SAS - PART I

Technology Support

ShortCourse Handout

Texas Tech University

Table of Contents

SAS 9.4 – Part I
ShortCourse Handout

Introduction

SAS is a computer application that can be used to perform statistical analyses. The system has statistical analysis tools, data presentation capabilities, procedures for data entry, and many other features.

In this ShortCourse we will focus on **Base SAS**, a quick brush-up on the SAS basics. Base SAS is like Microsoft Windows operating system in Microsoft products. It is the core of the SAS system, which includes **SAS language**, **SAS Procedures**, and the **SAS Windowing Environment**. Base SAS provides you with data access, management, statistical analysis, and presentation.

This ShortCourse will introduce you to basic SAS programming skills needed to perform elementary data manipulation and analytical tasks in SAS. In this course, we will cover only a few of many SAS statistical procedures. This ShortCourse assumes that the users are familiar with elementary statistics.

Course Objectives

- After completing this ShortCourse, you should be able to:
- Create a SAS dataset;
- Create New SAS Libraries;
- Perform some basic data and file management tasks;
- Write, Submit and store a SAS program; and
- Import an Excel file into a SAS dataset.

Starting SAS

- Click on **Start** button > **All Programs** > **SAS** > **SAS 9.4 (English)**

Accessing SAS Window Help

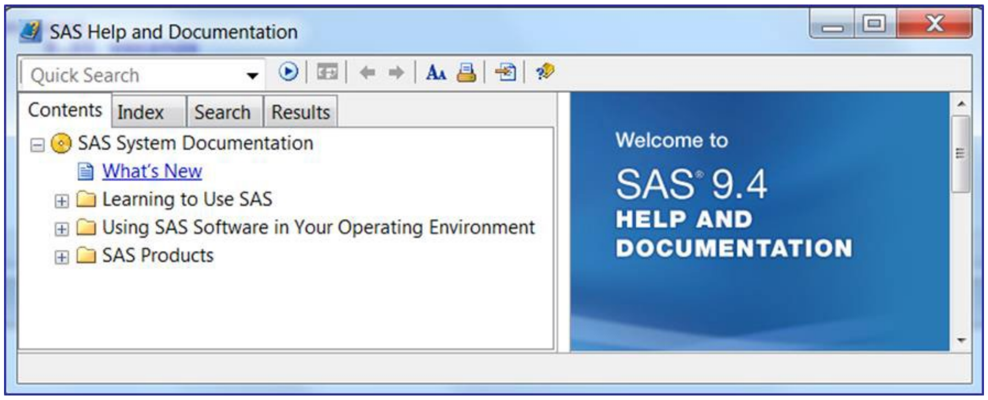
You can access SAS Help on an individual window in any of the following ways:

- Issue the **HELP** command from the **command line** of the Window.
- Select the **Help** icon on the toolbar.
- From the window for which you want help, select **Help Using This Window** (**F1**).

SAS/STAT Help

- On SAS Help and Documentation, on the **Contents** tab:
  - Expand **SAS products**
  - Expand the **SAS/STAT**
  - Expand **SAS/STAT 14.1 User's Guide**

Note: **SAS/STAT** includes more than **80 chapters**, and most chapters include several examples.



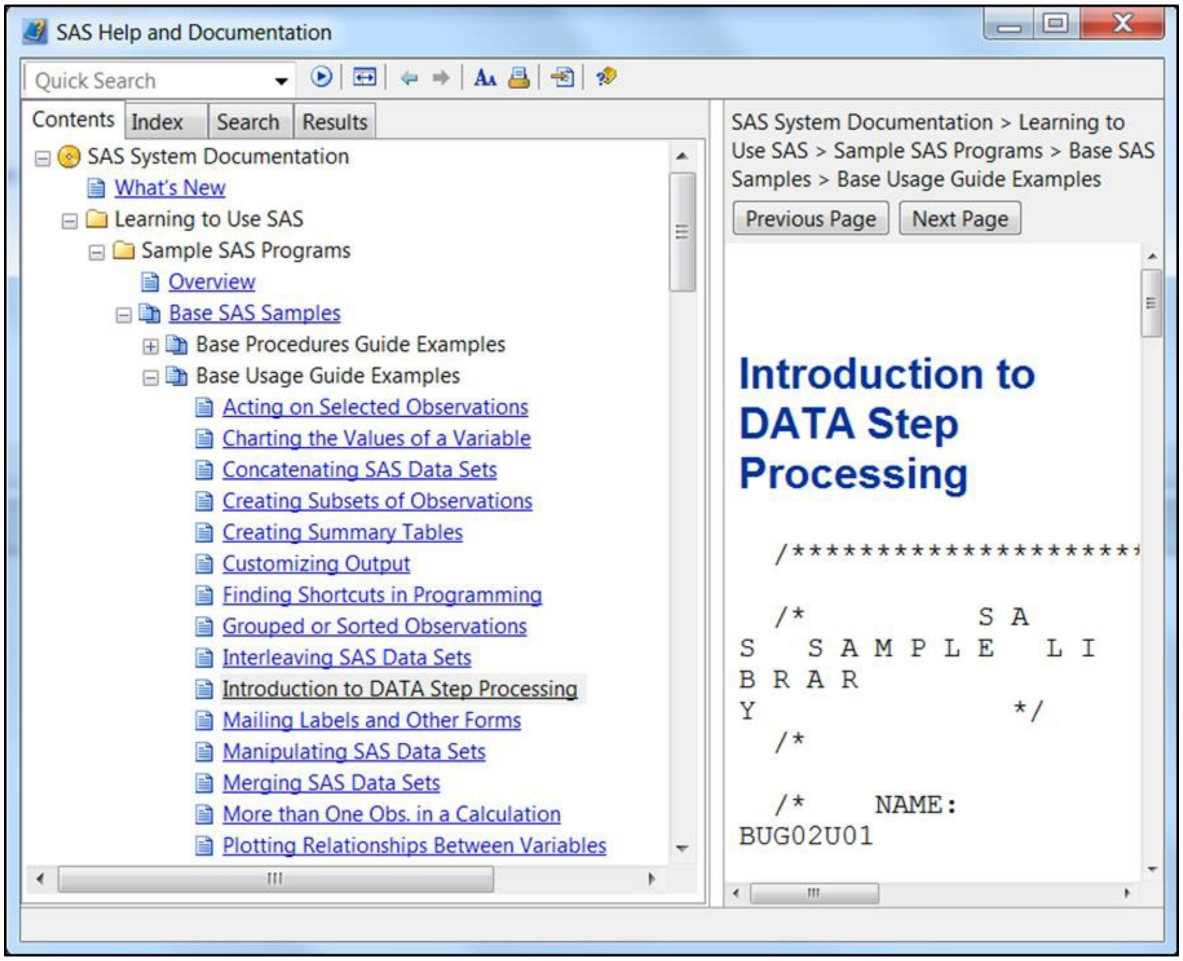
Exploring SAS Help

- In SAS, click **Help** > click **SAS Help and Documentation**
- Click the **Contents** tab, expand the **SAS Products folder**
- Expand the **Base SAS folder**
- Select **Base SAS 9.4 Procedures Guide**

Accessing the SAS/STAT Sample Library

- Expand the **Learning to Use SAS** folder.
- Expand the **Sample SAS Programs**.
- Expand the **Base SAS Samples**
- Expand the **Base Usage Guide Examples**.

- Click on **Introduction to DATA Step processing**.
- **Copy** the program from the **Options** line through the last **Run** line.
- **Paste** the program in the **Editor** window.
- **Type a title** statement at the end of the program to clear any title in effect.
- **Submit** this program.
- **Close** the **SAS Help and Documentation** window.



