The George Washington University
Department of Statistics

STAT 6197 – Spring 2019

Week 1 – January 18, 2019

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
8. List Reports
9. SAS User Resources

Readings:

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

2. Exercises from Relevant Chapters/Sections - Ottesen RA, Delwiche LD, and Slaughter SJ. *Exercises and Projects for The Little SAS Book*, Fifth Edition Paperback – July 1, 2015

3. SAS® 9.4 Language Reference: Concepts, Sixth Edition (here)

4. THE SAS SUPERVISOR. By Don Henderson & Merry Rabb ORI, Inc. (here)

Statistical Analysis System® or SAS® - An Overview

What Is SAS?

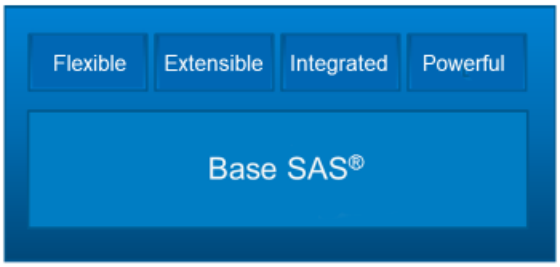
SAS is a suite of business solutions and technologies to help organizations solve business problems.

sas

4

What Is Base SAS?

Base SAS is the centerpiece of all SAS software.



Base SAS is a product within the SAS Foundation set of products. Base SAS provides the following:

- a highly flexible, highly extensible fourth-generation programming language
- a rich library of encapsulated programming procedures
- a graphical user interface for administering SAS tasks

6

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

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.

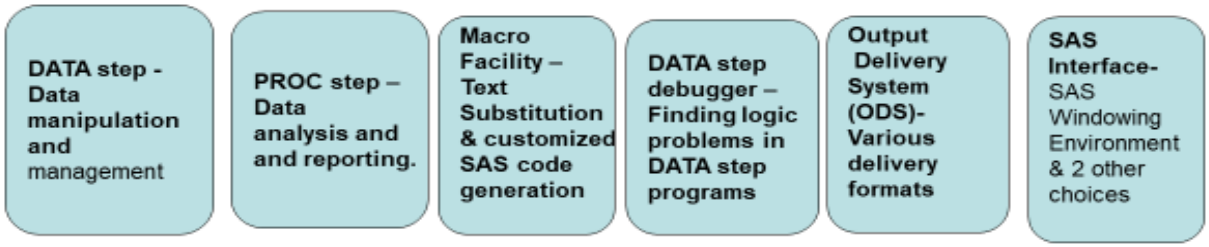
- **the SAS language**
- **Macro**
- **SQL**
- SCL
- SAS/C
- **IML**
- SAS Viya (Cloud Analytics Services(CAS) programming language)

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

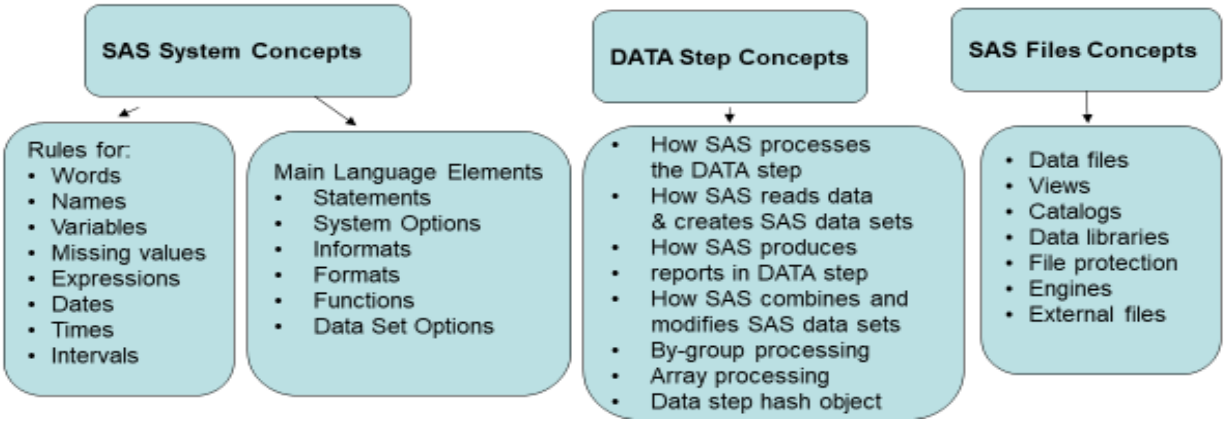
To see what products have been installed in your computer, run the following SAS code: **`PROC PRODUCT_STATUS; RUN;`**

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

Overview of Base SAS Software



Base SAS Software – Conceptual Information



Ways to Run SAS

Processing Modes

The following are two possible processing modes for submitting a SAS program:

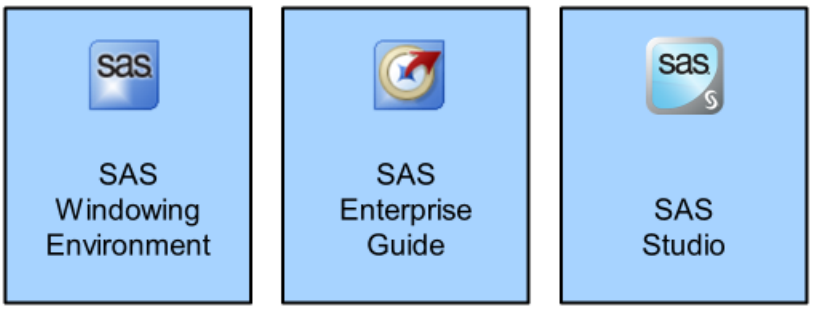
Interactive Mode	A SAS program is submitted within a SAS interface for foreground processing.
Batch Mode	A SAS program is submitted to the operating environment for background processing.

- In this course, interactive mode is used to process SAS programs.

13

SAS Interfaces for Interactive Processing

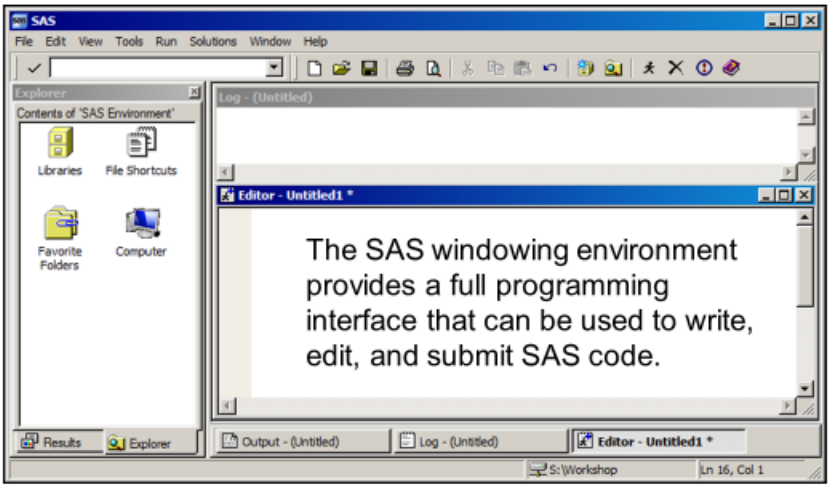
There are three SAS interfaces available for processing a SAS program in interactive mode.



14

SAS Windowing Environment

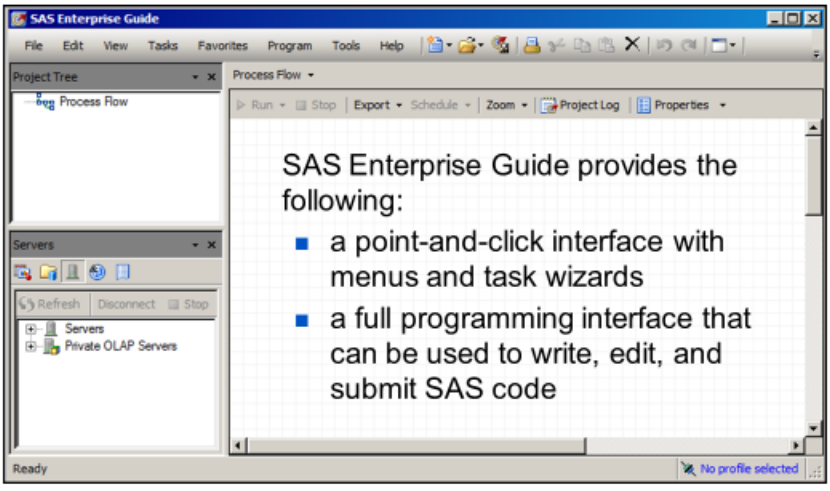
The *SAS windowing environment* is an application that is accessed from different operating environments.



