

Using Relational Database Software To Prepare An Archival Dataset For Importation Into A Statistical Analysis Package

Paul J. Lazarony, (Email: paul.lazarony@csun.edu), California State University, Northridge
Donna A. Driscoll, (Email: donna.driscoll@csun.edu), California State University, Northridge

Abstract

*Researchers frequently find that archival data that they need to test their theoretical models must undergo a variety of preparatory steps before the data are ready to be loaded into their favorite statistical analysis package. Although some powerful software tools and techniques to handle such data preparation do exist and are relatively easy to learn, many researchers remain unaware of them. The purpose of this paper is to introduce some of these data preparation tools and techniques by use of a hands-on demonstration. The demonstration begins with data embedded in a Web page and shows how a spreadsheet, a word processor, and the SQL language from a relational database can be used to make the extracted “ready for prime-time.” The original Web page data for the demonstration can be downloaded from the authors’ web page; the software packages used to prepare the data are **MS Excel**, **MS Word**, and **MS Access**. Although **SPSS** is used for the statistical analysis described in the final phase of the process, readers do not have to have this software to benefit from the demonstration. Any statistical analysis package could be used.*

Introduction

*R*esearch data from archival sources are often not directly usable by statistical analysis packages (e.g. **SPSS**, **SAS**, **MINITAB**). Common causes of this problem include:

1. *The presence of invisible formatting characters in the archive.* For example, the seemingly innocuous formatting characters that are introduced into archive files by the software package that created them can make it physically impossible to (a) copy and paste directly from the archive to the statistical package or (b) import the archive directly into the statistical package (e.g., the inconsistent number of “*mso-tab-count*” tab characters inserted between columns of a Web page by **MS FrontPage**).
2. *The frequent need of investigators to combine data from multiple sources.* For example, an investigator might need to use two types of student data: (a) 10,000 demographic data records (one for each student) and (b) 300,000 course grade records (an average of 30 per student). Before this data can be imported into a statistical analysis package, it must be reorganized into a 300,000 row matrix in which each row contains both information about the course and the grade earned as well as about the particular student who completed it. This reorganization requires that each of the 300,000 grade records be matched up with one of the 10,000 student demographic records. This matching would be done based on a field that is present in both of the datasets, most likely a student identification number.

3. *The fact that investigators' data sources are often created by someone other than the investigator with some non-research purpose in mind and thus are "messy" from a statistical analysis standpoint. For example, some common data "clean up" tasks are:*
- a. changing data types from text to numeric,
 - b. changing coding schemes (e.g., 2004 vs. 04 for a year field),
 - c. fixing inconsistencies in the treatment of special characters (e.g., 555-1212 vs. 5551212 in a phone number field),
 - d. creating new fields (e.g., recoding letter grades onto a 4-point scale), or
 - e. separating inappropriately merged fields (e.g., to separate an area code from a telephone number).

Although most active researchers seem to feel fairly confident about their ability to analyze data once it has been safely placed into their favorite statistical analysis package; many feel far less confident about their ability to get it there in the first place. Fortunately, there are some easy software tools and techniques that can be used to overcome all three of the archival dataset problems listed above. Unfortunately, many researchers are not aware of these tools and techniques and so they might end up: (a) shying away from a particular project because they are unsure about how to deal with the ill-behaved data, (b) paying someone else to deal with data problems that they could easily have solved themselves, or (c) typing in the dataset by hand when it's already available in electronic form. Clearly, none of these results is desirable, and that is why we have written this paper.

More specifically, our purpose is to introduce researchers to the process of taming unruly data by use of readily available software, specifically a spreadsheet, a word processor, and a relational database (all of which are available in the **MS Office** suite). This introduction takes the form of a hands-on demonstration of the steps involved in extracting data from a Web page, massaging it a bit, importing it into a statistical analysis package, and then running a multiple linear regression. The initial Web page used in the demonstration is available for download; the reader is invited to follow along.

Demonstration

In our example, the initial Web page data is a presentation schedule for an academic conference. We will use this data to test a rather odd hypothetical "early-bird" theory that can be described as follows. Suppose we believe that a person who chooses to give a conference presentation both (a) early in the day and (b) near the beginning of a conference would also send his/her paper to the conference chair soon after the opening of the submission period. (As far as we know there is no actual theory to support our belief. We are just pretending that there is for purposes of this demonstration.) Based on our "theory," we have developed the following model:

$$\mathbf{PaperNum} = \alpha + \beta_1 \mathbf{DayNum} + \beta_2 \mathbf{Hour}$$

Where:

PaperNum = A sequential integer assigned to a submitted paper when it is accepted for presentation at the conference.

DayNum = A sequential integer representing the day of the conference that the submitter chooses to present his paper (e.g., Thursday =1, Friday=2).

Hour = An integer representing the hour of the day that the submitter chooses to present his/her paper (e.g., 8:00-8:59=8, 9:00-9:59=9).

Our goal is to test this model using multiple linear regression on the schedule data that is embedded in the conference Web page (see Figure 1).

The data that we need for our regression are in the **Day**, **Time**, and **Paper #** columns of the Web page. Unfortunately, since all three of these columns are text and the three variables in our model (**Daynum**, **Hour**, and **PaperNum**) must be integers, each of the Web page columns needs a little work, specifically:

- The data in the **Day** column combines the number of the day that we need for our model with the name of the day. We need to split **Day** into two separate columns, **DayNum** and **DayName**.
- The data in the **Time** column contains both the beginning and the ending times of each session, as well as a hyphen character (-) that separates them. For purposes of our example, let's assume that we are interested only in when the presentation starts, measured in whole hours. To get this data, we need to truncate the **Time** column to include just the first two digits. We will call this new variable by the name, **Hour**.
- The data in the **Paper #** column contains both the number of the paper and an occasional suffix (-D) which indicates that the particular paper is based upon dissertation research. Since our model does not make a distinction between dissertation and non-dissertation research, we need to extract the first three digits of **Paper #** as our **PaperNum** variable.

The remainder of our demonstration consists of four *Phases* (numbered **I**, **II**, **III**, and **IV**) that describe the process of preparing and using the data in Figure 1 to test our model. The discussion will proceed as follows:

Phase I: Using a spreadsheet and a word processor, locate and remove the hidden formatting characters that have been inserted into our file by the Web page editor.

Phase II: Import the streamlined data into a relational database.

Phase III: Using the SQL (Structured Query Language) facility of the database, efficiently correct, recode, and reorganize the data.

Phase IV: Import the improved data from the database into a statistical analysis package and run the regression.

As previously mentioned, this example is designed so that the reader can follow along step-by-step. The Web page containing the data can be found at www.csun.edu/~pjl26399/iberj. The following software will also be required to complete the example: **MS Excel**, **MS Word**, **MS Access**, and **SPSS** (Statistical Package for the Social Sciences). If **SPSS** is not available, phase **IV** could also be accomplished using another statistical package such as **SAS** or **MINITAB**.

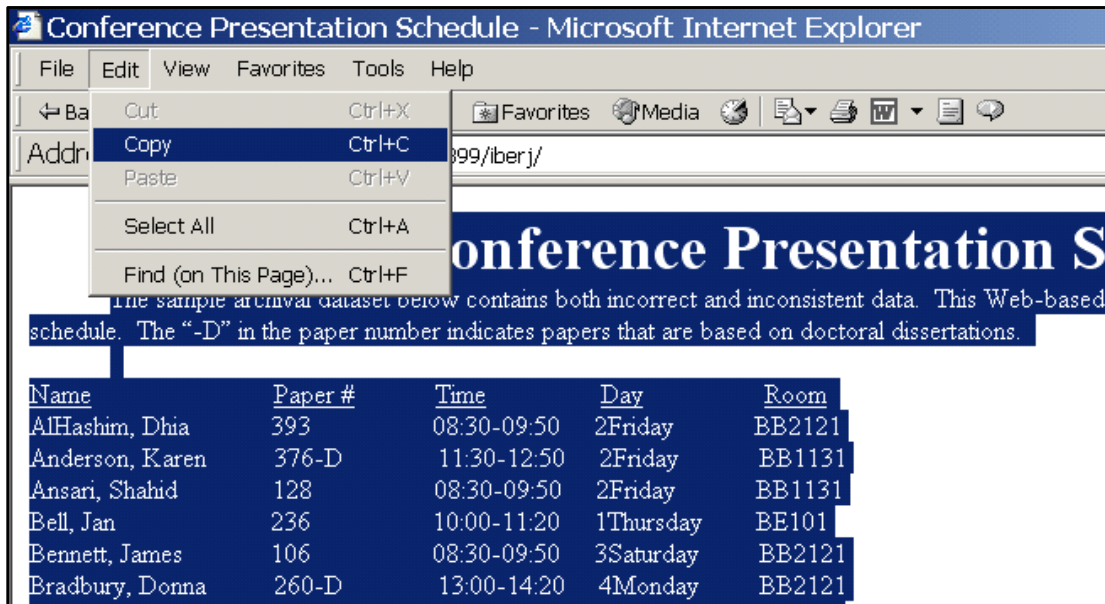
Phase I. Reformat Web-based Archival Data for Importation into a Database

The increasing use of Web page editors (e.g. **MS FrontPage**) often poses problems for investigators who need to extract embedded data for statistical analyses. Although such editors are very useful for creating Web pages that are pleasing to the eye, they also insert invisible formatting characters between columns of data that are not interpretable by some types of software (i.e., databases, statistical packages). We can, however, use the features of two other types of software (a spreadsheet and a word processor) to locate and remove the extra characters, thus making our file usable by our two target programs.