SAS System Documentation
What's New
Learning to Use SAS
Sample SAS Programs
Overview
Base SAS Samples
Base Procedures Guide Examples
Base Usage Guide Examples
Acting on Selected Observations
Charting the Values of a Variable
Concatenating SAS Data Sets
Creating Subsets of Observations
Creating Summary Tables
Customizing Output
Finding Shortcuts in Programming
Grouped or Sorted Observations
Interleaving SAS Data Sets
Introduction to DATA Step Processing
Mailing Labels and Other Forms
Manipulating SAS Data Sets
Merging SAS Data Sets
More than One Obs. in a Calculation
Plotting Relationships Between Variables

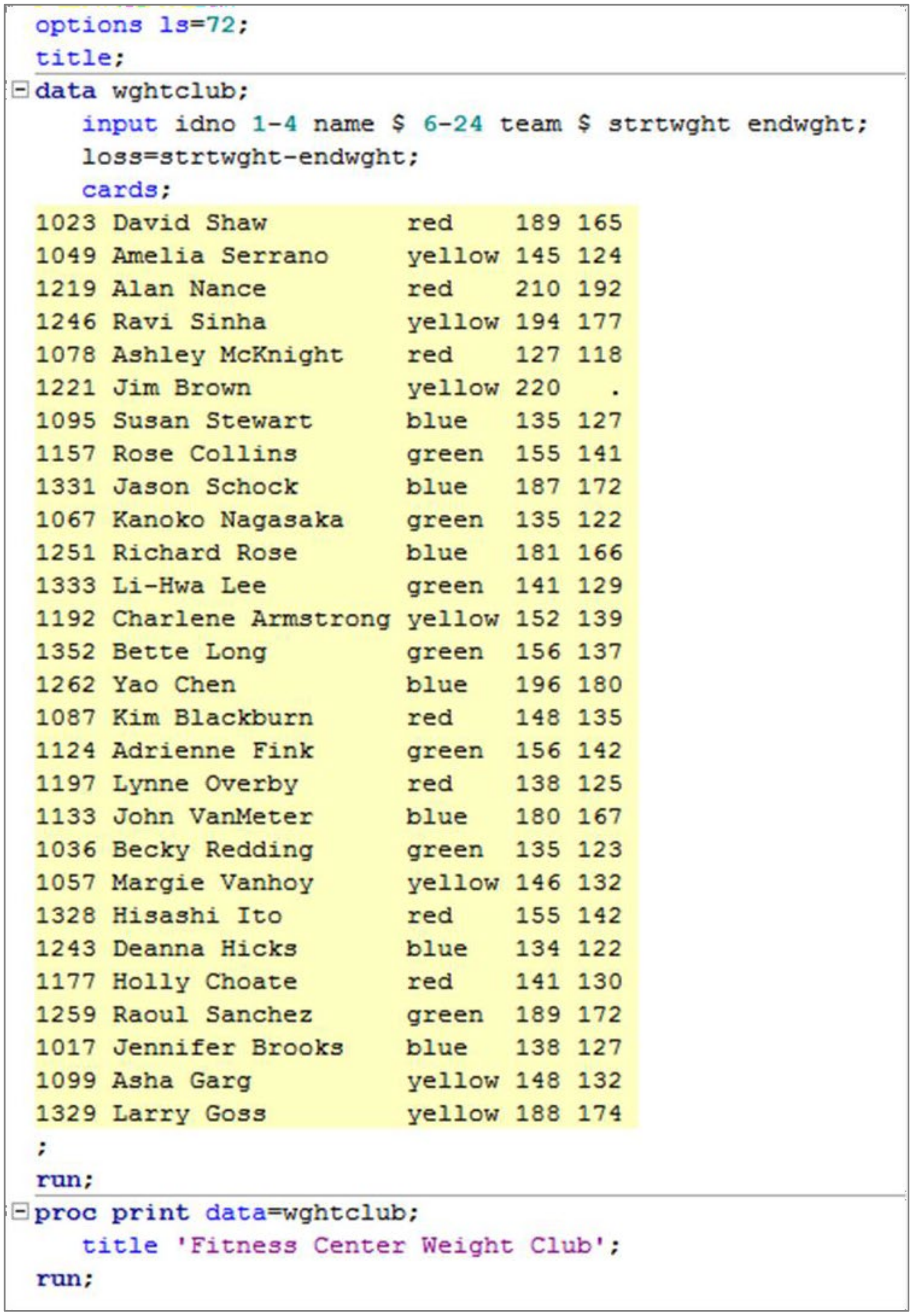
SAS System Documentation > Learning to Use SAS > Sample SAS Programs > Base SAS Samples > Base Usage Guide Examples

Introduction to DATA Step Processing

```
  /********************
  /*          S A
S   S A M P L E   L I
B R A R
Y                 */
  /*

  /*    NAME:
BUG02U01
```

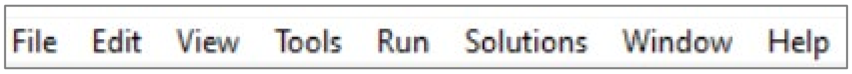
Exploring a Sample SAS Program: Introduction to DATA Step Processing



```
options ls=72;
title;
data wghtclub;
   input idno 1-4 name $ 6-24 team $ strtwght endwght;
   loss=strtwght-endwght;
   cards;
1023 David Shaw         red    189 165
1049 Amelia Serrano     yellow 145 124
1219 Alan Nance         red    210 192
1246 Ravi Sinha         yellow 194 177
1078 Ashley McKnight    red    127 118
1221 Jim Brown          yellow 220   .
1095 Susan Stewart      blue   135 127
1157 Rose Collins       green  155 141
1331 Jason Schock       blue   187 172
1067 Kanoko Nagasaka    green  135 122
1251 Richard Rose       blue   181 166
1333 Li-Hwa Lee         green  141 129
1192 Charlene Armstrong yellow 152 139
1352 Bette Long         green  156 137
1262 Yao Chen           blue   196 180
1087 Kim Blackburn      red    148 135
1124 Adrienne Fink      green  156 142
1197 Lynne Overby       red    138 125
1133 John VanMeter      blue   180 167
1036 Becky Redding      green  135 123
1057 Margie Vanhoy      yellow 146 132
1328 Hisashi Ito        red    155 142
1243 Deanna Hicks       blue   134 122
1177 Holly Choate       red    141 130
1259 Raoul Sanchez      green  189 172
1017 Jennifer Brooks    blue   138 127
1099 Asha Garg          yellow 148 132
1329 Larry Goss         yellow 188 174
;
run;
proc print data=wghtclub;
   title 'Fitness Center Weight Club';
run;
```

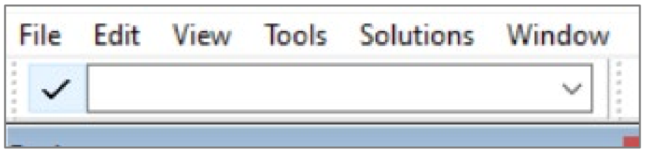
Menu Bar

- Each window has its own menu selections that reflect the actions you can perform using the window.
- The pop-up menus that appear when you right click inside an application window are customized for that window as well.
- Click the **Editor** window. Then click on the Log window. ***Notice*** that they offer different selections in the menu bar.



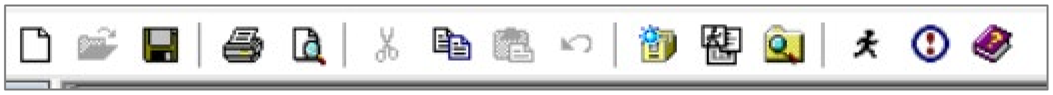
Command Bar

- Provides a way to quickly enter any SAS command.
- The command bar retains a list of the commands that you enter. To switch the keyboard focus to the **command bar, press F11**.
- Most of the commands that you can type in the command bar are also accessible through the pull-down menus or the toolbar.