16

SAS Enterprise Guide

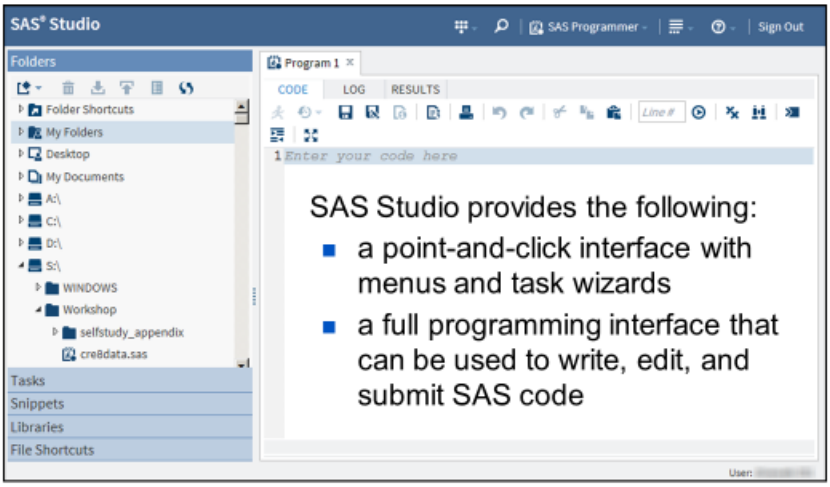
SAS Enterprise Guide is a client application that is accessed from the Windows operating environment.



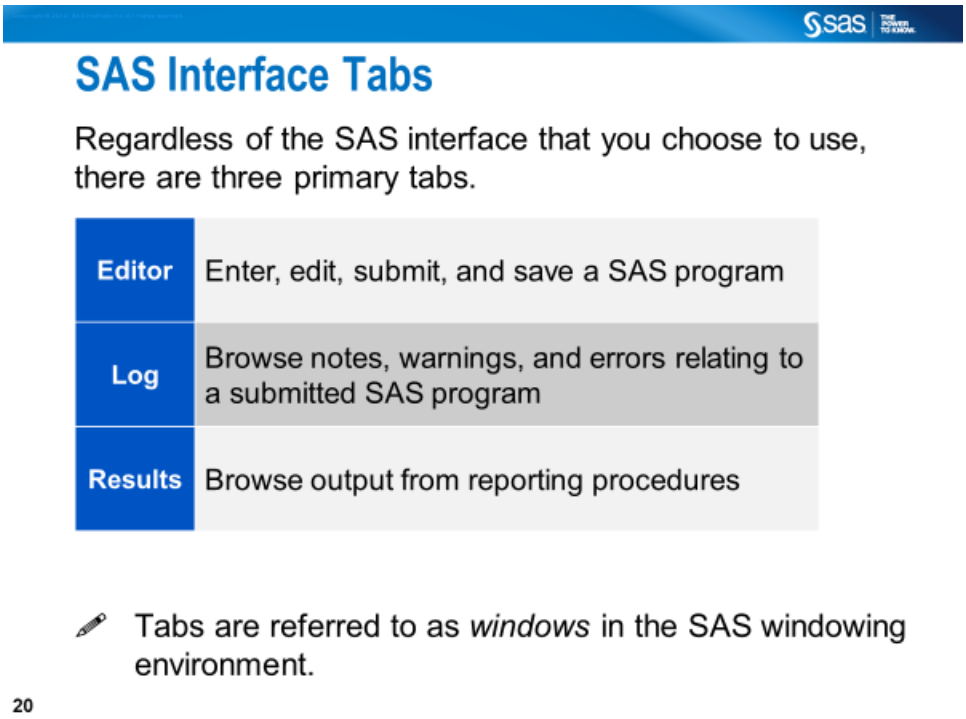
17

SAS Studio

SAS Studio is a web client that is accessed through an HTML5-compliant web browser.



18



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 obtain 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

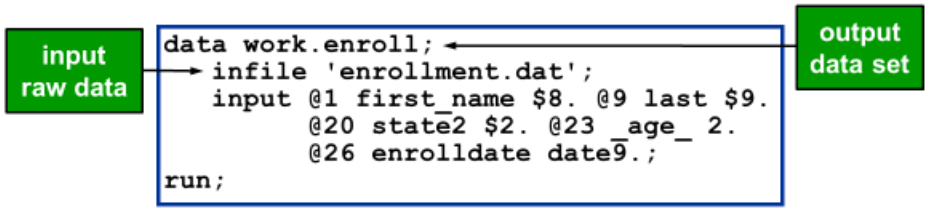
A SAS program, which is saved as text files with an extension sas, can be any combination of at least the following:

- DATA step
- PROC step
- Global statement
- SAS macro language code

DATA Step

In general, the DATA step manipulates data.

- The input for a DATA step can be of several types, such as raw data or a SAS data set.
- The output from a DATA step can be of several types, such as a SAS data set or a report.

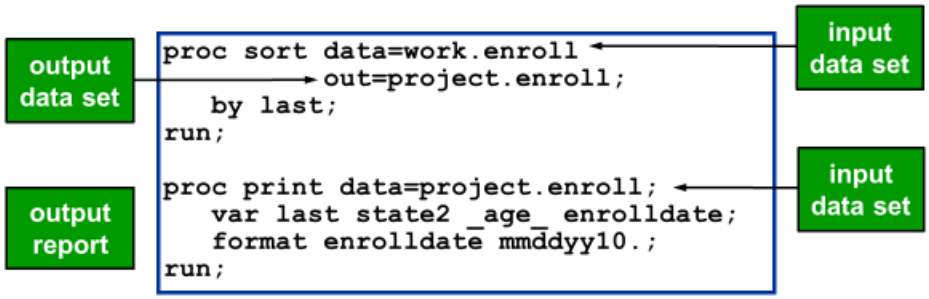


- 38

PROC Step

In general, the PROC step analyzes data, produces output, or manages SAS files.

- The input for a PROC step is usually a SAS data set.
- The output from a PROC step can be of several types, such as a report or an updated SAS data set.



- 41

Editing SAS Code

After you execute a SAS program, you might have to edit the code because of the following:

- program errors
- program specification changes
- the need to add extra code

17

Program Errors

A program might not run successfully, or at all, due to program errors.

Type of error	Occurs when…	Example
Typographical	File, variable, or other names are misspelled.	**pilt.dat** instead of **pilot.dat**
Syntax	Program statements do not conform to the rules.	Misspelling a SAS keyword or forgetting a semicolon
Logical	Specified actions to be carried out by a program are inconsistent, ineffective, or incorrect.	Multiplying instead of dividing

18

SAS Log

The SAS log is a record of your submitted SAS program.

```
125  libname project 'C:\workshop\winsas\lwcrb\data';

NOTE: Libref PROJECT was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\workshop\winsas\lwcrb\data

126  proc sort data=work.enroll
127            out=project.enroll;
128     by last;
129  run;

NOTE: There were 4 observations read from the data set WORK.ENROLL.
NOTE: The data set PROJECT.ENROLL has 4 observations and 5 variables.
```

- Original program statements are identified by line numbers.
- SAS messages can include the words NOTE, INFO, WARNING, or ERROR.