In this first phase of our demonstration, a spreadsheet package (**MS Excel**) will be used to locate the extra characters embedded in the Web page. A word processing package (**MS Word**) will then be used to remove them. The steps are as follows:

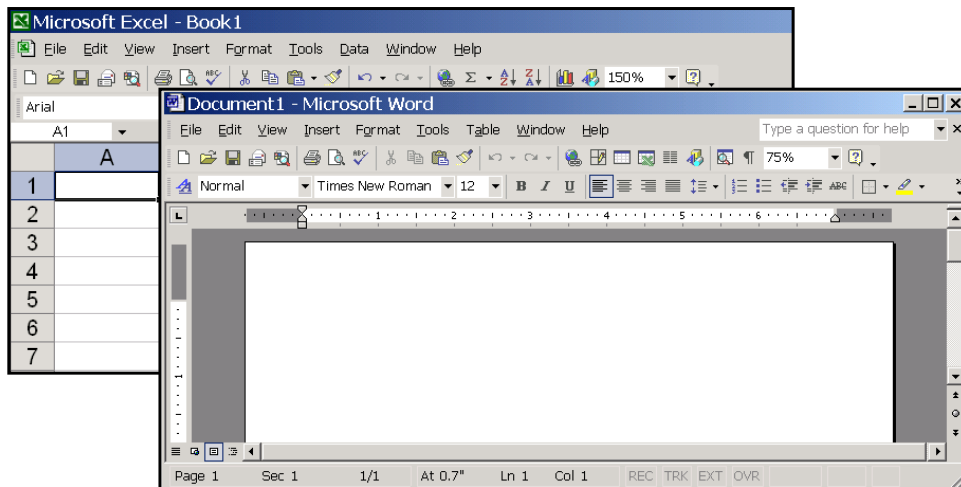
- A. Using an Internet browser (e.g., **Internet Explorer**, **Netscape Navigator**), display the Web page containing the data we'll be using in our example (see Web address above). Next, use either control keys (**CTRL+A**) or the menu (**EDIT/SELECT ALL**) to select the entire Web page and then copy it (**CTRL+C** or **EDIT/COPY**) to the clipboard (see Figure 1).

Figure 1: Web Page containing Schedule Data



- B. Open both **MS Excel** and **MS Word** (see Figure 2).

Figure 2: New MS Excel Workbook and New MS Word Document



- C. Paste (**CTRL+V** or **EDIT/PASTE**) the copy of the Web page that we stored in the clipboard into both **MS Excel** (see Figure 3) and **MS Word** (see Figure 4).

Figure 3: Web Page pasted into MS Excel

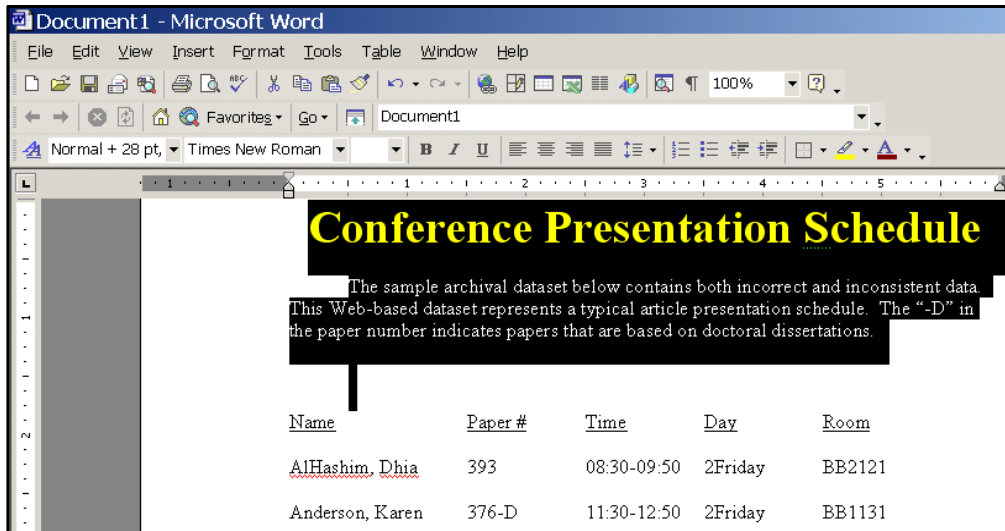
	A	B	C	D	E	F	G	H	I
1	Presentation Schedule								
2	The sample archival dataset below contains both incorrect and inconsistent data. This Web-based dataset r								
3									
4	<u>Name</u>			<u>Paper #</u>	<u>Time</u>		<u>Day</u>		<u>Room</u>
5	AlHashim, Dhia	393		08:30-09:2	Friday	BB2121			
6	Anderson, Karen	376-D		11:30-12:2	Friday	BB1131			
7	Ansari, Shahid		128	08:30-09:2	Friday	BB1131			
8	Bell, Jan		236	10:00-11:1	Thursday	BE101			
9	Bennett, James		106	08:30-09:3	Saturday	BB2121			
10	Bradbury, Donna	260-D		13:00-14:4	Monday	BB2121			
11	Call, Dwight		223	10:00-11:2	Friday	EE3131			

Figure 4: Web Page pasted into MS Word

<u>Name</u>	<u>Paper #</u>	<u>Time</u>	<u>Day</u>	<u>Room</u>
AlHashim, Dhia	393	08:30-09:50	2Friday	BB2121
Anderson, Karen	376-D	11:30-12:50	2Friday	BB1131
Ansari, Shahid	128	08:30-09:50	2Friday	BB1131

D. Since our final dataset only requires the column headings and the data beneath them, our next step is to remove all superfluous text above the column headings, as follows: In **MS Word**, highlight the text to be removed (see Figure 5) and press the **DELETE** key. To remove the corresponding text in **MS Excel**, select rows one and two and press **DELETE**.

Figure 5: Text Selected for Removal in MS Word



- E. When we copied the data from the clipboard into **MS Excel** and **MS Word** (Figures 3 and 4), you may have noticed that the columns appeared to be in complete disarray in **MS Excel**, yet perfectly aligned in **MS Word**. As a specific example of this discrepancy, compare the circled areas of Figures 6 and 7. Notice that in **MS Excel** there is extra space between the **Name** and **Paper #** headings, but there appears to be no such extra space

Figure 6: Extra Space Between Column Headings in MS Excel

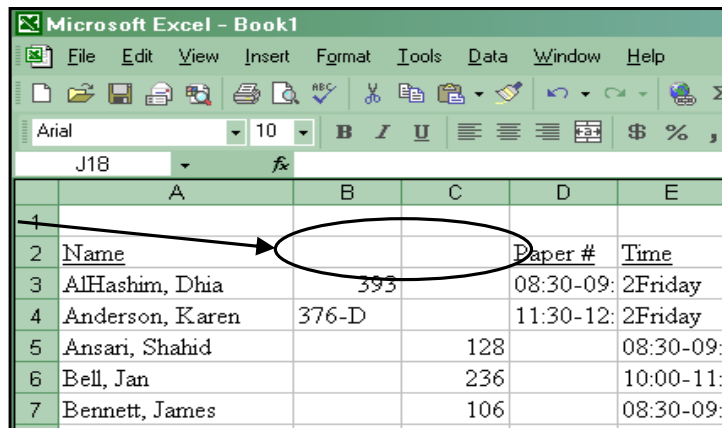
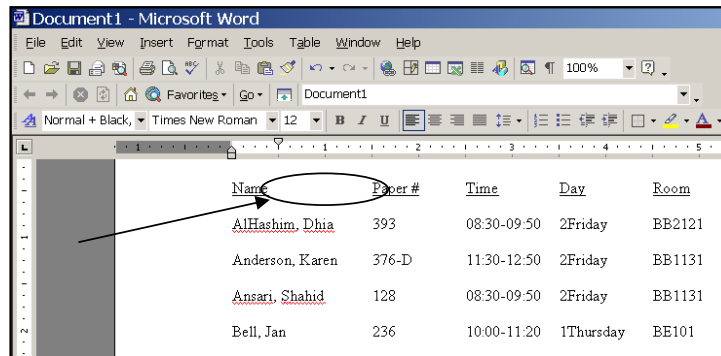


