SPSS: Data Manipulation and Advanced Topics

For Windows

Updated: August 2012

Table of Contents

Section 1: Introduction

This document is the fourth module of a four module tutorial series. This module describes the use of SPSS to do advanced data manipulation such as splitting files for analyses, merging two files, aggregating datasets, and combining multiple tables in a database into an SPSS dataset in the first section. Several advanced topics are included in the second section, including the use of SPSS syntax, the SPSS Visual Basic editor, and SPSS Macros.

If you are not familiar with SPSS or need more information about how to get SPSS to read your data, consult the first module, SPSS for Windows: Getting Started, of this four part tutorial. This set of documents uses a sample dataset, *Employee data.sav*, that SPSS provides. It can be found in the root SPSS directory. If you installed SPSS in the default location, then this file will be located in the following location: C:\Program Files\SPSS\Employee Data.sav.

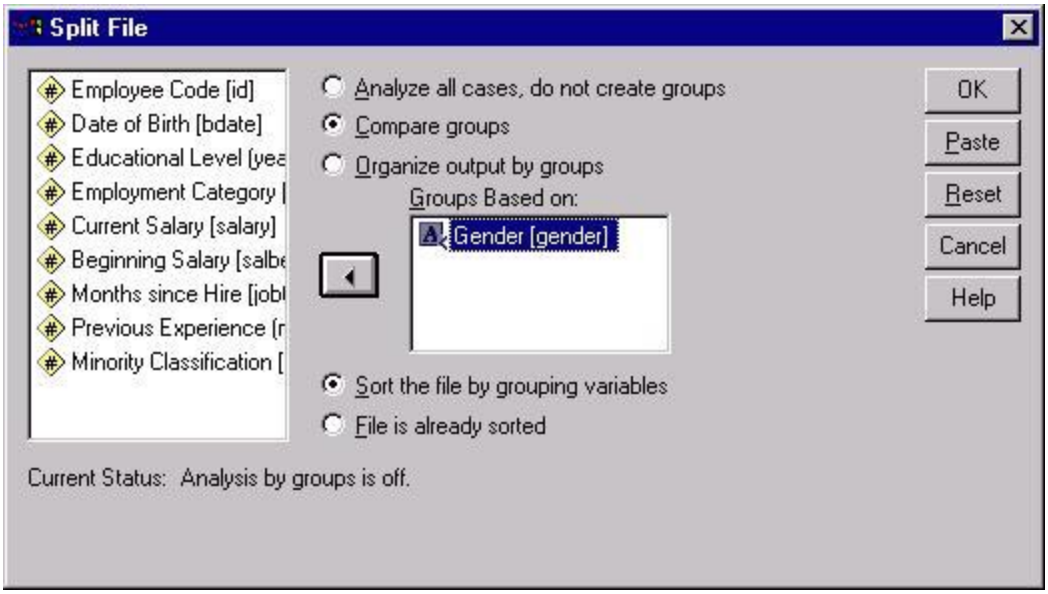
Section 2: Data Manipulation

2.1 Splitting Files

In some situations, you may want to perform the same analysis on different groups within the same dataset. For example, if you are comparing gender differences in salaries for men and women, you may want to first obtain separate sets of descriptive statistics such as the mean, standard deviation, etc. for men and women. Analyses such as these can be conducted by first selecting the *Split File* function from the *Data* menu in the Data Editor:

Data
Split File...

This will produce the following dialog box:



You have three possible choices for the organization of your analyses. The default choice is the *Analyze all cases, do not create separate groups*. With this option, all analyses are conducted on the entire dataset. Because the *split file* command remains in effect indefinitely, you should reset this option when you no longer want a split file analysis.

The next two choices, *Compare groups* and *Organize output by groups* result in the same values in the output, regardless of the analysis being performed, but they differ in the way in which the output is presented. These two options require you to select one or more variables from the variable list on the left side of the dialog box by first clicking on variables by which the file is to be split, then clicking on the arrow to the right of the variable list. This will display the variables by which the file is to be split in the *Groups Based on* box. Clicking the **OK** button is the final step to activate the split file. If the *Compare groups* option was specified, as shown above, then

all groups will be analyzed separately, but information on all groups will be contained in the same table in the output. Here is an example of some descriptive statistics with this option active:

Descriptive Statistics

Gender		N	Minimum	Maximum	Mean	Std. Deviation
Female	Current Salary	216	$15,750	$58,125	$26,031.92	$7,558.02
	Valid N (listwise)	216				
Male	Current Salary	258	$19,650	$135,000	$41,441.78	$19,499.21
	Valid N (listwise)	258				

The *Organize output by groups* option will produce separate tables for each group. The example below shows the result of selecting the identical options for descriptive statistics, as above, but with the *Organize output by groups* option active.

Gender = Female

Descriptive Statistics[a]

	N	Minimum	Maximum	Mean	Std. Deviation
Current Salary	216	$15,750	$58,125	$26,031.92	$7,558.02
Valid N (listwise)	216				

a. Gender = Female

Gender = Male

Descriptive Statistics[a]

	N	Minimum	Maximum	Mean	Std. Deviation
Current Salary	258	$19,650	$135,000	$41,441.78	$19,499.21
Valid N (listwise)	258				

a. Gender = Male

2.2 Merging Files

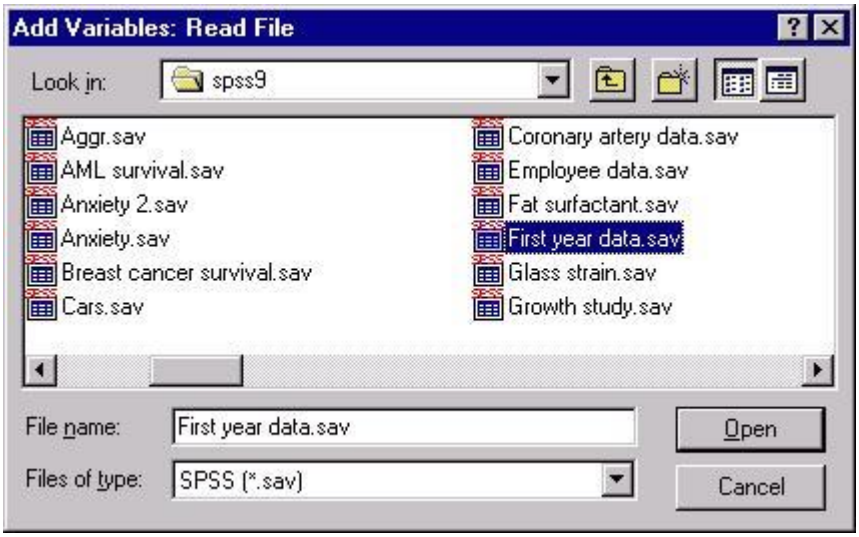
At times, you may want to conduct an analysis on data that is stored in more than one data file. Because SPSS only has one working data file at a time, you must first merge data into a single file for the purpose of analysis. For example, if there were another dataset that contained employee's salaries after one year and you wanted to analyze that information with the data in the current dataset, then you would first need to merge these two files.

But before merging files, keep these three considerations in mind: (1) There should be at least on identification variable in both files that contains the same values so that the files can be matched. (2) Variables with the same names in both datasets that are not being used to match cases will be need to be recode or they will be excluded. Therefore, if both datasets contain variables that have the same name but different values, one of these variables will be excluded. (3) You must sort your data in ascending order before performing a file merge.

