         VARCHAR2(50 BYTE),
 ERROR_MSG          VARCHAR2(200 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
            INITIAL          64K
            MINEXTENTS       1
            MAXEXTENTS       2147483645
            PCTINCREASE      0
            FREELISTS        1
            FREELIST GROUPS  1
            BUFFER_POOL      DEFAULT
           )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.JOB_CONTROL CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.JOB_CONTROL
(
 JOB_ID    NUMBER,
 JOB_NAME   VARCHAR2(50 BYTE),
 FREQUENCY  VARCHAR2(50 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
            INITIAL          64K
```

```
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.JOB_DETAIL_RUN CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.JOB_DETAIL_RUN
(
 JOB_RUN_ID            NUMBER,
 JOB_ID              NUMBER,
 LOAD_START_DATE        DATE,
 LOAD_END_DATE          DATE,
 STATUS                VARCHAR2(60 BYTE),
 TOTAL_NO_OF_ROWS        NUMBER,
 NUMBER_OF_ROWS_INSERTED  NUMBER,
 NUMBER_OF_ROWS_UPDATED   NUMBER,
 NUMBER_OF_ROWS_ERROR    NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
```

```
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_CHANNELS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_CHANNELS
(
 CHANNEL_ID      NUMBER,
 NAME            VARCHAR2(20 BYTE)       NOT NULL,
 DESCRIPTION     VARCHAR2(200 BYTE),
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY  NUMBER,
 SOURCE_RECORD_ID  VARCHAR2(50 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_CUSTOMERS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_CUSTOMERS
(
 CUSTOMER_ID      NUMBER,
 CUSTOMER_NAME    VARCHAR2(200 BYTE)        NOT NULL,
 ADDRESS          VARCHAR2(40 BYTE)         NOT NULL,
 ZIPCODE          VARCHAR2(10 BYTE)         NOT NULL,
 CITY             VARCHAR2(30 BYTE)         NOT NULL,
 STATE            VARCHAR2(40 BYTE),
 COUNTRY          CHAR(2 BYTE)              NOT NULL,
 PHONE_NUMBER     VARCHAR2(25 BYTE),
 EMAIL            VARCHAR2(30 BYTE),
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY NUMBER,
 SOURCE_RECORD_ID NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_INC_CHANGES CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_INC_CHANGES
(
 TABLE_NAME    VARCHAR2(50 BYTE),
 LAST_RUN_DATE DATE
)
TABLESPACE SYSTEM
```

```
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PRODUCTS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PRODUCTS
(
 PROD_ID          NUMBER(6),
 PRODUCT_NAME     VARCHAR2(50 BYTE)          NOT NULL,
 PRODUCT_DESC     VARCHAR2(4000 BYTE)        NOT NULL,
 PRODUCT_FAMILY   VARCHAR2(50 BYTE)          NOT NULL,
 PRODUCT_UOM      VARCHAR2(20 BYTE),
 PRODUCT_WEIGHT   VARCHAR2(30 BYTE),
 PRODUCT_PRICE    NUMBER(8,2)                NOT NULL,
 PRODUCT_MSRP     NUMBER(8,2)                NOT NULL,
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY NUMBER,
 SOURCE_RECORD_ID NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
```

```
STORAGE    (
        INITIAL        64K
        MINEXTENTS       1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS  1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PROMOTIONS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PROMOTIONS
(
 PROMOTION_ID         NUMBER,
 PROMOTION_NAME       VARCHAR2(60 BYTE),
 PROMOTION_DETAIL     VARCHAR2(60 BYTE),
 PROMOTION_START_DATE  DATE,
 PROMOTION_END_DATE    DATE,
 CREATE_DATE          DATE,
 CREATE_USER          VARCHAR2(50 BYTE),
 UPDATE_USER          VARCHAR2(50 BYTE),
 UPDATE_DATE          DATE,
 SOURCE_TABLE_KEY     NUMBER,
 SOURCE_RECORD_ID     NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS       1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS  1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PROMOTION_TRANSACTION CASCADE
CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PROMOTION_TRANSACTION
(
 CUSTOMER_NAME  VARCHAR2(30 BYTE),
 PRODUCT_NAME   VARCHAR2(50 BYTE),
 CHANNEL_NAME   VARCHAR2(50 BYTE)          NOT NULL,
 CITY           VARCHAR2(50 BYTE),
 PROMOTION_NAME VARCHAR2(50 BYTE),
 CAL_DATE       DATE,
 CREATE_USER    VARCHAR2(50 BYTE),
 CREATE_DATE    DATE,
 UPDATE_USER    VARCHAR2(50 BYTE),
 UPDATE_DATE    DATE
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;
```

```
DROP TABLE SALES_STG.STG_REGION CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_REGION
(
 REGION_KEY            NUMBER,
 CITY                VARCHAR2(50 BYTE),
 STATE                VARCHAR2(50 BYTE),
 COUNTRY               VARCHAR2(50 BYTE),
 CREATE_USER             VARCHAR2(50 BYTE),
 CREATE_DATE             DATE,
 UPDATE_USER             VARCHAR2(50 BYTE),
 UPDATE_DATE             DATE,
 SOURCE_SYSTEM_TABLE_KEY  NUMBER,
 SOURCE_SYSTEM_RECORD_KEY  NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_SALES_TRANSACTION CASCADE
CONSTRAINTS;

CREATE TABLE SALES_STG.STG_SALES_TRANSACTION
(
 PRODUCT_NAME   VARCHAR2(50 BYTE)          NOT NULL,
 CUSTOMER_NAME  VARCHAR2(30 BYTE),
 CHANNEL_NAME   VARCHAR2(50 BYTE)          NOT NULL,
 CITY         VARCHAR2(50 BYTE),
 QUANTITY_SOLD  NUMBER(3)                NOT NULL,
 AMOUNT_SOLD    NUMBER(10,2)              NOT NULL,
 CAL_DATE       DATE,
 CREATE_USER    VARCHAR2(50 BYTE),
 CREATE_DATE    DATE,
 UPDATE_USER    VARCHAR2(50 BYTE),
 UPDATE_DATE    DATE
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;
```

APPENDIX C

CODE FOR DEPLOYMENT STEP 6

```
CREATE TABLE SALES_STG.ERROR_DETAIL
(
 JOB_NAME          VARCHAR2(50 BYTE),
 TABLE_NAME          VARCHAR2(50 BYTE),
 TABLE_PK_KEY_VALUE  VARCHAR2(50 BYTE),
 ERROR_CODE          VARCHAR2(50 BYTE),
 ERROR_MSG          VARCHAR2(200 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.JOB_CONTROL CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.JOB_CONTROL
(
 JOB_ID    NUMBER,
 JOB_NAME  VARCHAR2(50 BYTE),
 FREQUENCY  VARCHAR2(50 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.JOB_DETAIL_RUN CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.JOB_DETAIL_RUN
(
 JOB_RUN_ID            NUMBER,
 JOB_ID              NUMBER,
 LOAD_START_DATE        DATE,
 LOAD_END_DATE         DATE,
 STATUS               VARCHAR2(60 BYTE),
 TOTAL_NO_OF_ROWS       NUMBER,
 NUMBER_OF_ROWS_INSERTED  NUMBER,
 NUMBER_OF_ROWS_UPDATED   NUMBER,
 NUMBER_OF_ROWS_ERROR   . NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
```

```
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_CHANNELS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_CHANNELS
(
 CHANNEL_ID      NUMBER,
 NAME           VARCHAR2(20 BYTE)        NOT NULL,
 DESCRIPTION     VARCHAR2(200 BYTE),
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY  NUMBER,
 SOURCE_RECORD_ID  VARCHAR2(50 BYTE)
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_CUSTOMERS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_CUSTOMERS
(
 CUSTOMER_ID      NUMBER,
 CUSTOMER_NAME    VARCHAR2(200 BYTE)       NOT NULL,
 ADDRESS          VARCHAR2(40 BYTE)        NOT NULL,
 ZIPCODE          VARCHAR2(10 BYTE)        NOT NULL,
 CITY             VARCHAR2(30 BYTE)        NOT NULL,
 STATE            VARCHAR2(40 BYTE),
 COUNTRY          CHAR(2 BYTE)             NOT NULL,
 PHONE_NUMBER     VARCHAR2(25 BYTE),
 EMAIL            VARCHAR2(30 BYTE),
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY  NUMBER,
 SOURCE_RECORD_ID  NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
            INITIAL          64K
            MINEXTENTS       1
            MAXEXTENTS       2147483645
            PCTINCREASE      0
            FREELISTS        1
            FREELIST GROUPS  1
            BUFFER_POOL      DEFAULT
           )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_INC_CHANGES CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_INC_CHANGES
(
 TABLE_NAME     VARCHAR2(50 BYTE),
 LAST_RUN_DATE  DATE
)
TABLESPACE SYSTEM
```

```
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS     1
        MAXEXTENTS     2147483645
        PCTINCREASE    0
        FREELISTS      1
        FREELIST GROUPS 1
        BUFFER_POOL    DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PRODUCTS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PRODUCTS
(
 PROD_ID          NUMBER(6),
 PRODUCT_NAME     VARCHAR2(50 BYTE)      NOT NULL,
 PRODUCT_DESC     VARCHAR2(4000 BYTE)    NOT NULL,
 PRODUCT_FAMILY   VARCHAR2(50 BYTE)      NOT NULL,
 PRODUCT_UOM      VARCHAR2(20 BYTE),
 PRODUCT_WEIGHT   VARCHAR2(30 BYTE),
 PRODUCT_PRICE    NUMBER(8,2)            NOT NULL,
 PRODUCT_MSRP     NUMBER(8,2)            NOT NULL,
 CREATE_DATE      DATE,
 CREATE_USER      VARCHAR2(50 BYTE),
 UPDATE_USER      VARCHAR2(50 BYTE),
 UPDATE_DATE      DATE,
 SOURCE_TABLE_KEY NUMBER,
 SOURCE_RECORD_ID NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
```

```
STORAGE    (
        INITIAL        64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PROMOTIONS CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PROMOTIONS
(
 PROMOTION_ID          NUMBER,
 PROMOTION_NAME        VARCHAR2(60 BYTE),
 PROMOTION_DETAIL      VARCHAR2(60 BYTE),
 PROMOTION_START_DATE  DATE,
 PROMOTION_END_DATE    DATE,
 CREATE_DATE           DATE,
 CREATE_USER           VARCHAR2(50 BYTE),
 UPDATE_USER           VARCHAR2(50 BYTE),
 UPDATE_DATE           DATE,
 SOURCE_TABLE_KEY      NUMBER,
 SOURCE_RECORD_ID      NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL        64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_PROMOTION_TRANSACTION CASCADE
CONSTRAINTS;