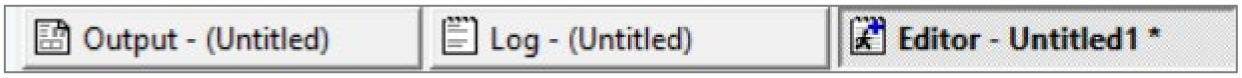
Toolbar

- Provides quick access to the commands that you perform most often.
- Some tools are specific to the active window.
- Click the Explorer window and view the icons that are available on the toolbar.
- Then click on the Log window and notice the tool differences.



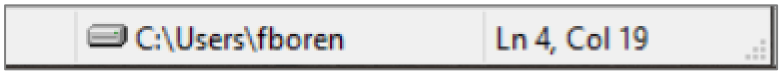
Window Bar

- Located at the bottom of the main SAS window and provides easy access to any window within the main SAS window.
- When a window opens, a button that represents that window is placed in the window bar.



Status Bar

- The area right below the Window Bar that contains a message (status) area, the current folder, and the Enhanced Editor insertion point position (where your cursor is currently active in the Editor Window).

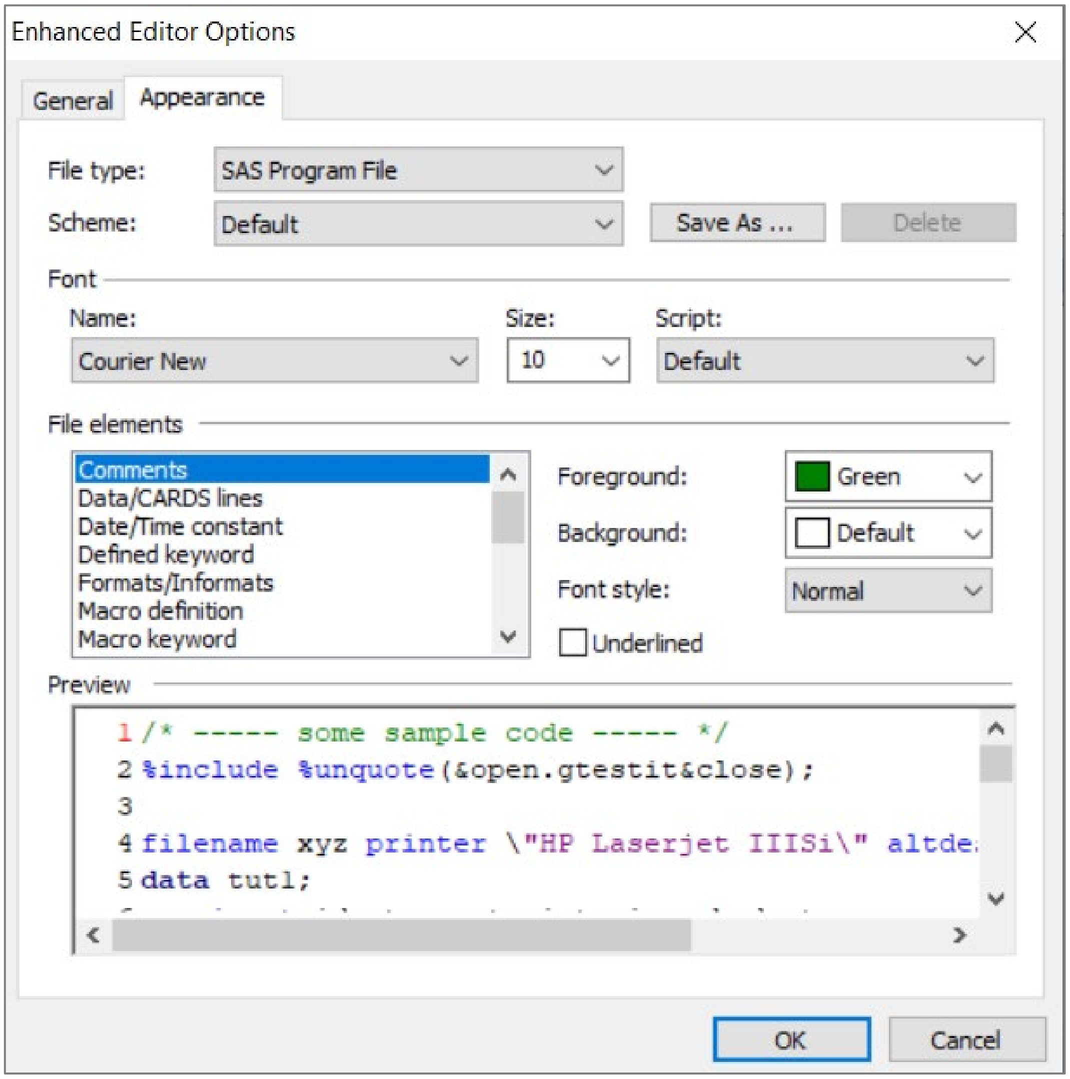


Enhanced Editor Window

- **Editor** or **Program Editor** is used to **create**, **edit**, and **execute** SAS programs.
- The **Editor window** provides several useful editing features, including:
  - Color coding and syntax checking of SAS language;
  - Support for keyboard shortcuts; and
  - Multilevel undo and redo.
- The initial Editor Window title is **Editor - Untitled*n***.
- When you open a file or save the contents of the Editor window to a file, the window title changes to reflect that file name.
- When the contents of the Editor window are modified, an asterisk **(*)** is added to the title, indicating that any changes to the file have not been saved.
- You can have multiple Editor windows open at the same time.
- You can **type**, **copy/paste**, or **drag and drop** the program lines into the Editor.

Changing the Appearance of the Enhanced Editor Window

- **Tools** -> **Options** -> **Enhanced Editor** …
- On the **General** Tab, check mark the "**Show line numbers**" box.
  - Or – type "**Nums**" in the **command bar** and press the **Enter** key.
- On the **Appearance** tab, choose a "File type:" and "Scheme:"
- Click **OK.**



```
/* ----- some sample code ----- */
%include %unquote(&open.gtestit&close);

filename xyz printer \"HP Laserjet IIISi\" altde
data tutl;
```

Log window

- The Log window displays messages about your **SAS session** and any SAS programs that you submit. Use this window to confirm that the code was correct.
- It displays separate messages for the **DATA** and **PROC steps.**
- **Error messages** appear in **red** and indicate that some portion of the program failed to work properly.
- Warnings appear in **green**.

Output window

- Output enables you to browse output from SAS programs that you submit. SAS Output is created by PROCs (or Procedures).
- The Output window is positioned behind the Log and Editor windows, **until there is output to display.**
- Use the **Taskbar** to navigate between windows.
- The Output window displays the output from SAS programs.

Results Window

- The Results window helps you navigate and manage output from SAS programs that you submit, much like a "table of contents."
- The Results pane displays a list of files that were created after the SAS program was executed.

Results Viewer

- The **Results Viewer** helps you view, save, print, or export your **HTML output** (which is the default format for results in SAS).
- The **Results Viewer** is empty **until** you **submit** a SAS program that creates HTML output. Then it opens or moves to the front of your display.
- Click on the **Results Viewer** button in the **Window Bar** to display your HTML results.
- Make sure you are clicked within the **Results Viewer** pane and
  - Click **File**, and then select **Save As** from the menu. This will allow you to save everything from your Results Viewer window.
  - To export an item from your results to Excel, **right-click** the item and select **Export to Microsoft Excel**.
  - To send an item from your results to OneNote, **right-click** the item and select **Send to OneNote**.
  - To print your results, **right-click** in the Results Viewer and select **Print**.

