**The George Washington University**

Department of Statistics

Stat 4197/6197 **–** Fall 2018

Week 1 – August 31, 2018

Major Topic: SAS Basics

Detailed Topics:

1. SAS Software and its Products
2. Ways to Run SAS
3. Components of SAS Programs
4. Rules for SAS Names and Language Elements
5. Data Step Concepts
6. Files Concepts
7. SAS Libraries, SAS Data Sets, and their Contents (Metadata)
8. List Reports
9. SAS User Resources

Readings:

# Relevant Chapters/Sections - Delwiche LD, and Slaughter SJ. *The Little SAS Book: A Primer*, Fifth Edition Paperback – November 7, 2012

# Exercises from Relevant Chapters/Sections - Ottesen RA, Delwiche [LD](http://www.amazon.com/s/ref=dp_byline_sr_book_2?ie=UTF8&text=Lora+D+Delwiche&search-alias=books&field-author=Lora+D+Delwiche&sort=relevancerank), and Slaughter [SJ](http://www.amazon.com/s/ref=dp_byline_sr_book_3?ie=UTF8&text=Susan+J+Slaughter&search-alias=books&field-author=Susan+J+Slaughter&sort=relevancerank). *Exercises and Projects for The Little SAS Book*, Fifth Edition Paperback – July 1, 2015

# SAS**®** 9.4 Language Reference: Concepts, Sixth Edition

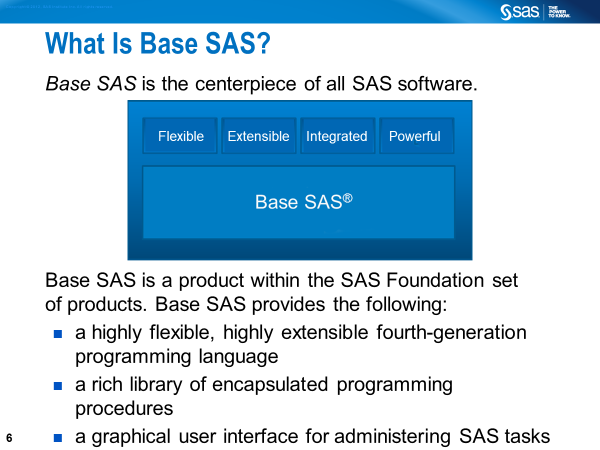
# <http://documentation.sas.com/?docsetId=lrcon&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en>

1. THE SAS SUPERVISOR. By Don Henderson & Merry Rabb ORI, Inc.

<https://www.lexjansen.com/nesug/nesug88/SAS_supervisor.pdf>

**Statistical Analysis System® or SAS® - An Overview**

****



**History**

1966: SAS was created by Anthony Barr (grad student at the UNC)  
1976: SAS Institute, Inc. was incorporated by Barr, Goodnight, Sall, and Helwig

2017: Revenue – US$3.24 billion

The SAS software suite includes more than 200 components, and the following languages.

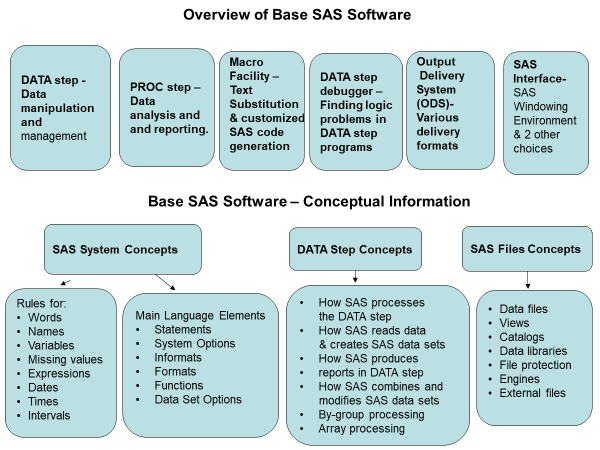
* **the SAS language**
* **Macro**
* **SQL**
* SCL
* SAS/C
* **IML**
* SAS Viya (Cloud Analytics Services(CAS) programming language)

To find the version and SAS products in your computer, run the following SAS code: **PROC** **SETINIT**; **run**;

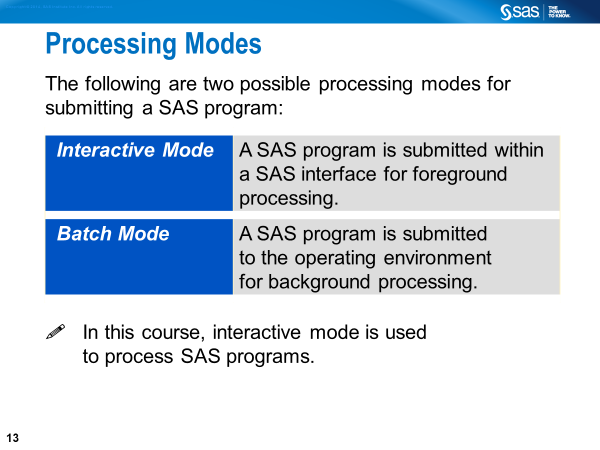
To see what products have been installed in your computer, run the following SAS code: **PROC** **PRODUCT\_STATUS**; **RUN**;

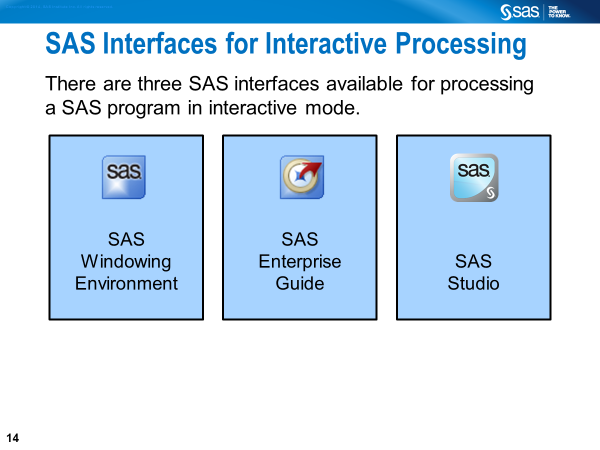
****

**You can run SAS on many platforms, including Windows and Unix.**

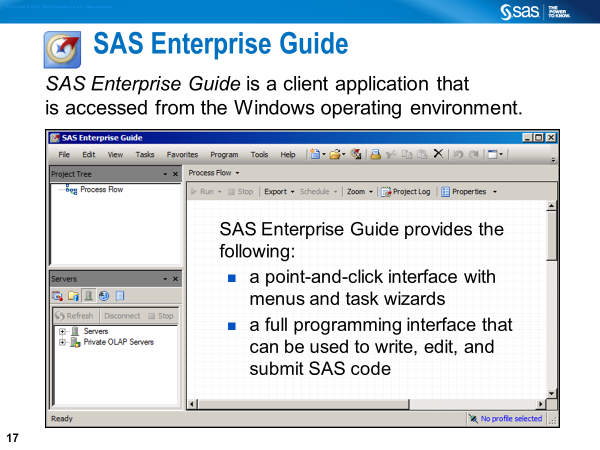


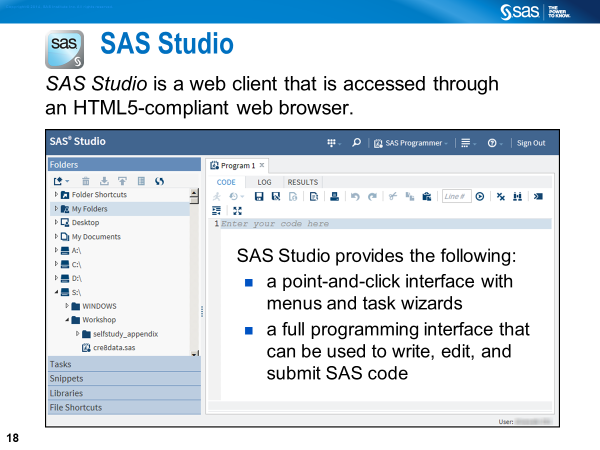
**Ways to Run SAS**

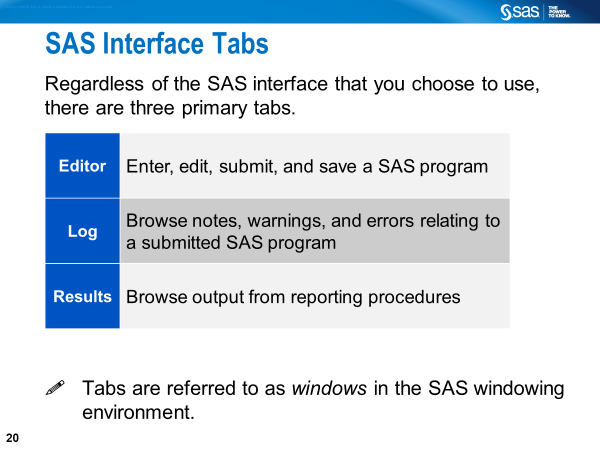
****

****

****

****

****



**Enhanced Editor provides** color-coding to identify SAS program elements. To run the code, you click the **Submit** button (“running person” icon) in the tool bar or select **Run** >> **Submit** in the main menu.

**Log** file can be saved as filename.log. Remember that the log window will always show activity when you submit a SAS program in the SAS windowing environment.

**Results tab** provides tree-like listing. Helps you locate items in the SAS Output Window and delete duplicate items.  Starting with SAS 9.3, HTML is the default method for viewing output (which can be saved as filenanme.mht). The default method is changeable as follows: From the menu bar, Select **Tool** >> **Option** >> **Preference >>**

**Additional Interface Tabs or Windows**

**Output** shows the results of the SAS output (Pre-SAS 9.3 default method for viewing output (which can be saved as filename.lst).

**Explorer** allows the user to access various resources including SAS libraries.

**The SAS Menus**

**File: Allows you to open and save SAS programs, import and export data, and print files.** From the menu bar, Select **File** >> **Save As** and specify the name of your program by assigning an extension (e.g., DataStep1.SAS). Assign the extension **.SAS** to your SAS programs to distinguish them from other types of SAS files (e.g., **.LOG** for log files; **.OUT** for output files, and **.DAT** for raw data files

**Edit: Allows you to copy, cut, and paste text, and find and replace text when writing a SAS program.** After you run your SAS program, you will find some text in the Log window, and may find text/results in the Output window. To clear the Output window, Program Editor window or Log window individually, Select **Edit ➩ Clear All**.

**View: Allows you to go back and forth between the Editor window, Log window, Output window, Results window, and Explorer window, etc.**