CREATE TABLE SALES_STG.STG_PROMOTION_TRANSACTION
(
 CUSTOMER_NAME   VARCHAR2(30 BYTE),
 PRODUCT_NAME    VARCHAR2(50 BYTE),
 CHANNEL_NAME    VARCHAR2(50 BYTE)          NOT NULL,
 CITY            VARCHAR2(50 BYTE),
 PROMOTION_NAME  VARCHAR2(50 BYTE),
 CAL_DATE        DATE,
 CREATE_USER     VARCHAR2(50 BYTE),
 CREATE_DATE     DATE,
 UPDATE_USER     VARCHAR2(50 BYTE),
 UPDATE_DATE     DATE
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS      1
        MAXEXTENTS      2147483645
        PCTINCREASE     0
        FREELISTS       1
        FREELIST GROUPS 1
        BUFFER_POOL     DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;
```

```
DROP TABLE SALES_STG.STG_REGION CASCADE CONSTRAINTS;

CREATE TABLE SALES_STG.STG_REGION
(
 REGION_KEY              NUMBER,
 CITY                 VARCHAR2(50 BYTE),
 STATE                 VARCHAR2(50 BYTE),
 COUNTRY                 VARCHAR2(50 BYTE),
 CREATE_USER              VARCHAR2(50 BYTE),
 CREATE_DATE              DATE,
 UPDATE_USER              VARCHAR2(50 BYTE),
 UPDATE_DATE              DATE,
 SOURCE_SYSTEM_TABLE_KEY   NUMBER,
 SOURCE_SYSTEM_RECORD_KEY  NUMBER
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL         64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;


DROP TABLE SALES_STG.STG_SALES_TRANSACTION CASCADE
CONSTRAINTS;