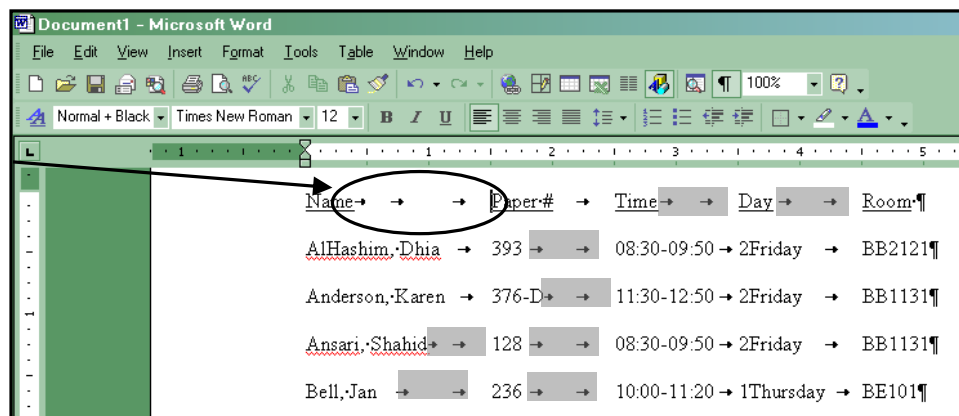
Figure 7: No Extra Space Between Column Headings in MS Word



Name	Paper #	Time	Day	Room
AlHashim, Dhia	393	08:30-09:50	2Friday	BB2121
Anderson, Karen	376-D	11:30-12:50	2Friday	BB1131
Ansari, Shahid	128	08:30-09:50	2Friday	BB1131
Bell, Jan	236	10:00-11:20	1Thursday	BE101

- F. How can this be? What is the cause of this drastic difference? The cause becomes clear when, in **MS Word**, we press the **Show/Hide** button (¶) to display the hidden formatting marks (see Figure 8). For any dataset to be successfully imported into an **MS Access** table (and then subsequently transferred to a statistical analysis package), all adjacent fields must be separated by one, and only one, tab. Figure 8 reveals that there are actually three tab markers (→) between the **Name** and **Paper #** column headings. A cursory examination of Figure 8 reveals eight other instances of multiple tab markers between fields (see eight shaded areas in Figure 8).

Figure 8: Display of Hidden Formatting Marks in MS Word

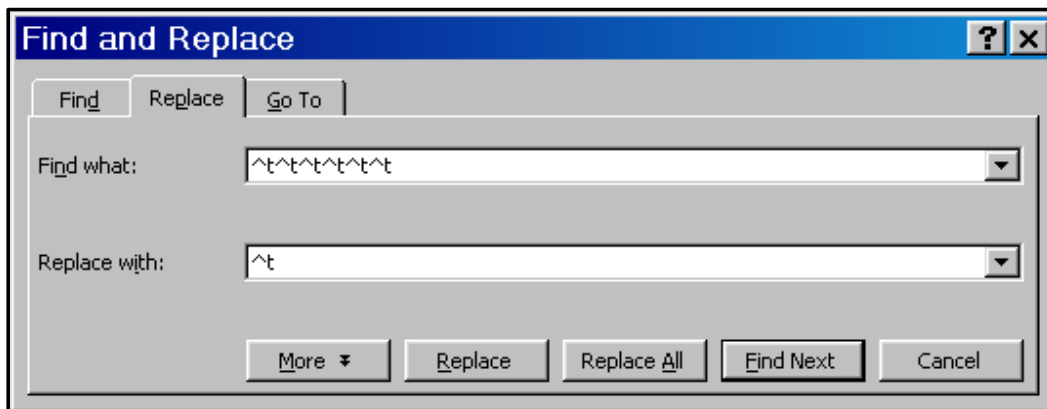


Name → → → Paper # → Time → Day → Room ¶
AlHashim, Dhia → → → 393 → → → 08:30-09:50 → 2Friday → BB2121 ¶
Anderson, Karen → → → 376-D → → → 11:30-12:50 → 2Friday → BB1131 ¶
Ansari, Shahid → → → 128 → → → 08:30-09:50 → 2Friday → BB1131 ¶
Bell, Jan → → → 236 → → → 10:00-11:20 → 1Thursday → BE101 ¶

- G. In **MS Word**, we use **FIND AND REPLACE** to convert each instance of multiple tabs into a single tab. The process of conversion is an iterative one. In the first iteration, we search for the highest number of consecutive tabs (say n) that is present in the document and replace that set of n tabs with a single tab. For each subsequent iteration, we search for one less tab than before until we are searching for just two tabs. The specific steps are as follows:
1. Position the cursor at the top of the Web page file.
 2. Select **EDIT/REPLACE** from the menu; the **FIND AND REPLACE** dialog box appears (see Figure 9).
 3. Next to **Replace with**, enter a single **^t**. (Note: “^t” is the code for a single tab marker.)

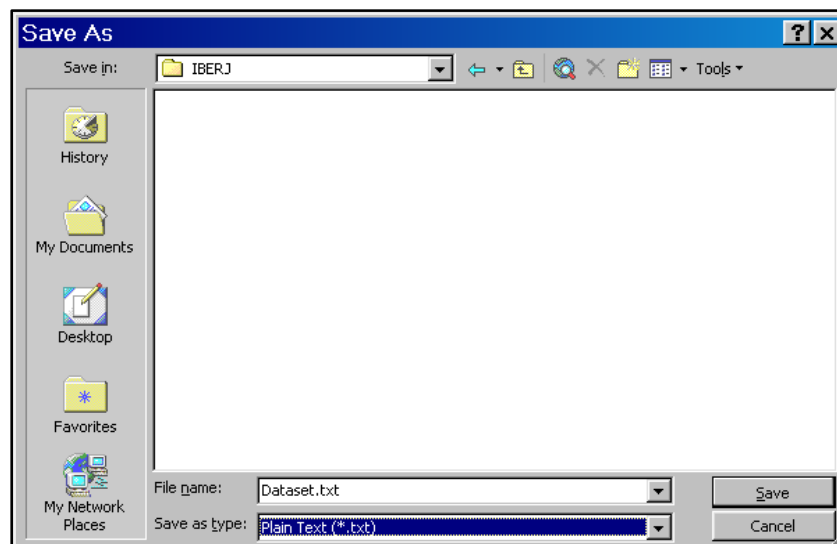
4. Next to **Find what**, enter **^t** for each tab in the search. (Figure 9 shows the dialog box after the completion of this step for six tabs.)
5. Click **Replace All**.
6. Repeat steps 4 and 5, searching for one less tab (**^t**) each time, until all instances of multiple tabs have been replaced by a single tab.

Figure 9: Find and Replace Dialog Box



- H. The dataset is now ready to be saved. Select **FILE/SAVE AS**. Click the down arrow next to **Save as type** to choose the **Plain Text** option (see Figure 10). Name the file **Dataset.txt**. Close both **MS Word** and **MS Excel**.

Figure 10: MS Word Save As Dialog Box

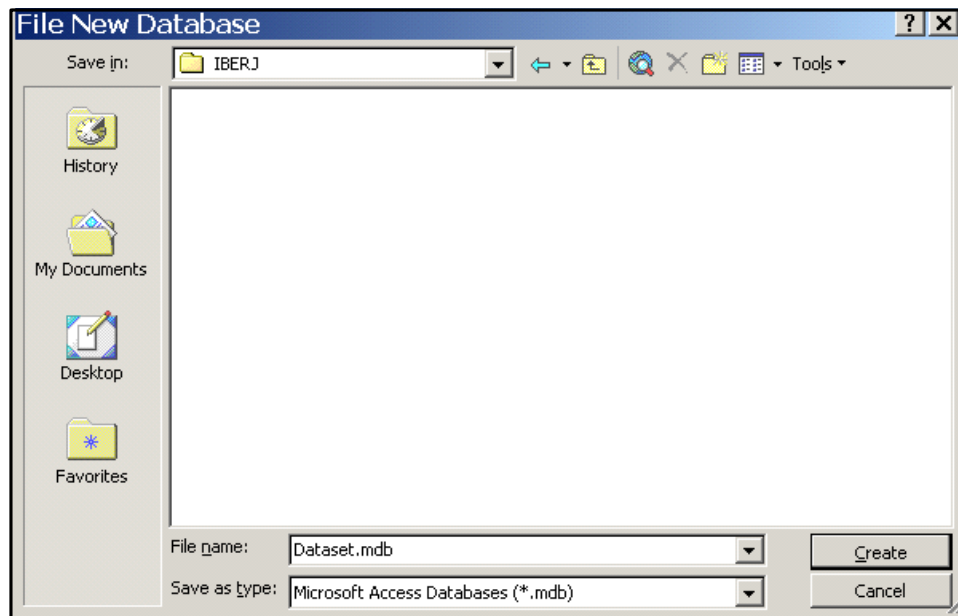


Phase II. Import the Streamlined Data into a Relational Database

Now that the Web page data has been streamlined and saved into a text file, the second phase is to import our new text file into a relational database (**MS Access**). The steps are as follows:

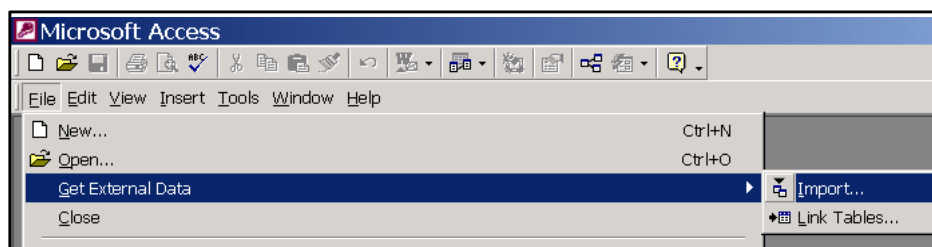
- A. Launch **MS Access** and create a new database; name it **Dataset.mdb** (see Figure 11).