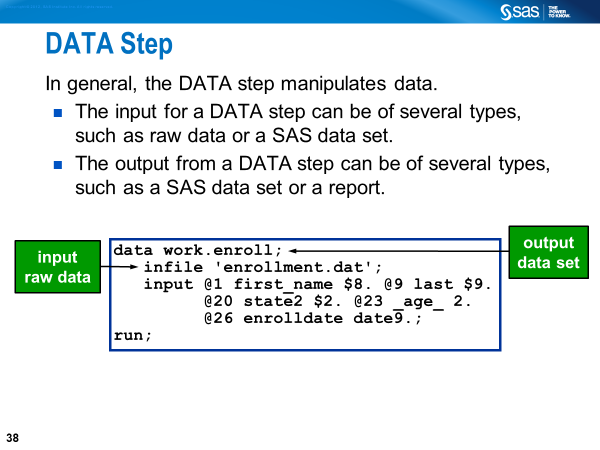
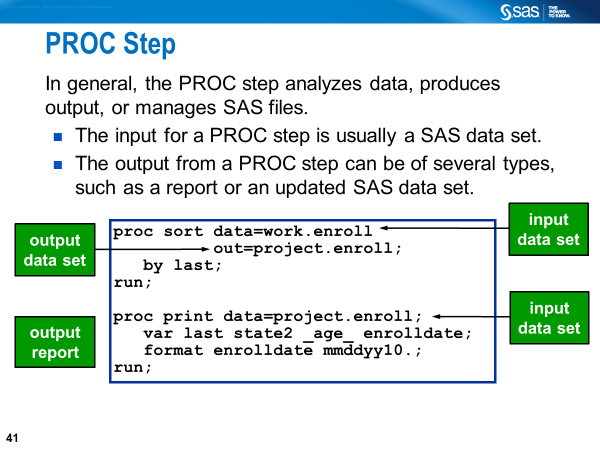
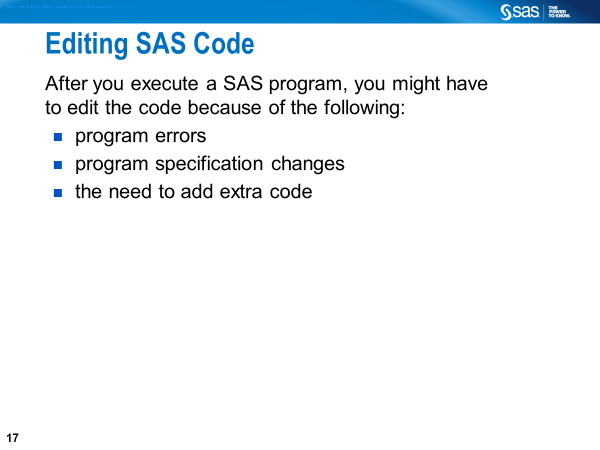
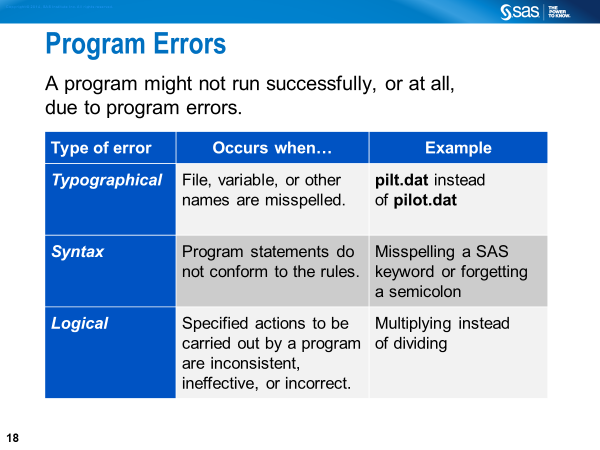
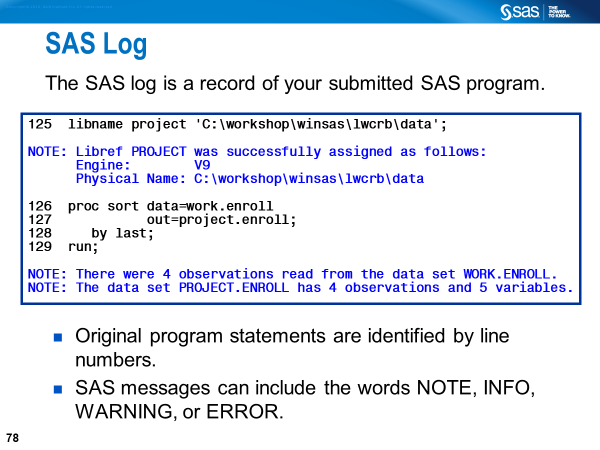
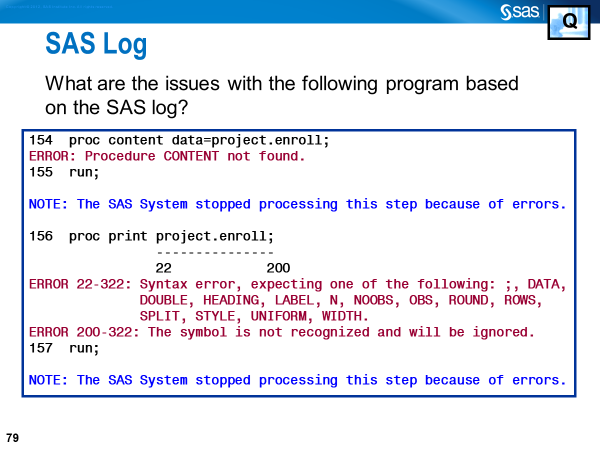
**Tools: Allows you to open programs for graphics and viewing tables, etc.**

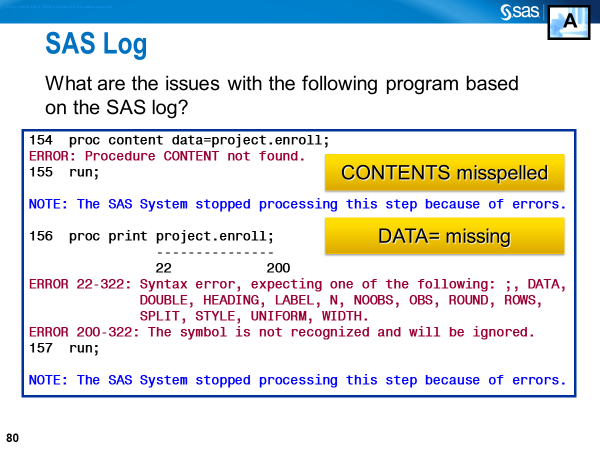
**Run: Allows you to submit SAS programs.** A SAS program when submitted by clicking the ‘running man’ icon disappears from the program editor window. You can recall it by selecting **Run ➩ Recall Last Submit**.

**Help: Allows to o**btain help for writing your SAS source code. Invoke the Help window by selecting **Help** >> **SAS Help and Documentation** in the main menu.

**Components of SAS Programs**

A SAS program, which is saved as text files with an extension sas, can be any combination of at least the following:

* DATA step,
* PROC step,
* Global statement,
* SAS macro language code
* 
* 
*  
* 
* 



Remember that the log window will always show activity when you submit a SAS program in the SAS windowing environment. To know more about this, read Sections 1.7 and 1.8 of *The Little SAS Book*.

**Beginning SAS 9.3, by default, SAS creates results in the HTML format.**

To view the results, you are required to turned on the creation of LISTING results using the

**Tools -> Options -> Preferences -> Results** choices as shown below. You may want to turn off the automatic creation of the HTML results, also shown below.

**Exploring SAS Programs**

**\*Example\_DSPS.sas;①**

**DATA** work.HAVE; ②

INPUT Name $ quiz1 quiz2 quiz3; ③

Ave\_Score = ROUND(MEAN(OF quiz1-quiz3),**.01**);④

DATALINES;⑤

Amy 78 84 82

Neil 90 85 86

John 82 79 89

Keya 78 86 78

;⑥

**PROC** **PRINT** data=work.HAVE; ⑦

**run**;⑧

①This line is commented out to prevent from execution.

②The DATA statement marks the beginning of the DATA step (**step boundary**).

③The INPUT statement lists variable names.

④This is an ASSIGNMENT statement that creates a new variable AVE\_SCORE.

⑤The DATALINES statement tells SAS that data records are located in the next lines.

⑥This is a null statement that marks the end of the input data (**step boundary**).

⑦The PROC PRINT statement marks the beginning of a new step.

⑧The RUN statement marks the end of the PRINT procedure (**step boundary**).

The DATA step is an implied loop. In the above example, SAS will execute the DATA step one time for each observation. The INPUT and ASSIGNMENT statements are executed 4 times. The loop stops when it gets to the INPUT statement a 5th time.

To submit the program, click on the icon of **“the little guy running”** at the top of the SAS tool bar. After the program is executed SAS will generate the following:

* Log Messages
* Results of Processing
  + - Data Step Output
    - Procedure Output

**SAS Step Boundary**

A step boundary may begin with

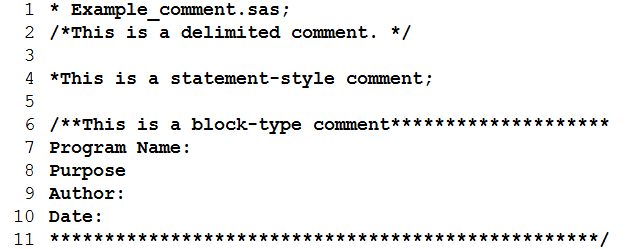
* a DATA statement
* a PROC statement

A step boundary may end with

* a DATA statement
* a PROC statement
* a RUN statement (for DATA steps and most PROCs)
* a QUIT statement (for some PROCs)
* a null (;) statement

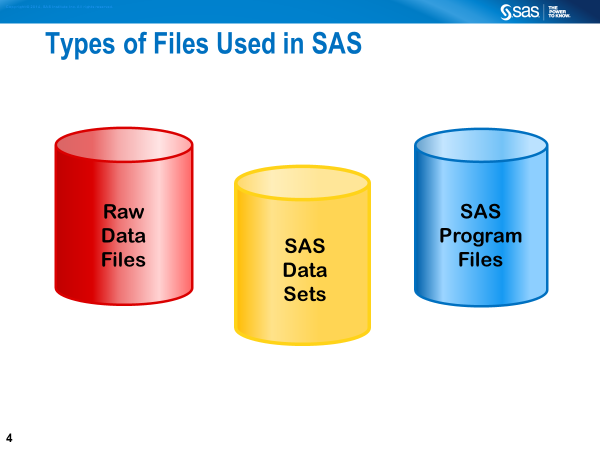
**Commenting Out in SAS Programs**

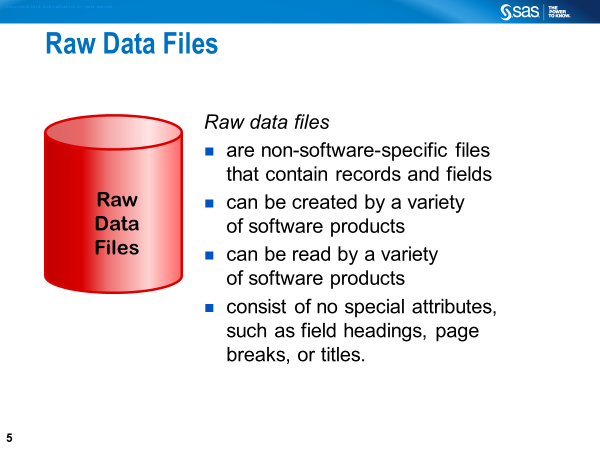
* Statement-type comment
* Delimited comment
* Block-type comment

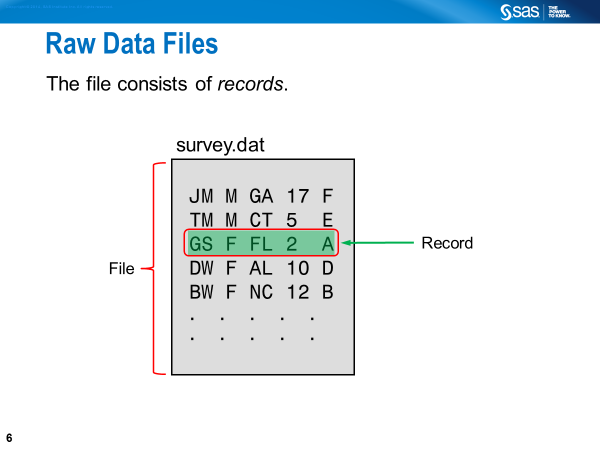


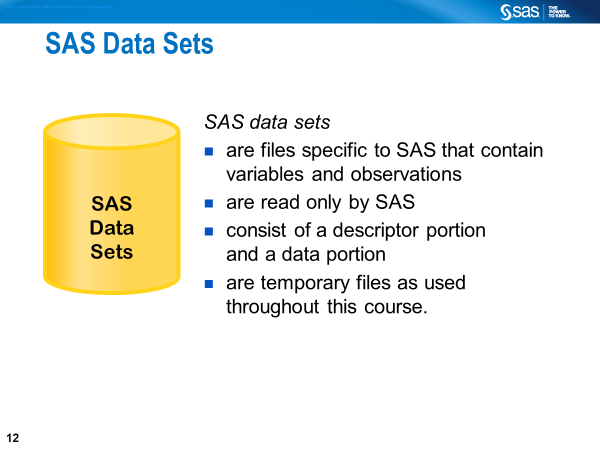
The above lines are commented out. None of the lines are valid SAS statements. When submitted, they are not compiled or executed.