CREATE TABLE SALES_STG.STG_SALES_TRANSACTION
(
 PRODUCT_NAME   VARCHAR2(50 BYTE)           NOT NULL,
 CUSTOMER_NAME  VARCHAR2(30 BYTE),
 CHANNEL_NAME   VARCHAR2(50 BYTE)          NOT NULL,
 CITY          VARCHAR2(50 BYTE),
 QUANTITY_SOLD  NUMBER(3)                  NOT NULL,
 AMOUNT_SOLD    NUMBER(10,2)               NOT NULL,
 CAL_DATE       DATE,
 CREATE_USER    VARCHAR2(50 BYTE),
 CREATE_DATE    DATE,
 UPDATE_USER    VARCHAR2(50 BYTE),
 UPDATE_DATE    DATE
)
TABLESPACE SYSTEM
PCTUSED    40
PCTFREE    10
INITRANS   1
MAXTRANS   255
STORAGE    (
        INITIAL          64K
        MINEXTENTS       1
        MAXEXTENTS       2147483645
        PCTINCREASE      0
        FREELISTS        1
        FREELIST GROUPS  1
        BUFFER_POOL      DEFAULT
       )
LOGGING
NOCACHE
NOPARALLEL
MONITORING;
```

APPENDIX D

DATA DICTIONARY

SALES_OL

PRA_CHANNELS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_CHANNELS	CHANNEL_ID	NUMBER	Chennal Identification Number, Primary Key
PRA_CHANNELS	NAME	VARCHAR2	NAME
PRA_CHANNELS	DESCRIPTION	VARCHAR2	DESCRIPTION
PRA_CHANNELS	CREATE_USER	VARCHAR2	CREATE USER
PRA_CHANNELS	CREATE_DATE	DATE	CREATE DATE
PRA_CHANNELS	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_CHANNELS	UPDATE_DATE	DATE	UPDATE DATE

PRA_CUSTOMERS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_CUSTOMERS	CUSTOMER_ID	NUMBER	Customer Identification Number, Primary Key
PRA_CUSTOMERS	CUSTOMER_NAME	VARCHAR2	CUSTOMER NAME
PRA_CUSTOMERS	ADDRESS	VARCHAR2	ADDRESS
PRA_CUSTOMERS	ZIPCODE	VARCHAR2	ZIPCODE
PRA_CUSTOMERS	CITY	VARCHAR2	CITY
PRA_CUSTOMERS	STATE	VARCHAR2	STATE
PRA_CUSTOMERS	COUNTRY	CHAR	COUNTRY
PRA_CUSTOMERS	PHONE_NUMBER	VARCHAR2	PHONE NUMBER
PRA_CUSTOMERS	EMAIL	VARCHAR2	EMAIL
PRA_CUSTOMERS	CREATE_USER	VARCHAR2	CREATE USER
PRA_CUSTOMERS	CREATE_DATE	DATE	CREATE DATE
PRA_CUSTOMERS	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_CUSTOMERS	UPDATE_DATE	DATE	UPDATE DATE

PRA_PRODUCTS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_PRODUCTS	PRODUCT_ID	NUMBER	PRODUCT Identification Number, Primary Key
PRA_PRODUCTS	PRODUCT_NAME	VARCHAR2	PRODUCT NAME
PRA_PRODUCTS	PRODUCT_DESC	VARCHAR2	PRODUCT DESC
PRA_PRODUCTS	PRODUCT_FAMILY	VARCHAR2	PRODUCT FAMILY
PRA_PRODUCTS	PRODUCT_UOM	VARCHAR2	PRODUCT UOM
PRA_PRODUCTS	PRODUCT_WEIGHT	VARCHAR2	PRODUCT WEIGHT
PRA_PRODUCTS	PRODUCT_PRICE	NUMBER	PRODUCT PRICE
PRA_PRODUCTS	PRODUCT_MSRP	NUMBER	PRODUCT MSRP
PRA_PRODUCTS	CREATE_USER	VARCHAR2	CREATE USER
PRA_PRODUCTS	CREATE_DATE	DATE	CREATE DATE
PRA_PRODUCTS	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_PRODUCTS	UPDATE_DATE	DATE	UPDATE DATE
PRA_PRODUCTS	SUPPLIER_ID	NUMBER	SUPPLIER ID

PRA_PROMOTIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_ PROMOTIONS	PROMOTION_ID	NUMBER	PROMOTION Identification Number, Primary Key
PRA_ PROMOTIONS	PROMOTION_NAME	VARCHAR2	PROMOTION NAME
PRA_ PROMOTIONS	PROMOTION_DETAIL	VARCHAR2	PROMOTION DETAIL
PRA_ PROMOTIONS	PROMOTION_START_DATE	DATE	PROMOTION START DATE
PRA_ PROMOTIONS	PROMOTION_END_DATE	DATE	PROMOTION END DATE
PRA_ PROMOTIONS	CREATE_USER	VARCHAR2	CREATE USER
PRA_ PROMOTIONS	CREATE_DATE	DATE	CREATE DATE
PRA_ PROMOTIONS	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_ PROMOTIONS	UPDATE_DATE	DATE	UPDATE DATE

PRA_REGIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_REGIONS	UPDATE_DATE	DATE	UPDATE DATE
PRA_REGIONS	REGION_KEY	NUMBER	REGION Identification Number,Primary Key
PRA_REGIONS	CITY	VARCHAR2	CITY
PRA_REGIONS	STATE	VARCHAR2	STATE
PRA_REGIONS	COUNTRY	VARCHAR2	COUNTRY
PRA_REGIONS	CREATE_USER	VARCHAR2	CREATE USER
PRA_REGIONS	CREATE_DATE	DATE	CREATE DATE
PRA_REGIONS	UPDATE_USER	VARCHAR2	UPDATE USER

PRA_PROMOTION_DETAILS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_PROMOTION_DETAIL	PRODUCT_ID	NUMBER	PRODUCT ID
PRA_PROMOTION_DETAIL	CUSTUMER_ID	NUMBER	CUSTUMER ID
PRA_PROMOTION_DETAIL	CHANNEL_ID	NUMBER	CHANNEL ID
PRA_PROMOTION_DETAIL	REGION_KEY	NUMBER	REGION KEY
PRA_PROMOTION_DETAIL	PROMOTION_ID	NUMBER	PROMOTION ID
PRA_PROMOTION_DETAIL	CAL_DATE	DATE	CAL DATE
PRA_PROMOTION_DETAIL	CREATE_USER	VARCHAR2	CREATE USER
PRA_PROMOTION_DETAIL	CREATE_DATE	DATE	CREATE DATE
PRA_PROMOTION_DETAIL	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_PROMOTION_DETAIL	UPDATE_DATE	DATE	UPDATE DATE

PRA_SALES_DETAIL

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PRA_SALES_DETAIL	PRODUCT_ID	NUMBER	PRODUCT ID
PRA_SALES_DETAIL	CUSTUMER_ID	NUMBER	CUSTUMER ID
PRA_SALES_DETAIL	CHANNEL_ID	NUMBER	CHANNEL ID
PRA_SALES_DETAIL	REGION_KEY	NUMBER	REGION KEY
PRA_SALES_DETAIL	QUANTITY_SOLD	NUMBER	QUANTITY SOLD
PRA_SALES_DETAIL	AMOUNT_SOLD	NUMBER	AMOUNT SOLD
PRA_SALES_DETAIL	CAL_DATE	DATE	CAL DATE
PRA_SALES_DETAIL	CREATE_USER	VARCHAR2	CREATE USER
PRA_SALES_DETAIL	CREATE_DATE	DATE	CREATE DATE
PRA_SALES_DETAIL	UPDATE_USER	VARCHAR2	UPDATE USER
PRA_SALES_DETAIL	UPDATE_DATE	DATE	UPDATE DATE

SALES_STG

STG_RIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_REGION	REGION_KEY	NUMBER	REGION Identification Number,Primary Key
STG_REGION	CITY	VARCHAR2	CITY
STG_REGION	STATE	VARCHAR2	STATE
STG_REGION	COUNTRY	VARCHAR2	COUNTRY
STG_REGION	CREATE_USER	VARCHAR2	CREATE USER
STG_REGION	CREATE_DATE	DATE	CREATE DATE
STG_REGION	UPDATE_USER	VARCHAR2	UPDATE USER
STG_REGION	UPDATE_DATE	DATE	UPDATE DATE
STG_REGION	SOURCE_SYSTEM_TABLE_KEY	NUMBER	SOURCE SYSTEM TABLE KEY
STG_REGION	SOURCE_SYSTEM_RECORD_KEY	NUMBER	SOURCE SYSTEM RECORD KEY

STG_PRODUCTS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_PRODUCTS	PROD_ID	NUMBER	PRODUCT Identification Number,Primary Key
STG_PRODUCTS	PRODUCT_NAME	VARCHAR2	PRODUCT NAME
STG_PRODUCTS	PRODUCT_DESC	VARCHAR2	PRODUCT DESC
STG_PRODUCTS	PRODUCT_FAMILY	VARCHAR2	PRODUCT FAMILY
STG_PRODUCTS	PRODUCT_UOM	VARCHAR2	PRODUCT UOM
STG_PRODUCTS	PRODUCT_WEIGHT	VARCHAR2	PRODUCT WEIGHT
STG_PRODUCTS	PRODUCT_PRICE	NUMBER	PRODUCT PRICE
STG_PRODUCTS	PRODUCT_MSRP	NUMBER	PRODUCT MSRP
STG_PRODUCTS	CREATE_DATE	DATE	CREATE DATE
STG_PRODUCTS	CREATE_USER	VARCHAR2	CREATE USER
STG_PRODUCTS	UPDATE_USER	VARCHAR2	UPDATE USER
STG_PRODUCTS	UPDATE_DATE	DATE	UPDATE DATE
STG_PRODUCTS	SOURCE_ TABLE_KEY	NUMBER	SOURCE TABLE KEY
STG_PRODUCTS	SOURCE_ RECORD_ID	NUMBER	SOURCE RECORD ID

STG_CUSTOMER

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_CUSTOMERS	CUSTOMER_ID	NUMBER	CUSTOMER Identification Number, Primary Key
STG_CUSTOMERS	CUSTOMER_NAME	VARCHAR2	CUSTOMER NAME
STG_CUSTOMERS	ADDRESS	VARCHAR2	ADDRESS
STG_CUSTOMERS	ZIPCODE	VARCHAR2	ZIPCODE
STG_CUSTOMERS	CITY	VARCHAR2	CITY
STG_CUSTOMERS	STATE	VARCHAR2	STATE
STG_CUSTOMERS	COUNTRY	CHAR	COUNTRY
STG_CUSTOMERS	PHONE_NUMBER	VARCHAR2	PHONE NUMBER
STG_CUSTOMERS	EMAIL	VARCHAR2	EMAIL
STG_CUSTOMERS	CREATE_DATE	DATE	CREATE DATE
STG_CUSTOMERS	CREATE_USER	VARCHAR2	CREATE USER
STG_CUSTOMERS	UPDATE_USER	VARCHAR2	UPDATE USER
STG_CUSTOMERS	UPDATE_DATE	DATE	UPDATE DATE
STG_CUSTOMERS	SOURCE_ TABLE_KEY	NUMBER	SOURCE TABLE KEY
STG_CUSTOMERS	SOURCE_ RECORD_ID	NUMBER	SOURCE RECORD ID

STG_PROMOTIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_ PROMOTIONS	PROMOTION_ID	NUMBER	PROMOION Identification Number, Primary Key
STG_ PROMOTIONS	PROMOTION_NAME	VARCHAR2	PROMOTION NAME
STG_ PROMOTIONS	PROMOTION_DETAIL	VARCHAR2	PROMOTION DETAIL
STG_ PROMOTIONS	PROMOTION_ START_DATE	DATE	PROMOTION START DATE
STG_ PROMOTIONS	PROMOTION_ END_DATE	DATE	PROMOTION END DATE
STG_ PROMOTIONS	CREATE_DATE	DATE	CREATE DATE
STG_ PROMOTIONS	CREATE_USER	VARCHAR2	CREATE USER
STG_ PROMOTIONS	UPDATE_USER	VARCHAR2	UPDATE USER
STG_ PROMOTIONS	UPDATE_DATE	DATE	UPDATE DATE
STG_ PROMOTIONS	SOURCE_ TABLE_KEY	NUMBER	SOURCE TABLE KEY
STG_ PROMOTIONS	SOURCE_ RECORD_ID	NUMBER	SOURCE RECORD ID

STG_PROMOTION_TRANSACTION

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_PROMOTION_ TRANSACTION	CUSTOMER_NAME	VARCHAR2	CUSTOMER NAME
STG_PROMOTION_ TRANSACTION	PRODUCT_NAME	VARCHAR2	PRODUCT NAME
STG_PROMOTION_ TRANSACTION	CHANNEL_NAME	VARCHAR2	CHANNEL NAME
STG_PROMOTION_ TRANSACTION	CITY	VARCHAR2	CITY
STG_PROMOTION_ TRANSACTION	PROMOTION_ NAME	VARCHAR2	PROMOTION NAME
STG_PROMOTION_ TRANSACTION	CAL_DATE	DATE	CAL DATE
STG_PROMOTION_ TRANSACTION	CREATE_USER	VARCHAR2	CREATE USER
STG_PROMOTION_ TRANSACTION	CREATE_DATE	DATE	CREATE DATE
STG_PROMOTION_ TRANSACTION	UPDATE_USER	VARCHAR2	UPDATE USER
STG_PROMOTION_ TRANSACTION	UPDATE_DATE	DATE	UPDATE DATE

STG_SALES_TRANSACTION

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
STG_SALES_ TRANSACTION	CAL_DATE	DATE	CAL DATE
STG_SALES_ TRANSACTION	CREATE_USER	VARCHAR2	CREATE USER
STG_SALES_ TRANSACTION	CREATE_DATE	DATE	CREATE DATE
STG_SALES_ TRANSACTION	UPDATE_USER	VARCHAR2	UPDATE USER
STG_SALES_ TRANSACTION	UPDATE_DATE	DATE	UPDATE DATE
STG_SALES_ TRANSACTION	PRODUCT_NAME	VARCHAR2	PRODUCT NAME
STG_SALES_ TRANSACTION	CUSTOMER_NAME	VARCHAR2	CUSTOMER NAME
STG_SALES_ TRANSACTION	CHANNEL_NAME	VARCHAR2	CHANNEL NAME
STG_SALES_ TRANSACTION	CITY	VARCHAR2	CITY
STG_SALES_ TRANSACTION	QUANTITY_SOLD	NUMBER	QUANTITY SOLD
STG_SALES_ TRANSACTION	AMOUNT_SOLD	NUMBER	AMOUNT SOLD

SLES_DW

SALES_DIM_CHANNELS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_DIM_ CHANNELS	CHANNEL_KEY	NUMBER	Channel Identification Number, Primary Key
SALES_DIM_ CHANNELS	NAME	VARCHAR2	NAME
SALES_DIM_ CHANNELS	DESCRIPTION	VARCHAR2	DESCRIPTION
SALES_DIM_ CHANNELS	CREATE_DATE	DATE	CREATE DATE
SALES_DIM_ CHANNELS	CREATE_USER	VARCHAR2	CREATE USER
SALES_DIM_ CHANNELS	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_DIM_ CHANNELS	UPDATE_DATE	DATE	UPDATE DATE
SALES_DIM_ CHANNELS	SOURCE_ TABLE_KEY	NUMBER	SOURCE TABLE KEY
SALES_DIM_ CHANNELS	SOURCE_ RECORD_ID	NUMBER	SOURCE RECORD ID

SALES_DIM_CUSTOMERS

TABLE_NAME	COLUMN_NAME	DATA TYPE	DESCRIPTION
SALES_DIM_ CUSTOMERS	CUST_KEY	NUMBER	CUSTOMER Identification Number, Primary Key
SALES_DIM_ CUSTOMERS	CUSTOMER_NAME	VARCHAR2	CUSTOMER NAME
SALES_DIM_ CUSTOMERS	ADDRESS	VARCHAR2	ADDRESS
SALES_DIM_ CUSTOMERS	ZIPCODE	VARCHAR2	ZIPCODE
SALES_DIM_ CUSTOMERS	CITY	VARCHAR2	CITY
SALES_DIM_ CUSTOMERS	STATE	VARCHAR2	STATE
SALES_DIM_ CUSTOMERS	COUNTRY	CHAR	COUNTRY
SALES_DIM_ CUSTOMERS	PHONE_NUMBER	VARCHAR2	PHONE NUMBER
SALES_DIM_ CUSTOMERS	EMAIL	VARCHAR2	EMAIL
SALES_DIM_ CUSTOMERS	CREATE_DATE	DATE	CREATE DATE
SALES_DIM_ CUSTOMERS	CREATE_USER	VARCHAR2	CREATE USER
SALES_DIM_ CUSTOMERS	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_DIM_ CUSTOMERS	UPDATE_DATE	DATE	UPDATE DATE
SALES_DIM_ CUSTOMERS	SOURCE_TABLE_KEY	NUMBER	SOURCE TABLE KEY
SALES_DIM_ CUSTOMERS	SOURCE_RECORD_ID	NUMBER	SOURCE RECORD ID

SALES_DIM_PRODUCTS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_DIM_PRODUCTS	PROD_KEY	NUMBER	PRODUCT Identification Number, Primary Key
SALES_DIM_PRODUCTS	PRODUCT_NAME	VARCHAR2	PRODUCT NAME
SALES_DIM_PRODUCTS	PRODUCT_DESC	VARCHAR2	PRODUCT DESC
SALES_DIM_PRODUCTS	PRODUCT_FAMILY	VARCHAR2	PRODUCT FAMILY
SALES_DIM_PRODUCTS	PRODUCT_UOM	VARCHAR2	PRODUCT UOM
SALES_DIM_PRODUCTS	PRODUCT_WEIGHT	VARCHAR2	PRODUCT WEIGHT
SALES_DIM_PRODUCTS	PRODUCT_PRICE	NUMBER	PRODUCT PRICE
SALES_DIM_PRODUCTS	PRODUCT_MSRP	NUMBER	PRODUCT MSRP
SALES_DIM_PRODUCTS	CREATE_DATE	DATE	CREATE DATE
SALES_DIM_PRODUCTS	CREATE_USER	VARCHAR2	CREATE USER
SALES_DIM_PRODUCTS	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_DIM_PRODUCTS	UPDATE_DATE	DATE	UPDATE DATE
SALES_DIM_PRODUCTS	SOURCE_TABLE_KEY	NUMBER	SOURCE TABLE KEY
SALES_DIM_PRODUCTS	SOURCE_RECORD_ID	NUMBER	SOURCE RECORD ID

SALES_DIM_PROMOTIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_DIM_ PROMOTIONS	CREATE_DATE	DATE	CREATE DATE
SALES_DIM_ PROMOTIONS	CREATE_USER	VARCHAR2	CREATE USER
SALES_DIM_ PROMOTIONS	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_DIM_ PROMOTIONS	UPDATE_DATE	DATE	UPDATE DATE
SALES_DIM_ PROMOTIONS	SOURCE_TABLE_KEY	NUMBER	SOURCE TABLE KEY
SALES_DIM_ PROMOTIONS	SOURCE_RECORD_ID	NUMBER	SOURCE RECORD ID
SALES_DIM_ PROMOTIONS	PROMO_KEY	NUMBER	PROMOTION Identification Number, Primary Key
SALES_DIM_ PROMOTIONS	PROMOTION_NAME	VARCHAR2	PROMOTION NAME
SALES_DIM_ PROMOTIONS	PROMOTION_DETAIL	VARCHAR2	PROMOTION DETAIL
SALES_DIM_ PROMOTIONS	PROMOTION_ START_DATE	DATE	PROMOTION START DATE
SALES_DIM_ PROMOTIONS	PROMOTION_ END_DATE	DATE	PROMOTION END DATE

SALES_DIM_REGIONS

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_DIM_ REGIONS	REGION_KEY	NUMBER	REGION Identification Number, Primary Key
SALES_DIM_ REGIONS	CITY	VARCHAR2	CITY
SALES_DIM_ REGIONS	STATE	VARCHAR2	STATE
SALES_DIM_ REGIONS	COUNTRY	VARCHAR2	COUNTRY
SALES_DIM_ REGIONS	CREATE_USER	VARCHAR2	CREATE USER
SALES_DIM_ REGIONS	CREATE_DATE	DATE	CREATE DATE
SALES_DIM_ REGIONS	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_DIM_ REGIONS	UPDATE_DATE	DATE	UPDATE DATE
SALES_DIM_ REGIONS	SOURCE_SYSTEM_ TABLE_KEY	NUMBER	SOURCE SYSTEM TABLE KEY
SALES_DIM_ REGIONS	SOURCE_SYSTEM_ RECORD_KEY	NUMBER	SOURCE SYSTEM RECORD KEY

SALES_FACT

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_FACT	PROD_KEY	NUMBER	PROD KEY
SALES_FACT	CUST_KEY	NUMBER	CUST KEY
SALES_FACT	CHANNEL_KEY	NUMBER	CHANNEL KEY
SALES_FACT	REGION_KEY	NUMBER	REGION KEY
SALES_FACT	QUANTITY_SOLD	NUMBER	QUANTITY SOLD
SALES_FACT	AMOUNT_SOLD	NUMBER	AMOUNT SOLD
SALES_FACT	TIME_KEY	NUMBER	TIME KEY
SALES_FACT	CREATE_USER	VARCHAR2	CREATE USER
SALES_FACT	CREATE_DATE	DATE	CREATE DATE
SALES_FACT	UPDATE_USER	VARCHAR2	UPDATE USER
SALES_FACT	UPDATE_DATE	DATE	UPDATE DATE

PROMOTION_FACT_DETAILES

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
PROMOTION_ FACT_DETAILES	CUST_KEY	NUMBER	CUST KEY
PROMOTION_ FACT_DETAILES	CHANNEL_KEY	NUMBER	CHANNEL KEY
PROMOTION_ FACT_DETAILES	PRODUCT_KEY	NUMBER	PRODUCT KEY
PROMOTION_ FACT_DETAILES	REGION_KEY	NUMBER	REGION KEY
PROMOTION_ FACT_DETAILES	PRMO_COUNTER	NUMBER	PRMO COUNTER
PROMOTION_ FACT_DETAILES	PROMOTION_KEY	NUMBER	PROMOTION KEY
PROMOTION_ FACT_DETAILES	TIME_KEY	NUMBER	TIME KEY
PROMOTION_ FACT_DETAILES	CREATE_USER	VARCHAR2	CREATE USER
PROMOTION_ FACT_DETAILES	CREATE_DATE	DATE	CREATE DATE
PROMOTION_ FACT_DETAILES	UPDATE_USER	VARCHAR2	UPDATE USER
PROMOTION_ FACT_DETAILES	UPDATE_DATE	DATE	UPDATE DATE

SALES_TIME_DIM

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DESCRIPTION
SALES_TIME_DIM	TIME_KEY	NUMBER	DATE Identification Number,Primary Key
SALES_TIME_DIM	CALENDAR_DATE	DATE	CALENDAR DATE
SALES_TIME_DIM	CALENDAR_DAY_ IN_YEAR_NUMBER	NUMBER	CALENDAR DAY IN YEAR NUMBER
SALES_TIME_DIM	CALENDAR_MTH_ END_DATE	DATE	CALENDAR MTH END DATE
SALES_TIME_DIM	CALENDAR_MTH_END_IND	VARCHAR2	CALENDAR MTH END IND
SALES_TIME_DIM	CALENDAR_MTH_NAME	VARCHAR2	CALENDAR MTH NAME
SALES_TIME_DIM	CALENDAR_MTH_START_DATE	DATE	CALENDAR MTH START DATE
SALES_TIME_DIM	CALENDAR_MTH_START_IND	VARCHAR2	CALENDAR MTH START IND
SALES_TIME_DIM	CALENDAR_QTR_END_DATE	DATE	CALENDAR QTR END DATE
SALES_TIME_DIM	CALENDAR_QTR_END_IND	VARCHAR2	CALENDAR QTR END IND
SALES_TIME_DIM	CALENDAR_QTR_IN_ YEAR_NUMBER	NUMBER	CALENDAR QTR IN YEAR NUMBER
SALES_TIME_DIM	CALENDAR_QTR_NAME	VARCHAR2	CALENDAR QTR NAME
SALES_TIME_DIM	CALENDAR_QTR_START_DATE	DATE	CALENDAR QTR START DATE
SALES_TIME_DIM	CALENDAR_QTR_START_IND	VARCHAR2	CALENDAR QTR START IND
SALES_TIME_DIM	CALENDAR_WEEK_NAME	VARCHAR2	CALENDAR WEEK NAME
SALES_TIME_DIM	CALENDAR_WEEK_IN_ YEAR_NUMBER	NUMBER	CALENDAR WEEK IN YEAR NUMBER
SALES_TIME_DIM	CALENDAR_YEAR_NAME	VARCHAR2	CALENDAR YEAR NAME

APPENDIX E

CODE

CODE