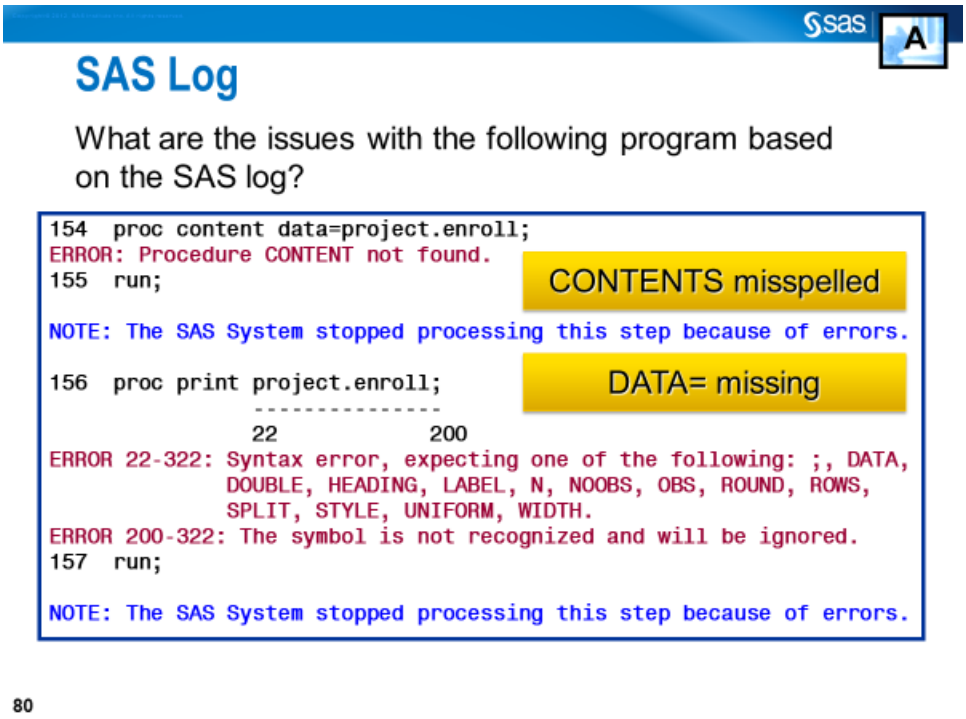
78

SAS Log

What are the issues with the following program based on the SAS log?

```
154  proc content data=project.enroll;
ERROR: Procedure CONTENT not found.
155  run;

NOTE: The SAS System stopped processing this step because of errors.

156  proc print project.enroll;
                ---------------
                22             200
ERROR 22-322: Syntax error, expecting one of the following: ;, DATA,
              DOUBLE, HEADING, LABEL, N, NOOBS, OBS, ROUND, ROWS,
              SPLIT, STYLE, UNIFORM, WIDTH.
ERROR 200-322: The symbol is not recognized and will be ignored.
157  run;

NOTE: The SAS System stopped processing this step because of errors.
```

79



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.

Run this SAS program: **Ex13_Syntax_Errors.sas.**

Exploring SAS Programs

```
*Ex1_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

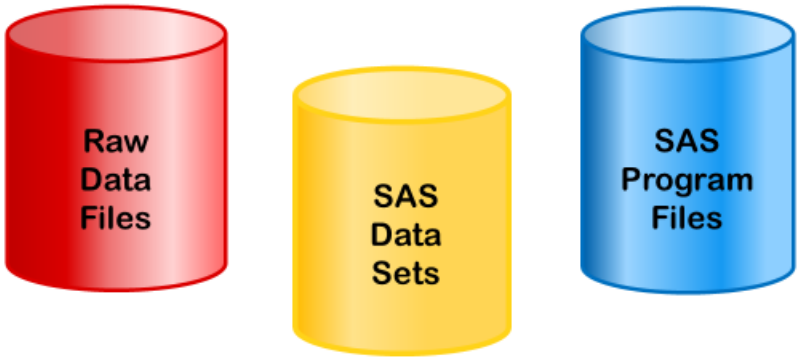
- Statement-type comment
- Delimited comment
- Block-type comment

```
1  *Ex2_Comments.sas;
2  *This is a statement-style comment;
3  /*This is a delimited comment. */
4  /**This is a block-type comment*********
5  Program Name:
6  Purpose
7  Author:
8  Date:
9  ****************************************/
```

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 2: Statement-style comments.
- Line 3: Delimited type comment.
- Lines 4-9: Block type comment.

Types of Files Used in SAS



4

Raw Data Files

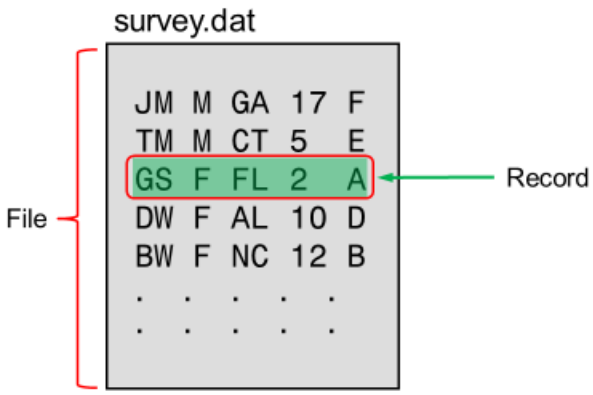
Raw data files

- are non-software-specific files that contain records and fields
- can be created by a variety of software products
- can be read by a variety of software products
- consist of no special attributes, such as field headings, page breaks, or titles.

5

Raw Data Files

The file consists of *records*.



6

SAS Data Sets

SAS data sets

- are files specific to SAS that contain variables and observations
- are read only by SAS
- consist of a descriptor portion and a data portion
- are temporary files as used throughout this course.

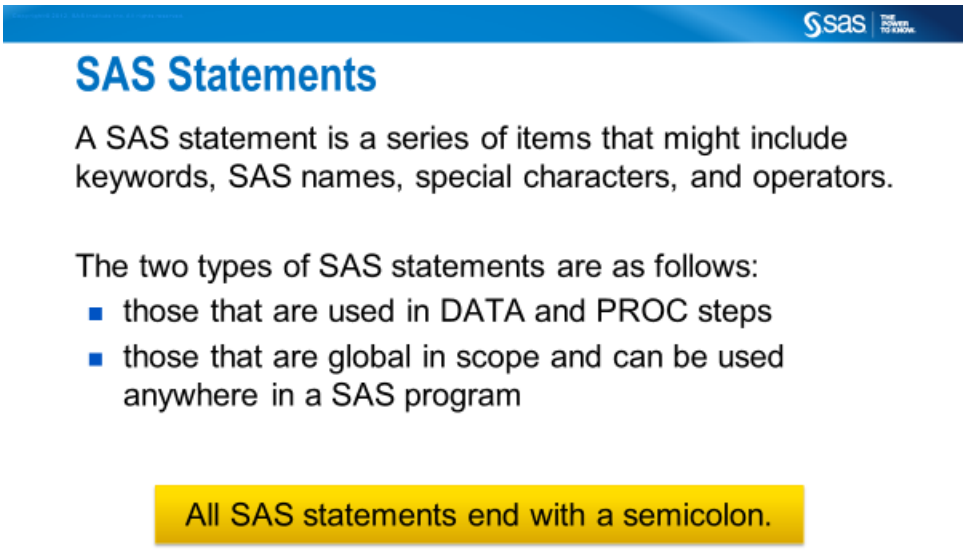
12

Terminology

Here is a comparison of data processing and SAS terms.

Data Processing	SAS
file	SAS data set or SAS table
record	observation or row
field	variable or column

16



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

Global Statements

Global statements

- are used anywhere in a SAS program
- stay in effect until changed or canceled, or until you end your SAS session.