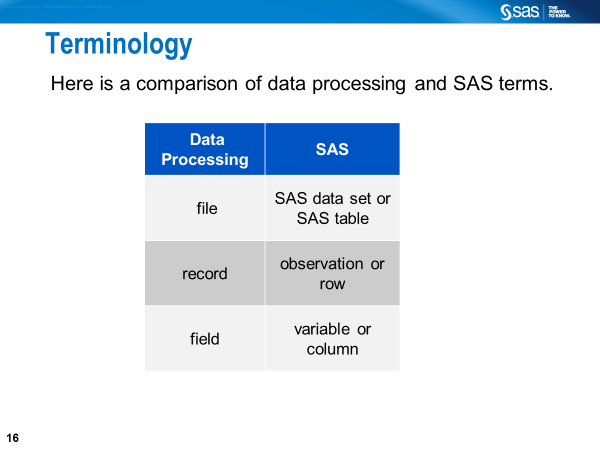
* Lines 1 and 4 are of statement-style.
* Line 2 is a delimited type comment.
* Lines 6-11 are of block type.

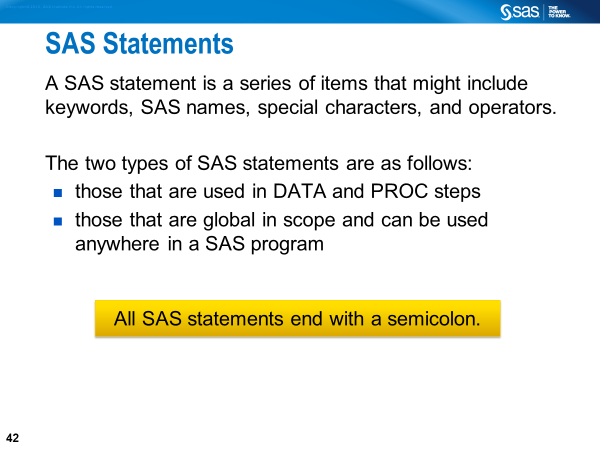


****

****

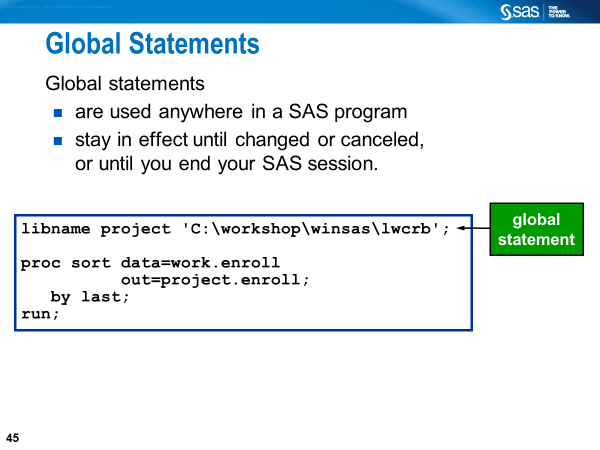
****

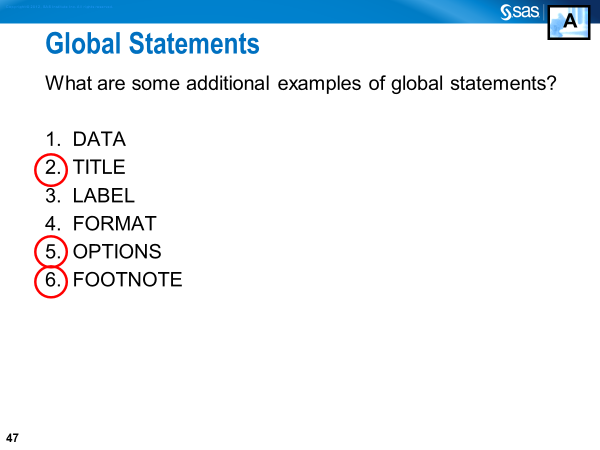
****

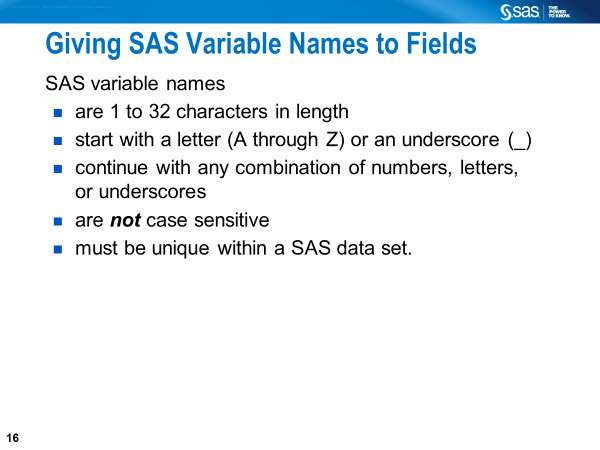
****

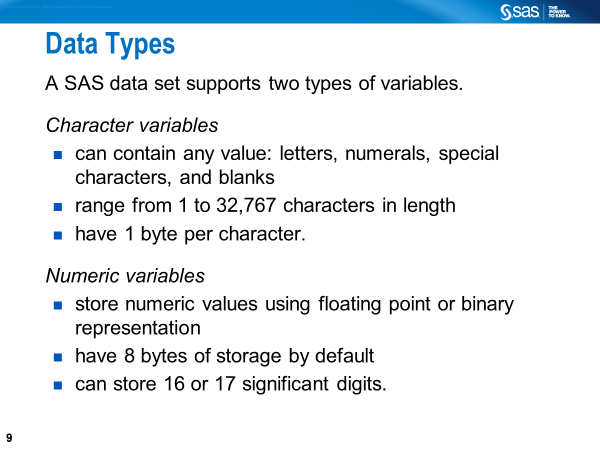
**Characteristics of SAS Statements**

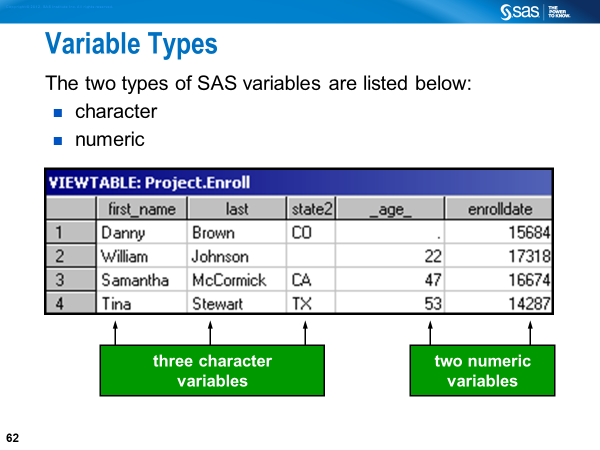
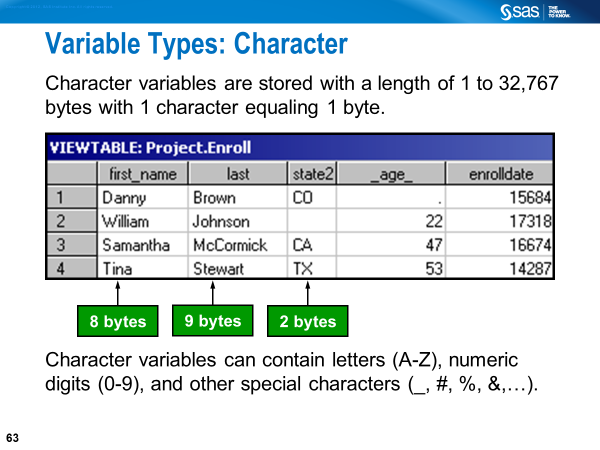
* Begin with a keyword (e.g., DATA, PROC) or keywords (e.g., CALL MISSING) and end with a semicolon
* Can begin and end anywhere on a line or over several lines (or several statements can be on the same line)
* Can be entered in uppercase or lowercase
* Can have blank or special characters that separate words
* Can have words that are separated by blanks or special characters, but cannot have words that are entered across lines





****



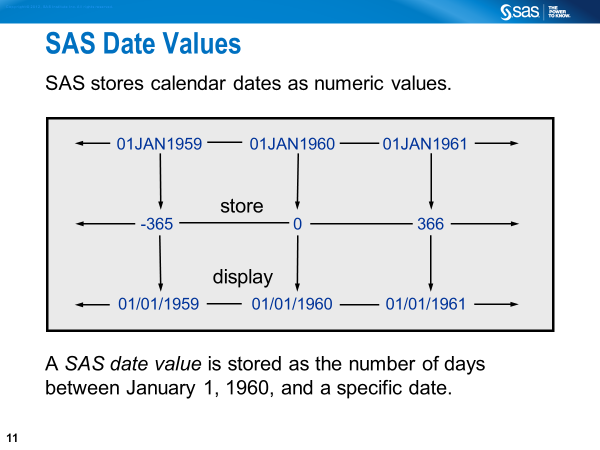


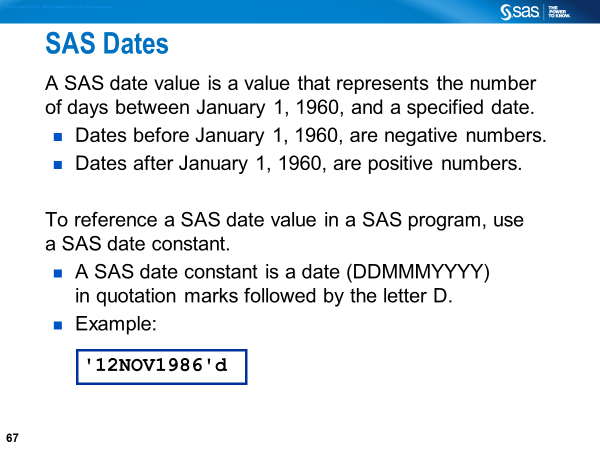
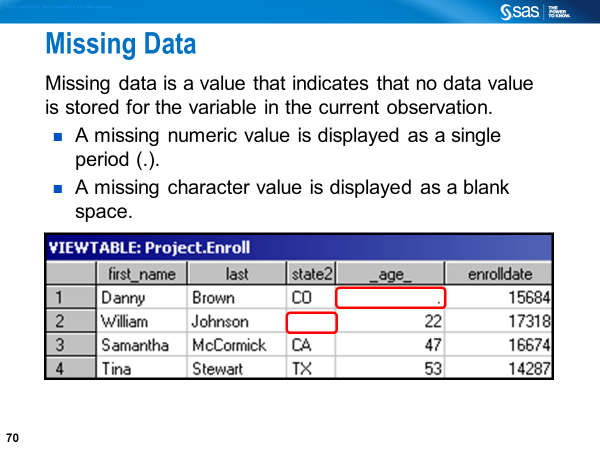
The following are the only acceptable characters in a standard numeric field:

**0 1 2 3 4 5 6 7 8 9 . E e D d - +**

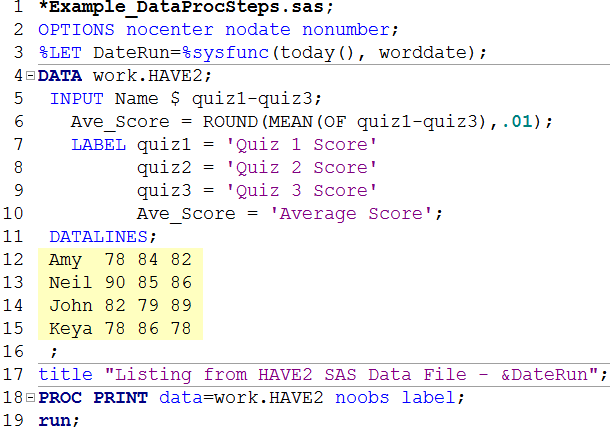
Leading or trailing blanks are also acceptable.

* **E**, **e**, **D**, and **d** represent exponential notation in a standard numeric field. An alternate way of writing **300000**, for example, is **3E5**.

****

**** ****

**SAS Statements Explained**



Line 1: This is a “Statement-Style Comment” preventing this line from execution.

Line 2: The OPTIONS statement is a global statement and used here to specify 3 options. NOCENTER prevents centering SAS output. NODATE prevents printing the date on the output. NONUMBER prevents printing page numbers on the output. Some Additional Common SAS System Options are shown below.