```
/* SALES_STG.Load_lookup_table.pkg */

CREATE OR REPLACE PACKAGE SALES_STG.Load_lookup_table AS

 PROCEDURE LOAD_CHANNEL;
 PROCEDURE Load_Product;
 Procedure Load_Customer;
 Procedure Load_region;
 Procedure Load_promotion;

 End;
/

/* SALES_STG.Load_lookup_table.pkb */


CREATE OR REPLACE PACKAGE BODY SALES_STG.Load_lookup_table IS

 PROCEDURE Initialize_Incremental_date IS
 V_run_date Date;
 BEGIN

     BEGIN

     SELECT LAST_RUN_DATE INTO V_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='PRODUCT';
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_run_date FROM SALES_OL.PRODUCTS;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_run_date WHERE TABLE_NAME='PRODUCT';
    END;



    BEGIN

     SELECT LAST_RUN_DATE INTO V_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='REGION';
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_run_date FROM SALES_OL.REGION;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_run_date WHERE TABLE_NAME='REGION';
    END;



 END;
```

```
/* Procedure LOAD_CHANNEL */

 PROCEDURE LOAD_CHANNEL  IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_channel_id NUmber;
 CURSOR channel_cur IS SELECT CHANNEL_ID,CHANNEL_DESC,CHANNEL_CLASS,update_date FROM
SALES_OL.CHANNELS WHERE
 Update_date > v_last_run_date order by update_date ;
 channel_rec channel_cur%Rowtype;

 BEGIN

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
      FROM Dual;

     SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Channel';

     SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

     util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
              'IN_PROCESS',NULL,NULL,NULL,NULL);
    BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='CHANNEL';

      IF (V_last_run_date IS NULL) THEN

        SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.channels;
        UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
        COMMIT;
      END IF;

     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.channels;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
     COMMIT;
    END;

    FOR channel_rec IN channel_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM STG_CHANNELS WHERE CHANNEL_DESC =
channel_rec.CHANNEL_DESC;

      IF (v_cnt = 0) THEN

        BEGIN
        V_last_run_date:=channel_rec.update_date;
         Insert INTO stg_channels VALUES(CHANNEL_KEY_SEQ.nextval,channel_rec.CHANNEL_DESC,
                             channel_rec.CHANNEL_CLASS,SYSDATE,USER,USER,SYSDATE,
                             null,channel_rec.channel_id);
         v_insert_cnt:=v_insert_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         COMMIT;
         EXCEPTION
         WHEN OTHERS THEN

           v_error_cnt:= v_error_cnt+1;
           v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
           DBMS_OUTPUT.PUT_LINE('ERROR');
           RAISE;

         END;

      ELSE
        V_last_run_date:=channel_rec.update_date;
         SELECT CHANNEL_ID INTO v_CHANNEL_ID FROM stg_channels WHERE
CHANNEL_DESC=channel_rec.CHANNEL_DESC;

         BEGIN

         DBMS_OUTPUT.PUT_LINE('ELSE');

         UPDATE stg_channels SET CHANNEL_CLASS=channel_rec.CHANNEL_CLASS,Update_user=user,
                        update_date=SYSDATE WHERE CHANNEL_ID=v_CHANNEL_ID;
                        COMMIT;
         v_Update_cnt:=v_Update_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

         EXCEPTION
         WHEN OTHERS THEN
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          v_error_cnt:= v_error_cnt+1;


         END;
      END IF;
    END LOOP;

        SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
       FROM Dual;

       UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                        TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                        NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                        NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                        NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
        UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
       Commit;
  END;

/* Procedure LOAD_PRODUCT */

 PROCEDURE Load_Product IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_product_KEY NUmber;
 CURSOR product_cur IS SELECT
PROD_ID,PROD_NAME,PROD_DESC,PROD_SUBCATEGORY,PROD_SUBCAT_DESC,
               PROD_CATEGORY,PROD_CAT_DESC,PROD_WEIGHT_CLASS,PROD_UNIT_OF_MEASURE,
               PROD_PACK_SIZE,PROD_STATUS,PROD_LIST_PRICE,PROD_MIN_PRICE,update_date
                FROM SALES_OL.products WHERE   Update_date > v_last_run_date ORDER BY update_date;
 product_rec product_cur%Rowtype;
 BEGIN

    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
       FROM Dual;

      SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Product';

      SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

      util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
               'IN_PROCESS',NULL,NULL,NULL,NULL);
  BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='PRODUCT';

      IF (V_last_run_date IS NULL) THEN

       SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.productS;
       UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
       commit;
      END IF;
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.productS;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
    END;

  FOR product_rec IN product_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM stg_products WHERE PROD_NAME = product_rec.PROD_NAME;

      IF (v_cnt = 0) THEN

        BEGIN

         V_last_run_date :=product_rec.update_date;
        Insert INTO stg_products VALUES(PROD_KEY_SEQ.nextval,product_rec.PROD_NAME,
                            product_rec.PROD_DESC,product_rec.PROD_SUBCATEGORY,
                            product_rec.PROD_SUBCAT_DESC,
                            product_rec.PROD_CATEGORY,product_rec.PROD_CAT_DESC,
                            product_rec.PROD_WEIGHT_CLASS,product_rec.PROD_UNIT_OF_MEASURE,
                            product_rec.PROD_PACK_SIZE,product_rec.PROD_STATUS,
                            product_rec.PROD_LIST_PRICE,product_rec.PROD_MIN_PRICE,SYSDATE,USER,
                            USER,SYSDATE,
                            null,product_rec.PROD_ID);
         v_insert_cnt:=v_insert_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         COMMIT;
         EXCEPTION
         WHEN OTHERS THEN

           v_error_cnt:= v_error_cnt+1;
           v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
           DBMS_OUTPUT.PUT_LINE('ERROR');
           RAISE;

         END;
      ELSE

        V_last_run_date :=product_rec.update_date;

        SELECT prod_id INTO v_product_KEY FROM stg_products WHERE PROD_NAME =
product_rec.PROD_NAME;