```
libname project 'C:\workshop\winsas\lwcrb';

proc sort data=work.enroll
          out=project.enroll;
   by last;
run;
```

global statement

45

Global Statements

What are some additional examples of global statements?

1. DATA
2. TITLE
3. LABEL
4. FORMAT
5. OPTIONS
6. FOOTNOTE

47

Giving SAS Variable Names to Fields

SAS variable names

- are 1 to 32 characters in length
- start with a letter (A through Z) or an underscore (_)
- continue with any combination of numbers, letters, or underscores
- are ***not*** case sensitive
- must be unique within a SAS data set.

16

Data Types

A SAS data set supports two types of variables.

Character variables

- can contain any value: letters, numerals, special characters, and blanks
- range from 1 to 32,767 characters in length
- have 1 byte per character.

Numeric variables

- store numeric values using floating point or binary representation
- have 8 bytes of storage by default
- can store 16 or 17 significant digits.

9

Variable Types

The two types of SAS variables are listed below:

- character
- numeric

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

three character variables

two numeric variables

62

Variable Types: Character

Character variables are stored with a length of 1 to 32,767 bytes with 1 character equaling 1 byte.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

8 bytes

9 bytes

2 bytes

Character variables can contain letters (A-Z), numeric digits (0-9), and other special characters (_, #, %, &,…).

63

Variable Types: Numeric

Numeric variables are stored as floating-point numbers with a default byte size of 8.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

8 bytes 8 bytes

To be stored as a floating point number, the numeric value can contain numeric digits (0-9), plus or minus sign, decimal point, and E for scientific notation.

64

Standard and Nonstandard Data (Review)

Standard data is data that SAS can read without any additional instruction.

- Character data is always standard.
- Some numeric values are standard and some are not.

Standard Numeric Data

58
67.23 -23
5.67E5 00.99
1.2E-2

Nonstandard Numeric Data

(23) $67.23
5,823 01/12/2010
12May2009

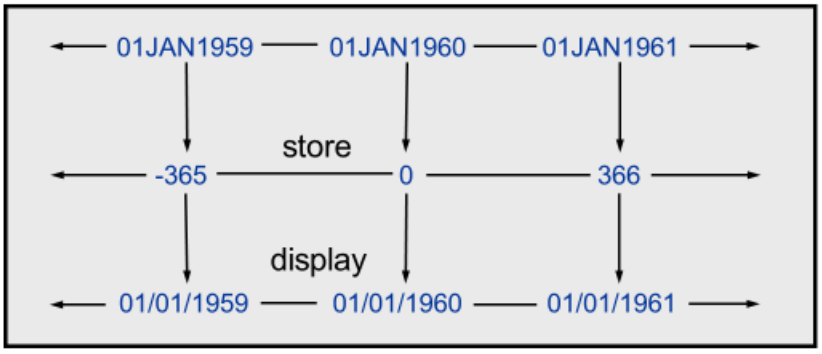
8

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 Date Values

SAS stores calendar dates as numeric values.



A *SAS date value* is stored as the number of days between January 1, 1960, and a specific date.

11

SAS Dates

A SAS date value is a value that represents the number of days between January 1, 1960, and a specified date.

- Dates before January 1, 1960, are negative numbers.
- Dates after January 1, 1960, are positive numbers.

To reference a SAS date value in a SAS program, use a SAS date constant.

- A SAS date constant is a date (DDMMMYYYY) in quotation marks followed by the letter D.
- Example:

```
'12NOV1986'd
```

67

Missing Data

Missing data is a value that indicates that no data value is stored for the variable in the current observation.

- A missing numeric value is displayed as a single period (.).
- A missing character value is displayed as a blank space.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

70

SAS Statements Explained

```
1  * Ex3_DataProcSteps.sas;
2  OPTIONS nocenter nodate nonumber;
3  %LET DateRun=%sysfunc(today(), worddate);
4  DATA work.HAVE2;
5    INPUT Name $ quiz1-quiz3;
6      Ave_Score = ROUND(MEAN(OF quiz1-quiz3),.01);
7      LABEL quiz1 = 'Quiz 1 Score'
8            quiz2 = 'Quiz 2 Score'
9            quiz3 = 'Quiz 3 Score'
10           Ave_Score = 'Average Score';
11   DATALINES;
12   Amy  78 84 82
13   Neil 90 85 86
14   John 82 79 89
15   Keya 78 86 78
16   ;
17 title "Listing from HAVE2 SAS Data File - &DateRun";
18 PROC PRINT data=work.HAVE2 noobs label;
19 run;
```

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)

```
1  * Ex3_DataProcSteps.sas;
2  OPTIONS nocenter nodate nonumber;
3  %LET DateRun=%sysfunc(today(), worddate);
4  DATA work.HAVE2;
5    INPUT Name $ quiz1-quiz3;
6      Ave_Score = ROUND(MEAN(OF quiz1-quiz3),.01);
7      LABEL quiz1 = 'Quiz 1 Score'
8            quiz2 = 'Quiz 2 Score'
9            quiz3 = 'Quiz 3 Score'
10           Ave_Score = 'Average Score';
11   DATALINES;
12   Amy  78 84 82
13   Neil 90 85 86
14   John 82 79 89
15   Keya 78 86 78
16   ;
17 title "Listing from HAVE2 SAS Data File - &DateRun";
18 PROC PRINT data=work.HAVE2 noobs label;
19 run;
```

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)

```
1  * Ex3_DataProcSteps.sas;
2  OPTIONS nocenter nodate nonumber;
3  %LET DateRun=%sysfunc(today(), worddate);
4  DATA work.HAVE2;
5   INPUT Name $ quiz1-quiz3;
6     Ave_Score = ROUND(MEAN(OF quiz1-quiz3),.01);
7     LABEL quiz1 = 'Quiz 1 Score'
8           quiz2 = 'Quiz 2 Score'
9           quiz3 = 'Quiz 3 Score'
10          Ave_Score = 'Average Score';
11  DATALINES;
12  Amy  78 84 82
13  Neil 90 85 86
14  John 82 79 89
15  Keya 78 86 78
16  ;
17 title "Listing from HAVE2 SAS Data File - &DateRun";
18 PROC PRINT data=work.HAVE2 noobs label;
19 run;
```

Lines 7-10: The LABEL statement 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 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)

```
1  * Ex3_DataProcSteps.sas;
2  OPTIONS nocenter nodate nonumber;
3  %LET DateRun=%sysfunc(today(), worddate);
4  DATA work.HAVE2;
5    INPUT Name $ quiz1-quiz3;
6      Ave_Score = ROUND(MEAN(OF quiz1-quiz3),.01);
7      LABEL quiz1 = 'Quiz 1 Score'
8            quiz2 = 'Quiz 2 Score'
9            quiz3 = 'Quiz 3 Score'
10           Ave_Score = 'Average Score';
11   DATALINES;
12   Amy  78 84 82
13   Neil 90 85 86
14   John 82 79 89
15   Keya 78 86 78
16   ;
17 title "Listing from HAVE2 SAS Data File - &DateRun";
18 PROC PRINT data=work.HAVE2 noobs label;
19 run;
```

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= 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.

Run this program: **Ex12_Data_step_without_datalines.sas**

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

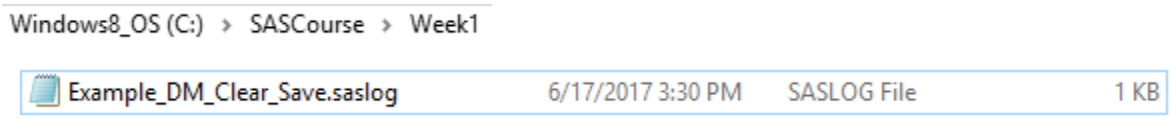
Saving a Log File Automatically

Ex4_DM_Clear.sas

```
*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

```
 1  *Ex5_Proc_Printto.sas;
 2  options symbolgen nocenter nodate nonumber;
 3  DM 'log;clear;output;clear odsresults; clear';
 4  FILENAME MYLOG 'C:\SASCourse\Week1\PP_log.TXT';
 5  FILENAME MYPRINT 'C:\SASCourse\Week1\PP_OUTPUT.TXT';
 6  PROC PRINTTO LOG=MYLOG PRINT=MYPRINT NEW;
 7  RUN;
 8  TITLE 'Listing from SASHELP.CLASS';
 9  PROC PRINT data=sashelp.class;
10  RUN;
11  PROC PRINTTO;
12  RUN;
```

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 it into 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 (Obtained from here)