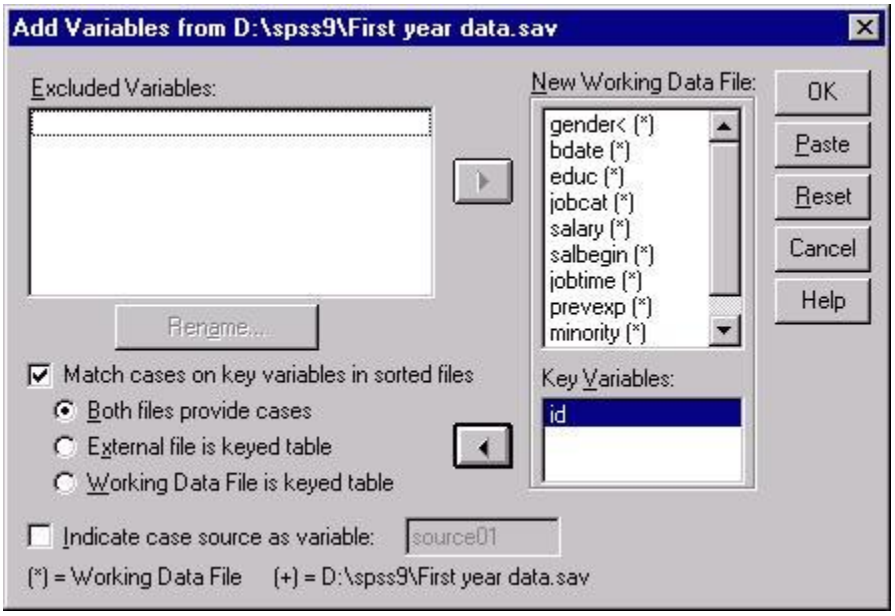
Data files can be merged using dialog boxes available under the *Data* menu item in the Data Editor:

Data
Merge Files

Selecting *Merge Files* will give you two options: *Add Cases* and *Add Variables*. The *Add Cases* option combines two files with different cases that have the same variables, whereas the *Add Variables* adds new variables on the basis of variables that are common to both files. The following example illustrates the use of the *Add Variables* option to combine two files that share a common variable. Here, the *Add Variables* option will be used to combine the working data file, *Employee data.sav* with an external file (*First year data.sav*) that contains two variables, employee ID (*id*) and salary after one year (*oneyear*). The variable *id* has identical values to the variable with the same name in the *Employee data.sav*, dataset whereas the variable *oneyear* is unique to the external data file. After selecting the *Add Variables* option, the following dialog box will appear:

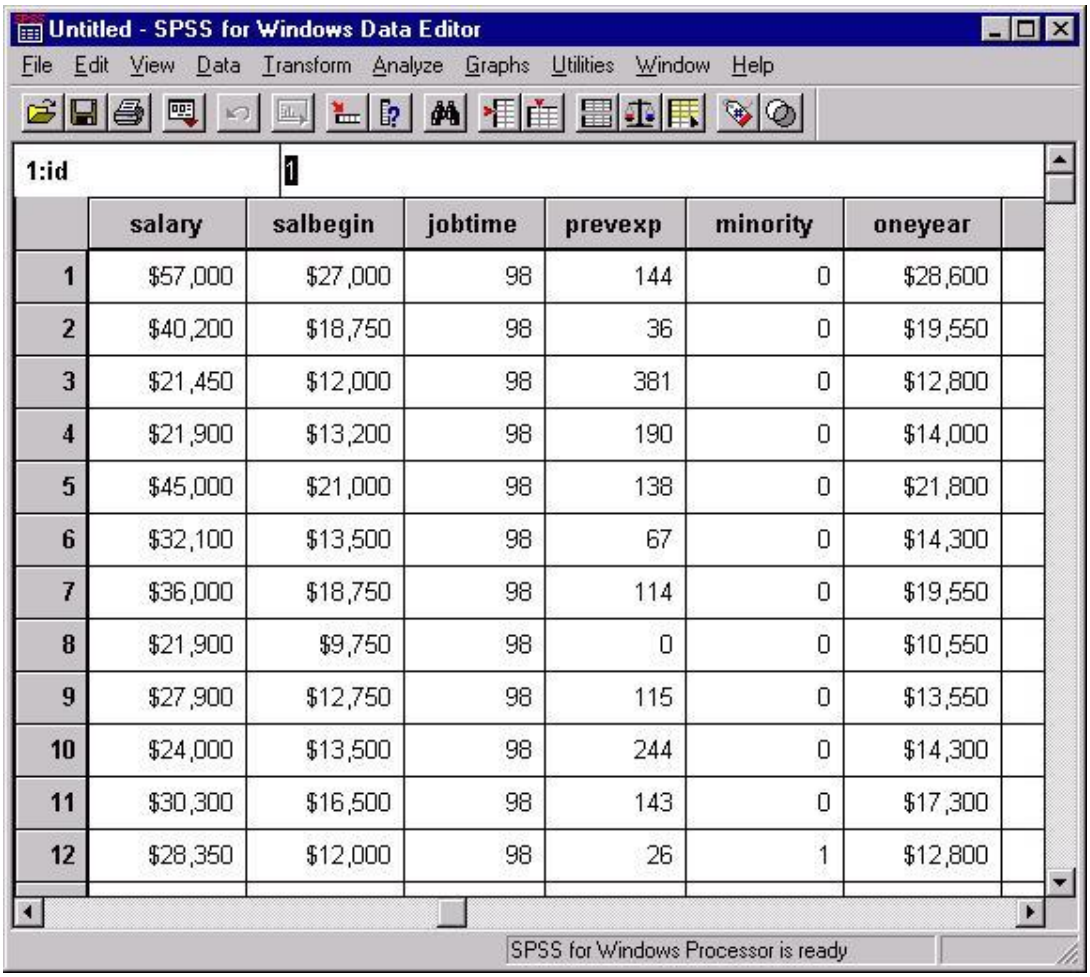


This is the first of two dialog boxes, in which you can select a file to merge with the working data file. Here, the dataset, *First year data.sav* has been selected to merge with the working dataset. This is the dataset containing the data on employees' salaries after one year. After selecting a file for the ensuing merge, click the **Open** button and the following dialog box will appear:



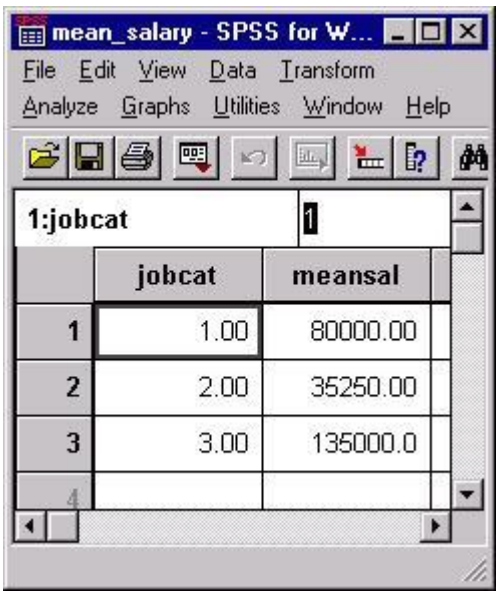
When this box first appears, variable names that appear in both datasets will be in the box labeled *Excluded Variables*. In this example, *id* is the only variable that fits this description. At least one variable must be common to both files in order to perform a merge. All variables that are unique to one of the data files will appear in the box labeled *New Working Data File*, indicating that they will appear in the file that is the merged product of the two files. The file in which these variables originated is indicated by the symbol following the variable name: variables from the working dataset are assigned the '*'; variables that originate from the external dataset are assigned a '+'. The default setting merges all of the nonduplicated variables. Any variables that you do not want in the merged file can be highlighted in the box labeled *New Working Data File* and moved to the *Excluded Variables* box by clicking on the arrow in between these two boxes.

To match files on the basis of one or more variables, click on the box labeled *Match cases on key variables in sorted files*. This will activate the options underneath this heading. Select the option *Both files provide cases* for a standard file merge. The arrow to the left of the box labeled *Keep Variables* will then become active, allowing you to select a variable from the box labeled *Excluded Variables*. In the above example, only the variable *id* appeared in this box because it was the only variable that is in both data files. To move it from the *Excluded Variables* box to the *Key Variables* box, as has already been done above, click on the arrow button to the left of the *Key Variables* box. You are now ready to run the procedure by clicking on the **OK** button. Doing so will produce the following dataset:



	salary	salbegin	jobtime	prevexp	minority	oneyear
1	$57,000	$27,000	98	144	0	$28,600
2	$40,200	$18,750	98	36	0	$19,550
3	$21,450	$12,000	98	381	0	$12,800
4	$21,900	$13,200	98	190	0	$14,000
5	$45,000	$21,000	98	138	0	$21,800
6	$32,100	$13,500	98	67	0	$14,300
7	$36,000	$18,750	98	114	0	$19,550
8	$21,900	$9,750	98	0	0	$10,550
9	$27,900	$12,750	98	115	0	$13,550
10	$24,000	$13,500	98	244	0	$14,300
11	$30,300	$16,500	98	143	0	$17,300
12	$28,350	$12,000	98	26	1	$12,800

In addition to matching files where there is a one-to-one match between cases in the files, you may also want to match two files on the basis of a particular variable. For example, you may want to add to your data file a data column containing the average salary for a person's job category. In this situation, you could match the employee dataset containing 474 cases with a dataset that contained only three rows. Each row in this dataset will represent one of the three possible job categories. The dataset for this example, *mean_salary.sav*, is shown below, containing a variable for job category, *jobcat*, and a variable representing the average salary for that job category, *meansal*.