        BEGIN

        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE stg_products SET PROD_NAME=product_rec.PROD_NAME,
                    PROD_DESC=product_rec.PROD_DESC,
                    PROD_SUBCATEGORY=product_rec.PROD_SUBCATEGORY,
                    PROD_SUBCAT_DESC=product_rec.PROD_SUBCAT_DESC,
                    PROD_CATEGORY=product_rec.PROD_CATEGORY,
                    PROD_CAT_DESC=product_rec.PROD_CAT_DESC,
                    PROD_WEIGHT_CLASS=product_rec.PROD_WEIGHT_CLASS,
                    PROD_UNIT_OF_MEASURE=product_rec.PROD_UNIT_OF_MEASURE,
                    PROD_PACK_SIZE=product_rec.PROD_PACK_SIZE,
                    PROD_STATUS=product_rec.PROD_STATUS,
                    PROD_LIST_PRICE=product_rec.PROD_LIST_PRICE,
                    PROD_MIN_PRICE=product_rec.PROD_MIN_PRICE,
                    Update_user=user,update_date=SYSDATE WHERE prod_id=v_product_key;
                    COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         v_error_cnt:= v_error_cnt+1;


        END;


      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                   TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                   NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                   NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                   NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
      UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
      Commit;

 END;

/* PROCEDURE LOAD_CUSTOMER */

 Procedure Load_Customer IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_customer_KEY NUmber;
 CURSOR customer_cur IS SELECT CUST_ID,CUST_FIRST_NAME,CUST_LAST_NAME,CUST_GENDER,
                    CUST_YEAR_OF_BIRTH,CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,
                    CUST_POSTAL_CODE,CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID,
                    CUST_MAIN_PHONE_NUMBER,CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,
                    CUST_EMAIL,update_date FROM SALES_OL.customers
                     WHERE   Update_date > v_last_run_date order by update_date;
 customer_rec customer_cur%Rowtype;
 BEGIN

    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
      FROM Dual;

     SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Customer';

     SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

     util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
               'IN_PROCESS',NULL,NULL,NULL,NULL);

   BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='CUSTOMER';

      IF (V_last_run_date IS NULL) THEN

       SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.CUSTOMERS;
       UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
      END IF;
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.CUSTOMERS;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
    END;

   FOR customer_rec IN customer_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM stg_customers WHERE CUST_EMAIL =
customer_rec.CUST_EMAIL;

      IF (v_cnt = 0) THEN

        BEGIN
         V_last_run_date:=customer_rec.update_date;
        Insert INTO stg_customers VALUES(customer_KEY_SEQ.nextval,customer_rec.CUST_FIRST_NAME,
                               customer_rec.CUST_LAST_NAME,
                               customer_rec.CUST_GENDER,
                               customer_rec.CUST_YEAR_OF_BIRTH,
                               customer_rec.CUST_MARITAL_STATUS,
                               customer_rec.CUST_STREET_ADDRESS,
                               customer_rec.CUST_POSTAL_CODE,
                               customer_rec.CUST_CITY,
                               customer_rec.CUST_STATE_PROVINCE,
                               customer_rec.COUNTRY_ID,
                               customer_rec.CUST_MAIN_PHONE_NUMBER,
                               customer_rec.CUST_INCOME_LEVEL,
                               customer_rec.CUST_CREDIT_LIMIT,
                               customer_rec.CUST_EMAIL,SYSDATE,USER,
                              USER,SYSDATE,
                              null,customer_rec.cust_ID);
        v_insert_cnt:=v_insert_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
        COMMIT;
        EXCEPTION
        WHEN OTHERS THEN

          v_error_cnt:= v_error_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          DBMS_OUTPUT.PUT_LINE('ERROR');
          RAISE;

        END;

      ELSE
        V_last_run_date:=customer_rec.update_date;

        SELECT cust_id INTO v_customer_KEY FROM stg_customers WHERE
CUST_EMAIL=customer_rec.CUST_EMAIL;
```

```
        BEGIN

        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE stg_customers SET CUST_FIRST_NAME=customer_rec.CUST_FIRST_NAME,
                        CUST_LAST_NAME=customer_rec.CUST_LAST_NAME,
                        CUST_GENDER=customer_rec.CUST_GENDER,
                        CUST_YEAR_OF_BIRTH=customer_rec.CUST_YEAR_OF_BIRTH,
                        CUST_MARITAL_STATUS=customer_rec.CUST_MARITAL_STATUS,
                        CUST_STREET_ADDRESS=customer_rec.CUST_STREET_ADDRESS,
                        CUST_POSTAL_CODE=customer_rec.CUST_POSTAL_CODE,
                        CUST_CITY=customer_rec.CUST_CITY,
                        CUST_STATE_PROVINCE=customer_rec.CUST_STATE_PROVINCE,
                        COUNTRY_ID=customer_rec.COUNTRY_ID,
                        CUST_MAIN_PHONE_NUMBER=customer_rec.CUST_MAIN_PHONE_NUMBER,
                        CUST_INCOME_LEVEL=customer_rec.CUST_INCOME_LEVEL,
                        CUST_CREDIT_LIMIT=customer_rec.CUST_CREDIT_LIMIT,
                        CUST_EMAIL=customer_rec.CUST_EMAIL,
                         Update_user=user,
                       update_date=SYSDATE WHERE cust_id=v_customer_key;
                       COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         v_error_cnt:= v_error_cnt+1;


        END;


       END IF;


     END LOOP;

      SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
       FROM Dual;

       UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                      TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                      NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                      NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                      NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
       UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
       Commit;
```

```
 END;

/* PROCEDURE LOAD_REGION */

 Procedure Load_region IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_REGION_KEY NUmber;
 CURSOR region_cur IS SELECT REGION_KEY,CITY,STATE,COUNTRY,update_date
   FROM SALES_OL.REGION WHERE Update_date > v_last_run_date ORDER BY update_date;
 region_rec region_cur%Rowtype;
 BEGIN

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
      FROM Dual;

      SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Region';

      SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

      util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
                'IN_PROCESS',NULL,NULL,NULL,NULL);

      BEGIN

      SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='REGION';
       IF (V_last_run_date IS NULL) THEN

          SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.REGION;
          UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
         COMMIT;
       END IF;
      EXCEPTION
      WHEN NO_DATA_FOUND THEN
      SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.REGION;
      UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
    END;

    FOR region_rec IN region_cur
```

```
LOOP

 SELECT COUNT(*) INTO v_cnt FROM STG_REGION WHERE CITY = region_rec.CITY;

 IF (v_cnt = 0) THEN

   BEGIN
   v_last_run_date:=region_rec.update_date;
    Insert INTO stg_region VALUES(region_KEY_SEQ.nextval,region_rec.city,
                          region_rec.state,region_rec.country,USER,SYSDATE,USER,SYSDATE,
                          null,region_rec.REGION_KEY);
    v_insert_cnt:=v_insert_cnt+1;
    v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
    COMMIT;
    EXCEPTION
    WHEN OTHERS THEN

       v_error_cnt:= v_error_cnt+1;
       v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
       DBMS_OUTPUT.PUT_LINE('ERROR');
       RAISE;

    END;

 ELSE

   v_last_run_date:=region_rec.update_date;

    SELECT REGION_KEY INTO v_REGION_KEY FROM stg_REGION WHERE city=region_rec.city;

    BEGIN

    DBMS_OUTPUT.PUT_LINE('ELSE');

    UPDATE stg_region SET state=region_rec.state,country=region_rec.country,Update_user=user,
                    update_date=SYSDATE WHERE Region_key=v_region_key;
                    COMMIT;
    v_Update_cnt:=v_Update_cnt+1;
    v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

    EXCEPTION
    WHEN OTHERS THEN
     v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
     v_error_cnt:= v_error_cnt+1;


    END;
```

```
      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
       FROM Dual;
       UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                      TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                      NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                      NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                      NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
       UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
       Commit;

  END;

/* PROCEDURE LOAD_PROMOTIONS */

  Procedure Load_promotion IS
  V_last_run_date DATE;
  V_status VARCHAR2(50);
  v_start_date DATE;
  v_End_date DATE;
  v_job_id NUMBER;
  v_insert_cnt NUMBER:=0;
  v_total_number_of_rec_cnt NUMBER:=0;
  v_Update_cnt NUMBER:=0;
  v_run_id NUMBER;
  v_cnt Number :=0;
  v_error_cnt Number:=0;
  v_promotion_KEY NUmber;
  CURSOR promotion_cur IS SELECT
PROMO_ID,PROMO_NAME,PROMO_SUBCATEGORY,PROMO_CATEGORY,
                 PROMO_COST,PROMO_BEGIN_DATE,PROMO_END_DATE,update_date FROM
SALES_OL.promotions
                 WHERE Update_date > v_last_run_date;
  promotion_rec promotion_cur%Rowtype;
  BEGIN

      SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
       FROM Dual;

      SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD promotion';

      SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

      util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
              'IN_PROCESS',NULL,NULL,NULL,NULL);
    BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='PROMOTION';
     IF (V_last_run_date IS NULL) THEN

        SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.PROMOTIONS;
        UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
        COMMIT;
      END IF;
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.PROMOTIONS;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
    END;

    FOR promotion_rec IN promotion_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM stg_promotions WHERE PROMO_NAME =
promotion_rec.PROMO_NAME;

      IF (v_cnt = 0) THEN

        BEGIN

          v_last_run_date:=promotion_rec.update_date;
        Insert INTO stg_promotions VALUES(PROMOTION_KEY_SEQ.nextval,promotion_rec.PROMO_NAME,
                          promotion_rec.PROMO_SUBCATEGORY,promotion_rec.PROMO_CATEGORY,
                          promotion_rec.PROMO_COST,promotion_rec.PROMO_BEGIN_DATE,
                          promotion_rec.PROMO_END_DATE,SYSDATE,USER,
                          USER,SYSDATE,
                          null,promotion_rec.PROMO_ID);
        v_insert_cnt:=v_insert_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
        COMMIT;
        EXCEPTION
        WHEN OTHERS THEN

          v_error_cnt:= v_error_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          DBMS_OUTPUT.PUT_LINE('ERROR');
          RAISE;

        END;

      ELSE

        v_last_run_date:=promotion_rec.update_date;
        SELECT promo_id INTO v_promotion_KEY FROM stg_promotions WHERE
PROMO_NAME=promotion_rec.PROMO_NAME;

        BEGIN

        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE stg_promotions SET PROMO_NAME=promotion_rec.PROMO_NAME,
                       PROMO_SUBCATEGORY=promotion_rec.PROMO_SUBCATEGORY,
                       PROMO_CATEGORY=promotion_rec.PROMO_CATEGORY,
                       PROMO_COST=promotion_rec.PROMO_COST,
                       PROMO_BEGIN_DATE=promotion_rec.PROMO_BEGIN_DATE,
                       PROMO_END_DATE=promotion_rec.PROMO_END_DATE,
                       Update_user=user,
                      update_date=SYSDATE WHERE promo_id=v_promotion_key;
                      COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         v_error_cnt:= v_error_cnt+1;


        END;


      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                    TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                    NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                    NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                    NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;

      UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
      Commit;
  END;