FIRSTOBS=1 First obs. of the Data file to be processed

LINESIZE=98 Line size for the printed output

MISSING=. Character printed for numeric missing values

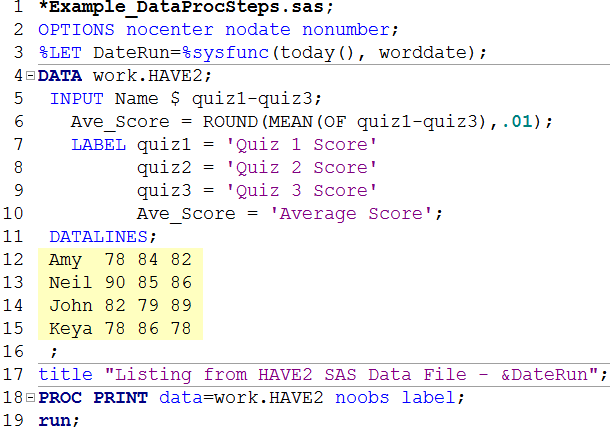
OBS=MAX Number of last observation to be processed

PAGENO=1 Resets the current page number on SAS output

PAGESIZE=58 Number of lines printed per page of output

Line 3: The %LET statement defines the macro variable (i.e., DateRun) before the SAS program is executed.

**SAS Statements Explained (continued)**

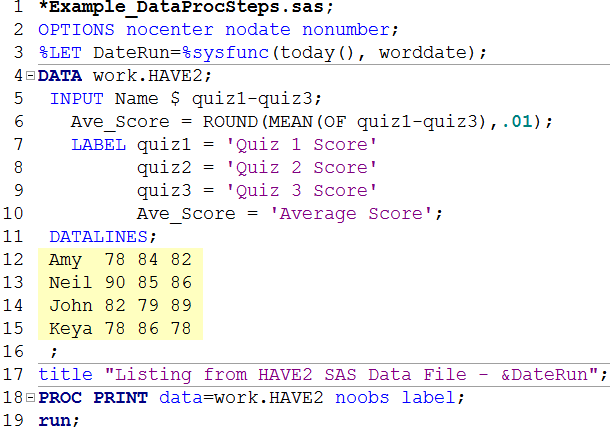


Line 4: The DATA statement marks the beginning of the data step. WORK.HAVE is a user-given SAS Data Set name (specifying work. is optional). After a successful run, WORK. HAVE gets created. Note that WORK is a temporary SAS library, which is automatically created by SAS in every data step session even if WORK is not specified. WORK. HAVE gets deleted after we exit the SAS session.

Line 5: The INPUT statement lists variable names. NAME is an alphanumeric variable, as indicated by the **$.** QUIZ1-QUIZ3 are numeric variables; the dash operator enables us to list variables that are numbered sequentially.

Line 6: This is an assignment statement used to create a new variable called AVE\_SCORE. AVE\_SCORE (Average Score) is on the left-hand side of the assignment statement. The right-hand side of the assignment statement includes two numeric functions: ROIUND () and MEAN(). The first argument of the ROUND() is the MEAN () with three numeric arguments (i.e., quiz1, quiz2, and quiz3). The second argument of the ROUND() is .01 to tell SAS to round-up the returned value to the nearest hundredth.

**SAS Statements Explained (continued)**

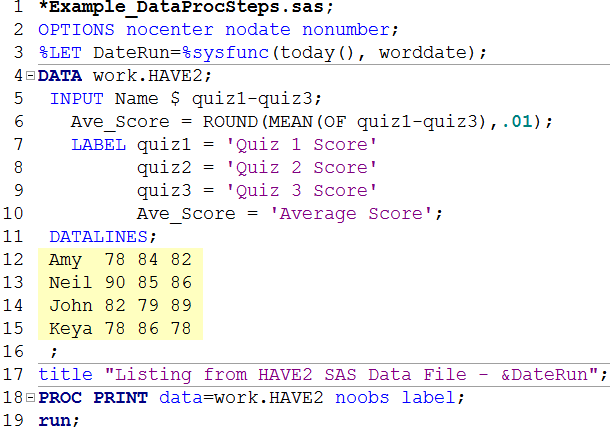


Lines 7-10: The [LABEL statement](javascript:%20return%20false;) defines permanent labels (up to 256 characters in length) for four variables namely, QUIZ1, QUIZ2, QUIZ3, and AVE\_SCORE in the DATA step. Alternatively, you can use the [LABEL statement](javascript:%20return%20false;) in a PROC PRINT step to define temporary labels for the variables created in the DATA step.

Line 11: The DATALINES statement tells SAS that the data are located in the next lines and that data records will continue to be read until a line with a semicolon is encountered.  Instead of the DATALINES statement you can also use the CARDS statement. There are also DATALINES4 and CARDS4 statements, which are enhanced versions of the DATALINES and CARDS statements, respectively. DATALINES4 and CARDS4 statements each allows semicolon to be placed in the instream data; however, you must use four semicolons to mark the end of the instream data.

Line 16: This is a null statement, which signals the end of the data lines that occur in the above program. The semi-colon after the data lines causes the DATA Step to execute.

**SAS Statements Explained (continued)**



Line 17: The TITLE statement is a global statement because the operations with this statement are not tied to a particular data or proc step. It remains in effect until you cancel or change it or until you end your SAS session. Although the regular text in the TITLE statement is put in single quotes, the text that includes the macro variable reference (i.e., &DateRun) must be put in double quotes in order to substitute the parameter value (i.e., today’s date) for the macro variable reference.

Line 18: The PROC PRINT statement marks the beginning of a new step. The NOOBS option suppresses the observation number. The LABEL option is specified to display descriptive labels that are saved in a SAS Data Set (or the labels that are temporarily defined in this proc step). Alternatively, you can use the [SPLIT=](javascript:%20return%20false;) option to display the labels as well as specify a split character to control line breaks in column headings.

Line 19: The RUN statement is the step boundary for the PRINT procedure that begins in the previous line.

**Saving a Log File Automatically**

**\*Save the log file automatically;**

DM 'log; file "C:\SASCourse\Week1\%sysget(SAS\_EXECFILENAME)log" replace;

**Explanation of the SAS Code**

The DM statement sends the SAS log to a file as shown below





SAS\_EXECFILENAME is an environment variable for the Enhanced Editor, and you can retrieve the name of the current program (that you are running interactively) by adding %sysget(SAS\_EXECFILENAME) to DM statement. Here the name of the program (Example\_DM\_Clear\_Save.sas) is concatenated with the text **log**.

**\*Clear the log window from a previous SAS session;**

DM log "clear";

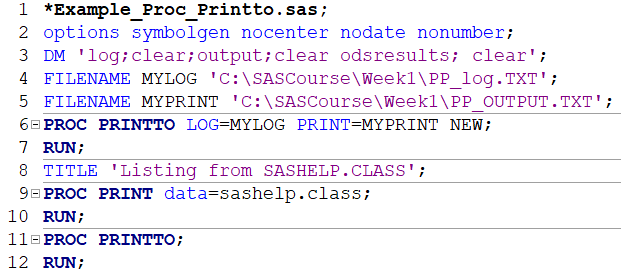
**\*Clear the output window from a previous SAS Session;**

DM output "clear";

**\*Clear the results window from a previous SAS Session;**

DM ODSRESULTS "clear";

**How to Automatically Create Log and Output Files**



Line 3: The DM statement automatically clears LOG, OUTPUT, and ODSRESULTS.

Line 4: The FILENAME statement associates a fileref (i.e., MYLOG) with an external file that is used for output (i.e., PP\_log.txt).

Line 5: The FILENAME statement associates a fileref (i.e., MYPRINT) with an external file that is used for output (i.e., PP\_OUTPUT.txt).

Lines 6-7: This PROC PRINTTO step writes the log and print output to disk for the PROC PRINT STEP in lines 9-10.

Lines 11-12: This "null" or "dummy" PROC PRINTTO step is required to close the log and print files.

**Exploring SAS Data Step Processing**

* After the SAS program is submitted, the codes are copied into a memory area called **input stack**
* The **word scanner** reads the text from the input stack and breaks itinto fundamental units called tokens, which are of four types:
  + **Literal** - a string of characters enclosed in quotation marks (e.g., “GWU”)
  + **Number** - digits, date values, time values, and hexadecimal numbers (e.g., 1234 ‘13mar2016’dv 20e4v 14.5)
  + **Name** - a string of characters beginning with an underscore or letter (\_N\_ Descending, means)
  + **Special characters** – other than a letter, number, or underscore that have a special meaning to the SAS system (e.g., \* / + - % &)
* The word scanner**passes tokens**, one at a time, to the appropriate compiler (regular SAS compiler, or macro compiler) as appropriate. The compiler **requests tokens until it receives a semicolon.** The word scanner repeats this process for each SAS statement.
* The compilation is suspended after the step boundary is encountered.
* SAS statements are **scanned for syntax errors**
  + missing semicolons
  + misspelled keywords
  + unmatched quotation marks
  + invalid options
* If there is **no compilation error** for the step, SAS **executes the compiled code**.

**Compilation Phase**

* The **input buffer** is created to hold a data record from the raw data file
* The **program data vector** (PDV) is created for two automatic variables (\_N\_ and \_ERROR\_)
* The **descriptor portion** of the SAS data file is created: data file name, # of observations, and the number, names and attributes of variables

**Execution Phase**

* **Variables in the PDV are initialized to missing** before each execution of the data step
* The DATA step **executes once for each record in the input file**, unless otherwise directed by additional statements
* Each record in the raw data file is **read into the input buffer**, **copied to the** **PDV**, and then **written to the** **SAS data file** at the end of the DATA step

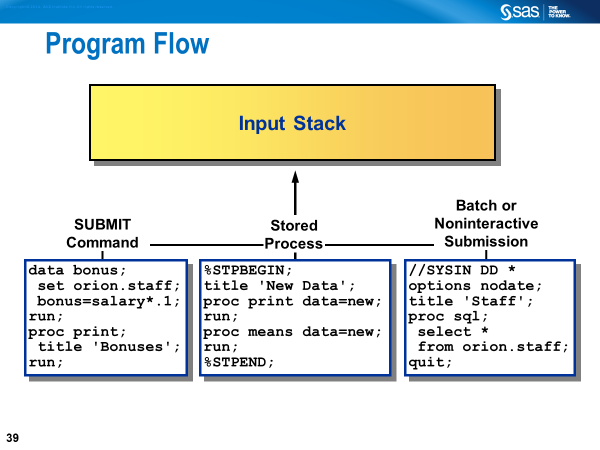
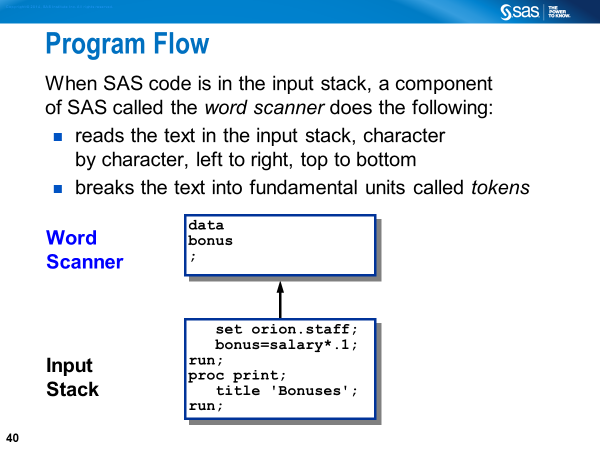
There are two types of names in SAS:

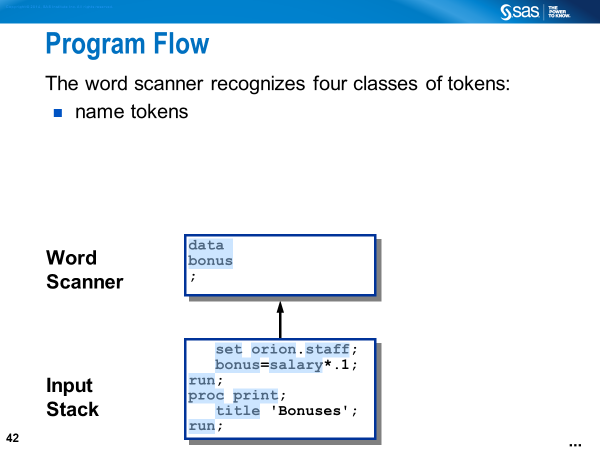
* names of elements of the SAS language
* names supplied by SAS users