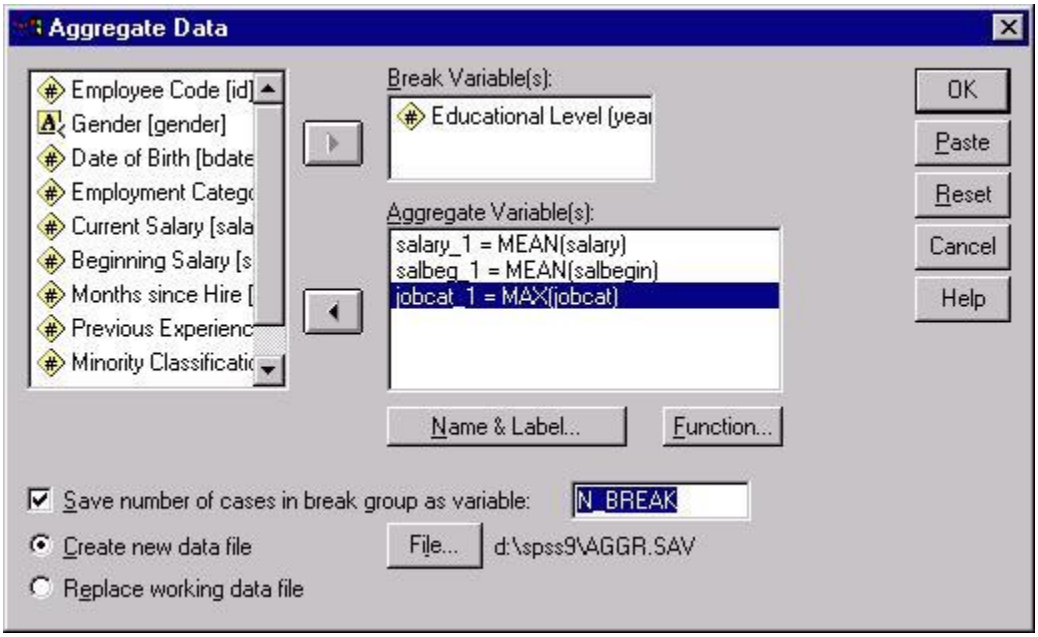
To do the type of file merge described above, select the *External file is keyed table* option in the *Merge* dialog box (this assumes that the *Employee data.sav* file is open in the Data Editor as opposed to the *mean_salary.sav* dataset--in which case you would have selected the *Working data file is keyed table* option). This option will require that you specify a variable on which the two files can be matched. Here, you should match on the basis of *jobcat*, as it is the one variable that the two files have in common. To designate a key variable, highlight the variable you want to use for your match in the *Excluded Variables* box and move it into the *Key Variables* box by clicking on the arrow next to that box. Clicking **OK** at this point will execute the match. You will receive a dialog box reminding you that both files need to be sorted on the key variable. If the files were not sorted on the key variable, *jobcat* in the above example, the file merge will fail. The file resulting from this process will contain the variable *meansal* for all cases, and the value will be the average salary for each person's job category.

2.3 Aggregating Files

Aggregating files is another frequently used data manipulation procedure. The *Aggregate* procedure allows you to condense a dataset by collapsing the data on the basis of one or more variables. For example, to investigate the characteristics of people in the company on the basis of the amount of their education, you could collapse all of the variables you want to analyze into rows defined by the number of years of education. To access the dialog boxes for aggregating data, choose the menu item:

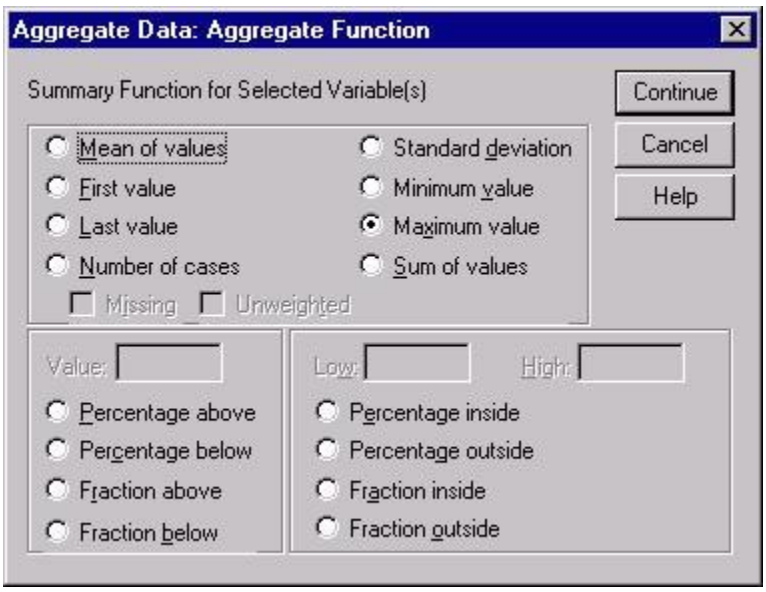
Data
Aggregate...

This will produce the following dialog box:

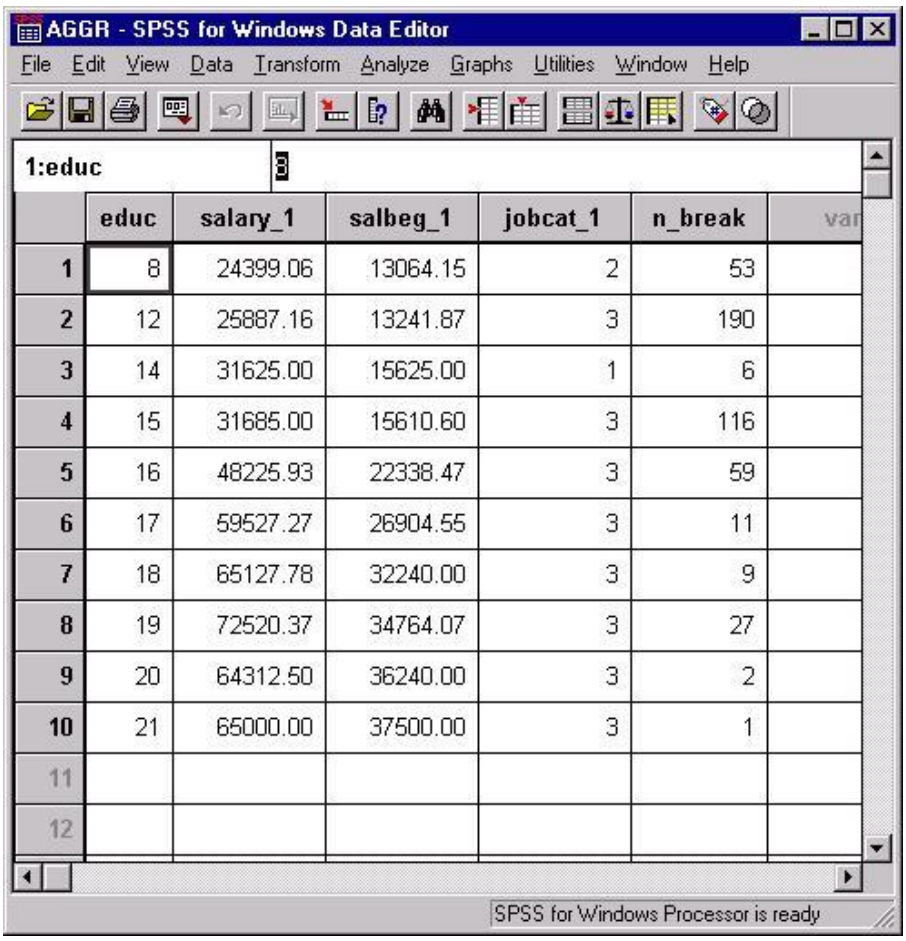


This dialog box shows an example aggregation. The top box, labeled *Break Variable(s)*, contains the variable within which other variables are summarized. Here, the variable contains a value representing a person's years of education. The box below, labeled *Aggregate Variable(s)*, contains the variables that will be collapsed. The variables *salary*, *salbegin*, and *jobcat* will all be summarized for each level of the variable *educ*. Two options available in the bottom left corner of the dialog box should be considered each time you conduct an aggregate function. The first allows you to save the number of cases that were collapsed at each level of the break variable or variables as a new variable. The second option determines whether the aggregated working data file is saved as a new file or replaces the working dataset. If you click on the button labeled *Create new data file*, you can then select the location and name of the file by clicking on **File**. If you click on the button labeled *Replace working data file*, then the resulting data file will be written over the original data file. These changes will be permanent if you save the ensuing aggregated dataset.

There are several options for summarizing variables. In the above example, the aggregated file will contain the means for *salary* and *salbegin* and the maximum value for *jobcat* within each level of *educ*. The summary function is indicated in the *Aggregate Variable(s)* dialog box in front of the original variable name. To the left of this expression is a new variable name that is assigned automatically. To change the summary function, first click on the name of the variable whose function you want to change, then click on the button labeled *Function*, which will produce the following dialog box:



This dialog box allows you to change the summary function for each variable individually. For example, here the function for the variable *jobcat* was changed from the default function, *Mean of values*, to the *Maximum value* function, which returns the maximum value of the variable within each level of the break variable. To select the specific function that you want to use for summarizing a variable, click on the radio button next to it. If you select one of the percentage or fraction functions in the bottom half of the box, you will also have to supply a value or two. When you finish defining your function using those dialog boxes, you can execute the aggregation by clicking on the **OK** button in the main aggregate dialog box. Doing so will write the data shown in the following figure to the file *D:\spss9\AGGR.SAV*.



	educ	salary_1	salbeg_1	jobcat_1	n_break	var
1	8	24399.06	13064.15	2	53	
2	12	25887.16	13241.87	3	190	
3	14	31625.00	15625.00	1	6	
4	15	31685.00	15610.60	3	116	
5	16	48225.93	22338.47	3	59	
6	17	59527.27	26904.55	3	11	
7	18	65127.78	32240.00	3	9	
8	19	72520.37	34764.07	3	27	
9	20	64312.50	36240.00	3	2	
10	21	65000.00	37500.00	3	1	
11						
12						

The first column, *educ*, is the break variable that represents the number of years of education an employee has had. The next two columns represent the mean current salary and mean beginning salary of employees within each educational level. These values were obtained by using the mean function to summarize these variables. The *Maximum value* option was used to produce the value in the fourth column, labeled *jobcat_1*. In this column, the value 3 represents a managerial position. The last column, labeled *n_break*, contains the number of cases that were collapsed into each level of *educ*.

One common use of aggregation is to collapse a dataset into a single row that contains a summary statistic for every variable in the dataset. To do this, you will need a break variable that is identical for every case in the dataset. Thus, when you collapse across this break variable that is the same for all cases, the resulting aggregated dataset will be reduced to a single row. To create this break variable, use the compute procedure. The use of dialog boxes to obtain a new variable is described in Section 3 of the first module of this SPSS tutorial series, "Getting Started." If you use the dialog boxes to obtain a new break variable, you would simply type in the name of a new variable in the *Target Variable* box and type 1 in the *Numeric Expression* box. Or, you could use the following SPSS syntax to compute a new break variable:

```
COMPUTE breakvar = 1.
EXECUTE.
```

This will produce a new variable, named *breakvar*, containing the value 1 for all cases in the dataset. This variable would then be used as the break variable for the *Aggregate* procedure.

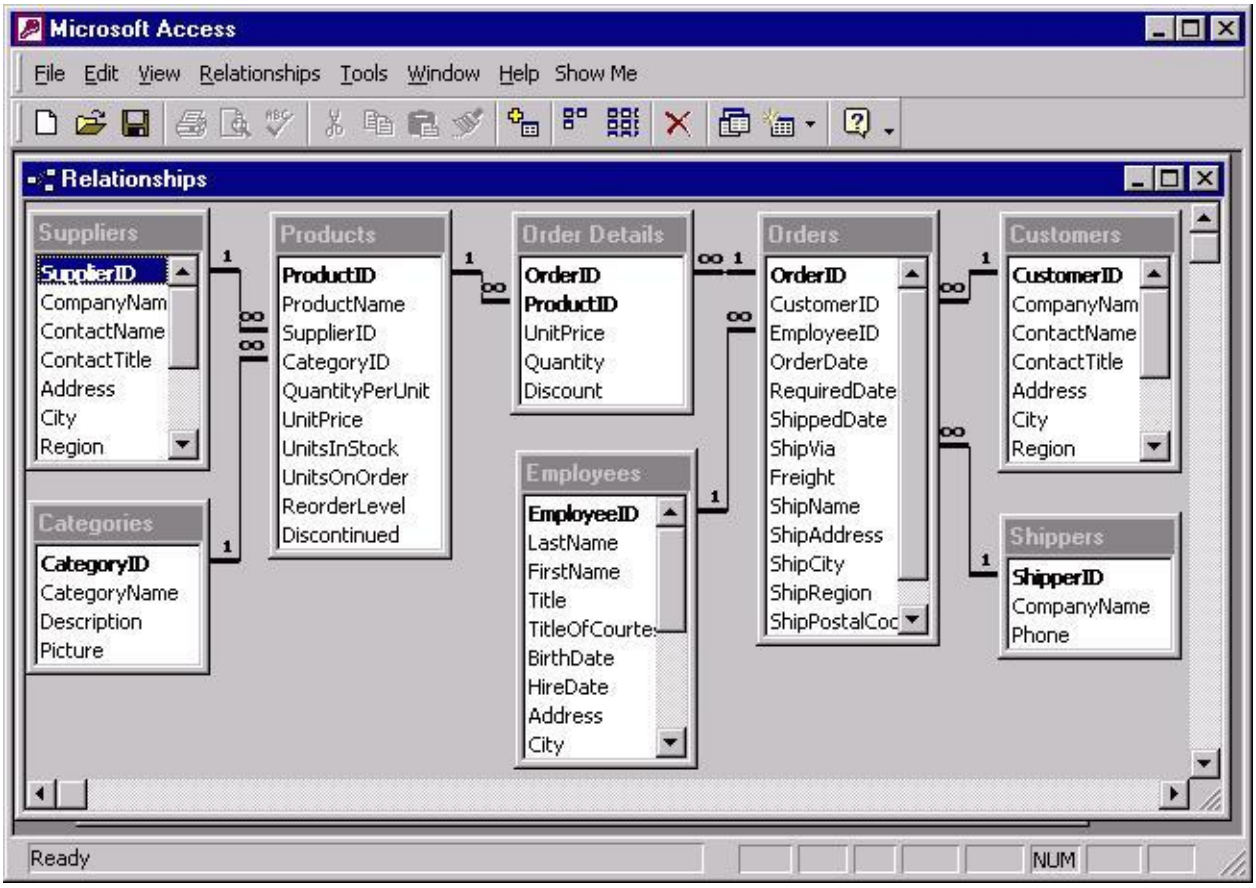
2.4 Database Capture

SPSS can be used to import data stored in databases into the Data Editor. The SPSS *Data Capture Wizard* can import variables that are stored in different tables in a database. To illustrate the use of the Data Capture Wizard, we will leave the *Employee data.sav* dataset used so far and introduce a database that is a sample database included with Microsoft Access: *Northwind.mdb*. It includes variables containing information about a company's inventory, products, suppliers, etc.