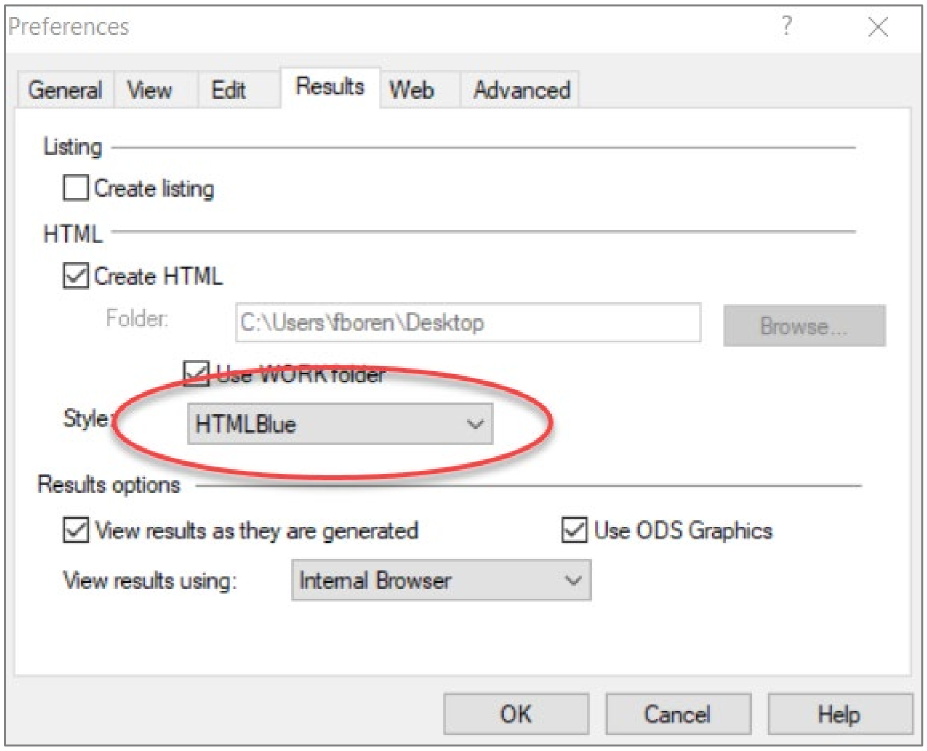
Note: It is a good practice to clear the Log and Output windows between program submissions. **Right-click** in the window > **Edit** > **Clear ALL** (Ctrl + E).

HTML Output in the SAS 9.4 Windowing Environment

- In SAS 9.4, the default destination in the SAS windowing environment is

HTML, and ODS Graphics is enabled by default.

- Graphs are integrated with tables, and all output is displayed in the same HTML file using the style, **HTMLBlue**.
- You can view and modify the default settings by selecting:
  - **Tools** -> **Options** -> **Preferences**
  - Then on the **Results** tab, make your selections.
  - You can also change the results format from **HTML** to **Listing** using this menu.



Listing

☐ Create listing

HTML

☑ Create HTML

Folder: C:\Users\fboren\Desktop

Browse...

☑ Use WORK folder

Style: HTMLBlue

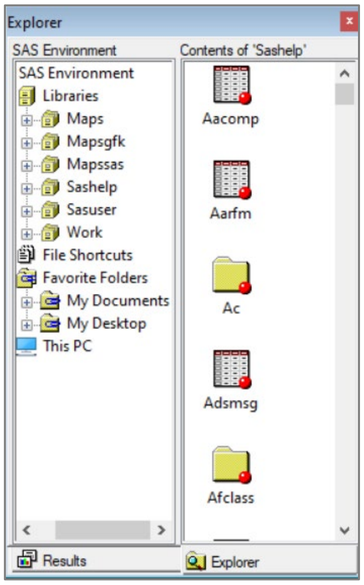
Results options

☑ View results as they are generated

☑ Use ODS Graphics

View results using: Internal Browser

Explorer window



The Explorer window provides a central access point to SAS **tables (datasets), libraries**, and other files.

- To open an Explorer window using:
  - **Menus:** Click View > Explorer
  - **Commands:** Type Explorer
  - **Toolbar:** Click the Explorer's icon
- You can use this window to:
  - **Create new** SAS libraries and SAS files
  - **Open** and **Edit** existing SAS files
- The Explorer window has two sides, the tree view

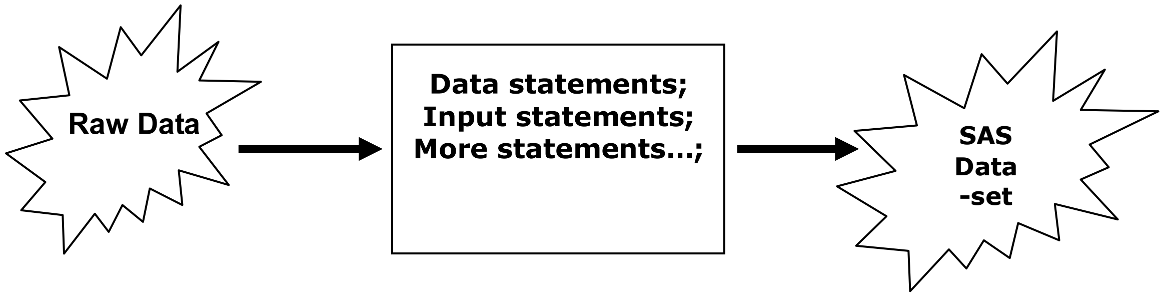
on the left side and the contents view on the right side.

- You can use the toggle tree button in the Toolbar to toggle the tree view on and off.

Components of a SAS Program

SAS programs consist of two basic building blocks: **DATA steps** and **PROC steps**.

- **The DATA step**
  - Starts with a keyword **DATA**
  - Reads and modifies data
  - Assigns a name to the dataset
  - Assigns names to variables
  - Creates a new variable from an existing variable



- **The PROC step (procedures)**
  - Starts with a keyword **PROC**
  - Carries out statistical analysis
  - Displays information about a SAS dataset
  - Must appear after a **DATA** step (you must have data before you can run an analysis on it).

SAS Program Structure

- A SAS program is a sequence of statements executed in order that gives instructions to the SAS system.
- A SAS program has a series of **Data steps** and **Procedure steps** (called **PROCs**).
- A **Data step** begins with a key word, **DATA**, followed by a collection of statements that defines the **source of data**, the **variables to be read**, etc.
- A **Procedure step** begins with a key word, **PROC**, followed by the **procedure name,** statements, and proc options.

- Each SAS statement ends with a **semicolon (;)**, a reserved character in SAS.
- Almost all SAS statements begin with a SAS keyword such as **DATA**, **SET**, **PROC**, **INPUT**, **Title**, **IF**, **Options**, etc.
- A **RUN** statement is inserted after DATA steps or PROC steps.
- It is helpful to indent the statements that go together.
- **Quotes** can be single or double, but they must match.
- **Data** definitions and **Options** are at the top of most SAS programs.
- **Title** – puts a title line on the output. For additional lines of title, use title <*n*> (n up to 10).

  TITLE <n> <'text' | “text”>;

  n specifies the relative line that contains the title line.
  'text' | “text” specifies text that is enclosed in single or double quotation marks.

- **Footnote** – Writes up to 10 lines of text at the bottom of the output.

  FOOTNOTE<1...10> <text-argument(s)>;

SAS Syntax Rules

- Words in statements must be separated by **at least one blank**.
- SAS statements can be in upper, lower, or mixed cases, *except inside quotation marks* (gender = 'm'; is not the same as gender = 'M';).
- SAS statements can continue on the next line.
- SAS statements can start in any column.

Using Comments in a SAS Program

- **Comments** are lines of text that are ignored by the program. Comments either:
  - Start with an **asterisk (*)** and end with a **semicolon (;)**
  - Or- starts with a **slash asterisk (/*)** and ends with **asterisks slash (*/)**
- In the enhanced editor, you can highlight a statement or a block of statements and press the **CTRL and forward slash** (**/**) to place comments

around that statement in the block of statements.

- Press the **CTRL** and **SHIFT** and **/** to remove line comments from any lines that are highlighted. This method does not work on non-Windows systems or in the (non-enhanced) Program Editor on Windows.

Note: You can use comment statements anywhere in a SAS program to **document the purpose of the program**, or document information about the author or the date program was created, etc. However, **avoid** placing (**/***) comment symbols in columns 1 and 2. In some operating environments, SAS may interpret it as a request to end the SAS session.