Figure 11: MS Access New Database Dialog Box



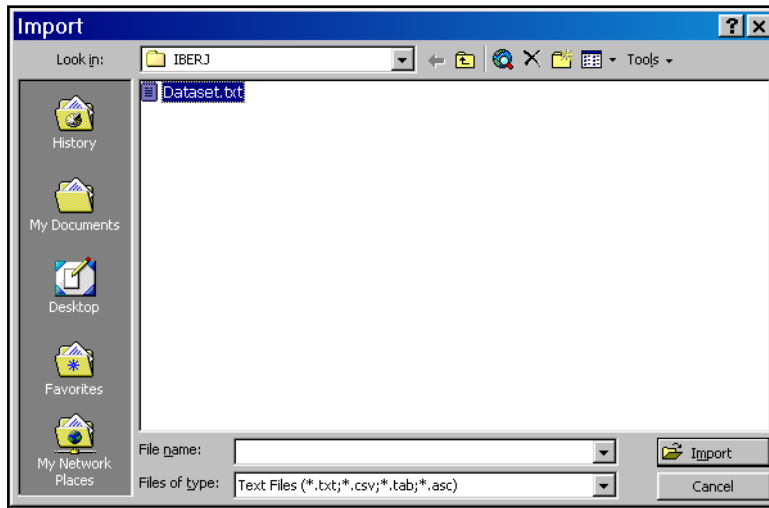
- B. The **Dataset.txt** file can now be imported as a new table in this database. Select **FILE/GET EXTERNAL DATA/IMPORT** (see Figure 12).

Figure 12: Get External Data



- C. The default **MS Access** file type is .mdb. To allow the text file that we are trying to import to appear in the list of possible files, click on the down arrow next to **Files of type** and choose **Text Files** (see Figure 13). Locate and select the **Dataset.txt** file; click **Import**.

Figure 13: Import Dialog Box



D. The **Import Text Wizard** dialog box appears.

1. Choose **Delimited**; click **Next** (see Figure 14).
2. Choose **Tab** as the field delimiter and check **First Row Contains Field Names**. Click **Next** (see Figure 15).

Figure 14: Import Text Wizard Step 1

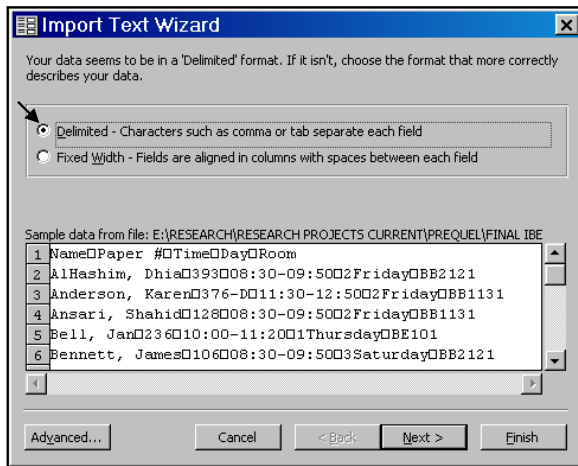
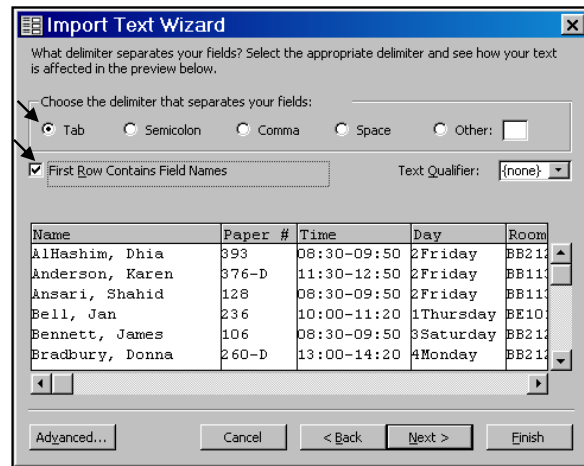


Figure 15: Import Text Wizard Step 2



3. Click the radio button next to **In a New Table** (which is the default); click **Next** (see Figure 16).
4. Leave all fields with the default **Data Type** of "Text" by clicking **Next** (see Figure 17).

Figure 16: Import Text Wizard Step 3

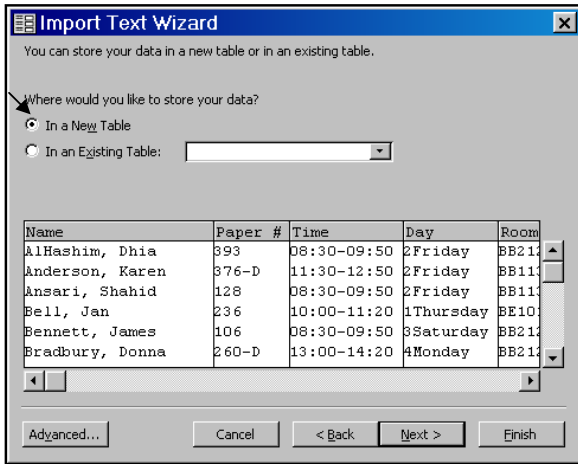
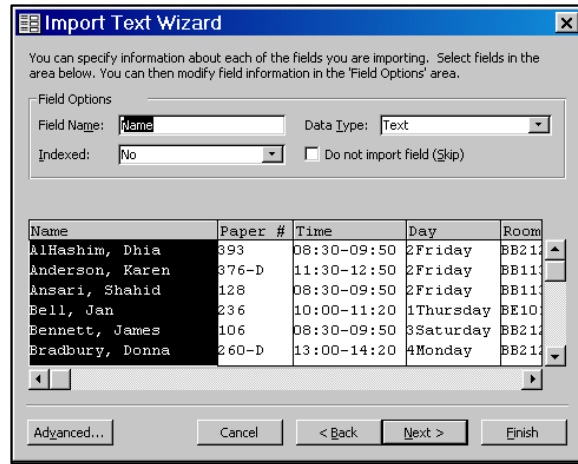


Figure 17: Import Text Wizard Step 4



5. Click the radio button next to **No Primary Key**; click **Next** (see Figure 18).
6. Name the table **Dataset**. Click **Finish** (see Figure 19).

Figure 18: Import Text Wizard Step 5

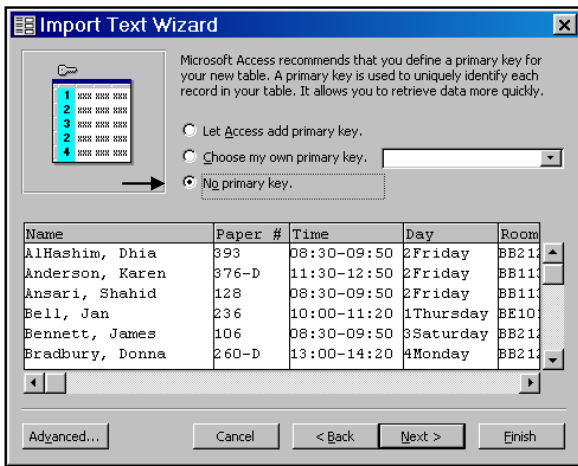
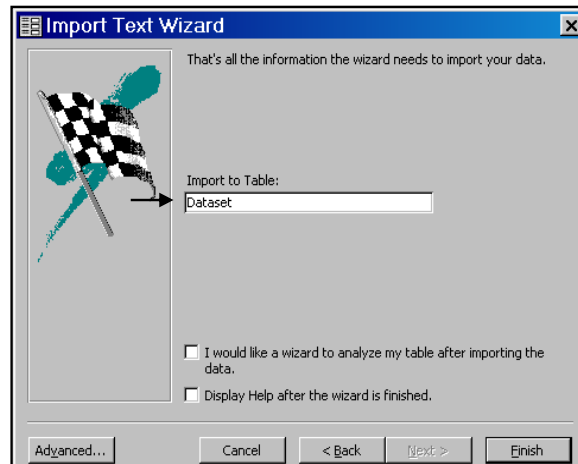


Figure 19: Import Text Wizard Step 6



7. The **Dataset** table now appears in the **Database Window** (see Figure 20).
8. Here is the **Dataset** table in datasheet view. Notice all five columns that were present in the Web page (see Figure 1) have been successfully imported into the new table (see Figure 21).