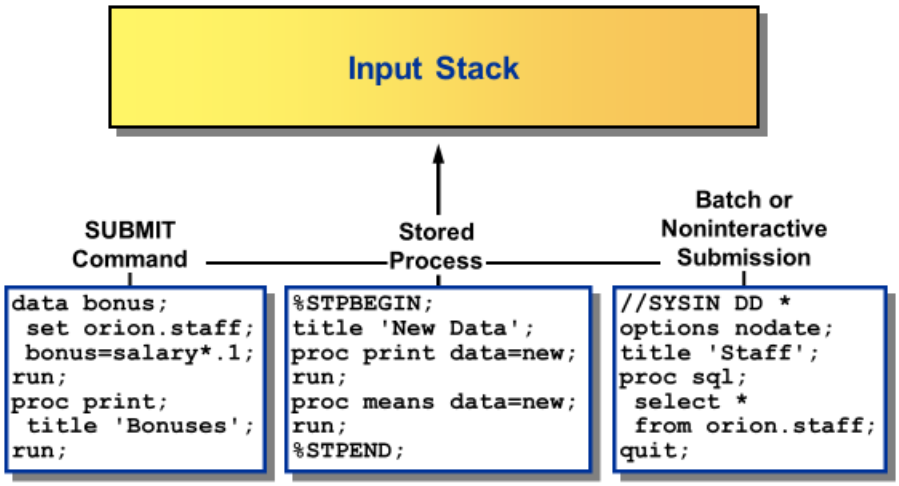
• variables	• SAS data sets	• formats or informats	• SAS procedures
• options	• arrays	• statement labels	• SAS macros or macro variables
• SAS catalog entries	• Librefs	• filerefs	• component objects

There are two types of SAS statements that are used in a data step:

- declarative statements that provide information and do their work during the compilation phase (e.g., ARRAY, BY, DROP, FORMAT/INFORMAT, KEEP, LABEL, LENGTH, RENAME, RETAIN)

- executable statements that result in some action during the individual iteration of the data step (e.g., ABORT, CALL, CONTINUE, DELETE, DESCRIBE, DISPLAY, DO, DO UNTIL, DO WHILE, ERROR, EXECUTE, FILE, IF-THEN/ELSE, INPUT, INFILE, GO TO, LEAVE, LINK, LIST, LOSTCARD, MERGE, MODIFY, OUTPUT, PUT, REDIRECT, REMOVE, REPLACE, RETURN, MERGE, RETURN, SELECT, SET, STOP, and UPDATE)

Program Flow

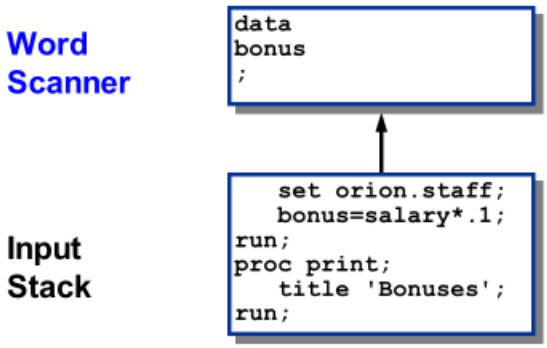


39

Program Flow

When SAS code is in the input stack, a component of SAS called the *word scanner* does the following:

- reads the text in the input stack, character by character, left to right, top to bottom
- breaks the text into fundamental units called *tokens*

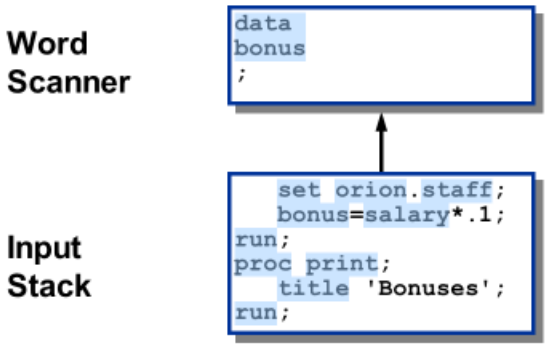


40

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens

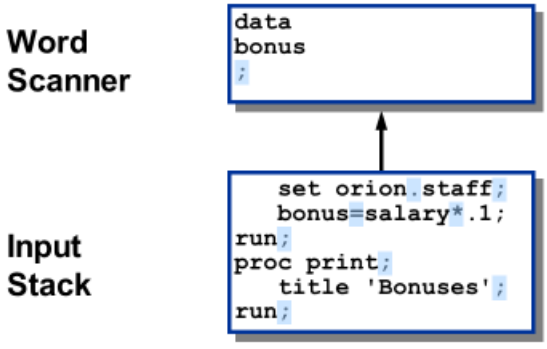


42 ...

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens

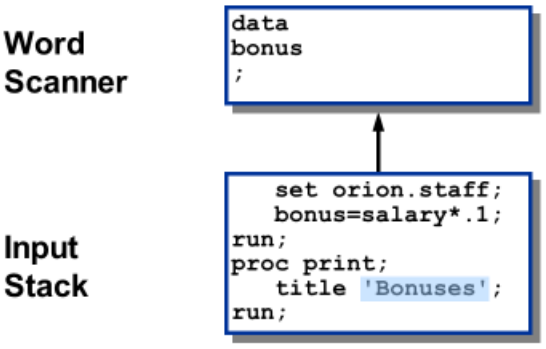


43 ...

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens
- literal tokens



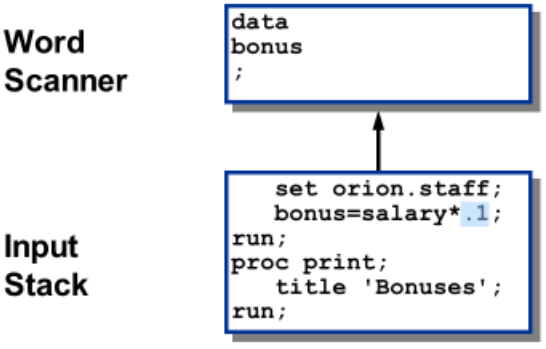
44

...

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens
- literal tokens
- number tokens

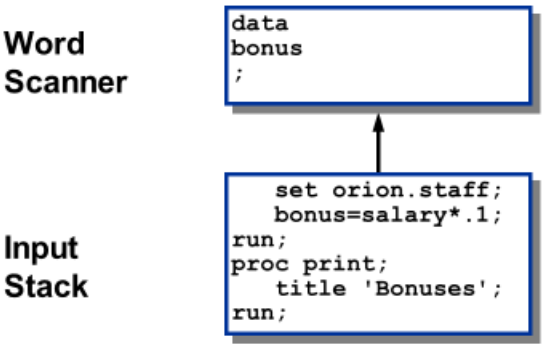


45

Tokenization

A token ends when the word scanner detects

- a blank
- the beginning of another token.

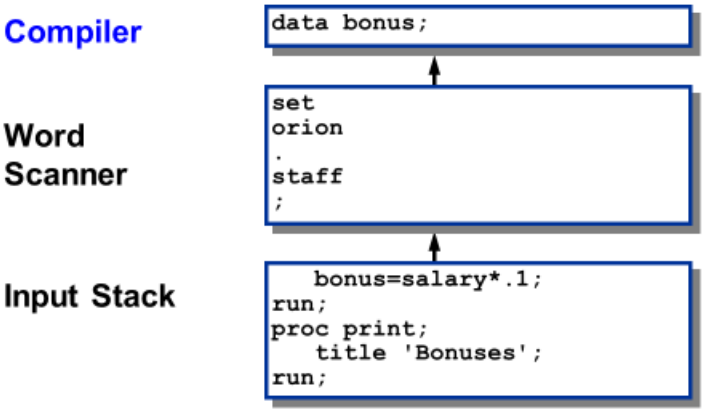


The maximum length of a token is 32,767 characters.