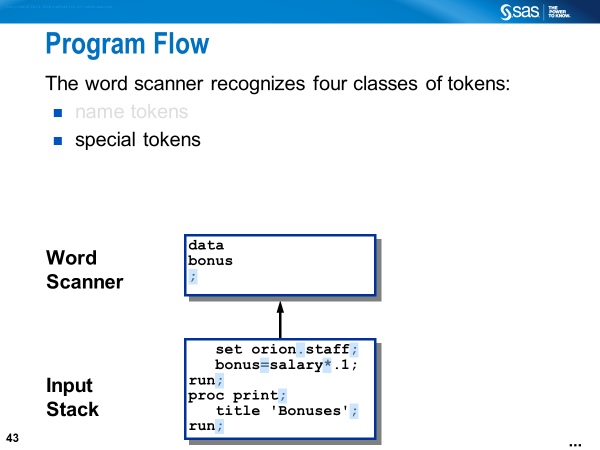
Below are some of the SAS name tokens that represent:

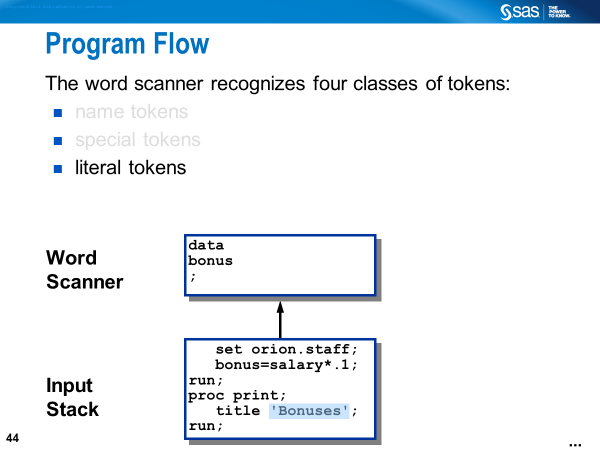
|  |  |  |  |
| --- | --- | --- | --- |
| * variables | * SAS data sets | * formats or informats | * SAS procedures |
| * options | * arrays | * statement labels | * SAS macros or macro variables |
| * SAS catalog entries | * librefs | * filerefs | * component objects |

<https://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000998953.htm>

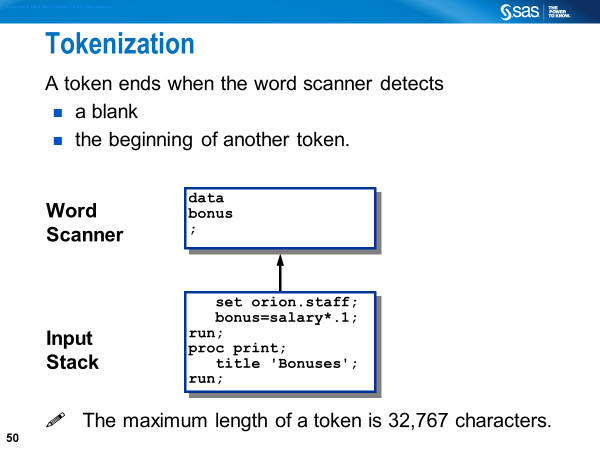
**** ****

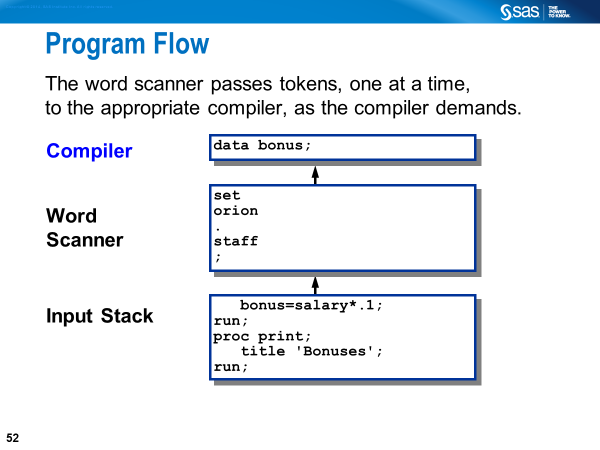
****

****

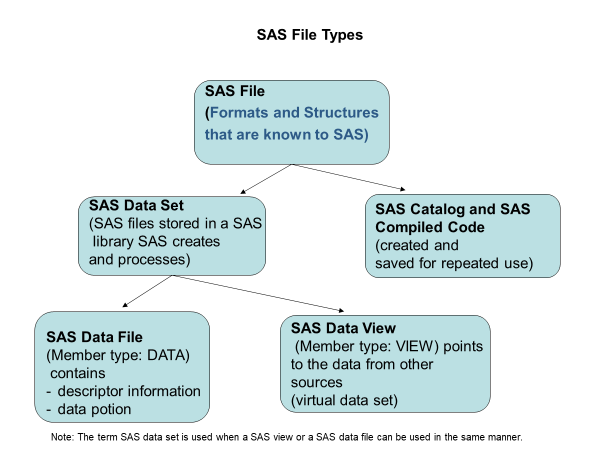
****

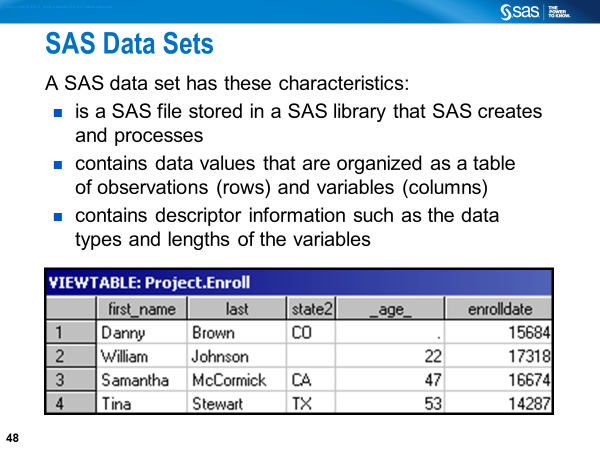
****

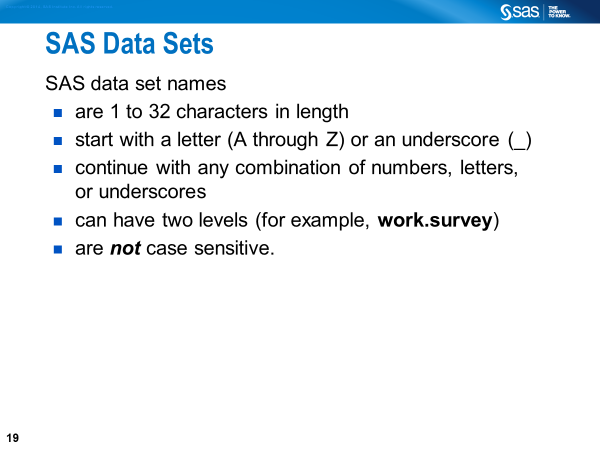
****

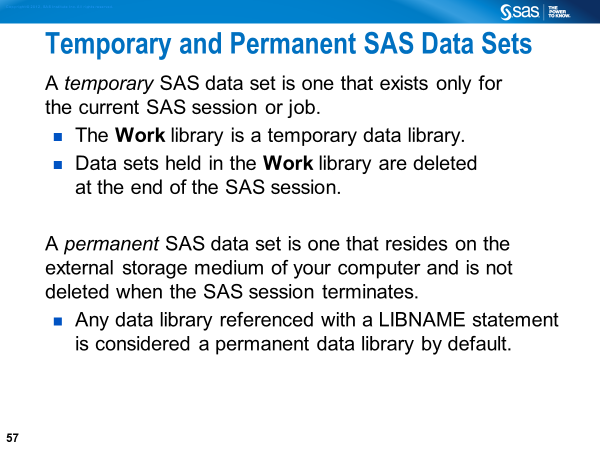
****

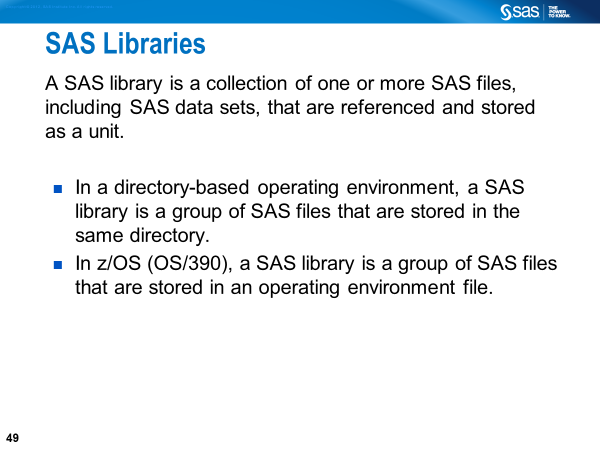
**SAS Files Concepts**

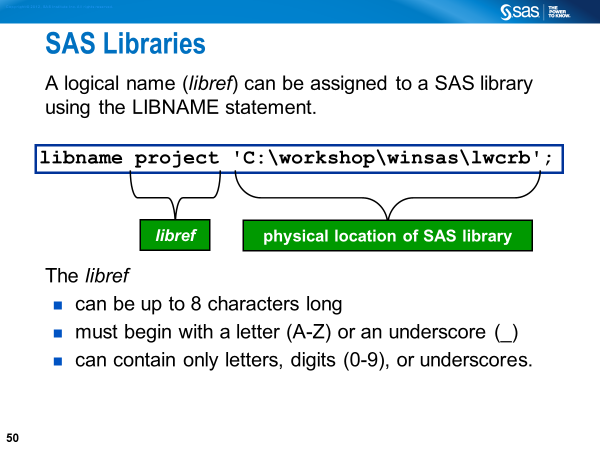
****



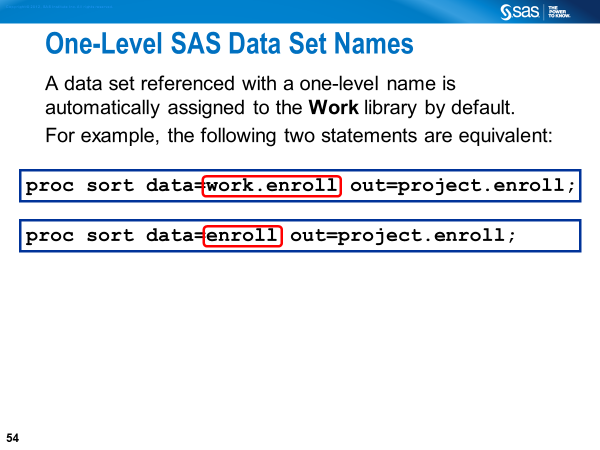
****





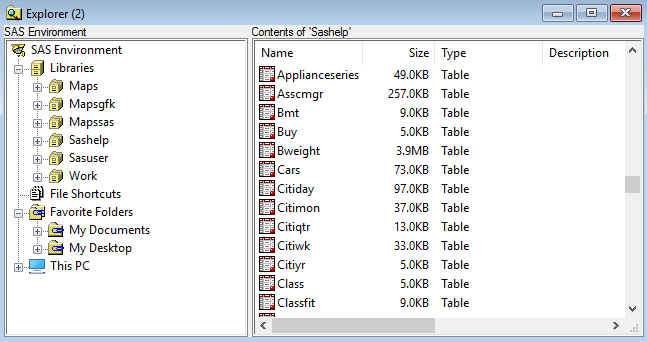


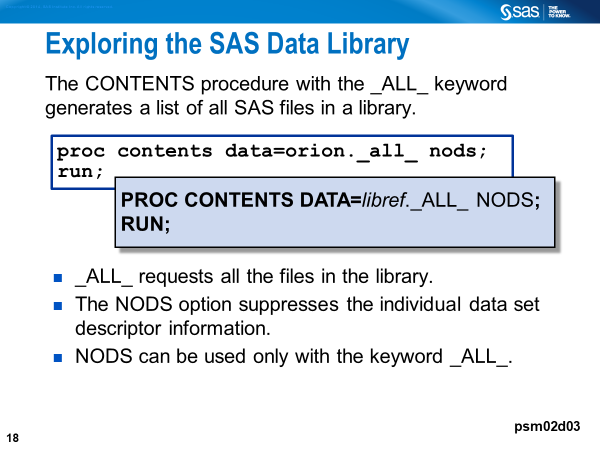
“A libref is a name that you associate with the physical location of the SAS library. You should not use SASHELP, SASUSER or SASWORK as librefs.” – SAS Documentation.