Before learning the process for importing a database into SPSS, it is important to understand some database terms and their corresponding terms in SPSS datasets:

- A *relational database* consists of several individual datasets called *tables.* Relational databases are called that because they consist of several related databases.
- Tables are roughly equivalent to individual datasets: they contain several variables, organized in columns.
- Variables are called *fields* in relational databases.

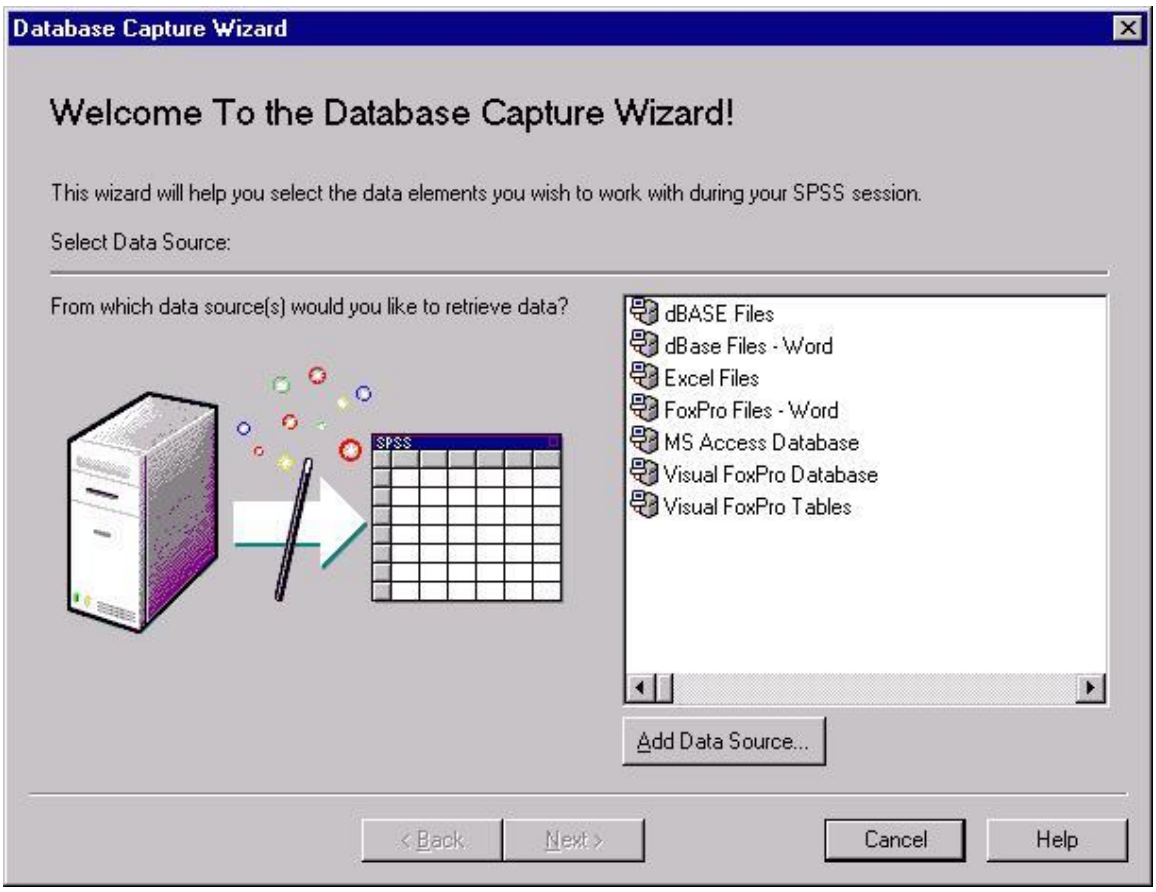
Below is a figure containing a graphical display of the Microsoft Access relational database, *Northwind.mdb*, that illustrates the above terms. Each box represents an individual table in the database. Within each of the tables is a list of the fields contained within each table. The relations in the database below are illustrated by the lines connecting fields having the same name in different tables.



The Database Capture Wizard can be accessed from the SPSS *File* menu:

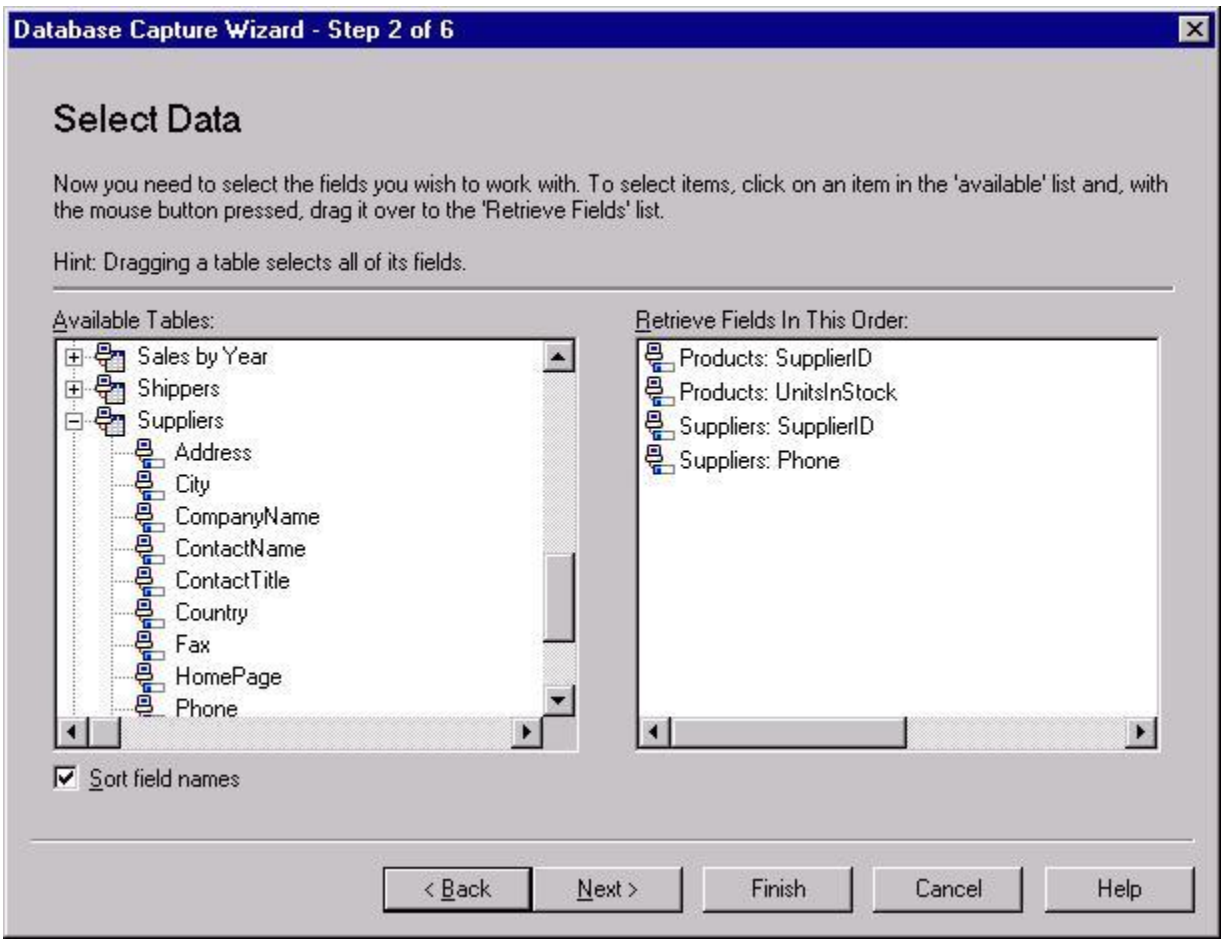
File
Database Capture
New Query...