SAS Variables

In SAS there are two types of variables:

1. **Numeric Variables**
   - Can contain only numeric values (the digits 0 through 9, +, -, ., and E for scientific notation).
   - Use a single **period (.)** to represent missing values.
2. **Character Variables**
   - They may contain numerals, letters, and can be up to 32,767 characters long.
   - Use a **blank space** to represent missing values.

Rules for SAS Variable and Dataset Names

- Variable or File Names can be up to **32 characters** long.
- Names must begin with a letter or an **underscore (_).**
- Names **cannot** include special characters such **as %, $, !, *, &, #,** and **@** .
- Names may contain any combination of numbers, letters, or underscores.
- Names can contain **upper** and **lowercase** letters.
- However, you cannot use the same letters with different combinations of lower case and upper case to represent different variables. For example, **cat**, **Cat**, and **CAT** will all be read as the same variable.

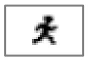
Analyzing the Wghtclub Program Example

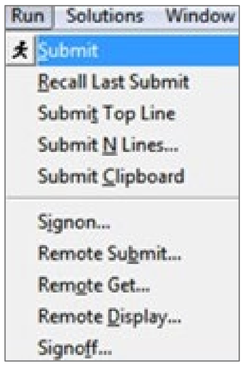
- **Options ls=n (or LINESIZE=n)** specifies the line size (printer line width) for the SAS log and the SAS output. Valid values range between **64** and **256**.

However, we don't want the line size to be larger than **80**.

- The **DATA** statement creates the SAS dataset and names the table (output) you want to create.
- The **INPUT** statement creates variable names (column names in the SAS dataset). A dollar sign ($) should follow the variable name if the variable is a character variable (not numeric).
- **Loss = StartWeight – EndWeight** is an **assignment statement** and calculates the value for the variable **Loss.**
- The **DATALINES** statement tells SAS that you are finished with the instructions on how to create the dataset. **CARDS** is used instead in old versions of SAS.

Note: the **datalines** statement is used *only if* the data is embedded within the SAS program.

- The single semicolon (**;**) marks the end of the input data and the DATA step.
- The **RUN** statement tells SAS to run (submit) the program. The **RUN** statement is included as the last line of the **DATA Step**.
- The **PROC** (short for procedure) statement marks the beginning of the **PROC Step**. The PRINT procedure, **PROC PRINT,** prints out the contents of the dataset to the Results Viewer window. *Don't* do this with a very big dataset.
- The **RUN** statement is included as the last line of the **PROC Step**.
- To execute the program, type "**submit**" on the **SAS command line** and press the **Enter** key. You can also select the **Run** menu and choose **Select** or click on the "running man" button in the **Toolbar**. 



Common SAS Programing Options

- **PAGESIZE (or PS) =n** - Specifies the number of lines that can be printed per page of SAS output (15 to 32767). The default is 50. ***Note:*** we don't want this to be larger than 200.
- **OBS** - limits the number of observations processed. Use **NOOBS** (in Proc Print), if you want to suppress the observation number ("Obs" column).
- **NOCENTER** – writes all the output in the log and listing files (left justified

output).

- **DATE** (default) - Prints the date and time to the log and output window. Use NODATE if you don't want the time and date to be printed on your log and output files. ***Note***: this option does not affect output in the Results Viewer.

Naming SAS Datasets

- All SAS datasets have a two-level name such as **work.bikesales**.
- The first level of a SAS dataset name (work) is called is its **libref** (short for library reference).
- The second level (bikesales) is the SAS "member" name (the name you assign the dataset).
- Libref **cannot** be more than 8 characters. However, the member name can be up to 32 characters.
- Names of **temporary** SAS datasets can have only one level and are stored in the **Work** data library.
- For example:

```
proc print data=work.bikesales; /*this is a two-level data name*/
run;
```

Or-

```
proc print data=bikesales;
   /*This is a one-level data name and can be used only when the SAS
   dataset bikesales is located in the work library*/
run;
```

Editing SAS Programs

- **To insert a new line**, place the cursor at the end of the line and then press the Enter key.
- **To move a line**, create a blank line first, right-click, and use the copy/paste feature.
- **To undo commands**, click undo from the Edit menu.

Submitting Code from the Enhanced Editor

- You can submit either a complete program or a specified number of lines of your program by highlighting part of the program.
- To submit SAS code that you have typed into the Enhanced Editor or Program

Editor window, you issue the SUBMIT command. SAS provides several ways to do this:

- Press **F8** when the editor window is active.
- Click the **Submit** button.
- Type **submit** in the command bar and press the **Enter** key.
- From the **Run** menu, select **Submit**.

- To **interrupt** your SAS Session
  - Click the **Break** exclamation point in the Toolbar.

Note: after submitting a program, activate your **Log** and **Results Viewer** window, and from the **Edit** menu, choose **Clear All (Ctrl + E)** or right-click in the window and select **Clear All**.

SAS Errors, Warnings, and Notes

- The most important rule in debugging SAS programs is to check the **SAS log**, after submitting a SAS program.
- SAS logs contain three types of messages: **Errors**, **Warnings**, and **Notes**.
- A **missing semicolon** is by far the most common error.
- It will cause SAS to misinterpret not only the statement where the semicolon is missing, but possibly several statements that follow.
- SAS will **underline** the error where it detects it, but sometimes the actual error is in a different place in your program, typically the preceding line.
- **Always** look at the statements immediately above the line with the error.
- **Sometimes** SAS will correct your spelling mistakes for you by making its best guess at what you meant to do. But do not rely on this feature.
- **Never** assume that a program that has run without errors is correct.
- **Always** review the SAS log for notes and warnings as well as errors.
- Check all SAS **Warnings** to see if they are harmless.
- SAS **Notes** inform you of the status of your program, and they should be checked too.

Saving SAS Files

- To save your SAS file:
  - Select **File** > **Save As**
- Select a folder in which to save your file

- Type a file name in the **File name:** field
- Select a file type from the **Save as type:** field
    - **.sas** for SAS programs
    - **.lst** for output files
    - **.log** for log files
    - **.sas7bdat** for data files (in table format)
- Click on **OK.**

Using SAS Data Libraries

A SAS data library is a physical collection of files and folders.

In the **Explorer** window, double-click **Libraries**, and then use Expand (+) and collapse (-) icons to see the files.

These libraries are automatically assigned each time you start SAS:

Sashelp Sasuser Work

- **Sashelp** - a permanent library that contains more than 200 sample data and other files that control how SAS works at your site.
- **Sasuser** - a permanent library that stores your personal settings.
- **Work** - a temporary library for files that do not need to be saved from session to session.
    - To access a library, you assign it a name (also known as a **libref**, or library reference).