**Built-in SAS Libraries**

SASHELP and SASUSER are built-in libraries and will always show up when SAS is invoked. The SASHELP library is where SAS has stored all the demonstration data files and catalogs; there are about 200 SAS data sets (i.e. Tables) in this library. Below is the snapshot of selected Tables from the **Explorer** sub-window.



****

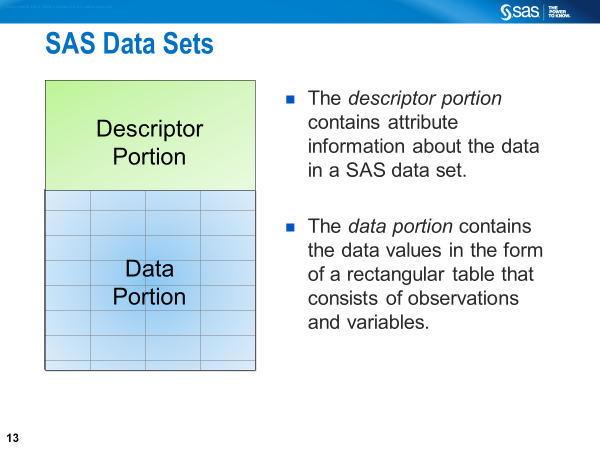
The following example code will generate listing of the contents of the SAS library.

**Proc** **contents** data=sashelp.\_ALL\_ nods;

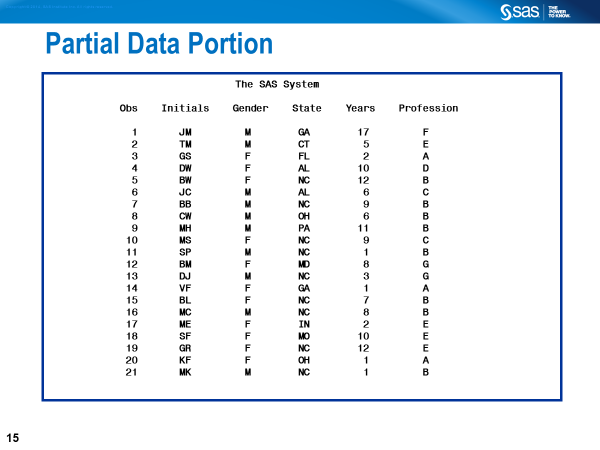
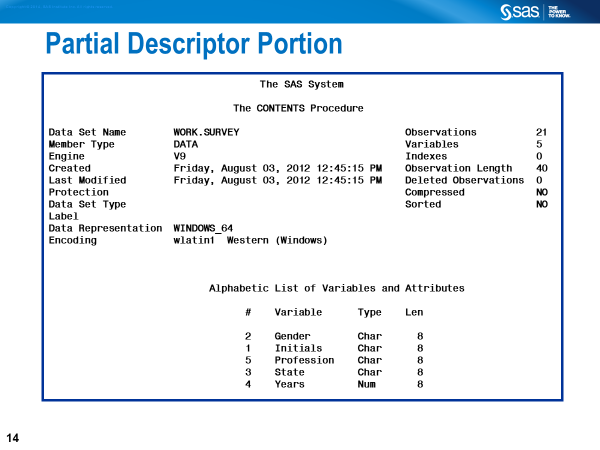
**run**;

**Contents of SAS Data Sets**

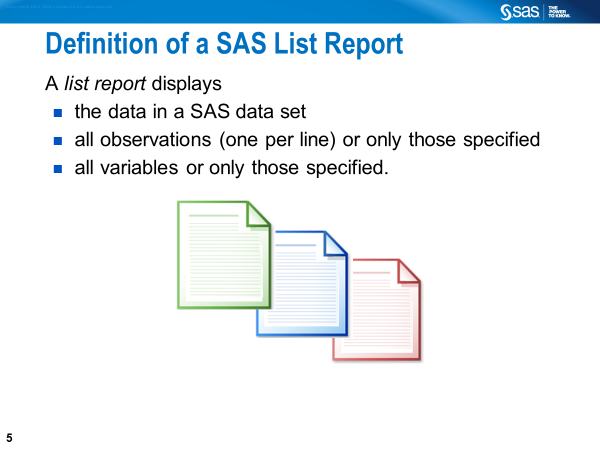
* Descriptor portion - contains information including the
  + - * Data file name
      * member type
      * date and time the data file was created
      * number of observations
      * attributes of the variables
* Data portion - contains the data values in the form of a rectangular table that consists of observations and variables

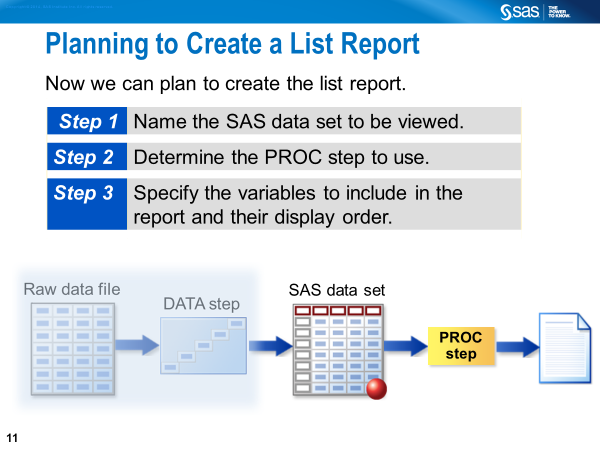


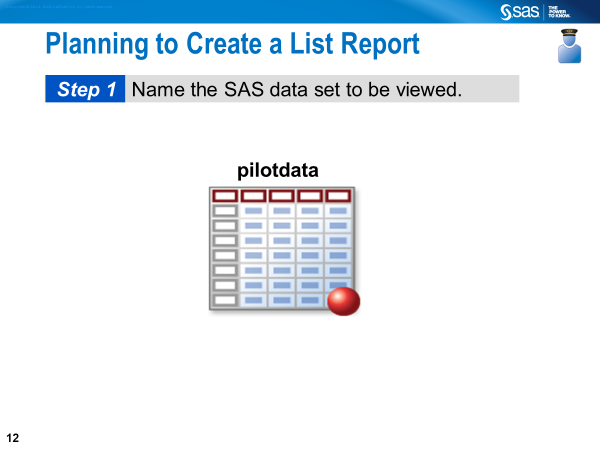
.

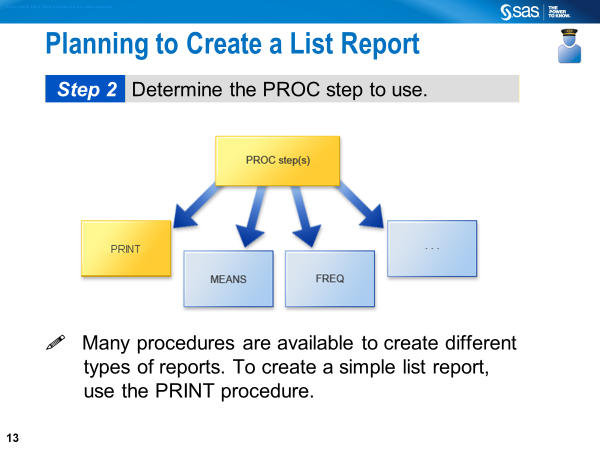


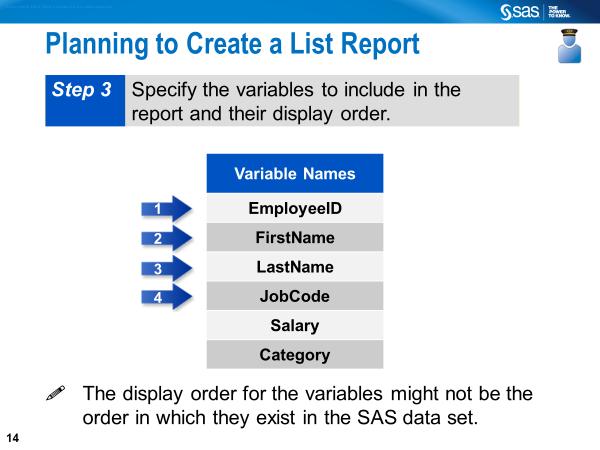
**List Reports**

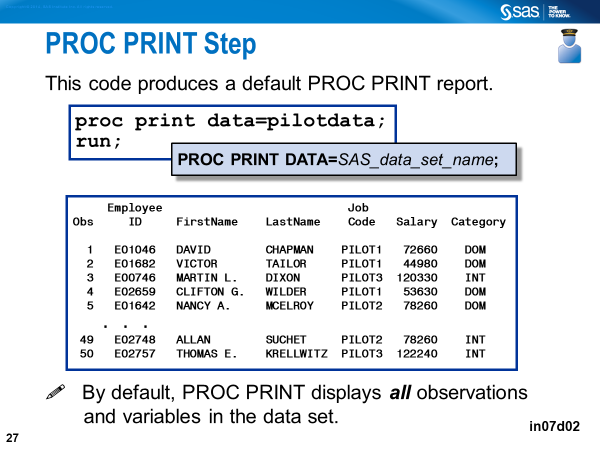
****

****

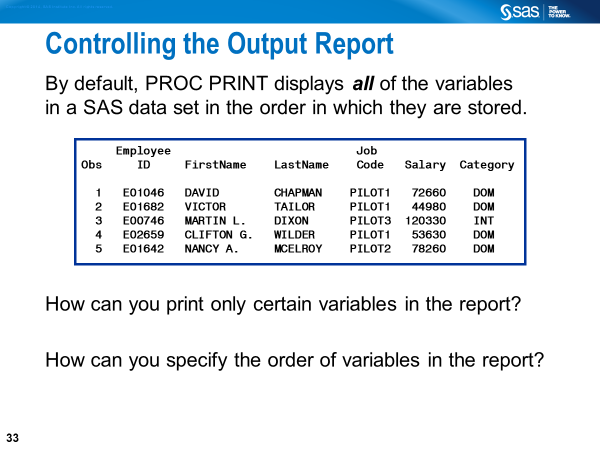
****

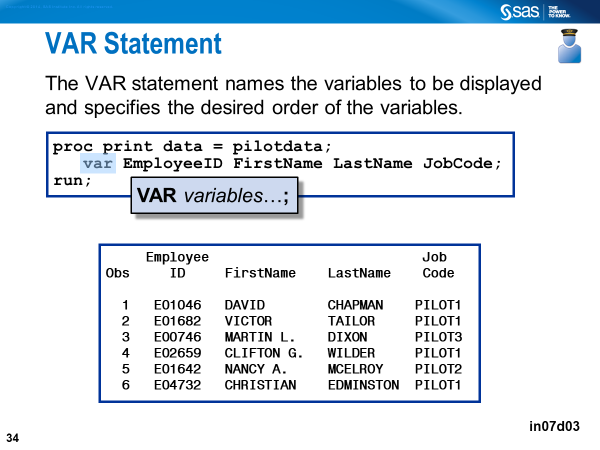
****

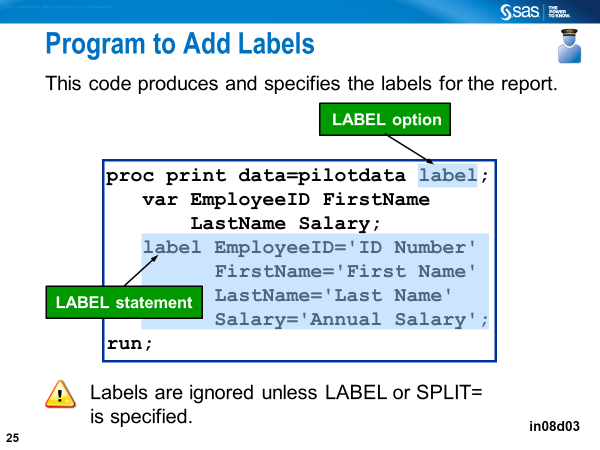
****

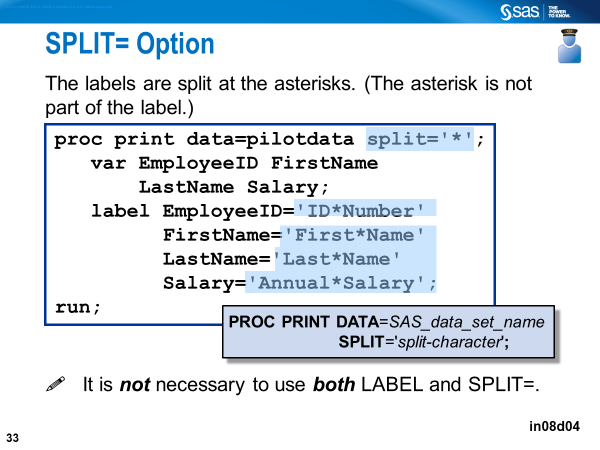
****

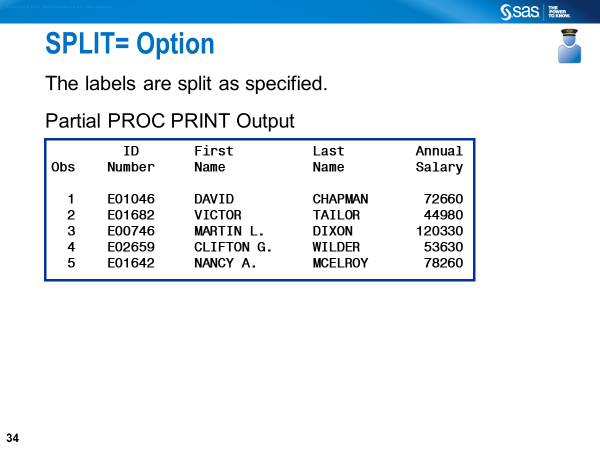
**OBS is a column that is automatically displayed in a PROC PRINT report. It represents the number of an observation.**