Figure 20: Database Window

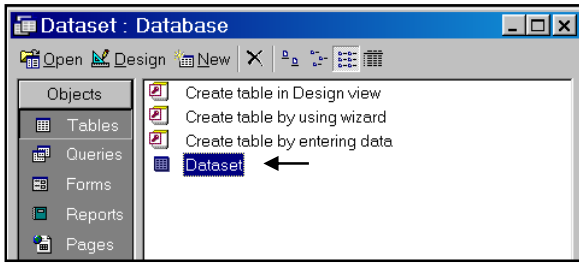
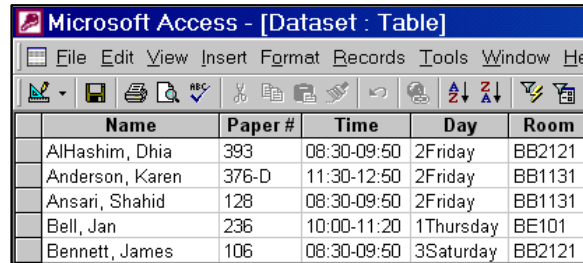


Figure 21: Dataset Table in Datasheet View



Phase III. Use SQL to Efficiently Correct, Recode, and Reorganize the Data

The data is now safely stored in a database table. In phase three, we will use SQL (Structured Query Language) to further clean up our data. For readers who might not be familiar with SQL, it is an easy to use language for communicating with a database. It can be used to create new tables, alter existing table structures, do searches of tables based on specified criteria, insert new records into tables, delete old records from tables, create new columns based on calculations, change existing data, and combine data from multiple tables. Most SQL statements have the following basic syntax:

```
SELECT Field1Name, Field2Name, Field3Name, ...
FROM TableName;
```

Here are some sample SQL statements needed to display specific sets of data from our newly imported **Dataset** table.

To Display:	Use this SQL Statement:
Every field from the Dataset table.	SELECT * FROM Dataset;
<u>Only</u> the Name and Room fields.	SELECT Name, Room FROM Dataset;
A field with a column heading that is different from the official field name, such as referring to the Room field as a “Location”, the keyword AS is used.	SELECT Room AS Location FROM Dataset;
Part of a field, such as displaying only the first two characters of the Room field, the function LEFT is used.	SELECT LEFT(Room,2) AS BuildingCode FROM Dataset;
To convert a field such as Quantity from text to integer (so that it is usable in calculations), the function CINT (convert to integer) is used.	SELECT CINT(Quantity) AS Qty FROM OrderLine;
To count the number of characters in the Name field, the LEN (length) function is used.	SELECT LEN(Name) AS FieldLength FROM Dataset;

All of the SQL commands that will be used in our ongoing demonstration are defined below:

SQL Command	Description
SELECT	Lists the field(s) that will be displayed in the resulting query
FROM	Lists the data source(s) -- one or more tables or queries
AS	Gives a field/table an alias
*	Selects all fields in the table/query
CINT(FieldName)	Converts the field type to integer
LEFT(FieldName,n)	Extracts <i>n</i> characters from the field starting from the left
RIGHT(FieldName,n)	Extracts <i>n</i> characters from the field starting from the right
LEN(FieldName)-n	Determines the length of the field and subtracts <i>n</i> from that number
Note: If a field or table name contains characters other than letters or digits (or is a reserved word in the SQL language) it must be enclosed in square brackets (e.g., [Building Code] vs. BuildingCode).	

Now that we have a basic understanding of how SQL statements are constructed, we can put our knowledge to work to deal with: (A) the incorrectly combined **DayNum** and **DayName** components of the **Day** field, (B) the need for recoding of the **Paper #** and **Time** fields, and (C) a second separate hypothetical example showing how to use a SQL statement to combine data from multiple tables.

A. Use SQL to Split the Incorrectly Combined Day Field into Two New Fields

In part A, we will deal with two problems inherent in the **Day** field:

- It is currently two fields (**DayNum** and **DayName**) combined into one. The two fields need to be separated.
- The resulting **DayNum** field is formatted as text. It needs to be an integer to be usable in the regression.

The steps are as follows:

1. Start a New Query
 - a. Click on the **Queries** button in the **Database Window** and then click on **New** (see Figure 22).
 - b. Select **Design View** from the list, then click **OK**.
 - c. In the **Show Table** dialog box (see Figure 23); click **Close**. In the upper left-hand corner of the toolbar; click **SQL**.
 - d. The SQL editor should now be displayed on the screen. This is where we will type our SQL code (see Figure 24).

Figure 22: New Query Dialog Box

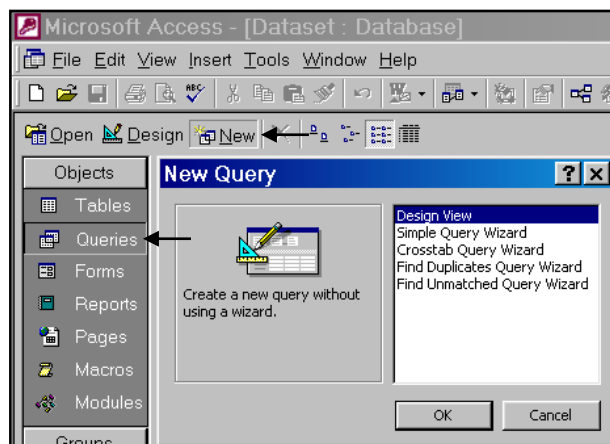


Figure 23: Show Table Dialog Box

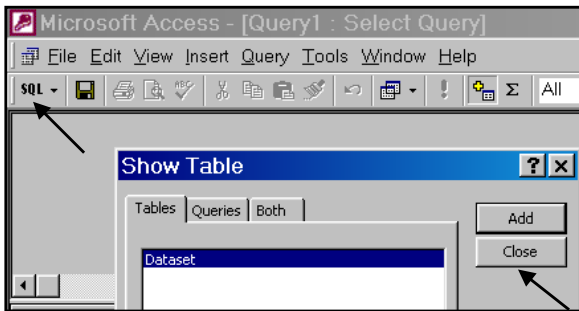
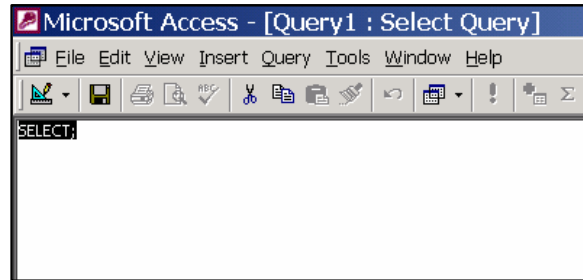


Figure 24: SQL Editor View

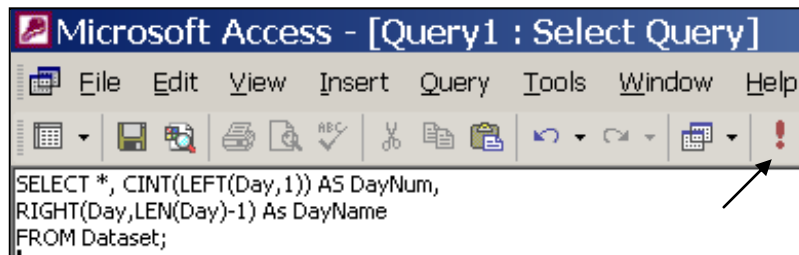


2. Type in and run a SQL statement to create the two new fields.

a. Enter the following statement:
**SELECT *, CINT(LEFT(Day,1)) AS DayNum,
 RIGHT(Day,LEN(Day)-1) As DayName
 FROM Dataset;**

b. Click the **Run (!)** button to execute the query (see Figure 25).

Figure 25: SQL Statement One in the Editor



c. Here, in datasheet view, are the results of the query (see Figure 26). Notice the two new fields on the right: **DayNum** and **DayName**.