50

Program Flow

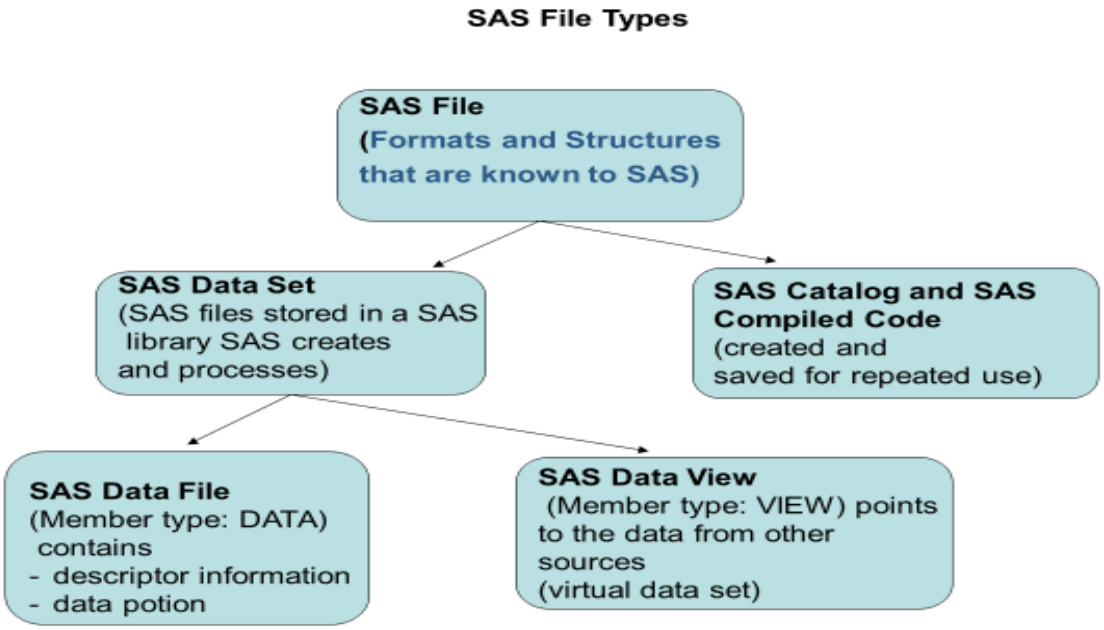
The word scanner passes tokens, one at a time, to the appropriate compiler, as the compiler demands.



52

Run this SAS program: **Ex6_Conpilation_Execution.sas.**

SAS Files Concepts



Note: The term SAS data set is used when a SAS view or a SAS data file can be used in the same manner.

SAS Data Sets

A SAS data set has these characteristics:

- is a SAS file stored in a SAS library that SAS creates and processes
- contains data values that are organized as a table of observations (rows) and variables (columns)
- contains descriptor information such as the data types and lengths of the variables

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

48

SAS Data Sets

SAS data set names

- are 1 to 32 characters in length
- start with a letter (A through Z) or an underscore (_)
- continue with any combination of numbers, letters, or underscores
- can have two levels (for example, **work.survey**)
- are ***not*** case sensitive.

19

Temporary and Permanent SAS Data Sets

A *temporary* SAS data set is one that exists only for the current SAS session or job.

- The **Work** library is a temporary data library.
- Data sets held in the **Work** library are deleted at the end of the SAS session.

A *permanent* SAS data set is one that resides on the external storage medium of your computer and is not deleted when the SAS session terminates.

- Any data library referenced with a LIBNAME statement is considered a permanent data library by default.

57

SAS Libraries

A SAS library is a collection of one or more SAS files, including SAS data sets, that are referenced and stored as a unit.

- In a directory-based operating environment, a SAS library is a group of SAS files that are stored in the same directory.
- In z/OS (OS/390), a SAS library is a group of SAS files that are stored in an operating environment file.

49

SAS Libraries

A logical name (*libref*) can be assigned to a SAS library using the LIBNAME statement.

```
libname project 'C:\workshop\winsas\lwcrb';
```

libref — physical location of SAS library

The *libref*

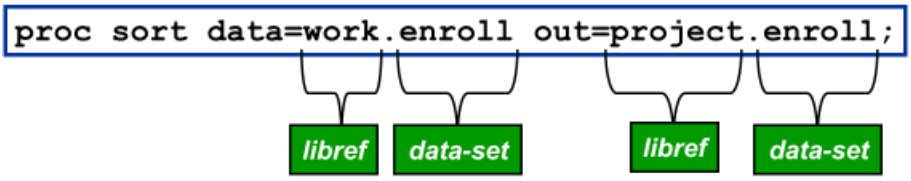
- can be up to 8 characters long
- must begin with a letter (A-Z) or an underscore (_)
- can contain only letters, digits (0-9), or underscores.

50

"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.

Two-Level SAS Data Set Names

A SAS data set can be referenced using a two-level SAS data set name.



- *libref* is the logical name that is associated with the physical location of the SAS library.
- *data-set* is the data set name, which can be up to 32 characters long, must begin with a letter or an underscore, and can contain letters, digits, and underscores.

53

One-Level SAS Data Set Names

A data set referenced with a one-level name is automatically assigned to the **Work** library by default.

For example, the following two statements are equivalent:

```
proc sort data=work.enroll out=project.enroll;
```

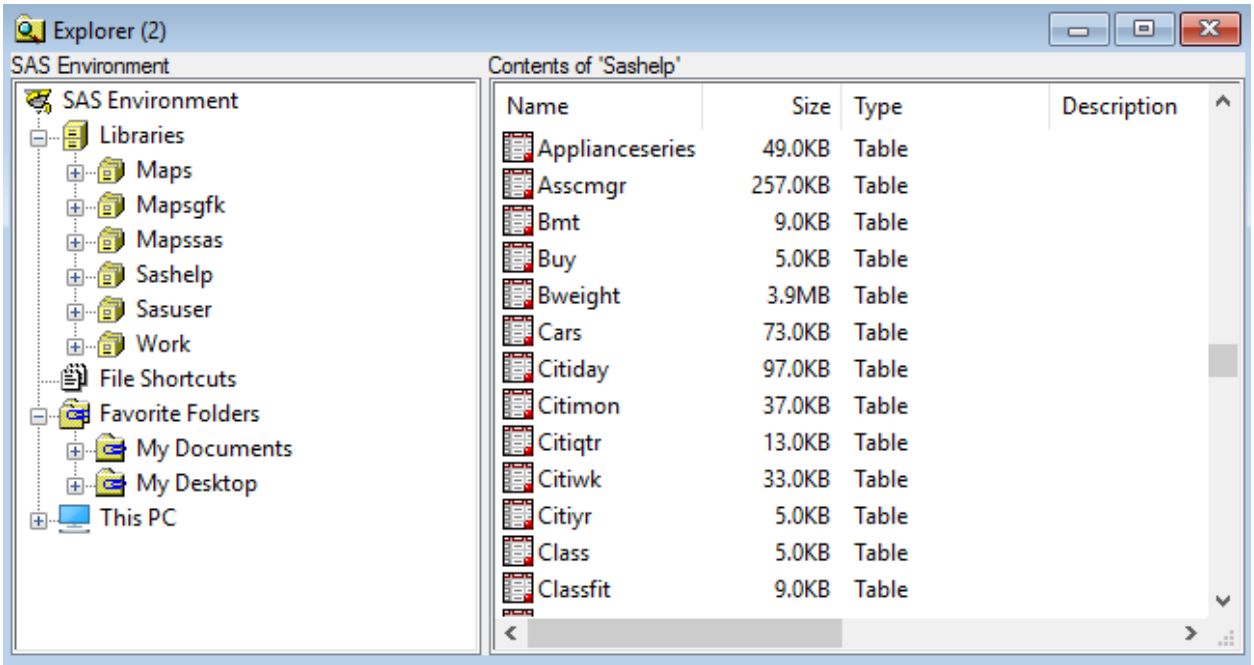
```
proc sort data=enroll out=project.enroll;
```

54

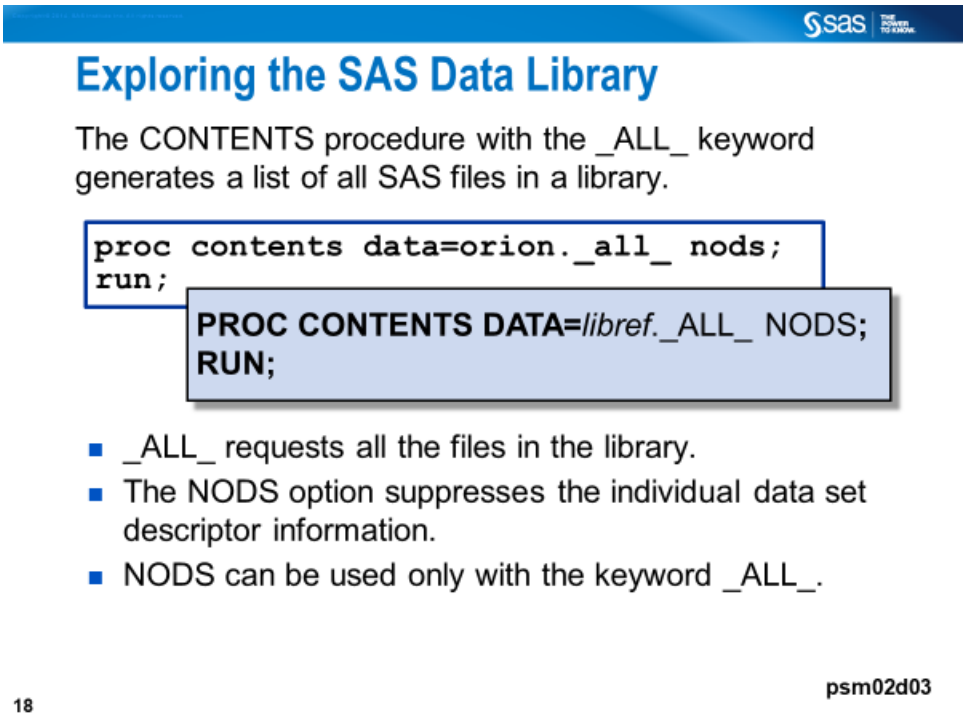
Run this program: **Ex7_Referencing_SAS_Data_Sets.sas.**

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.