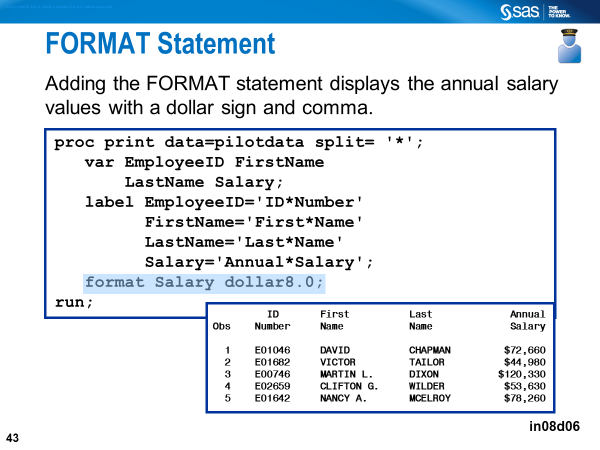
****

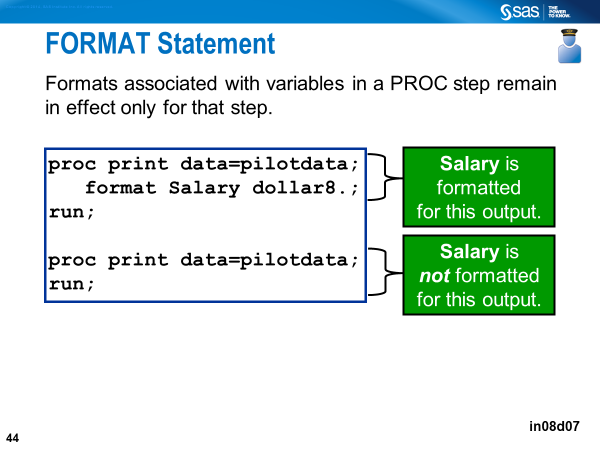
****

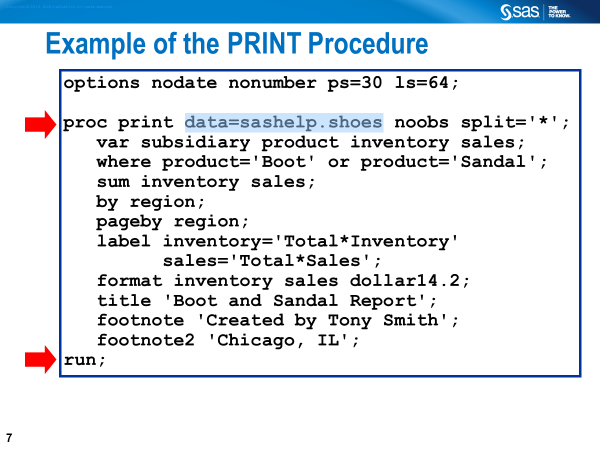
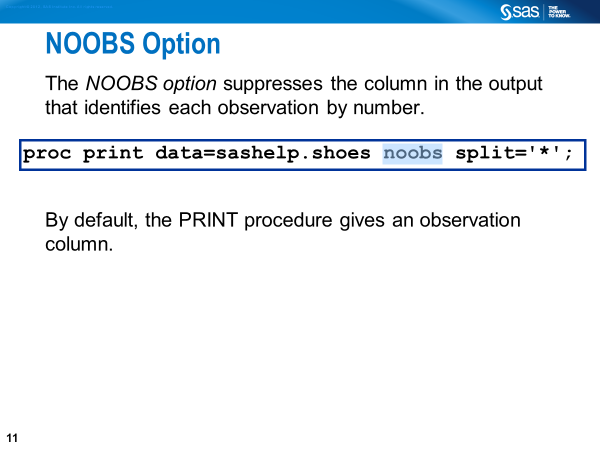
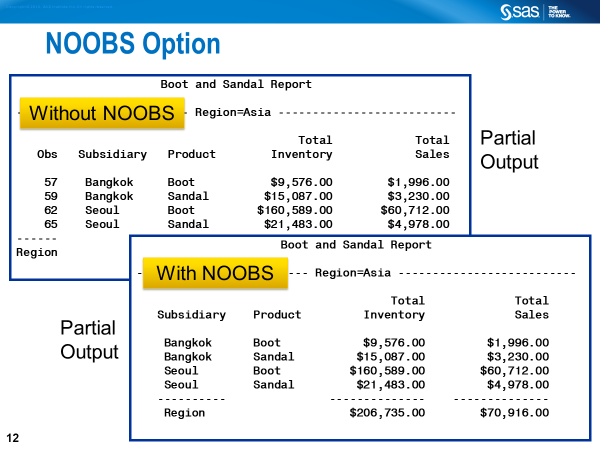


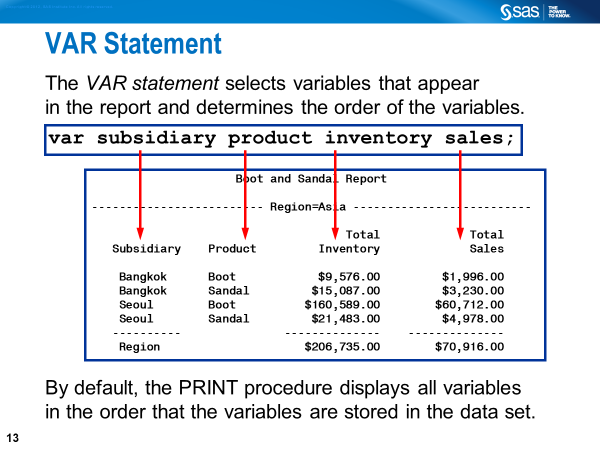


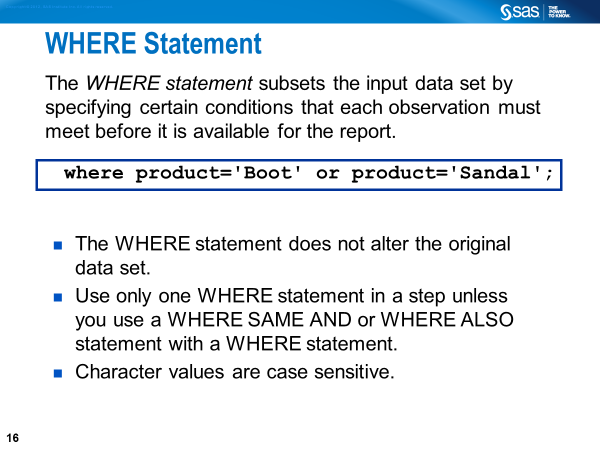
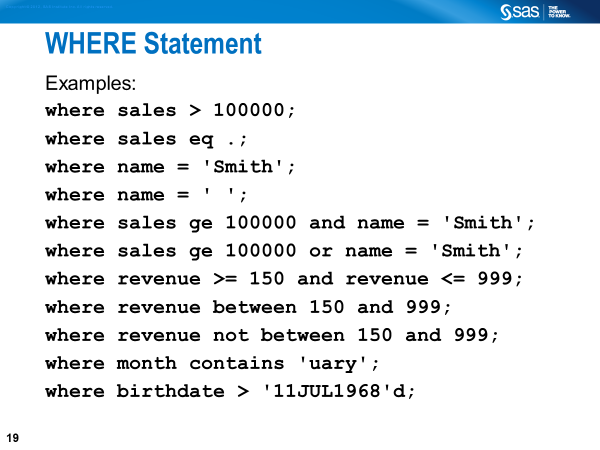
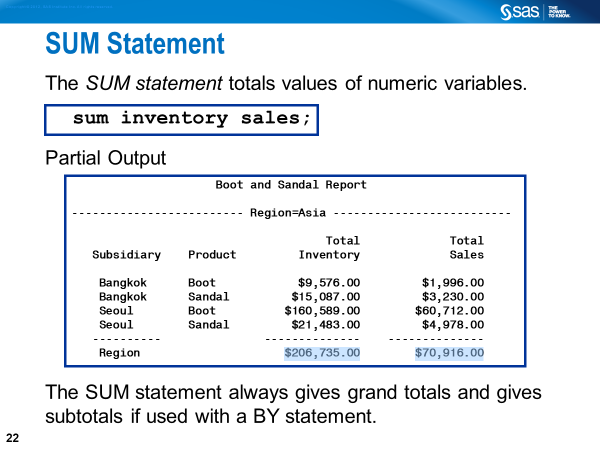


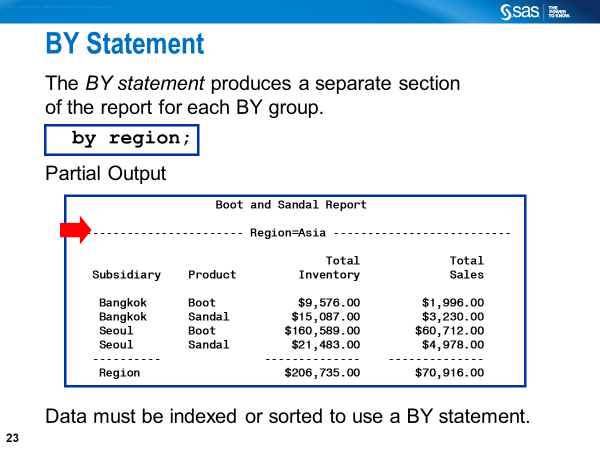




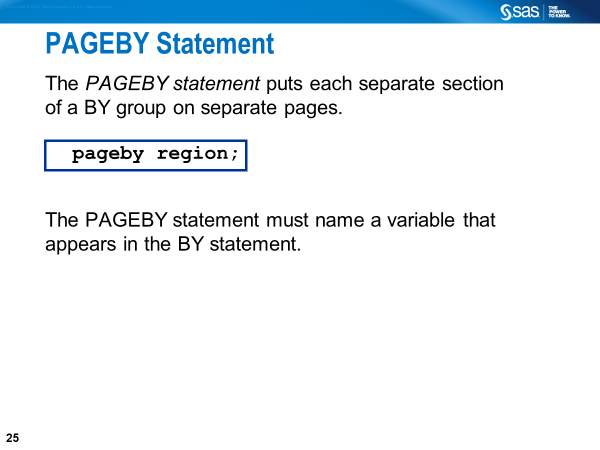
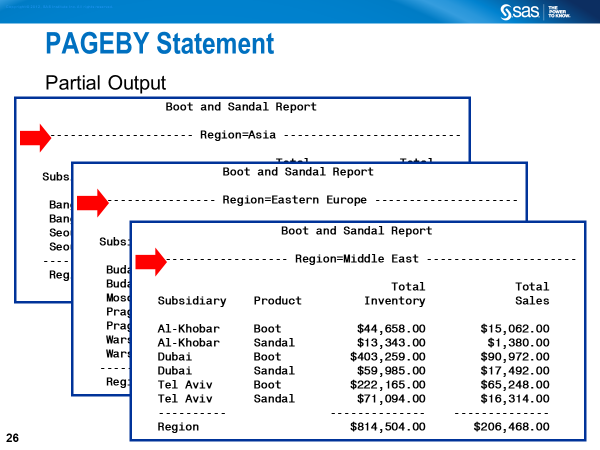
  