Selecting this menu item will produce the following dialog box:

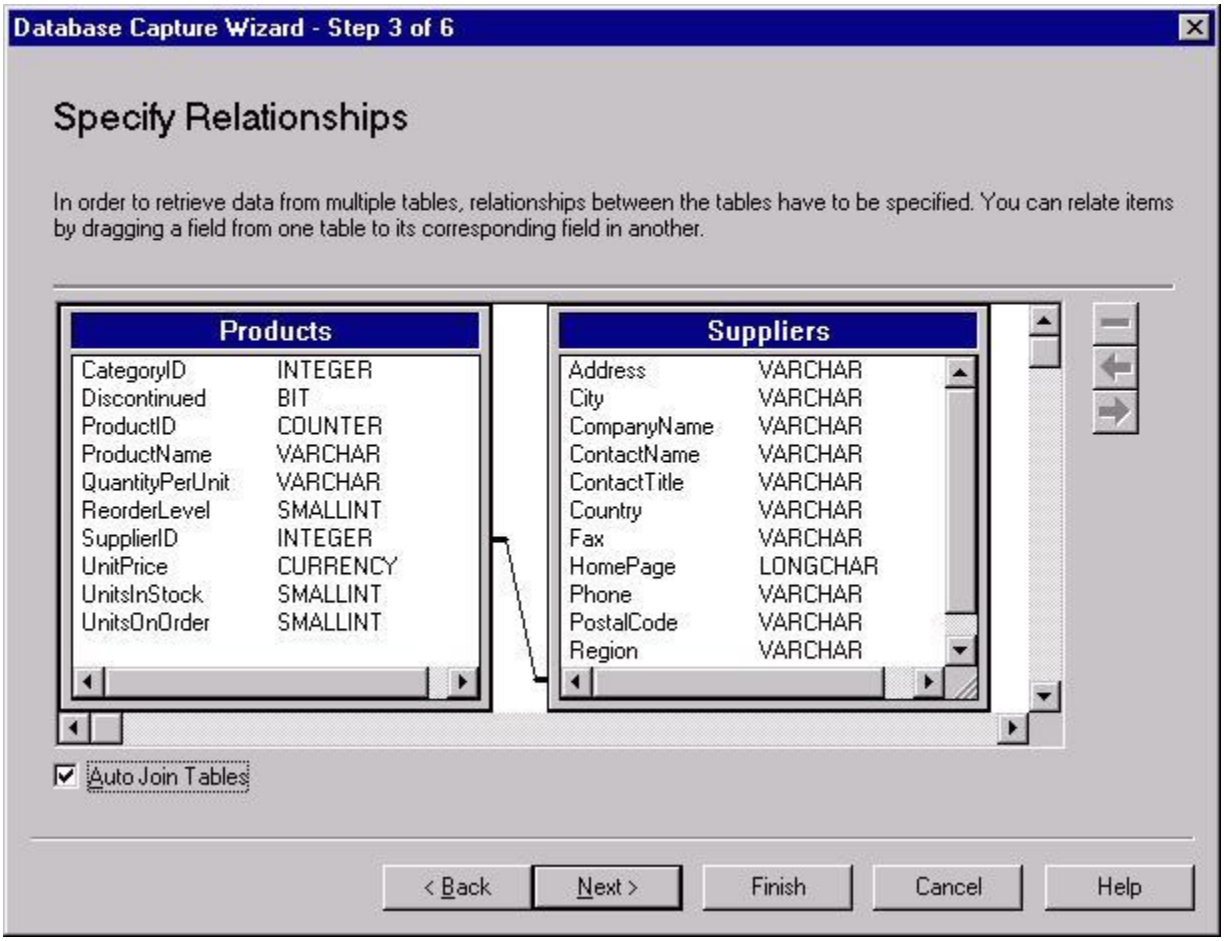


The Wizard consists of six dialog boxes, each of which performs a single function. While there are numerous features of the Database Capture Wizard, only the basics of importing data into a SPSS dataset are discussed here. Some useful features of the Wizard not covered here include: selection of a subset of variables that contain particular values, renaming variables, and saving syntax for the import process.

The first dialog box, shown above, will prompt you to select the type of file that you want to import into SPSS. Here, select the *MS Access database* option. After selecting the type of database and clicking **Next**, you will be provided with a dialog box from which you can select a database located on disk. The second dialog box in the series, shown below, will appear automatically once you have selected the database you want to import.



This dialog box is critical for selecting the information you want to include in the SPSS dataset you are creating. The dialog box will display a list of all of the tables contained within a relational database. You can click the plus sign next to the name of an individual table to expand it so that you have a list of the fields contained within that table. To create a new dataset, you can select either entire tables or you can get a list of individual fields in a table by clicking on the plus sign next to the table name and then selecting fields from within tables. To select items, double-click on them or use your mouse to drag them from the left window to the right window. In the example above, the plus sign next to the *Suppliers* table has been selected. Four fields have been selected, two of which are from the *Products* table and two from the *Suppliers* table. These include the *SupplierID* and *UnitsInStock* fields from the *Products* table and *SupplierID* and *Phone* fields from the *Suppliers* table. The SPSS dataset being created in this example will consist of these four fields. Clicking **Next** advances you to the next dialog box, shown below:



This dialog box illustrates the relationship between tables from which you are importing data. The type of line that connects the tables defines their relationship. There are three possibilities: (1) rows in the tables are included only if a common value appears in both fields (e.g., rows are included only if a particular *SupplierID* exists in both tables), (2) All rows from the table on the left are included and rows on the right table are included only if the value of an identifier variable exists in the left table (e.g., all rows in the *Products* table are included, but only rows from the *Suppliers* table that contain a *SupplierID* value that exists in the *Products* table are included in the SPSS dataset), and (3) All rows from the table on the right are included, and rows on the left table are included only if the value of a identifier variable exists in the left table (e.g., all rows in the *Suppliers* table are included, but only rows from the *Products* table that contain a *SupplierID* value that exists in the *Suppliers* table are included in the SPSS dataset). Double-clicking on the line connecting the tables will produce a dialog box from which you can select one of the above relationships between each table in your database. Clicking **Finish** here or at any other point will complete the process and import the data as you've defined it to the SPSS Data Editor.

The final three steps of the Database Capture Wizard are optional and are beyond the scope of this tutorial. The fourth step allows you to select only records with certain values. The fifth step lets you rename your variables. The sixth step allows you to print the SPSS syntax to the Syntax

Editor, which will replicate the process which you accomplished with the Database Capture Wizard.

Section 3: Advanced Topics

3.1 Syntax

SPSS syntax was first mentioned in the first module of this tutorial series, "Getting Started," in the discussion of the Syntax Editor in Section II. This section will introduce some generalizations about the use of SPSS syntax and resources for creating SPSS syntax, including options for executing syntax, important syntactic characters, and two modes for running syntax in SPSS: batch and interactive. With these core concepts, you should be able to navigate SPSS syntax and generate your own syntax.

The SPSS Syntax Editor section in the "Getting Started" module illustrates the use of the **Paste** button, which is available in most dialog boxes. The **Paste** button can generate syntax which can then be used to send commands to the SPSS processor.

The example from the "Getting Started document illustrates some features of SPSS syntax that generalize to virtually all syntax. The example syntax can be used to generate descriptive statistics:

```
DESCRIPTIVES
  VARIABLES=educ
  /STATISTICS=MEAN STDDEV MIN MAX .
```

First, know the difference between commands and subcommands. The command name is always on the first line. If it is a procedure that generates statistical output, then the names of variables to be included in the analysis will immediately follow. Here, the command name is **DESCRIPTIVES** followed by the subcommand, **VARIABLES**, on the following line. After the **VARIABLES** subcommand, the variable name *educ* appears, which tells the SPSS processor the variable on which to calculate descriptive statistics. This could also be a list of variables if you want to generate statistics on more than one variable at a time. The **VARIABLES** subcommand is a typical subcommand for specifying variables; however, many procedures have special syntax that is used to specify different types of variables, such as fixed factors and covariates.

The next line contains the subcommand, **/STATISTICS**. It shows the slash character that is common to all subcommands other than those which specify variable information. The /**STATISTICS** subcommand also illustrates the use of options in a subcommand. Following the equals sign, there is a list of four statistics, **MEAN STDDEV MIN MAX**, which are four of several possible descriptive statistics that can be specified in this command. Notice the period following the last line of the above code. It indicates the end of a command and is always placed after the last subcommand for a particular procedure.