Run this program: `Ex14_Pathname_Library.sas.`



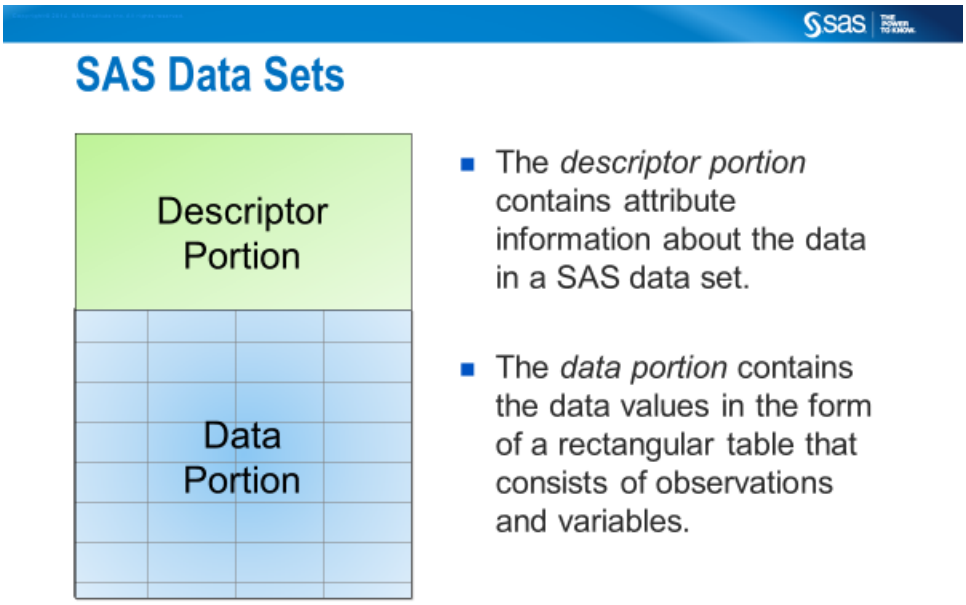
The following example code will generate listing of the contents of the SAS library.

```
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



Run these SAS programs:

Ex8_Contents_all_ods.sas

Ex9_Contents_many_ways.sas

Ex10_Data_step_view_etc.sas

Partial Descriptor Portion

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.SURVEY	Observations	21
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Friday, August 03, 2012 12:45:15 PM	Observation Length	40
Last Modified	Friday, August 03, 2012 12:45:15 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	Gender	Char	8
1	Initials	Char	8
5	Profession	Char	8
3	State	Char	8
4	Years	Num	8

14

Partial Data Portion

The SAS System

Obs	Initials	Gender	State	Years	Profession
1	JM	M	GA	17	F
2	TM	M	CT	5	E
3	GS	F	FL	2	A
4	DW	F	AL	10	D
5	BW	F	NC	12	B
6	JC	M	AL	6	C
7	BB	M	NC	9	B
8	CW	M	OH	6	B
9	MH	M	PA	11	B
10	MS	F	NC	9	C
11	SP	M	NC	1	B
12	BM	F	MD	8	G
13	DJ	M	NC	3	G
14	VF	F	GA	1	A
15	BL	F	NC	7	B
16	MC	M	NC	8	B
17	ME	F	IN	2	E
18	SF	F	MO	10	E
19	GR	F	NC	12	E
20	KF	F	OH	1	A
21	MK	M	NC	1	B

15

List Reports

Definition of a SAS List Report

A *list report* displays

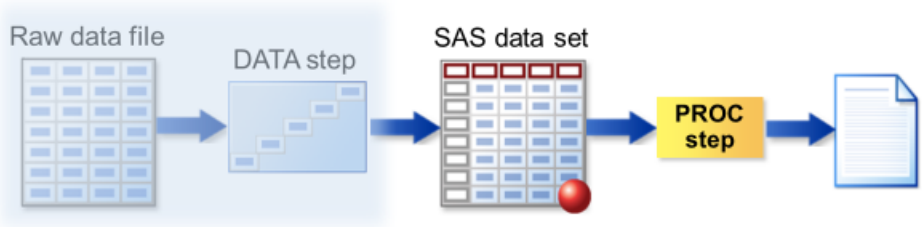
- the data in a SAS data set
- all observations (one per line) or only those specified
- all variables or only those specified.

5

Planning to Create a List Report

Now we can plan to create the list report.

Step 1	Name the SAS data set to be viewed.
Step 2	Determine the PROC step to use.
Step 3	Specify the variables to include in the report and their display order.



11

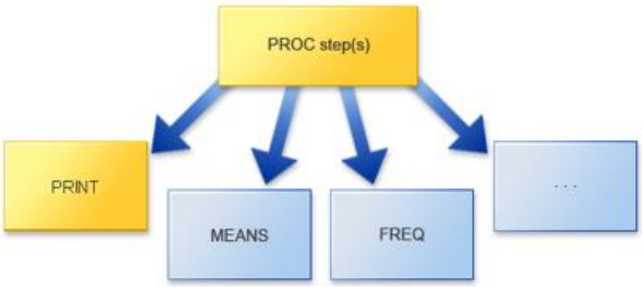
Planning to Create a List Report

Step 1 Name the SAS data set to be viewed.

12

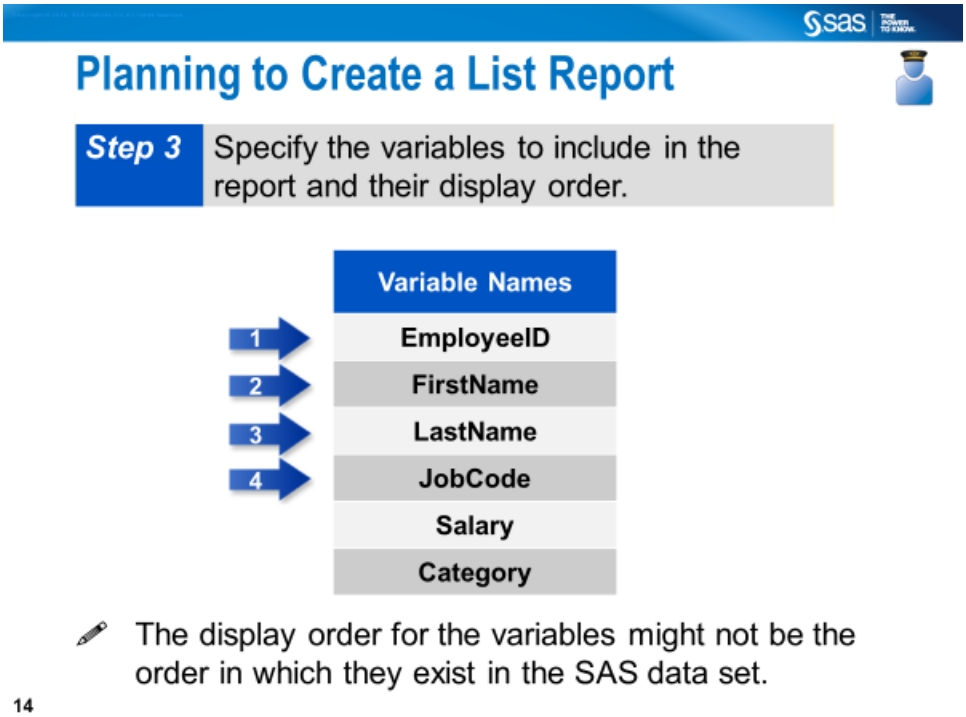
Planning to Create a List Report

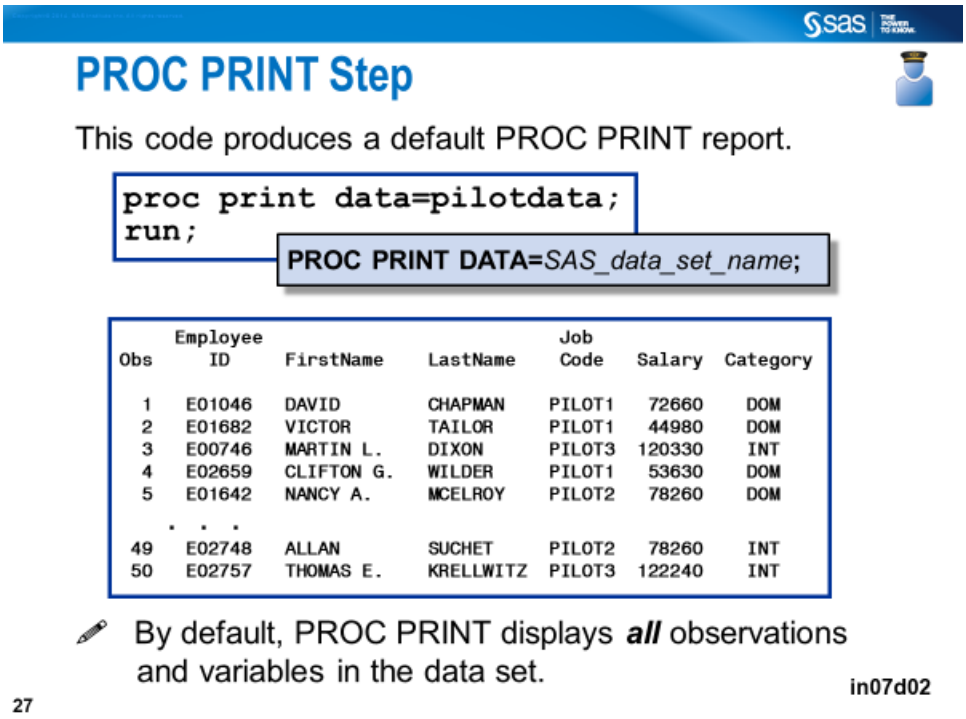
Step 2 Determine the PROC step to use.



Many procedures are available to create different types of reports. To create a simple list report, use the PRINT procedure.

13





PROC PRINT Step

This code produces a default PROC PRINT report.