  End;
/
```

LOAD_TRANSACTION_TABLE.pkg

```
CREATE OR REPLACE PACKAGE SALES_STG.LOAD_TRANSACTION_TABLE AS

  PROCEDURE Load_Sales_Transaction;

END;
/

LOAD_TRANSACTION_TABLE.pkb


CREATE OR REPLACE PACKAGE BODY SALES_STG.LOAD_TRANSACTION_TABLE AS

  PROCEDURE Load_Sales_Transaction IS

 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_channel_id NUmber;
 CURSOR C1 IS SELECT
PROD_NAME,CUST_EMAIL,CHANNEL_ID,CITY,QUANTITY_SOLD,AMOUNT_SOLD,CAL_DATE,
        CREATE_USER,CREATE_DATE,UPDATE_USER,UPDATE_DATE FROM
SALES_OL.SALES_TRANSACTION WHERE update_date >
V_last_run_date;
 C1_Rec c1%ROWTYPE;


  Begin


     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
     FROM Dual;

    SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Sales_transaction';

    SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

    util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,NULL,
             'IN_PROCESS',NULL,NULL,NULL,NULL);
   BEGIN

    SELECT LAST_RUN_DATE INTO V_last_run_date FROM STG_INC_CHANGES WHERE
TABLE_NAME='Sales_Transaction';
```

```
      IF (V_last_run_date IS NULL) THEN

        SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.Sales_Transaction;
        UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
        COMMIT;
      END IF;

     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_OL.Sales_Transaction;
     UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
     COMMIT;
    END;


    FOR c1_rec IN c1
    LOOP

    BEGIN

    INSERT INTO
STG_SALES_TRANSACTION(PROD_NAME,CUST_EMAIL,CHANNEL_ID,CITY,QUANTITY_SOLD,AMOUNT_SO
LD,
                          CAL_DATE,CREATE_USER,CREATE_DATE,UPDATE_USER,UPDATE_DATE)
                          VALUES(c1_rec.PROD_NAME,c1_rec.CUST_EMAIL,c1_rec.CHANNEL_ID,
                          c1_rec.CITY,c1_rec.QUANTITY_SOLD,c1_rec.AMOUNT_SOLD,c1_rec.CAL_DATE,
                          USER,SYSDATE,USER,SYSDATE);

        v_insert_cnt:=v_insert_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
        COMMIT;

     EXCEPTION
     WHEN OTHERS THEN

          v_error_cnt:= v_error_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          RAISE;


     END;




    END Loop;
      SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;
      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                        TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                        NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                        NUMBER_OF_ROWS_UPDATED=0,
                        NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
       UPDATE  STG_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
      Commit;


   End;

END;
/
```

UTIL.pkg

```
CREATE OR REPLACE PACKAGE SALES_STG.UTIL  AS

PROCEDURE LOAD_ERROR_TABLE(P_Job_name Varchar2,P_table_name varchar2,P_key_value varchar2,
p_error_code varchar2,p_error_msg varchar2);

Procedure analyze_table;

Procedure Load_Job_Detail(p_run_id Number,p_job_id Number,p_Load_start_date Date,p_load_end_date
varchar2,
p_status Varchar2,p_no_of_rec Number,p_no_of_rec_Rows_inserted Number,p_no_of_records_updated number,
p_no_of_rows_error number);

END;
/
```

UTIL.pkb

```
CREATE OR REPLACE PACKAGE BODY SALES_STG.UTIL  AS

PROCEDURE LOAD_ERROR_TABLE(P_Job_name Varchar2,P_table_name varchar2,P_key_value varchar2,
p_error_code varchar2,p_error_msg varchar2) IS
 PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN

    INSERT INTO ERROR_DETAIL(Job_name,table_name,TABLE_PK_KEY_VALUE,error_code,error_msg)
     Values(P_Job_name,P_table_name,P_key_value,p_error_code,p_error_msg);

     Commit;


END;

Procedure analyze_table IS

 BEGIN

 null;
 END;

Procedure Load_Job_Detail(p_run_id Number,p_job_id Number,p_Load_start_date Date,p_load_end_date
varchar2,
p_status Varchar2,p_no_of_rec Number,p_no_of_rec_Rows_inserted Number,p_no_of_records_updated number,
p_no_of_rows_error number) IS

  BEGIN

  INSERT INTO Job_Detail_run(JOB_RUN_ID,
                        JOB_ID,
                        LOAD_START_DATE,
                LOAD_END_DATE,
                STATUS,
                TOTAL_NO_OF_ROWS,
                NUMBER_OF_ROWS_INSERTED,
                NUMBER_OF_ROWS_UPDATED,
                NUMBER_OF_ROWS_ERROR) VALUES(p_run_id,p_job_id,
                            p_Load_start_date,p_load_end_date,
                            p_status,p_no_of_rec,
                            p_no_of_rec_Rows_inserted,
                            p_no_of_records_updated,
                            p_no_of_records_updated);

                   commit;


 EXCEPTION
 WHEN OTHERS THEN
 Raise;

 END;

END;
/
```

LOOKUP.PKg

```
CREATE OR REPLACE PACKAGE SALES_DW.LOOKUP AS

  PROCEDURE get_channel_key(p_channel_name VARCHAR2,p_channel_key OUT NUMBER);

  PROCEDURE get_Product_key(p_prod_name Varchar2,p_prod_key OUT NUMBER);

  PROCEDURE get_Time_key(p_cal_date Varchar2,p_time_key OUT NUMBER);

  PROCEDURE get_Region_key(p_City Varchar2,p_Region_key OUT NUMBER);

  PROCEDURE get_promotion_key(p_promo_name Varchar2,p_promotion_key OUT NUMBER);

  PROCEDURE get_customer_key(p_cust_email Varchar2,p_customer_key OUT NUMBER);


  END;
/
```

LOOKUP.PKb

```
CREATE OR REPLACE PACKAGE BODY SALES_DW.LOOKUP IS

  PROCEDURE get_channel_key(p_channel_name VARCHAR2,p_channel_key OUT NUMBER) IS
  BEGIN

  SELECT channel_key INTO p_channel_key FROM SALES_DIM_CHANNELS WHERE
CHANNEL_NAME=p_channel_name;
  EXCEPTION
  WHEN OTHERS THEN

    p_channel_key:=-888;


  END;

  PROCEDURE get_Product_key(p_prod_name Varchar2,p_prod_key OUT NUMBER) IS
  BEGIN

     SELECT prod_key INTO p_prod_key FROM SALES_DIM_PRODUCTS  WHERE PROD_NAME =
p_prod_name;
     EXCEPTION
     WHEN OTHERS THEN

        p_prod_key:=-888;

  END;

  PROCEDURE get_Time_key(p_cal_date Varchar2,p_time_key OUT NUMBER) IS
  BEGIN
```

```
      SELECT time_key INTO p_time_key FROM SALES_TIME_DIM  WHERE CALENDAR_DATE = p_cal_date;
      EXCEPTION
      WHEN OTHERS THEN
        p_time_key:=-888;


   END;

   PROCEDURE get_Region_key(p_City Varchar2,p_Region_key OUT NUMBER) IS
   BEGIN

      SELECT Region_key INTO p_Region_key FROM SALES_DIM_REGIONS  WHERE city = p_City;
      EXCEPTION
      WHEN OTHERS THEN
        p_Region_key := -888;


   END;

   PROCEDURE get_promotion_key(p_promo_name Varchar2,p_promotion_key OUT NUMBER) IS
   BEGIN

      SELECT promo_key INTO p_promotion_key FROM SALES_DIM_PROMOTIONS  WHERE PROMO_NAME
 = p_promo_name;
      EXCEPTION
      WHEN OTHERS THEN
        p_promotion_key := -888;

   END;


   PROCEDURE get_customer_key(p_cust_email Varchar2,p_customer_key OUT NUMBER) IS
   BEGIN

      SELECT cust_key INTO p_customer_key FROM SALES_DIM_CUSTOMERS  WHERE cust_email =
 p_cust_email;
      EXCEPTION
      WHEN OTHERS THEN
        p_customer_key := -888;

   END;


   END;
/
```

Load_Dim_table.pkg

```
CREATE OR REPLACE PACKAGE SALES_DW.Load_Dim_table AS

 PROCEDURE LOAD_DIM_CHANNELS;
 PROCEDURE Load_DIM_ProductS;
 Procedure Load_DIM_CustomerS;
 Procedure Load_DIM_regionS;
 Procedure Load_DIM_promotionS;

 End;
/
```

Load_Dim_table.pkb

```
CREATE OR REPLACE PACKAGE BODY SALES_DW.Load_Dim_table IS


 PROCEDURE LOAD_DIM_CHANNELS  IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_channel_id NUmber;
 CURSOR channel_cur IS SELECT CHANNEL_ID,SOURCE_RECORD_ID
channel_name,CHANNEL_DESC,CHANNEL_CLASS,update_date FROM SALES_STG.STG_CHANNELS
WHERE
 Update_date > v_last_run_date order by update_date ;
 channel_rec channel_cur%Rowtype;

 BEGIN

    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
     FROM Dual;

    SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Channel';

    SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

    util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');
   BEGIN
```

```
     SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='CHANNEL';

      IF (V_last_run_date IS NULL) THEN

        SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_stg.stg_channels;
        UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
        COMMIT;
      END IF;

     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.stg_channels;
     UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
     COMMIT;
    END;

    FOR channel_rec IN channel_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM SALES_DIM_CHANNELS  WHERE CHANNEL_NAME =
channel_rec.CHANNEL_NAME;

      IF (v_cnt = 0) THEN

        BEGIN
        V_last_run_date:=channel_rec.update_date;
         Insert INTO SALES_DIM_CHANNELS
VALUES(CHANNEL_KEY_SEQ.nextval,channel_rec.CHANNEL_NAME,
                                channel_rec.CHANNEL_DESC,
                                channel_rec.CHANNEL_CLASS,SYSDATE,USER,USER,SYSDATE,
                             null,channel_rec.channel_id);
         v_insert_cnt:=v_insert_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         COMMIT;
         EXCEPTION
         WHEN OTHERS THEN

           v_error_cnt:= v_error_cnt+1;
           v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
           DBMS_OUTPUT.PUT_LINE('ERROR');
           RAISE;

         END;

      ELSE
        V_last_run_date:=channel_rec.update_date;
         SELECT CHANNEL_KEY INTO v_CHANNEL_ID FROM SALES_DIM_CHANNELS WHERE
CHANNEL_DESC=channel_rec.CHANNEL_DESC;
         BEGIN
        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE SALES_DIM_CHANNELS SET
CHANNEL_CLASS=channel_rec.CHANNEL_CLASS,Update_user=user,
                         update_date=SYSDATE WHERE CHANNEL_KEY=v_CHANNEL_ID;
                         COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         v_error_cnt:= v_error_cnt+1;


        END;


      END IF;




    END LOOP;

       SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                        TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                        NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                        NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                        NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
       UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CHANNEL';
      Commit;
  END;