Another important distinction is that between *batch* and *interactive* modes. In interactive mode, you can maintain interaction with the SPSS processor while commands are being processed. In batch mode, the SPSS processor cannot be interrupted once commands have been submitted. If it

is interrupted by a user or even by an error in the syntax, then it stops processing commands. Interactive mode is more frequently used and is the mode in which commands executed from dialog boxes or syntax windows are processed. Batch mode is the default mode when using the SPSS Production Facility and can be called from the Syntax Editor by using the **INCLUDE** command.

There are a few differences in the way commands are processed in batch and interactive modes, and there are some critical differences in their syntax. Batch mode has some features that are not in interactive mode. Interactive mode will print the line numbers of code in the output and will print the level of control when you are using **DO** loops or other redundant processes. Key differences in syntax are covered here; for details about other differences, see a consultant.

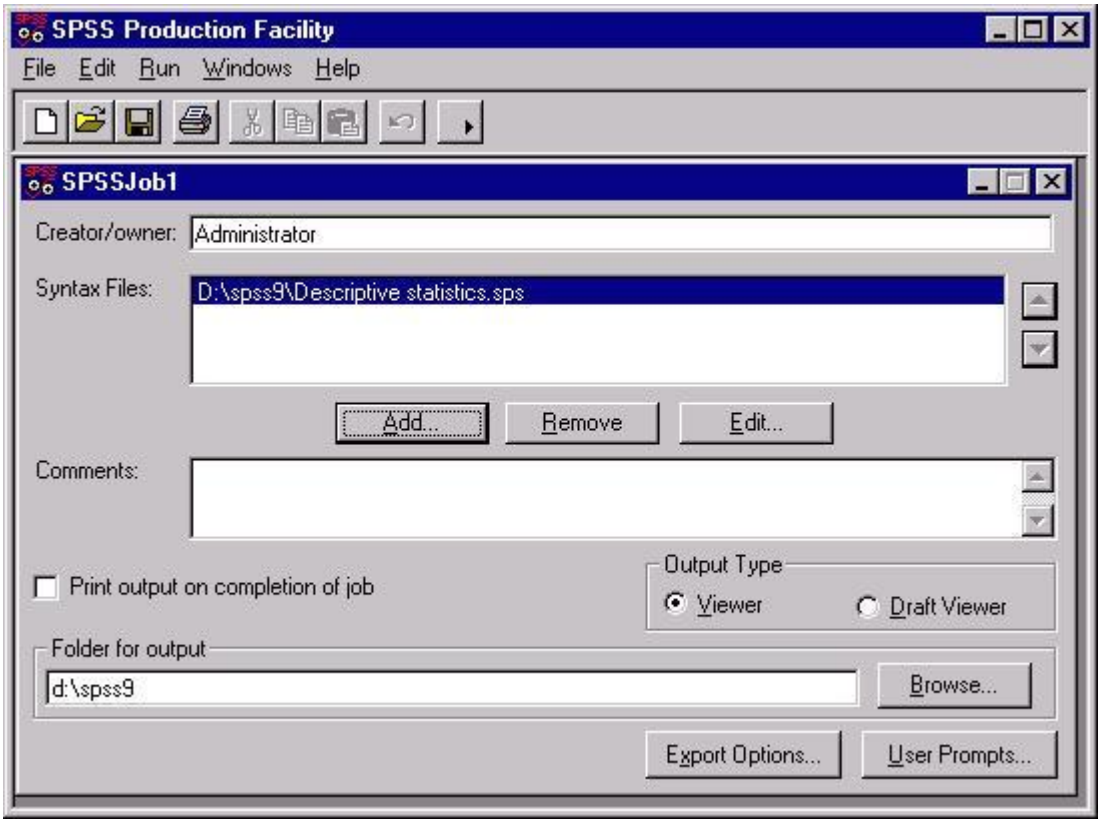
Generally, batch mode has stricter conventions about the format of syntax than does interactive mode. There are three key difference between the modes:

- Commands must start in the first column in batch mode.
- Continuation lines must be indented at least one column in batch mode.
- Periods at the ends of commands are optional in batch mode.

Our above syntax example conforms to the conventions of batch mode, as does all syntax generated using the **Paste** button: the command name, **DESCRIPTIVES**, begins in the first column, and all continuation lines are indented at least one space. The period at the end of the commands is not necessary but will not interrupt processing. In interactive mode, you could put all of the commands on a single line and submit them to get the same results.

3.2 The Production Facility

The *Production Facility* is an application that stands apart from the standard SPSS application. It can be accessed from the Window's *Start* menu. The Production Facility's primary purpose is to process syntax files in batch mode. You can process as many syntax files as you like by using the **Add** button to add syntax files to the list of files in the *Syntax Files* box. Clicking on the **Add** button, will present you with a dialog box from which you can select a syntax file. This process has been done for the file *Descriptive statistics.sps* in the example below. Repeat this process until you have added all of the files that you want to include in the production job.



The production job can be saved to include all of the options and a record of the syntax files included in that job. Selecting *Save* from the *File* menu will save the entire dialog box as it appears above. Doing so will include options such as the *Print output on completion of job* which causes your entire output file to be printed when the job has been completed. Saving a production job is useful for situations where you want to repeat an analysis on a regular basis. If you need to make changes in a syntax file in a saved production job, click on the **Edit** button, which will open the presently highlighted syntax file in a text editor.

3.3 Scripts using Visual Basic

The SPSS scripting facility enables you to add numerous custom features to SPSS. It uses the Visual Basic programming language to operate on most any feature of the SPSS environment. You can use any commands available in Visual Basic in addition to the Visual Basic commands that are unique to SPSS's scripting facility. While you may not be proficient in Visual Basic or interested in learning it, there are a number of scripts available that you may find useful. For example, one sample script will change the text, "Total," in a pivot table so that it is blue and in a bold font. This script contains comments that will indicate how to change the script so that it could find text other than "Total" and change it to colors other than blue.

There a few key concepts with which you should be familiar if you intend to use scripts in SPSS. Visual Basic operates on *objects*, which can refer to virtually any definable entity in the SPSS environment. For example, everything from the SPSS application to a cell in a pivot table are

objects in the SPSS environment. For a useful overview of such objects, select the *Objects* menu item from the *Help* menu:

Help
  Objects

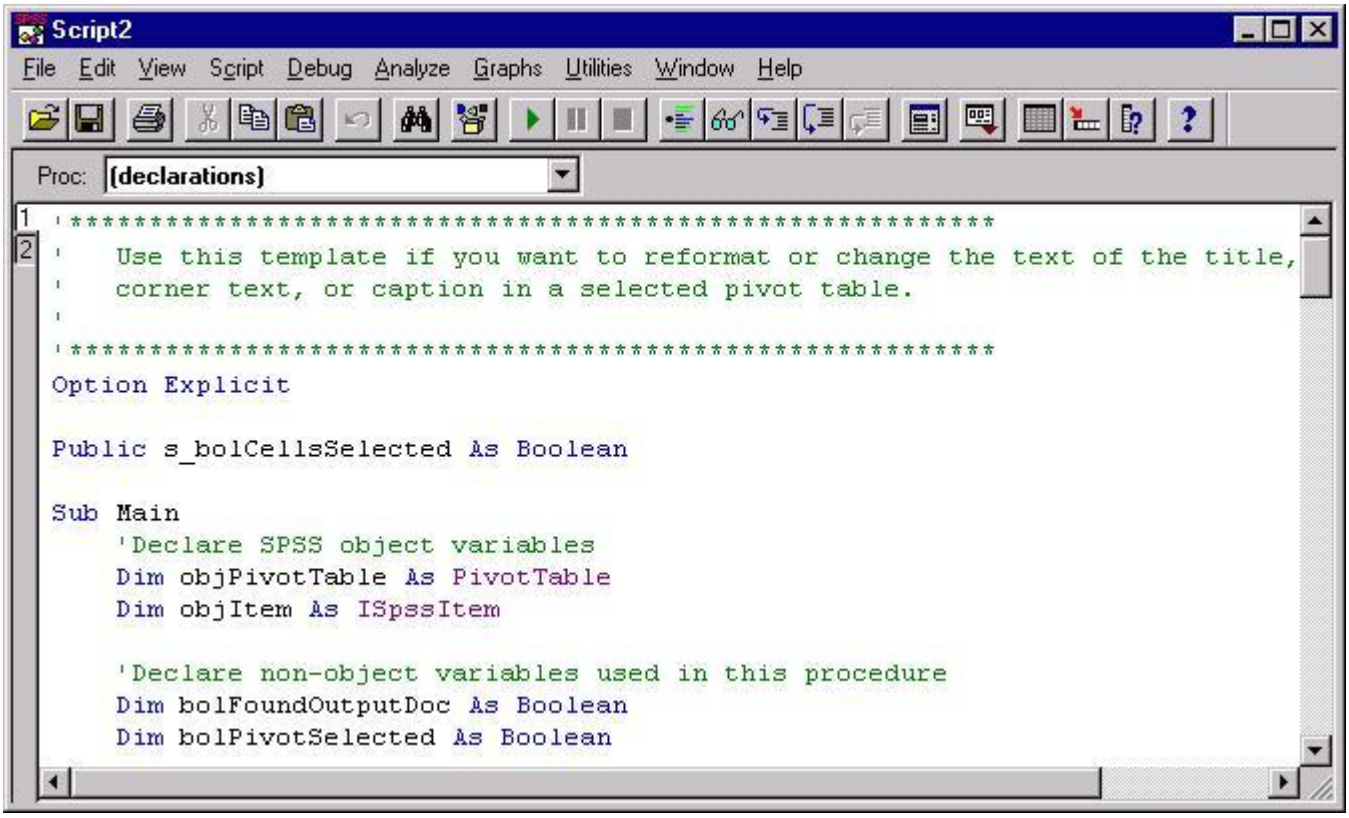
Two other key concepts in the Visual Basic environment are *methods* and *properties.* Methods refer to operations that can be performed on an object. For example, the **ResizeColumn** method will change the width of a column in a pivot table. A property refers to features of an object. For example, cells in a pivot table have properties such as height, width, and color, which can be changed with various methods. One such property is **TextFontAt**, which will return the type of font in a user-defined cell.