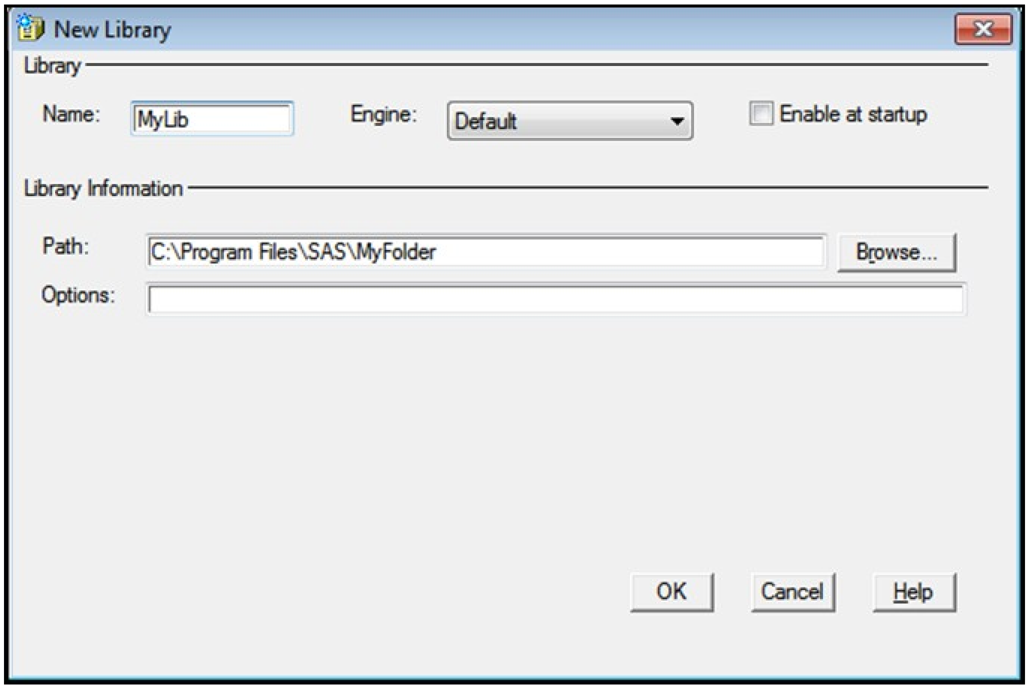
Note: The **Sashelp** and **Sasuser** libraries are ***permanent*** SAS libraries. But **Work** is a **temporary** SAS library. Files in the Work library *are not saved* once you end your SAS session.

Assigning a Library

When you define a library, you indicate the location (the path) of your SAS files to SAS.

- On the toolbar, click the **New Library** icon. The New Library window opens.
- In the **Name** box, type **MyLib** (for example).
- **Library names** are:
    - Limited to **8 characters**
    - Must start with a **letter or underscore (_)**

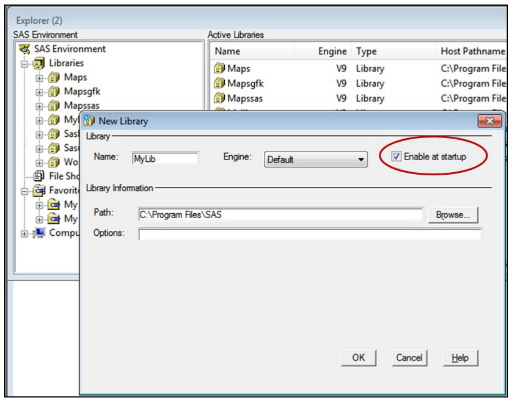
- Can contain **only letters, numbers, or underscores**
- Check the **Enable at Startup** check box, if you want the library to be created at the beginning of each SAS session.
- Click **Browse** to select a location in the **Path:** box. This will create a physical area for your files. For example, C:\Users\fboren\Desktop
- Click **OK.**



Name: MyLib
Engine: Default
Enable at startup
Path: C:\Program Files\SAS\MyFolder
Options:

Using the Point-and-Click Method to Add Libraries

- Click **View > Explorer** (or just click within the Explorer window).
- Click **File > New**.
- In the New Library window, specify information for the new library. If you want the library to be created at the beginning of each SAS session, check the **Enable at startup** box.
- Click **OK**.



Name: MyLib
Engine: Default
☑ Enable at startup
Path: C:\Program Files\SAS
Options:

Working with SAS Permanent Libraries

- On the **Toolbar**, click the **New Library** icon.
- In the **Name** box, type **MyLib**.
- Click **Browse** and select a location in the **Path:** box to create a

physical location for the library. For example, C:\Users\fboren\Desktop.

- Click **OK.**
- Type the following program in the **Editor** window.

```
DATA Mylib.Grade; /* Mylib is the library reference, or libref,
   that is created before submitting this program, and Grade is
   the name of the SAS dataset we are creating */
   Score=95;
Run;
PROC Print DATA=Mylib.Grade;
Run;
```

- **Submit** this program (select this program and then click on the Running man).
- Open the Mylib library in the Explorer and view your SAS dataset.

Examples of Some SAS Procedure Syntax

Univariate:

```
PROC UNIVARIATE DATA=mydata NORMAL PLOT;
   VAR v1 v2;
RUN;
```

Correlations:

```
PROC CORR DATA=mydata;
   VAR v1 v2;
RUN;
```

ANOVA- balanced design:

```
PROC ANOVA Data=mydata;
   CLASS x1 x2;
   MODEL y1= x1 x2 x1*x2;
RUN;
```

One-Sample T-test:

```
PROC ttest data=mydata;
   VAR v1;
RUN;
```

Paired T-test:

```
PROC ttest data=mydata;
   paired v1*v2;
RUN;
```

Two-Sample independent T-test:

```
PROC ttest data=mydata;
   Class v1;
   VAR v2;
RUN;
```

Using the @@ Symbol

The **double trailing @ Symbol** at the end of an ***input statement*** in the data step tells SAS not to move to the next data line after reading the values for the variables in the current input statement. For example, assume that a dataset contains three variables, X, Y, and Z. A DATA step could look like:

```
Data TEST;
   Input x y z;
Datalines;
1  1     12.4
1  2     11.3
1  3     1.4
2  1     2.1
2  2     19.4
2  3     10.0
;
Run;
```

Alternatively, you can read multiple observations in each line of data by adding the **@@ symbols** at the end of the input statement:

```
Data TEST;
   Input x y z @@;
Datalines;
1  1     12.4  1     2     11.3  1     3     1.4
2  1     2.1   2     2     19.4  1     3     10.0
;
Run;
```

In this program, SAS will go through each line of data and fill the variables X, Y, and Z in turn. Without the **@@** symbols, it would switch to the next line as soon as the last variable was filled. With the **@@** symbols, SAS will continue filling the variables until it reaches the end of the line.

SAS Datasets

- Before you can work with your data in SAS, it must be in a special form called a **SAS dataset** (also called a table).
- Each dataset consists of observations and variables.
- Observations are sets of variable values for a single entity (a person or an object).

	Variable 1	Variable 2	Variable 3	Variable 4
Observation 1				
Observation 2				
Observation 3				

Creating a SAS Dataset

Before you run an analysis or create a report, SAS must be able to read your data.
For SAS to be able to read your data, **it must be in SAS dataset format.**
There are many ways to get your data into a SAS dataset format:

1. Enter data directly into a SAS dataset using the **VIEWTABLE** window (Tools -> Table Editor);
2. Import raw data into a SAS dataset using the **Import wizard;** or
3. Use SAS programming statements, such as **Infile:**

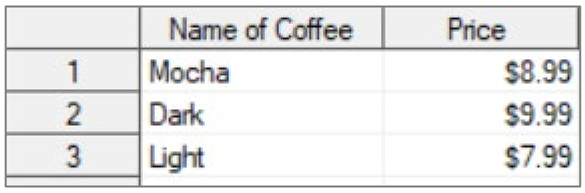
```
Infile ‘C:\Documents\test.sas7bdat’;
```

Exercise #1: Using the Table Editor to Create a SAS Dataset

- From the **Tools** menu select **Table Editor**.
- Click on the column heading **A** and type the word **Coffee** (your cursor will not be visible); then tab to column heading **B** and type **Price**.
    1. **Right-click** on column name, Coffee, to open the Column **A** Attributes window.
    2. Type **Name of Coffee** for the label, and then click on the **Apply** button.

3. When finished, **close** this window.
4. Right-click on the **Price** heading and select **Column Attributes**.
5. Under **Type**, select **Numeric**.
6. Then, click the **"…"** to the right of the **Format** box. In the **Available Formats** box, scroll until you find **dollar** and select it.
7. Under **Format Details**, change the **Decimal** to 2. Click **Ok**.
8. When back to the Column Attributes window, click **Apply**.
9. When finished, **close** this window.

- Type the data below into the new table you have created. SAS will automatically figure out if your columns are numeric or character based on the first row of data you enter (if you did not already specify the type in the column attributes window).