/* LOAD_DIM_PRODUCT */

 PROCEDURE Load_DIM_ProductS IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_product_KEY NUmber;
 CURSOR product_cur IS SELECT
PROD_ID,PROD_NAME,PROD_DESC,PROD_SUBCATEGORY,PROD_SUBCAT_DESC,
             PROD_CATEGORY,PROD_CAT_DESC,PROD_WEIGHT_CLASS,PROD_UNIT_OF_MEASURE,
             PROD_PACK_SIZE,PROD_STATUS,PROD_LIST_PRICE,PROD_MIN_PRICE,update_date
              FROM SALES_stg.stg_products WHERE  Update_date > v_last_run_date ORDER BY
update_date;
 product_rec product_cur%Rowtype;
 BEGIN

   SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
     FROM Dual;

    SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Product';

    SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

    util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');

   BEGIN

    SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='PRODUCT';

     IF (V_last_run_date IS NULL) THEN

      SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_stg.stg_products;
      UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
      commit;
     END IF;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_PRODUCTS;
    UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
   END;

  FOR product_rec IN product_cur
   LOOP

     SELECT COUNT(*) INTO v_cnt FROM SALES_DIM_products WHERE PROD_NAME =
product_rec.PROD_NAME;

     IF (v_cnt = 0) THEN

       BEGIN

        V_last_run_date :=product_rec.update_date;
        Insert INTO SALES_DIM_products VALUES(PROD_KEY_SEQ.nextval,product_rec.PROD_NAME,
                        product_rec.PROD_DESC,product_rec.PROD_SUBCATEGORY,
                        product_rec.PROD_SUBCAT_DESC,
                        product_rec.PROD_CATEGORY,product_rec.PROD_CAT_DESC,
                        product_rec.PROD_WEIGHT_CLASS,product_rec.PROD_UNIT_OF_MEASURE,
                        product_rec.PROD_PACK_SIZE,product_rec.PROD_STATUS,
                        product_rec.PROD_LIST_PRICE,product_rec.PROD_MIN_PRICE,SYSDATE,USER,
                        USER,SYSDATE,
                        null,product_rec.PROD_ID);
        v_insert_cnt:=v_insert_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
        COMMIT;
        EXCEPTION
        WHEN OTHERS THEN

          v_error_cnt:= v_error_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          DBMS_OUTPUT.PUT_LINE('ERROR');
          RAISE;

        END;

      ELSE

        V_last_run_date :=product_rec.update_date;

        SELECT PROD_KEY INTO v_product_KEY FROM SALES_DIM_products WHERE PROD_NAME =
    product_rec.PROD_NAME;

        BEGIN

        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE SALES_DIM_products SET PROD_NAME=product_rec.PROD_NAME,
                    PROD_DESC=product_rec.PROD_DESC,
                    PROD_SUBCATEGORY=product_rec.PROD_SUBCATEGORY,
                    PROD_SUBCAT_DESC=product_rec.PROD_SUBCAT_DESC,
                    PROD_CATEGORY=product_rec.PROD_CATEGORY,
                    PROD_CAT_DESC=product_rec.PROD_CAT_DESC,
                    PROD_WEIGHT_CLASS=product_rec.PROD_WEIGHT_CLASS,
                    PROD_UNIT_OF_MEASURE=product_rec.PROD_UNIT_OF_MEASURE,
                    PROD_PACK_SIZE=product_rec.PROD_PACK_SIZE,
                    PROD_STATUS=product_rec.PROD_STATUS,
                    PROD_LIST_PRICE=product_rec.PROD_LIST_PRICE,
                    PROD_MIN_PRICE=product_rec.PROD_MIN_PRICE,
                    Update_user=user,update_date=SYSDATE WHERE prod_key=v_product_key;
                    COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
        v_error_cnt:= v_error_cnt+1;


        END;


      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                      TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                      NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                      NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                      NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
      UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PRODUCT';
      Commit;

  END;

/* PROCEDURE LOAD_DIM_CUSTOMERS */


 Procedure Load_DIM_CustomerS IS
 V_last_run_date DATE;
 V_status VARCHAR2(50);
 v_start_date DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_Update_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_customer_KEY NUmber;
 CURSOR customer_cur IS SELECT CUST_ID,CUST_FIRST_NAME,CUST_LAST_NAME,CUST_GENDER,
                    CUST_YEAR_OF_BIRTH,CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,
                    CUST_POSTAL_CODE,CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID,
                    CUST_MAIN_PHONE_NUMBER,CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,
                    CUST_EMAIL,update_date FROM SALES_STG.stg_customers
                     WHERE   Update_date > v_last_run_date order by update_date;
 customer_rec customer_cur%Rowtype;
 BEGIN
```

```
    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
      FROM Dual;

     SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Customer';

     SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

     util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');

    BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='CUSTOMER';

      IF (V_last_run_date IS NULL) THEN

       SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_CUSTOMERS;
       UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
      END IF;
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_CUSTOMERS;
     UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
    END;

   FOR customer_rec IN customer_cur
    LOOP

      SELECT COUNT(*) INTO v_cnt FROM SALES_DIM_customers WHERE CUST_EMAIL =
customer_rec.CUST_EMAIL;

      IF (v_cnt = 0) THEN

        BEGIN
          V_last_run_date:=customer_rec.update_date;
         Insert INTO SALES_DIM_customers
VALUES(customer_KEY_SEQ.nextval,customer_rec.CUST_FIRST_NAME,
                              customer_rec.CUST_LAST_NAME,
                              customer_rec.CUST_GENDER,
                              customer_rec.CUST_YEAR_OF_BIRTH,
                              customer_rec.CUST_MARITAL_STATUS,
                              customer_rec.CUST_STREET_ADDRESS,
                              customer_rec.CUST_POSTAL_CODE,
                              customer_rec.CUST_CITY,
                              customer_rec.CUST_STATE_PROVINCE,
                              customer_rec.COUNTRY_ID,
                              customer_rec.CUST_MAIN_PHONE_NUMBER,
                              customer_rec.CUST_INCOME_LEVEL,
                              customer_rec.CUST_CREDIT_LIMIT,
                        customer_rec.CUST_EMAIL,SYSDATE,USER,
                        USER,SYSDATE,
                        null,customer_rec.cust_ID);
    v_insert_cnt:=v_insert_cnt+1;
    v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
    COMMIT;
    EXCEPTION
    WHEN OTHERS THEN

      v_error_cnt:= v_error_cnt+1;
      v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
      DBMS_OUTPUT.PUT_LINE('ERROR');
      RAISE;

    END;

  ELSE
    V_last_run_date:=customer_rec.update_date;

    SELECT cust_key INTO v_customer_KEY FROM SALES_DIM_customers WHERE
CUST_EMAIL=customer_rec.CUST_EMAIL;

    BEGIN

    DBMS_OUTPUT.PUT_LINE('ELSE');

    UPDATE SALES_DIM_customers SET CUST_FIRST_NAME=customer_rec.CUST_FIRST_NAME,
                CUST_LAST_NAME=customer_rec.CUST_LAST_NAME,
                CUST_GENDER=customer_rec.CUST_GENDER,
                CUST_YEAR_OF_BIRTH=customer_rec.CUST_YEAR_OF_BIRTH,
                CUST_MARITAL_STATUS=customer_rec.CUST_MARITAL_STATUS,
                CUST_STREET_ADDRESS=customer_rec.CUST_STREET_ADDRESS,
                CUST_POSTAL_CODE=customer_rec.CUST_POSTAL_CODE,
                CUST_CITY=customer_rec.CUST_CITY,
                CUST_STATE_PROVINCE=customer_rec.CUST_STATE_PROVINCE,
                COUNTRY_ID=customer_rec.COUNTRY_ID,
                CUST_MAIN_PHONE_NUMBER=customer_rec.CUST_MAIN_PHONE_NUMBER,
                CUST_INCOME_LEVEL=customer_rec.CUST_INCOME_LEVEL,
                CUST_CREDIT_LIMIT=customer_rec.CUST_CREDIT_LIMIT,
                CUST_EMAIL=customer_rec.CUST_EMAIL,
                 Update_user=user,
               update_date=SYSDATE WHERE cust_key=v_customer_key;
               COMMIT;
    v_Update_cnt:=v_Update_cnt+1;
    v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

    EXCEPTION
    WHEN OTHERS THEN
     v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
     v_error_cnt:= v_error_cnt+1;
```

```
        END;


      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                   TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                   NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                   NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                   NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
       UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='CUSTOMER';
      Commit;


  END;

/* PROCEDURE LOAD_DIM_REGIONS  */

  Procedure Load_DIM_regionS IS
  V_last_run_date DATE;
  V_status VARCHAR2(50);
  v_start_date DATE;
  v_End_date DATE;
  v_job_id NUMBER;
  v_insert_cnt NUMBER:=0;
  v_total_number_of_rec_cnt NUMBER:=0;
  v_Update_cnt NUMBER:=0;
  v_run_id NUMBER;
  v_cnt Number :=0;
  v_error_cnt Number:=0;
  v_REGION_KEY NUmber;
  CURSOR region_cur IS SELECT REGION_KEY,CITY,STATE,COUNTRY,update_date
   FROM SALES_STG.STG_REGION WHERE Update_date > v_last_run_date ORDER BY update_date;
  region_rec region_cur%Rowtype;
  BEGIN

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
      FROM Dual;

     SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Region';

     SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;
     util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');

     BEGIN

     SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='REGION';
      IF (V_last_run_date IS NULL) THEN

          SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_REGION;
          UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
         COMMIT;
      END IF;
     EXCEPTION
     WHEN NO_DATA_FOUND THEN
     SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_REGION;
     UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
    END;

    FOR region_rec IN region_cur
    LOOP

       SELECT COUNT(*) INTO v_cnt FROM SALES_DIM_REGIONS WHERE CITY = region_rec.CITY;

       IF (v_cnt = 0) THEN

         BEGIN
         v_last_run_date:=region_rec.update_date;
          Insert INTO SALES_DIM_REGIONS VALUES(region_KEY_SEQ.nextval,region_rec.city,
                               region_rec.state,region_rec.country,USER,SYSDATE,USER,SYSDATE,
                               null,region_rec.REGION_KEY);
          v_insert_cnt:=v_insert_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          COMMIT;
          EXCEPTION
          WHEN OTHERS THEN

             v_error_cnt:= v_error_cnt+1;
             v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
             DBMS_OUTPUT.PUT_LINE('ERROR');
             RAISE;

          END;

       ELSE

         v_last_run_date:=region_rec.update_date;

          SELECT REGION_KEY INTO v_REGION_KEY FROM SALES_DIM_REGIONS WHERE
city=region_rec.city;
```

```
        BEGIN

        DBMS_OUTPUT.PUT_LINE('ELSE');

        UPDATE SALES_DIM_REGIONS SET
state=region_rec.state,country=region_rec.country,Update_user=user,
                    update_date=SYSDATE WHERE Region_key=v_region_key;
                    COMMIT;
        v_Update_cnt:=v_Update_cnt+1;
        v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