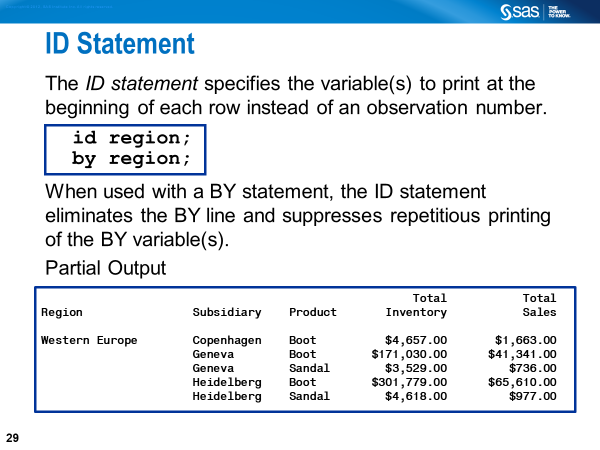
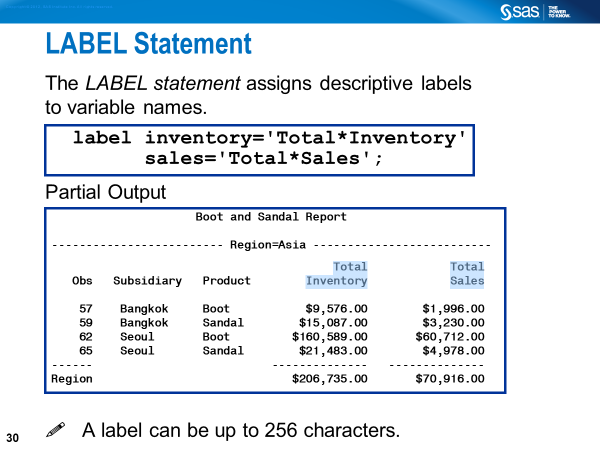


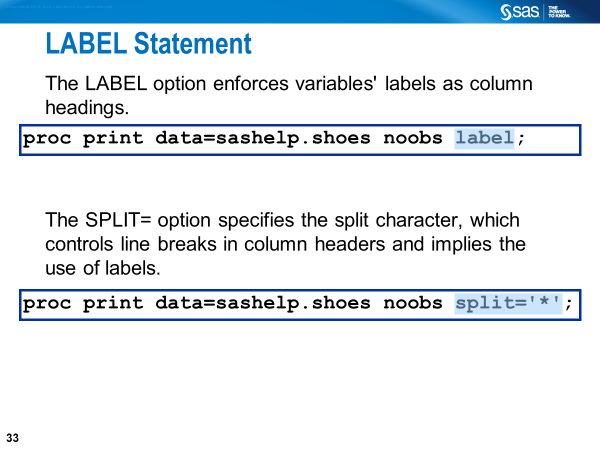
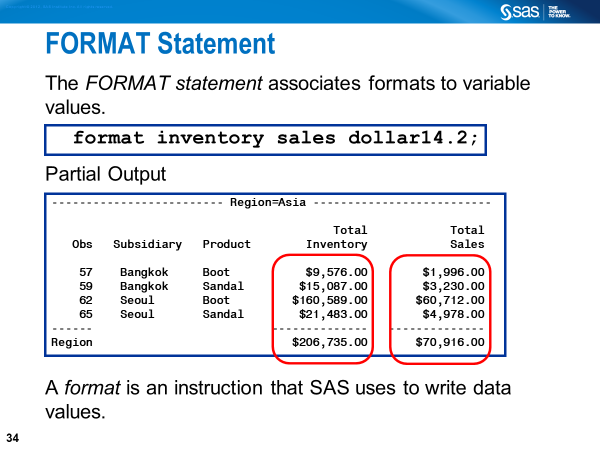
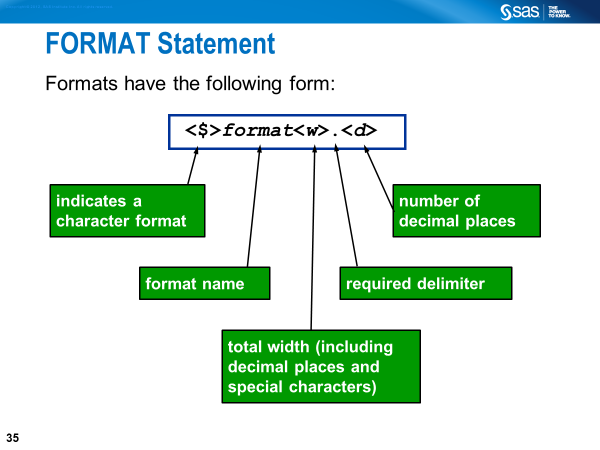
  

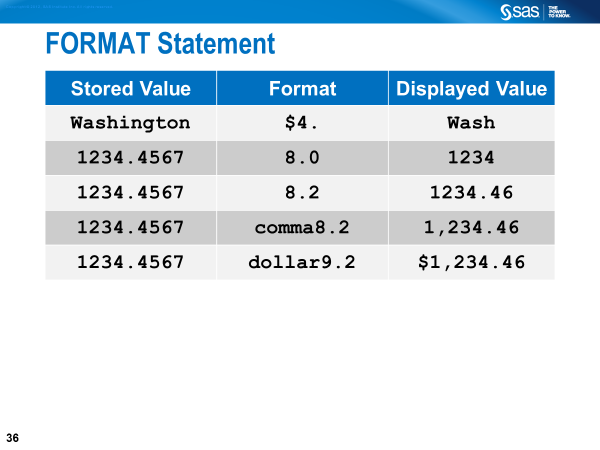
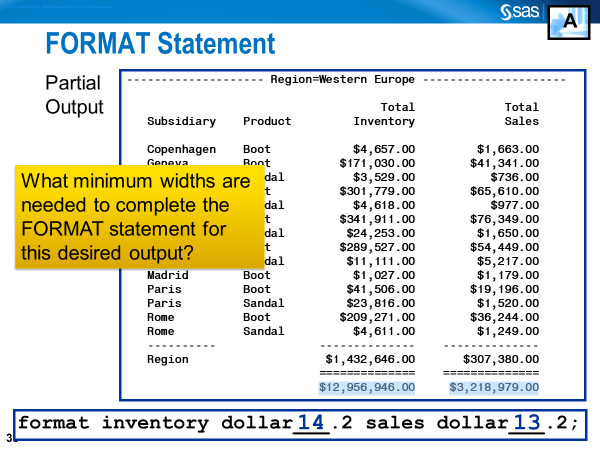


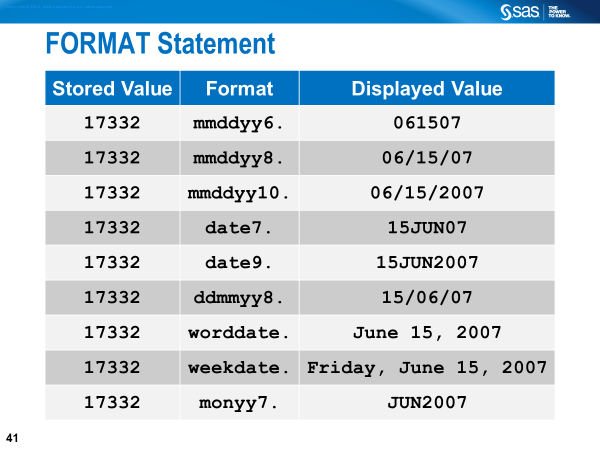


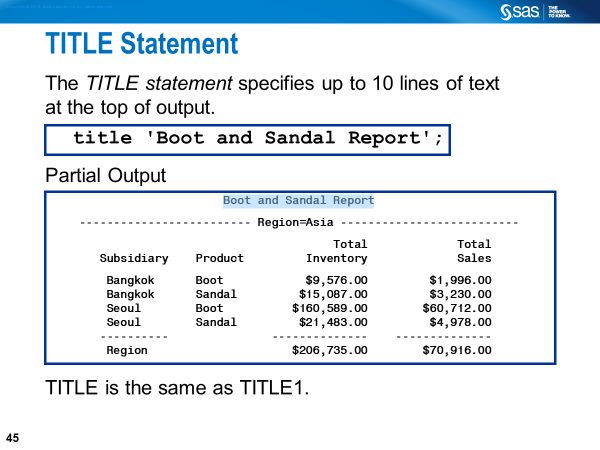
 

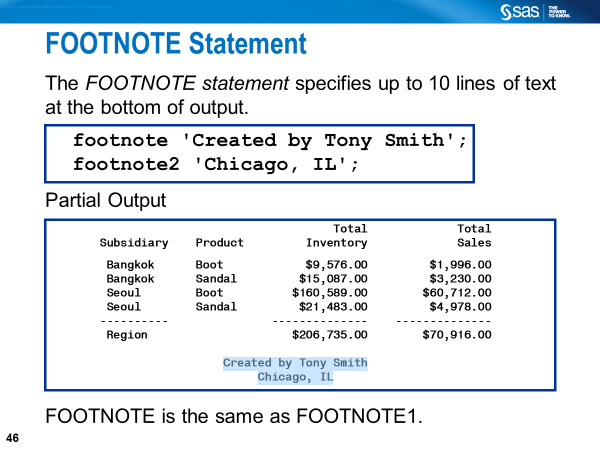
 

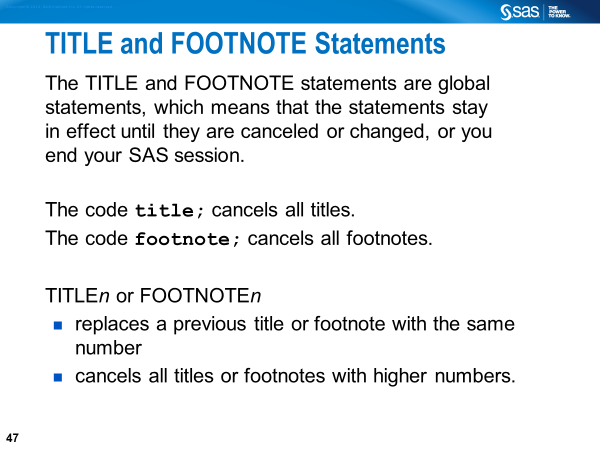
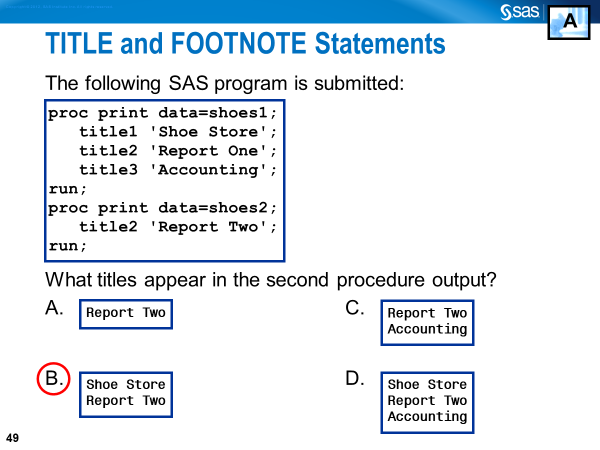
  





SAS User Resources

1. For starters, SAS documentation has a new location on the web: [http://documentation.sas.com](http://documentation.sas.com/)
2. SAS Help Menu in Display Manager
3. Free e-learning SAS courses: <https://support.sas.com/edu/elearning.html?ctry=us&productType=library>
4. SAS blogs (SAS Programming): <https://blogs.sas.com/content/tag/sas-programming/>
5. SAS Support Communities: <https://communities.sas.com/>
6. SAS-L: <http://www.sascommunity.org/wiki/SAS-L>
7. SAS Global Forum Papers: <https://lexjansen.com/>