	Name of Coffee	Price
1	Mocha	$8.99
2	Dark	$9.99
3	Light	$7.99

- Once you have created this table, click **File** > **Save As.** In the Save As dialog box, select a library (**Work** for example) and then specify the member name (file name) of your dataset - **coffee** for example.
- Close the Table.
- To print the contents of this table to the Results Viewer, write the following program and then submit it:

```
PROC print data=coffee;
Run;
```

To Edit a SAS Dataset with the Table Editor

- **Tools** > **Table Editor**
- **File** > **Open**> Find your file and open it
- From the **Edit** menu, choose **Edit Mode** and use the Table Editor's menus.

Exercise #2: Basic DATA Steps to Create the Dataset, "backpain"

Type the following program into the Editor window to create the "backpain" dataset:

```
data backpain;
   input Gender $ Age LostDays Cost;   /*input statement assigns name
   to variables */
datalines;
Female    35     10     995
Male      45     10     1115
Female    34     12     1225
Male      23     2      225
Male      50     1      175
;
run;
Proc Print data=backpain;
run;
```

Note: It is a good idea to indent everything between the DATA and Run statements, or between the PROC and Run statements, except the data itself.

- We begin with a DATA step (keyword **data**) to create a temporary SAS dataset called ***backpain***.
- The semicolon ends the data statement.
- The keyword **input** defines the variables to be read from each line of data (column names).
- We have **four variables** following the keyword **input**.

Notice that the Gender variable is followed by a dollar sign (**$**). This tells SAS that the information for that variable is not numeric but a **character variable**. Without specific instructions, SAS assumes all variables are numeric.

- The **DATALINES** statement tells SAS that the next lines contain the data.

Notice that the lines of data do not end in a semicolon. Once SAS reads the DATALINES statement, SAS assumes all subsequent lines are data until it finds a semicolon or either the keywords **proc** or **run**.

- **Proc print** ends the DATA step and tells SAS to process the new dataset (backpain) by listing it in the output window. The final **run** statement tells SAS to process the previous statements.
- Your final statement in a SAS program must always be a **run** statement.

- **Submit** this program, and view the Log, and the Output windows.

Exercise #3: Adding Label and Format Statements to a Program

A format statement is a SAS special instruction for **formatting** variables. Submit this program, and then view the Log and Output files.

Note: The **dollar10.2 format** specifies the display format of the value of the cost variable: Using a total of **10 characters** with **commas, two character spaces** after a decimal point, and a **$** sign in front of the value.

```
Data backpain;
   input Gender $ Age LostDays Cost;

   label LostDays =‘Number of missed workdays’
         Cost = ‘Cost of treatment in US dollars’;
datalines;
Female   35    10    995
Male     45    10    1115
Female   34    12    1225
Male     23    2     225
Male     50    1     175
;
run;
Proc print data =backpain label;
   Format Cost Dollar10.2; /* 10 digits with comma and 2 decimal places*/
Run;
```

Notice: that the DATA step includes a **LABEL** statement. For each variable, you can specify a label **up to 256 characters** long.

Exercise #4: Performing the Means Procedure (Proc Means)

The Means procedure computes descriptive statistics for numeric variables (such as the mean value, standard deviation, and minimum and maximum values). The MEAN procedure starts with the key words, PROC MEANS, followed by Options listing the statistics you want printed to the Results Viewer. Submit the following program:

```
proc means data=backpain;
```

```
   class Gender;
/*The class statement performs analyses for each level of variables in the
list*/
   var Age LostDays Cost;
/*The var statement specifies which numeric variables to use in the
analysis; if it is not specified, then SAS uses all numeric variables;*/
run;
```

Exercise #5: Performing the Frequency Procedure (PROC Freq)

Type the following program and submit it to create frequencies for the backpain dataset, and then view the output:

```
proc freq data=bakpain;
   tables Gender Cost Gender*Cost; /*Produces one-way and two-way tables*/
run;
```

Exercise #6: Performing the Sort Procedure

To sort the backpain dataset by Gender, type the following procedures and submit them. Then view the log and output files:

```
Proc sort data=backpain out=sorted_backpain;
   by Gender;
run;
Proc print data=sorted_backpain;
run;
```

Exercise #7: Performing the Contents Procedure

Proc Contents lists properties for any given dataset. The description of the dataset will be printed by **PROC Contents** to the Results Viewer. Type the following program and submit:

```
Proc contents data=sorted_backpain;
run;
```

Exercise #8: Performing the Univariate Procedure (Proc Univariate)

To run the univariate procedure, type the following program and submit it:

```
proc univariate data=backpain plots; /*This option will create box plots
and normal probability plots*/
   var Age LostDays Cost;
   title 'Summary Statistics for Backpain';
```

```
    title2 '--------------------------------';
  run;
  title; *to clear any title
```

The Output lists basic information about your distribution:

- **N** – the number of observations in the dataset.
- **Mean** – the arithmetic mean across all observations.
- **Standard deviation -** the square root of the variance. It measures the spread of a set of observations.
- **Skewness** – indicates how symmetrical the distribution is (whether it is more spread to one side).
- **Kurtosis** – indicates how flat or peaked the distribution is.

Note: A Normal distribution has values of ZERO for both Skewness and Kurtosis.

Exercise #9: Importing an Excel file

Suppose that you have the results of two tests for a group of five students in the following table (named **Roster**):

Student	Exam1	Exam2
1	80	84
2	85	90
3	70	55
4	94	94
5	88	84

To import an Excel file created from this data table (with no formatting) into SAS 9.4, you must save and close the Excel file before you can import it into SAS.

Note:

When Importing an Excel file (or an Access file) into a SAS program, you'll get a notice that the 64-bit version of Microsoft Windows and SAS 9.4 are not compatible. This isn't limited to importing Excel files. It can also happen when you use PROC EXPORT to export Excel files, or when you try to use LIBNAME EXCEL to reference a local Excel spreadsheet as data.

The Fix:

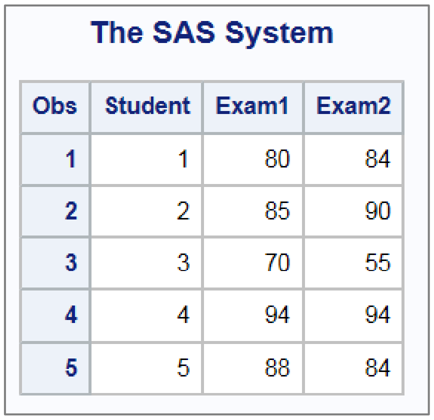
- Use DBMS=EXCELCS for Excel files or DBMS=ACCESSCS for Microsoft Access files.
- Then the syntax is:

```
proc import datafile="path\file.xlsx"
   out=work.newSASfile dbms=xlsx replace;
run;
```

From the Roster dataset, instead of using File > Import Data > Import Wizard, we can write the following codes to import data:

```
proc import datafile='C:\Users\fboren\Desktop\roster.xlsx' OUT=
   work.roster DBMS=xlsx replace;
run;
proc print data=roster;
run;
```

And here is the output in the Results Viewer:



The SAS System

Obs	Student	Exam1	Exam2
1	1	80	84
2	2	85	90
3	3	70	55
4	4	94	94
5	5	88	84

Exercise #10: Using Logical Operators to Compute in SAS

Write the SAS program below to compute a final grade as the average of the two Exams given in problem #9 and assigning a letter grade as:

Grade < 65 = F

65 <= Grade < 75 = C

75<= Grade < 85 = B

Grade >= 85 = A

```
Data Myclass;
```

```
SET Roster;/* SET statement is like the INPUT statement, with the
            difference being that a SET statement reads records
            from SAS dataset in a SAS library, while an INPUT
            statement reads raw data from an external file */
Final=(Exam1+Exam2)/2;
   If 65>final>=0 then grade='F';
   If 75>final>=65 then grade='C';
   If 85>final>=75 then grade='B';
   If final >= 85 then grade='A';
Run;
Proc print data=myclass noobs;
   var student final grade;  /* in that order*/
   title 'SAS - Part I  SC Exercise';
   title3 'Texas Tech University';
Run;
```