        EXCEPTION
        WHEN OTHERS THEN
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         v_error_cnt:= v_error_cnt+1;


        END;


      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                  TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                  NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                  NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                  NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;

       UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='REGION';
      Commit;

  END;

/* LOAD_DIM_PROMOTIONS */

  Procedure Load_DIM_promotionS IS
  V_last_run_date DATE;
  V_status VARCHAR2(50);
  v_start_date DATE;
  v_End_date DATE;
  v_job_id NUMBER;
  v_insert_cnt NUMBER:=0;
  v_total_number_of_rec_cnt NUMBER:=0;
  v_Update_cnt NUMBER:=0;
```

```
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_promotion_KEY NUmber;
 CURSOR promotion_cur IS SELECT
PROMO_ID,PROMO_NAME,PROMO_SUBCATEGORY,PROMO_CATEGORY,
                PROMO_COST,PROMO_BEGIN_DATE,PROMO_END_DATE,update_date FROM
SALES_STG.STG_promotions
                WHERE Update_date > v_last_run_date;
 promotion_rec promotion_cur%Rowtype;
 BEGIN

    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
     FROM Dual;

    SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD promotion';

    SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

    util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');


   BEGIN

    SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='PROMOTION';
    IF (V_last_run_date IS NULL) THEN

       SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_PROMOTIONS;
       UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
      COMMIT;
     END IF;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_STG.STG_PROMOTIONS;
    UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
   END;

   FOR promotion_rec IN promotion_cur
   LOOP

     SELECT COUNT(*) INTO v_cnt FROM SALES_DIM_promotions WHERE PROMO_NAME =
promotion_rec.PROMO_NAME;

     IF (v_cnt = 0) THEN

       BEGIN

         v_last_run_date:=promotion_rec.update_date;
         Insert INTO SALES_DIM_promotions
VALUES(PROMOTION_KEY_SEQ.nextval,promotion_rec.PROMO_NAME,
                              promotion_rec.PROMO_SUBCATEGORY,promotion_rec.PROMO_CATEGORY,
                              promotion_rec.PROMO_COST,promotion_rec.PROMO_BEGIN_DATE,
                              promotion_rec.PROMO_END_DATE,SYSDATE,USER,
                              USER,SYSDATE,
                              null,promotion_rec.PROMO_ID);
         v_insert_cnt:=v_insert_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
         COMMIT;
         EXCEPTION
         WHEN OTHERS THEN

            v_error_cnt:= v_error_cnt+1;
            v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
            DBMS_OUTPUT.PUT_LINE('ERROR');
            RAISE;

         END;

      ELSE

         v_last_run_date:=promotion_rec.update_date;

         SELECT promo_key INTO v_promotion_KEY FROM SALES_DIM_promotions WHERE
PROMO_NAME=promotion_rec.PROMO_NAME;

         BEGIN

         DBMS_OUTPUT.PUT_LINE('ELSE');

         UPDATE SALES_DIM_promotions SET PROMO_NAME=promotion_rec.PROMO_NAME,
                         PROMO_SUBCATEGORY=promotion_rec.PROMO_SUBCATEGORY,
                         PROMO_CATEGORY=promotion_rec.PROMO_CATEGORY,
                         PROMO_COST=promotion_rec.PROMO_COST,
                         PROMO_BEGIN_DATE=promotion_rec.PROMO_BEGIN_DATE,
                         PROMO_END_DATE=promotion_rec.PROMO_END_DATE,
                         Update_user=user,
                        update_date=SYSDATE WHERE promo_key=v_promotion_key;
                        COMMIT;
         v_Update_cnt:=v_Update_cnt+1;
         v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;

         EXCEPTION
         WHEN OTHERS THEN
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          v_error_cnt:= v_error_cnt+1;


         END;
      END IF;


    END LOOP;

     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
      FROM Dual;

      UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                     TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                     NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                     NUMBER_OF_ROWS_UPDATED=v_Update_cnt,
                     NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;

      UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='PROMOTION';
      Commit;
 END;

 End;
/
```

LOAD_FACT.pkg

```
CREATE OR REPLACE PACKAGE SALES_DW.LOAD_FACT AS

  PROCEDURE LOAD_SALES_FACT;


  END;
/
```

LOAD_FACT.pkb

```
CREATE OR REPLACE PACKAGE BODY SALES_DW.LOAD_FACT AS

  PROCEDURE LOAD_SALES_FACT IS
 V_last_run_date DATE;
 V_status      VARCHAR2(50);
 v_start_date   DATE;
 v_End_date DATE;
 v_job_id NUMBER;
 v_insert_cnt NUMBER:=0;
 v_total_number_of_rec_cnt NUMBER:=0;
 v_run_id NUMBER;
 v_cnt Number :=0;
 v_error_cnt Number:=0;
 v_channel_KEY NUmber;
 v_product_key NUMBER;
 v_customer_KEY NUmber;
 v_time_KEY NUmber;
 v_region_KEY NUmber;
 CURSOR C1 IS SELECT
PROD_NAME,CUST_EMAIL,CHANNEL_ID,CITY,QUANTITY_SOLD,AMOUNT_SOLD,CAL_DATE
 FROM SALES_STG.STG_SALES_TRANSACTION WHERE update_date > V_last_run_date;
 C1_Rec c1%ROWTYPE;
 Begin


     SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_start_date
     FROM Dual;

    SELECT JOB_ID INTO v_job_id FROM JOB_CONTROL WHERE JOB_NAME='LOAD Sales_transaction';

    SELECT JOB_RUN_ID_SEQ.NEXTVAL INTO v_run_id FROM DUAL;

    util.Load_Job_Detail(v_run_id,v_job_id,v_start_date,'IN_PROCESS');
    BEGIN
```

```
      SELECT LAST_RUN_DATE INTO V_last_run_date FROM SALES_INC_CHANGES WHERE
TABLE_NAME='Sales_Transaction';

       IF (V_last_run_date IS NULL) THEN

          SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_stg.stg_Sales_Transaction;
          UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
         COMMIT;
       END IF;

      EXCEPTION
      WHEN NO_DATA_FOUND THEN
      SELECT min(UPDATE_DATE)-1 INTO V_last_run_date FROM SALES_stg.Stg_Sales_Transaction;
      UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
      COMMIT;
    'END;

     FOR c1_REC IN C1
     LOOP

     LOOKUP.get_channel_key(c1_REC.channel_id,v_channel_key);
     LOOKUP.get_Product_key(c1_REC.prod_name,v_product_key);
     LOOKUP.get_Time_key(c1_REC.cal_date,v_time_KEY);
     LOOKUP.get_customer_key(c1_rec.CUST_EMAIL,v_customer_KEY);
     LOOKUP.get_Region_key(c1_rec.city,v_region_key);

     BEGIN

     INSERT INTO SALES_FACT VALUES(v_product_key,v_customer_KEY,v_channel_key,v_region_key,
                       c1_rec.QUANTITY_SOLD,c1_rec.AMOUNT_SOLD,v_time_KEY,USER,SYSDATE,
                       USER,SYSDATE);


     v_insert_cnt:=v_insert_cnt+1;
          v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
          COMMIT;

      EXCEPTION
      WHEN OTHERS THEN

            v_error_cnt:= v_error_cnt+1;
            v_total_number_of_rec_cnt:=v_total_number_of_rec_cnt+1;
            RAISE;


      END;

     END LOOP;
    SELECT to_date(TO_CHAR(SYSDATE,'mm/dd/yyyy HH24:MI:SS'),'mm/dd/yyyy HH24:MI:SS') INTO
v_end_date
     FROM Dual;

     UPDATE JOB_DETAIL_RUN SET LOAD_END_DATE=v_end_date,STATUS='COMPLETED',
                    TOTAL_NO_OF_ROWS=v_total_number_of_rec_cnt,
                    NUMBER_OF_ROWS_INSERTED=v_insert_cnt,
                    NUMBER_OF_ROWS_UPDATED=0,
                    NUMBER_OF_ROWS_ERROR=v_error_cnt WHERE JOB_RUN_ID=v_run_id;
      UPDATE  SALES_INC_CHANGES SET LAST_RUN_DATE=V_last_run_date WHERE
TABLE_NAME='Sales_Transaction';
     Commit;


  END;


  END;
/
```

APPENDIX F

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Definitions, Acronyms and Abbreviations

Cache
A method for improving system performance by creating a secondary memory area with access speeds closer to the processor speed. A memory cache is a section of high speed memory to cache data from main memory

Composite Key
A key for the database table made up of more than one attribute or field.

Confirmed Dimensions
Two sets of business dimensions represented in dimensional table are said to be conformed if both sets are identical in their attributes of if one set is an exact subset of the other.

Database
A repository where an ordered and integrated collection of the enterprise data is stored for computer processing and information sharing

Data Mart
A collection of related data from internal and external sources, transformed, integrated and stored for the purpose of providing strategic information for a specific set of users in an enterprise.

Data Warehouse
A collection of transformed and integrated data, stored for the purpose of providing strategic information to the entire enterprise.

DBMS
Database Management system. It is a software system to store, access, maintain, manage and safeguard data in the databases.

DD
Data Dictionary, a catalog or directory in database management systems used for defining data structures and relationships.

DDL
Data Definition Language.

Dimension tables
In the dimension data model, each dimension table contains the attributes of a single business dimension.

E-R Data Modeling
It is a popular data modeling technique used for representing business entities and their relationships.

Fact Table
In dimensional data model, the center table which contains the facts or metrics of the business as attributes in the table.

Granularity
Indicates the level or grain of data.

Indexing
The method for speeding up database access by creating index files that point to data files.

OLTP
Online Transaction processing is an application that collects data online during the execution of business transactions.

Operational systems
An application that supports the day–to–day operations of a business

Partitioning
It is a method for dividing a database into manageable parts for the purpose of easier management and better performance.

Primary Key
One or more fields or attributes that uniquely identify each record in a database table

RDBMS
It is abbreviated as Relational database management system.

Schema
A collection of tables that forms a database

Snowflake schema
It is a normalized version of star schema in which dimension tables are partially or fully normalized.

Star Schema
It is an arrangement of the collection of fact and dimensional tables in the dimensional data model, resembling a star formation, with the fact table placed at the centre surrounded by dimensional tables. Each dimension table is in a one-to-many relationship with the fact table.

Surrogate Key
It is an artificial key field, usually system-assigned sequential numbers, used in the dimensional model to link a dimensional table to the fact tables.

Table Space
It refers to an area on a physical medium where one or more relational database tables can exist.

REFERENCES

Adamson, C. *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance*, Indianapolis, IN: Wiley, 2006.

Ponniah, P. *Data Warehousing Fundamentals: a Comprehensive Guide for IT Professionals.* New York: Wiley, 2004.

Sperley, E. Enterprise *Data Warehouse: Planning, Building, and Implementation*, vol.1. Upper Saddle River, NJ: Prentice Hall, 1999.