Figure 26: Results of First Query

Name	Paper #	Time	Day	Room	DayNum	DayName
AlHashim, Dhia	393	08:30-09:50	2Friday	BB2121	2	Friday
Anderson, Karen	376-D	11:30-12:50	2Friday	BB1131	2	Friday
Ansari, Shahid	128	08:30-09:50	2Friday	BB1131	2	Friday
Bell, Jan	236	10:00-11:20	1Thursday	BE101	1	Thursday
Bennett, James	106	08:30-09:50	3Saturday	BB2121	3	Saturday

3. **Save** the Query.
 - a. Select **File/Save As**.
 - b. In the **Save As** dialog box, type the name of the query, **qryDataset01-ErrorCorrection** (see Figure 27).
 - c. Click **OK**.
 - d. The saved query should now be listed in the **Database Window** (see Figure 28).

Figure 27: Save As Dialog Box

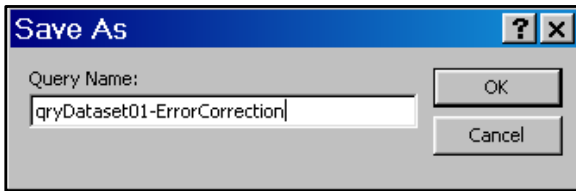
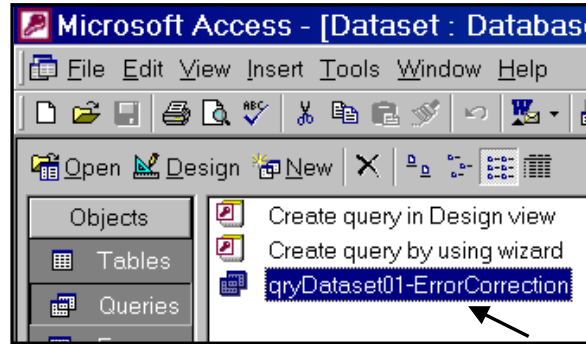


Figure 28: Database Window



B. Use SQL to Recode the Currently Unusable Paper # and Time Fields

The dataset now has two new fields: **DayNum** and **DayName**. Our planned statistical analysis requires two additional fields, **Time** and **Paper #**, neither of which is currently in usable form. We will now use SQL to create:

- An **Hour** field that includes the first two characters of **Time**, converted to an integer.
- A **PaperNum** field that includes only the first three characters of **Paper #**, converted to an integer. (Extracting the first three characters eliminates any “-D” suffixes.)

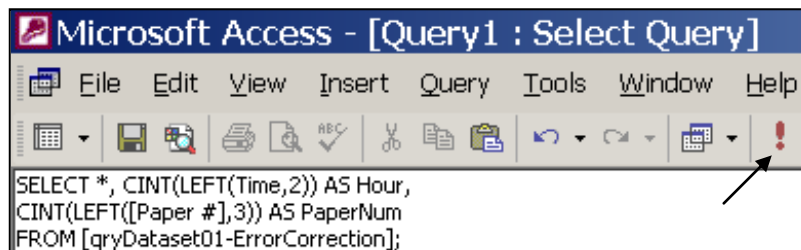
The steps are as follows:

1. Start a new query in SQL View.
2. Type and execute:

```
SELECT *, CINT(LEFT(Time,2)) AS Hour,
CINT(LEFT([Paper #],3)) AS PaperNum
FROM [qryDataset01-ErrorCorrection];
```

To create the two new fields (see Figure 29). The reader should note that this second query is based on the first query (**qryDataset01-ErrorCorrection**), rather than on the original **Dataset** table.

Figure 29: SQL Statement Two in the Editor



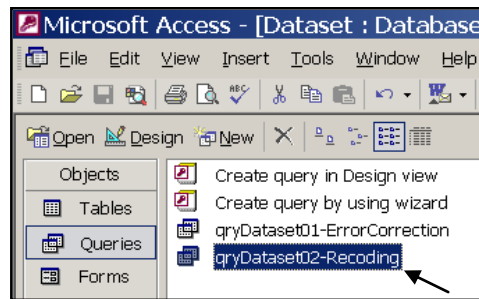
- Here are the results (see Figure 30). Notice the two new fields on the right: **Hour** and **PaperNum**.

Figure 30: Results of Second Query

Name	Paper #	Time	Day	Room	DayNum	DayName	Hour	PaperNum
AlHashim, Dhia	393	08:30-09:50	2Friday	BB2121	2	Friday	8	393
Anderson, Karen	376-D	11:30-12:50	2Friday	BB1131	2	Friday	11	376
Ansari, Shahid	128	08:30-09:50	2Friday	BB1131	2	Friday	8	128
Bell, Jan	236	10:00-11:20	1Thursday	BE101	1	Thursday	10	236
Bennett, James	106	08:30-09:50	3Saturday	BB2121	3	Saturday	8	106
Bradbury, Donna	260-D	13:00-14:20	4Monday	BB2121	4	Monday	13	260

- Save the new query as: **qryDataset02-Recoding** (see Figure 31).
- Close **MS Access**.

Figure 31: Main Database Window



C. Use SQL to Combine Data from Multiple Tables

Although our demonstration only involves one table, the reader should also note the extreme usefulness of SQL in integrating data from two or more tables. As an example, consider the situation of needing to combine student demographic and course grade data described in problem #2 of the **Introduction** to this paper. Although this task would be overwhelming if handled manually (matching 300,000 records), it is a simple task for SQL.

Suppose that we want to extract the high school code (**HSCode**) from a demographic table (**Student**) and the course grade on a 4-point scale (**GP**) from a course grade table (**Grades**). The two tables have a field in common, **StudentID**, which can be used as a link, and that we would like to include as a column in the result.

To combine the information from the two tables, we simply modify our original query syntax to: (a) include a second table on the FROM line and (b) add a new WHERE line that indicates which fields are to be used to link the tables. The revised syntax, which is referred to as a “join,” is as follows:

```
SELECT Field1Name, Field2Name, Field3Name,
FROM Table1Name, Table2Name
WHERE Table1Name.LinkingFieldName = Table2Name.LinkingFieldName;
```


Using this syntax, the query to join our **Student** and **Grades** tables would be as follows:

```
SELECT HSCode, Grades.StudentID, GP
FROM Student, Grades
WHERE Student.StudentID = Grades.StudentID;
```

This query follows the revised syntax exactly except for the inclusion of the table name prefix on the **StudentID** field (i.e., Grades.StudentID). Inclusion of the table name is required here because the **StudentID** field exists in both the **Student** and **Grades** tables. Omission of a table name prefix in this situation would cause the statement to be ambiguous and would result in a query execution error. Although a table name prefix is required for the linking field, the choice of tables is arbitrary (Student.StudentID would also have been acceptable).

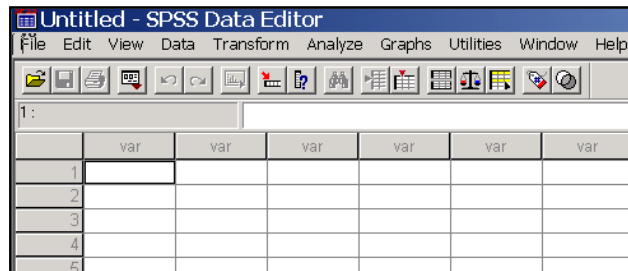
Phase IV. Import the Improved Data into a Statistical Analysis Package and Run the Regression

Part C of *Phase III* was a bit of a detour to a situation involving multiple tables. We will now be returning to our ongoing single-table demonstration. As previously stated, our goal is to use our data to run a multiple linear regression with two independent variables. Now that our data has been corrected and recoded (and is available in the query named *qryDataset02-Recoding*), it is ready to be: (A) imported into a statistical analysis package and then (B) used in a regression of **PaperNum** on **DayNum** and **Hour**. As mentioned earlier, although we will be using **SPSS**, the same process could be accomplished in another statistical analysis package. The steps are as follows:

A. Import the Query into SPSS.

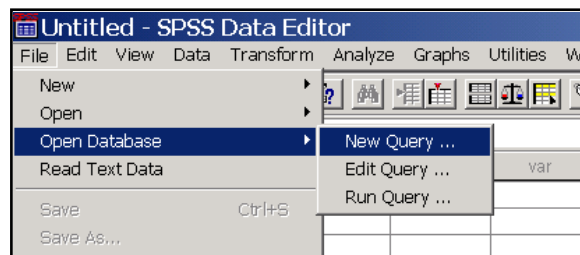
1. Open a new **SPSS Data Editor Window** (see Figure 32).

Figure 32: New SPSS Data Editor Window



2. Before we can import our query we must first establish it as a “Microsoft Data Source.” This is done by use of **SPSS’s Database Wizard**. To access this wizard, select **File/Open Database/New Query** (see Figure 33).

Figure 33: Accessing the Database Wizard



3. The **Database Wizard** dialog box appears.
 - a. Click the **Add Data Source** button (see arrow in Figure 34).
 - b. Click the **Add** button (see Figure 35).