Using Logical Operators

Here are the most common logical operators that are used to generate arithmetic or logical expressions:

Expression	Operator	Definition
EQ	=	Equal
LT	<	Less than
LE	<=	Less than or equal
GT	>	Greater than
GE	>=	Greater than or equal
AND	&	Logical and

Exercise #11: Appending Two Datasets

CLASS1			CLASS2		
Name	**Age**	**Height**	**Name**	**Age**	**Height**
Andrew	15	67	Andrea	16	60
Philip	14	70	Linda	13	55
Robert	15	78	Sandra	17	65
Stephen	17	72			

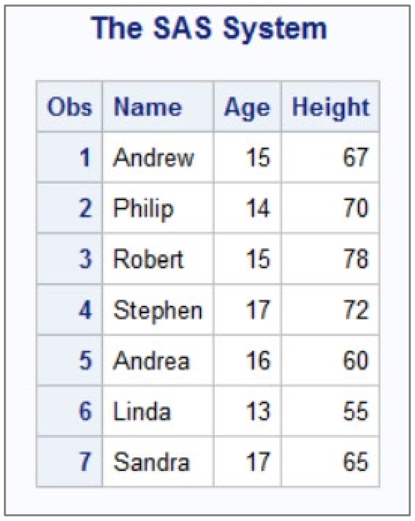
Note: These datasets have the same variable names.

The following SAS program uses the **PROC APPEND** procedure to add observations from one dataset, **CLASS2**, to the end of another dataset (called the **BASE** dataset), **CLASS1**:

```
data CLASS1;
   input Name $ Age Height;
datalines;
Andrew    15    67
Philip    14    70
Robert    15    78
Stephen   17    72
;
run;
Data CLASS2;
   input Name $ Age Height;
datalines;
Andrea    16    60
Linda     13    55
Sandra    17    65
;
run;
proc print data=CLASS1;
run;
proc print data=CLASS2;
run;
PROC APPEND  BASE=CLASS1  DATA=class2;
```

```
RUN;
proc print data=CLASS1;
run;
```

The PROC APPEND procedure above adds the observations from SAS dataset CLASS2 to the end of SAS dataset CLASS1. The SAS dataset CLASS1 *is changed* and now will have the observations to the right.



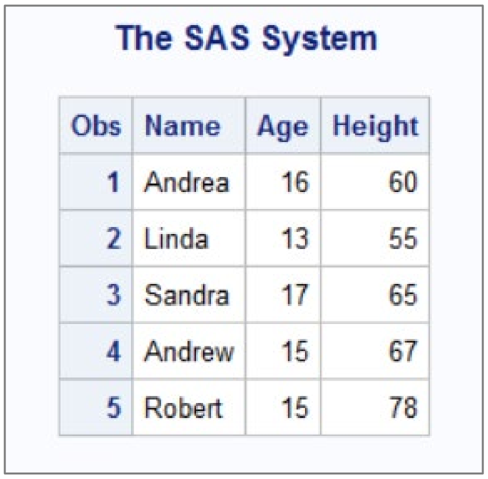
The SAS System

Obs	Name	Age	Height
1	Andrew	15	67
2	Philip	14	70
3	Robert	15	78
4	Stephen	17	72
5	Andrea	16	60
6	Linda	13	55
7	Sandra	17	65

You can also specify which observations will be added to the other SAS dataset. Here's an example:

```
PROC APPEND BASE=class2 DATA=class1
    (WHERE=(age=15));
RUN;
```

The **PROC APPEND** procedure above adds observations from the dataset CLASS1 to the end of the dataset CLASS2. The dataset option **WHERE=** is used to choose the observations with age equal to 15. Thus, it will select all observations in which age equals 15 from the dataset CLASS1 and add them to the end of the dataset CLASS2. The dataset CLASS2 will now have the following observations.



The SAS System

Obs	Name	Age	Height
1	Andrea	16	60
2	Linda	13	55
3	Sandra	17	65
4	Andrew	15	67
5	Robert	15	78

Ending Your SAS Session

You can use one of the methods below to end your SAS session:

- From the **File** menu click on **Exit**.
- Or- Double-click the SAS control button (the small icon in the upper-left corner of the main SAS window).

- Or- Click the **X** (in the upper right corner).

SAS e-Learning

As a member of Texas Tech University, you have access to **self-paced e-Learning** from SAS.

Please follow the steps below to activate your e-Learning. You will be prompted to insert your unique activation code in Step 2. Your activation code is the **prefix G followed by Texas Tech University Site number for SAS.**

To activate your e-Learning:

1. Go to http://support.sas.com/myelearn and log into your **SAS profile**. If you do not currently have a profile, please **create** one using the "Create Profile" option at the above link.
2. When you create an account, it will email you an activation code. Enter this code into the activation code box, and then click **Submit.** Review the license agreement and select **I agree** to accept it.
3. You should now see the training in your list of active courses and the date on which your access expires. Select a course title to start your learning.

Here is the list of SAS e-Learning available to TTU community to Activate:

- SAS(R) Programming Introduction: Basic Concepts
- SAS(R) Programming 1: Essentials
- SAS(R) Programming 2: Data Manipulation Techniques
- SAS(R) SQL 1: Essentials
- SAS(R) Programming 3: Advanced Techniques and Efficiencies
- SAS(R) Macro Language 1: Essentials
- SAS(R) Enterprise Guide(R) 1: Querying and Reporting
- Querying, Reporting, and Analyzing Data Using SAS(R) Enterprise Guide(R)

To access your e-Learning once it has been activated:

Once your e-Learning has been activated, visit http://support.sas.com/myelearn or select **My Training** from any page on the SAS training Web site and log into your profile to access your training. Please e-mail eLearn@sas.com if you have any questions.

Additional Online SAS Resources

- LEARNING CENTER / TRAINING – including SAS Self-Paced e-Learning and free tutorials: http://www.sas.com/apps/elearning/elearning_category_welcome.jsp
- SAS on Demand for Academics: http://support.sas.com/learn
- SAS Knowledge Base & Product Documentations: http://support.sas.com/documentation/onlinedoc/sas9doc.html
- SAS Technical Support and Online Samples: http://support.sas.com
- SAS Press and SAS Documentation Example Code and Data: http://support.sas.com/documentation/onlinedoc/code.samples.html
- SAS Annotated Output: http://www.ats.ucla.edu/stat/sas/output/univ.htm

SAS Resources for students

- SAS for Students: http://www.sas.com/govedu/edu/programs/studentresources.html
- SAS Job Network: https://www.sas.com/en_us/careers.html
- Suggested TTU Safari e-Books at http://library.ttu.edu
    - SAS Programming by Example: a Programmer's Guide, 2007. By Ron Cody & Ray Pass
    - A Step-by-Step Approach to Using SAS for Univariate & Multivariate Statistics, Second Edition, 2005. By Norm O'Rourke; Larry Hatcher; Edward
      J. Stepanski
    - The little SAS Book, 5th edition, 2012. By Lora Delwiche, & Susan Slanghter
- SAS 9.4 Documentation by Title: http://support.sas.com/documentation/cdl_main/94/docindex.html

Where to Get Help

- Texas Tech University has site licenses for SAS software, and as a member of the TTU community, you have the right to have full 24-hour support from SAS. You can do that via email: mailto:support@sas.com . You can contact this email with any problem you have. When you send an email message, remember to insert at the beginning of the message **TTU's SAS site number.**
- If you need a Statistical Consultant, please visit the **Statistical Consulting Lab** (**SCL**) of the Department of Mathematics & Statistics www.math.ttu.edu/~SCL

Please e-mail your comments or suggestions to: Faith.L.Harper@ttu.edu