One final concept that is critical to using scripts in SPSS is the concept of *active* objects. At any given time, only one object is active in the SPSS environment. For a method or property command to be performed on an object, that object must be active. There are two ways to activate an object. The first is to use the Visual Basic language. This topic is not covered here. The second way is by clicking on that object. Thus, when you move your Output Viewer to the foreground, you have activated that object. Or when you highlight a single cell in a pivot table, that object is activated. This feature can be especially useful if you only want to run a brief program on a specific object.

To run a script, open the SPSS scripting facility from the *File* menu:

File
  New
    Script

This will produce a dialog box containing several script files that are included with SPSS. To use one of these, select it from the dialog box. To create your own script, select the **Cancel** button to move into the SPSS scripting facility. Here is a dialog box containing the script, *Reformat misc pivot.sbs*:



If you are using one of the scripts provided with SPSS, you will often have several choices for easily modifying the programs. Usually, these involve removing a comment from the text. Comments are lines with green text that begin with the ' character. By removing the ' character, the line becomes a line of code rather than a comment. One example of a commented line of code in the above program is a line that allows you to change the title of the active pivot table. If you scroll down in the script shown above, you will eventually see the following commented lines:

'Remove the ' before the next line to change the Title of the selected PivotTable
'objPivotTable.TitleText = "PUT YOUR TEXT HERE"

The first line is a comment that instructs you to remove the comment character from the following line if you want to alter a pivot table's title. The second line will actually alter the title. Thus, to replace the default title for a descriptive statistics table from *Descriptive Statistics* to *New Title*, alter the above code to change the title by removing the ' character and replacing the text, "PUT YOUR TEXT HERE", with "New Title". This altered code would look like this:

'Remove the ' before the next line to change the Title of the selected PivotTable
objPivotTable.TitleText = "New Title"

To run this code, first click the pivot table that you want to alter to make sure that it is active (there will be a box around it when it is active). Next, click on the icon on your toolbar with the green arrow on it to run the program. These steps will alter the title of your pivot table.

One other useful source of SPSS script code are the examples available through the object diagram. Open the object diagram from the *Help* menu as described above. Next, click on the object on which you want to perform an operation. This will produce a text box that contains a description of the object. Click the **Methods** or **Properties** buttons to get a list of the methods and properties that can be performed on the selected object. Then select one of the methods or properties from the list by clicking on it, and click the **Display** button to get a description of the command you have selected. This will provide you with a syntax diagram. To get example syntax, click the **Example** button.

3.4 Macros

SPSS provides the capability to create macros that contain text and SPSS command syntax. Macros can perform the same command repeatedly, combine several procedures, or calculate user-defined statistics. Any functions you can perform using SPSS syntax can be incorporated into a macro. When a macro is called, it is expanded so that any text including commands will be used in its place.

There are two commands used to create a macro: **DEFINE** and **!ENDDEFINE**. To begin the definition of a macro, type the **DEFINE** command in the Syntax Editor followed immediately by the user-defined macro name. Be careful in choosing this name as it will activate the macro whenever it is used. It should be a name that does not appear elsewhere in your syntax. Following the user-defined macro name are parentheses. These required parentheses may or may not contain parameters. Next, you can type syntax or any other text that will be called each time your macro is called. After you have entered all of the text or syntax that you want to include in your macro, type **!ENDDEFINE** to end the definition. The following syntax illustrates the use of a macro that contains a list of variables names. The macro is then used in the **DESCRIPTIVES** procedure.

```
DEFINE varlist ()
     educ jobcat salary salbegin jobtime
!ENDDEFINE.

DESCRIPTIVES
  VARIABLES=varlist .
```

This syntax defines a macro named *varlist*. The text to be called each time *varlist* appears later in the syntax is a list of variable names: *educ jobcat salary salbegin jobtime*. For example, *varlist* appears in the **VARIABLES** subcommand of the **DESCRIPTIVES** command in the above syntax, at which time it is replaced with the text contained in the macro definition and descriptive statistics are calculated for all five of those variables. SPSS macros can also include SPSS syntax, as illustrated by the following macro:

```
DEFINE varlist ()

educ jobcat salary salbegin jobtime
!ENDDEFINE.

DEFINE descrip ()
DESCRIPTIVES
     VARIABLES=varlist
!ENDDEFINE.

DESCRIP.
```

The above macro is identical to the prior example except that the **DESCRIPTIVES** command has been included in a new macro called *descrip*, which is then called after the macro definition. The following processes take place in this case: First, the *varlist* macro is defined so that each time *varlist* appears, it is replaced by a variable list. Second, the *descrip* macro is defined to execute the **DESCRIPTIVES** command for the variables contained in the *varlist* macro. Last, the *descrip* macro is called in the line containing the macro's name, which then executes the **DESCRIPTIVES** command for the variables defined in the *varlist* command.

Thus far, none of our macro examples have contained *arguments*, which are additional terms such as variable names that you may want to call with a macro. The example below illustrates the use of a macro in which one macro is a variable list that is inserted in two statistical procedures.

```
DEFINE varlist ()
     educ jobcat salary salbegin jobtime
!ENDDEFINE.

DEFINE stats (arg = !CHAREND ('/')).
DESCRIPTIVES
     VARIABLES=!arg .
FREQUENCIES
 nbsp;VARIABLES=!arg
  /STATISTICS=MIN MAX.
!ENDDEFINE.

STATS arg = varlist /.
```

Here, the first step again begins with the varlist macro which defines variables. The second step defines a second macro with the name *stats*. It contains a user-defined argument, *arg*, which will be replaced with variable names when the macro is called. The argument is followed by the SPSS keyword, **!CHAREND**, which defines the character that signifies the end of the argument list. This is one of several possible keywords (See the *Syntax Guide* in the *Help* menu for more options). The third step is to execute a series of statistics: **DESCRIPTIVES** and

FREQUENCIES. Notice in the **VARIABLES** subcommand that *!arg* is in the place where a variable list normally is. After the **!ENDDEFINE** command, the *stats* macro is called with the argument varlist followed by the '/.' The argument varlist replaces the arg in the stats macro with the variable list which it defines and the '/' signifies the end of the arguments. Thus, the output for the above example will include descriptive statistics and frequency tables for the variables, *educ*, *jobcat*, *salary*, *salbegin*, and *jobtime.*