Figure 34: Database Wizard Step a

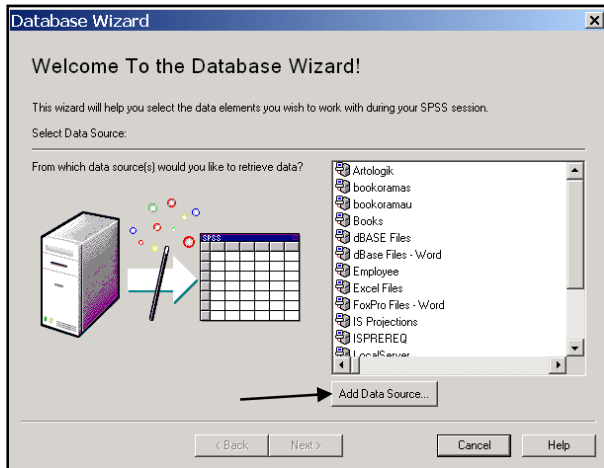
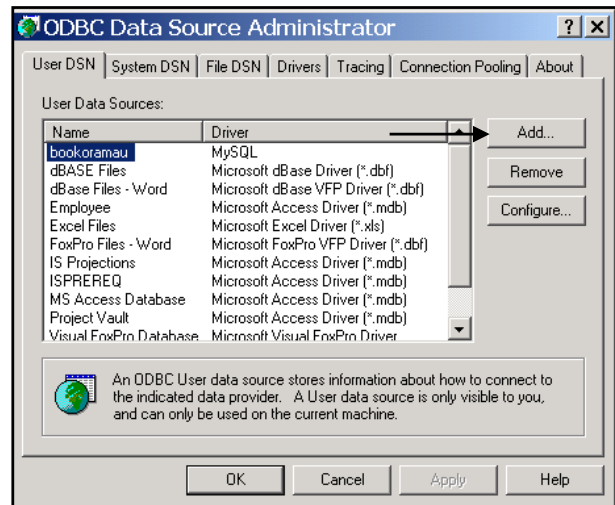


Figure 35: Database Wizard Step b



- c. Select **Microsoft Access Driver (*.mdb)**; click **Finish** (see Figure 36).
- d. For the **Data Source Name**, type in: **Dataset**; click **Select** (see Figure 37).
- e. Locate the **Dataset.mdb** file that we saved at the end of *Phase III*, part B; click **OK** (see Figure 38).
- f. Note that the **Dataset.mdb** file has been recorded as the database associated with the Data Source Name "Dataset," click **OK** (see Figure 39).
- g. Note that **Dataset** is now in the list of available User Data Sources; click **OK** (see Figure 40).

Figure 36: Database Wizard Step c

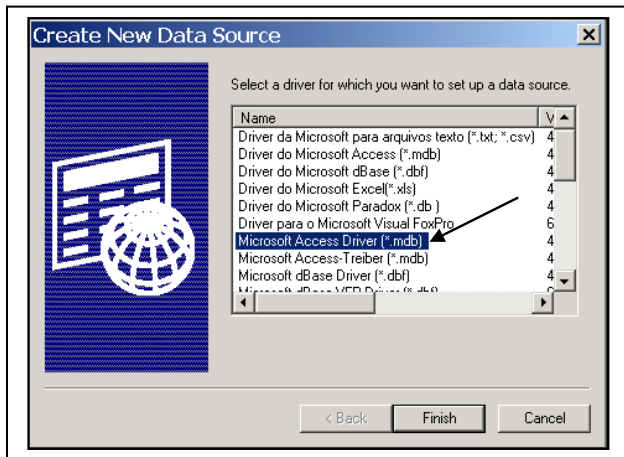


Figure 37: Database Wizard Step d

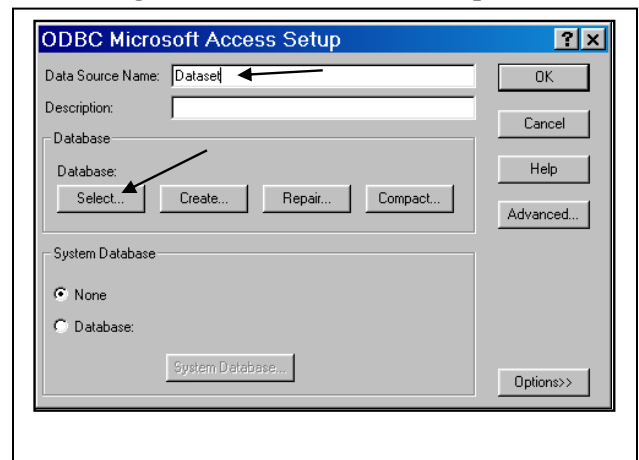


Figure 38: Database Wizard Step e

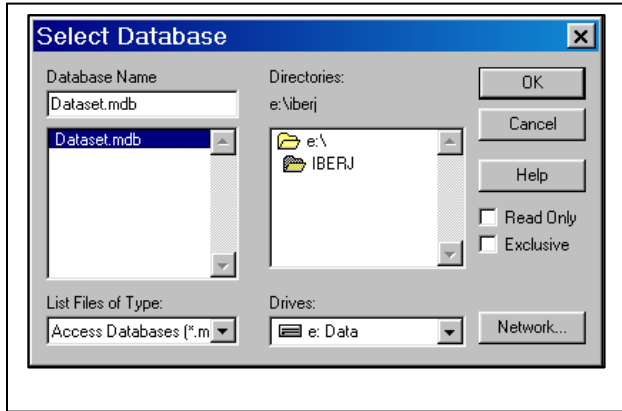


Figure 39: Database Wizard Step f

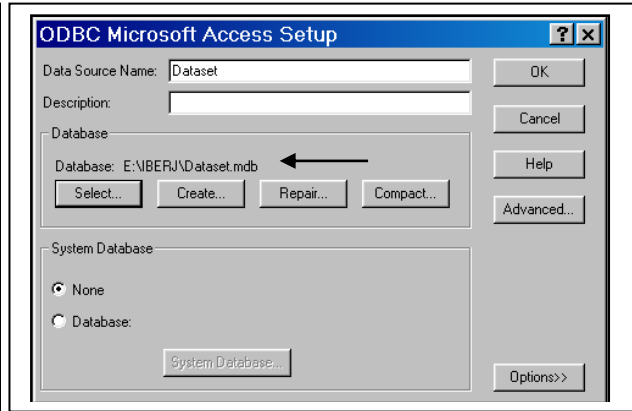
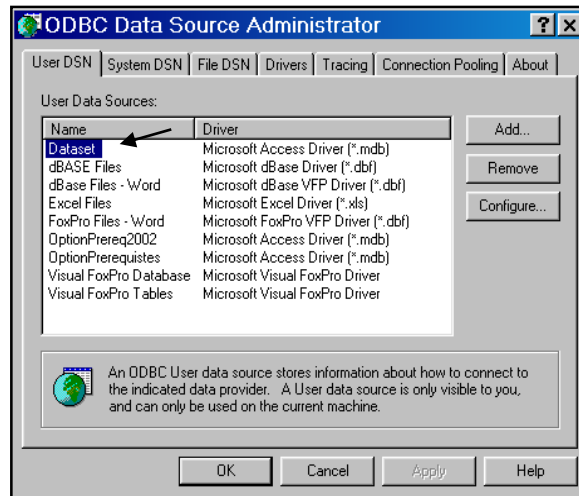
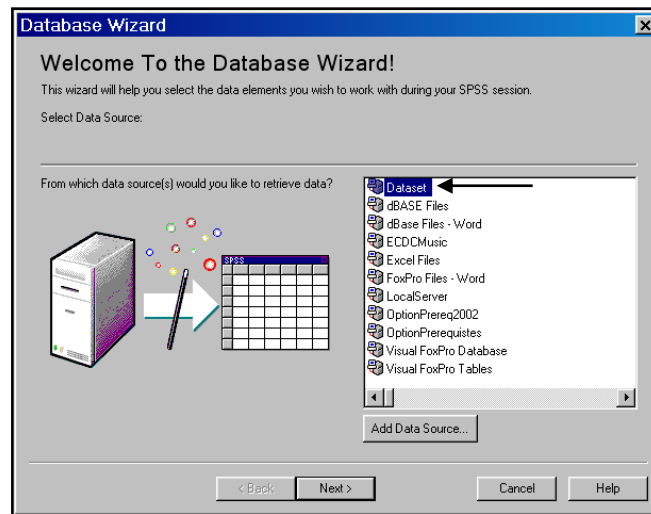


Figure 40: Database Wizard Step g



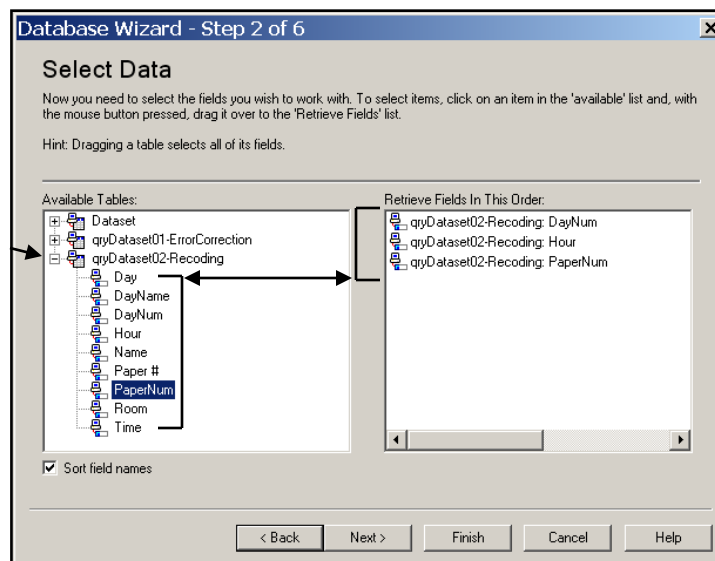
4. Now that we have established our Microsoft Data Source, our next task is to import the fields required for our regression from that Data Source into **SPSS**:
 - a. Select the **Dataset** database as the data source; click **Next** (see Figure 41).

Figure 41: Selecting the Data Source



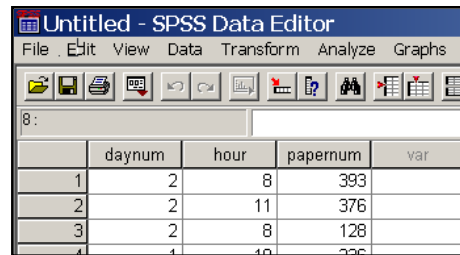
- b. Under **Available Tables** in the left-hand list, click the plus (+) sign next to **qryDataset02-Recoding**. A list of nine available fields should appear (see Figure 42).
- c. Double-click on the **DayNum**, **Hour**, and **PaperNum** fields that we need for our analysis; they will move over to the right-hand list.
- d. Click **Finish**.

Figure 42: Selecting the Desired Fields



5. The data for our three fields is now in the **SPSS Data Editor** (see Figure 43).

Figure 43: Data in SPSS Data Editor

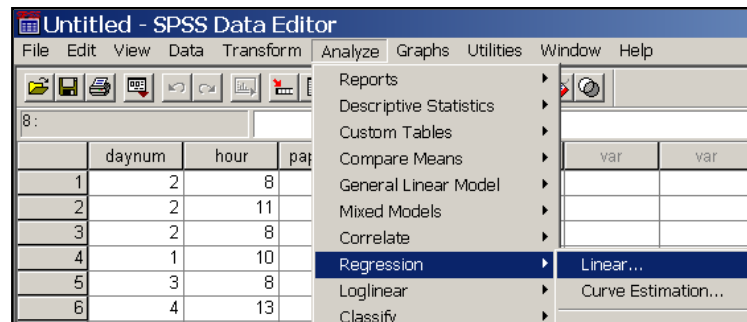


	daynum	hour	papernum	var
1	2	8	393	
2	2	11	376	
3	2	8	128	
4	1	10	326	

B. Run the Regression

1. Our final task is to do the multiple regression. From the menu, select **Analyze/Regression/Linear** (see Figure 44).

Figure 44: Starting a Linear Regression in SPSS



2. The **Linear Regression** dialog box appears (see Figure 45).
 - a. From the field list on the left-hand side, select the **PaperNum** field; click the **arrow** to the left of **Dependent**.
 - b. Select the **DayNum** field from the field list; click the **arrow** next to **Independent**.
 - c. Select the **Hour** field in the same way. Click **OK**.

Figure 45: Linear Regression Dialog Box

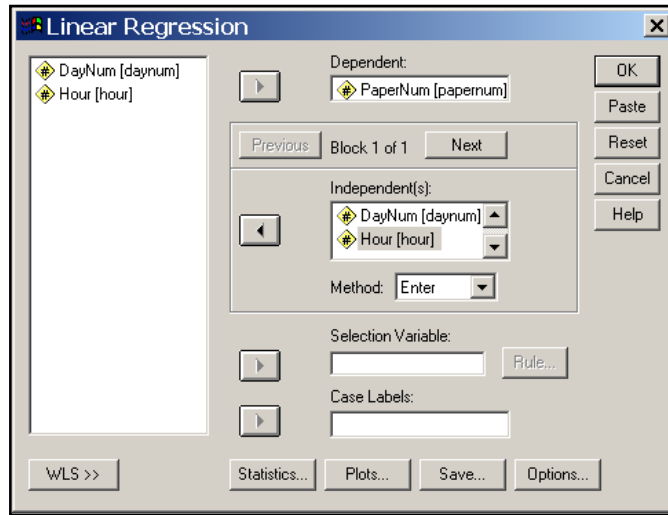


Figure 46: Regression Results in SPSS Viewer

3. The results of the regression are now displayed in the **SPSS Viewer** (see Figure 46). As can be seen from the significance levels in the right-hand column of the Coefficients table, only one of the two independent variables, **Hour**, is shown to have a significant relationship (i.e., .000) with the dependent variable, **PaperNum**. This result could be interpreted as meaning that people who volunteer to give their presentations earlier in the day (as measured by a smaller hour number) also tend to submit their papers earlier in the submission period (as measured by a numerically smaller assigned paper number).
4. Now that our regression is complete, it is time to store our results. Save the SPSS Viewer file as **Dataset.spo**; save the data file as **Dataset.sav**. Close **SPSS**.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.760 ^a	.578	.552	61.562

a. Predictors: (Constant), Hour, DayNum

ANOVA ^b						
Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	166242.1	2	83121.072	21.932	.000 ^a
	Residual	121276.8	32	3789.901		
	Total	287519.0	34			

a. Predictors: (Constant), Hour, DayNum
b. Dependent Variable: PaperNum

Coefficients ^a						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	-51.918	53.971		-.962	.343
	DayNum	1.028	8.233	.014	.125	.901
	Hour	26.339	3.982	.761	6.615	.000

a. Dependent Variable: PaperNum

Conclusion

The stated purpose of this paper was to help researchers deal with ill-behaved data through the use of a variety of software packages, with particular focus on relational database software. We believe we have accomplished this goal through a demonstration consisting of the following steps:

- Using a word processor and a spreadsheet to reformat data copied from a web page.
- Importing the reformatted data into a database.
- Using SQL (Structured Query Language) to efficiently clean up, recode, and reorganize the data.
- Importing the improved data from the database into a statistical analysis package and running a regression.

We hope that our hypothetical research project will provide the reader with some tools and techniques that will make him/her unafraid of dealing with data that are "not ready for prime time" and that might otherwise go unused.

Beyond this, we have a larger purpose. As information system (IS) professionals, we have come to: (a) have a great deal of respect for the power of the relational database model and (b) be impressed by the relative ease of use of database software. We can also envision a myriad of uses for databases in a multitude of contexts.

Unfortunately, the potential of database software has not been fully realized in areas outside the IS field. Possible reasons for the minimal use of database software outside of IS are that people in most other professions: (a) have never been exposed to it, (b) have only had minimal exposure to it, or (c) have used it, but still find it somewhat intimidating.

We want to change this. The broader purpose of this paper is to demystify database software and encourage the reader to learn more about the usefulness of database tools and techniques in their own field of expertise. To this end, we have included in our reference list a number of SQL, database theory, and MS Access texts that could be used for further exploration of various database topics. We hope that they will prove helpful.

Suggestions for Future Research

Suggestions for future research in this area of study include how to: (a) use various software packages to remedy other types of imperfections in archival datasets, and (b) use SQL to combine datasets from multiple sources into one file that can be analyzed by a statistical package.

References

1. Bowman, J., and Emerson, S., *The Practical SQL Handbook: Using SQL Variants, Fourth Edition*, Addison-Wesley, San Francisco, CA, 2001.
2. Dietrich, S., *Understanding Relational Database Query Languages*, Prentice Hall, Upper Saddle River, NJ, 2001.
3. Hernandez, M., and Viescas, J., *SQL Queries for Mere Mortals: A Hands on Guide to Data Manipulation in SQL*, Addison Wesley, San Francisco, CA, 2000.
4. Irwin, M., and Prague, C., *Microsoft Access 2002 Bible Gold Edition*, Hungry Minds Inc., New York, NY, 2002.
5. Morgan, B. and Perkins, J., *Teach Yourself SQL in 14 Days*, SAMS Publishing, Indianapolis, IN, 1995.
6. Patrick, J., *SQL Fundamentals, Second Edition*, Prentice Hall, Upper Saddle River, NJ, 2002.
7. Pratt, P., *A guide to SQL, Fifth Edition*, Course Technology, Cambridge, MA, 2001.
8. Rob, P., and Coronel, C., *Database Systems Design, Implementation, & Management, Fifth Edition*, Course Technology, Cambridge, MA, 2001.

Notes