```
proc print data=pilotdata;
run;
```

PROC PRINT DATA=*SAS_data_set_name*;

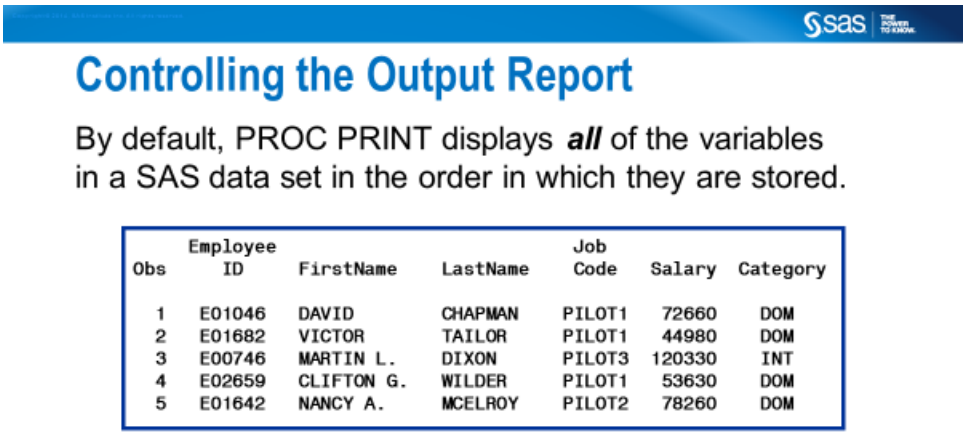
Obs	Employee ID	FirstName	LastName	Job Code	Salary	Category
1	E01046	DAVID	CHAPMAN	PILOT1	72660	DOM
2	E01682	VICTOR	TAILOR	PILOT1	44980	DOM
3	E00746	MARTIN L.	DIXON	PILOT3	120330	INT
4	E02659	CLIFTON G.	WILDER	PILOT1	53630	DOM
5	E01642	NANCY A.	MCELROY	PILOT2	78260	DOM
.	.	.	.	.	.	.
49	E02748	ALLAN	SUCHET	PILOT2	78260	INT
50	E02757	THOMAS E.	KRELLWITZ	PILOT3	122240	INT

By default, PROC PRINT displays ***all*** observations and variables in the data set.

in07d02

27

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



Controlling the Output Report

By default, PROC PRINT displays ***all*** of the variables in a SAS data set in the order in which they are stored.

Obs	Employee ID	FirstName	LastName	Job Code	Salary	Category
1	E01046	DAVID	CHAPMAN	PILOT1	72660	DOM
2	E01682	VICTOR	TAILOR	PILOT1	44980	DOM
3	E00746	MARTIN L.	DIXON	PILOT3	120330	INT
4	E02659	CLIFTON G.	WILDER	PILOT1	53630	DOM
5	E01642	NANCY A.	MCELROY	PILOT2	78260	DOM

How can you print only certain variables in the report?

How can you specify the order of variables in the report?

33

VAR Statement

The VAR statement names the variables to be displayed and specifies the desired order of the variables.

```
proc print data = pilotdata;
   var EmployeeID FirstName LastName JobCode;
run;
```

VAR *variables…;*

Obs	Employee ID	FirstName	LastName	Job Code
1	E01046	DAVID	CHAPMAN	PILOT1
2	E01682	VICTOR	TAILOR	PILOT1
3	E00746	MARTIN L.	DIXON	PILOT3
4	E02659	CLIFTON G.	WILDER	PILOT1
5	E01642	NANCY A.	MCELROY	PILOT2
6	E04732	CHRISTIAN	EDMINSTON	PILOT1

in07d03

34

Program to Add Labels

This code produces and specifies the labels for the report.

LABEL option

LABEL statement

```
proc print data=pilotdata label;
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID Number'
         FirstName='First Name'
         LastName='Last Name'
         Salary='Annual Salary';
run;
```

Labels are ignored unless LABEL or SPLIT= is specified.

in08d03

25

SPLIT= Option

The labels are split at the asterisks. (The asterisk is not part of the label.)

```
proc print data=pilotdata split='*';
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID*Number'
         FirstName='First*Name'
         LastName='Last*Name'
         Salary='Annual*Salary';
run;
```

PROC PRINT DATA=*SAS_data_set_name*
SPLIT='*split-character*';

- It is ***not*** necessary to use ***both*** LABEL and SPLIT=.

in08d04

33

SPLIT= Option

The labels are split as specified.

Partial PROC PRINT Output

Obs	ID Number	First Name	Last Name	Annual Salary
1	E01046	DAVID	CHAPMAN	72660
2	E01682	VICTOR	TAILOR	44980
3	E00746	MARTIN L.	DIXON	120330
4	E02659	CLIFTON G.	WILDER	53630
5	E01642	NANCY A.	MCELROY	78260

34

FORMAT Statement

Adding the FORMAT statement displays the annual salary values with a dollar sign and comma.

```
proc print data=pilotdata split= '*';
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID*Number'
         FirstName='First*Name'
         LastName='Last*Name'
         Salary='Annual*Salary';
   format Salary dollar8.0;
run;
```

Obs	ID Number	First Name	Last Name	Annual Salary
1	E01046	DAVID	CHAPMAN	$72,660
2	E01682	VICTOR	TAILOR	$44,980
3	E00746	MARTIN L.	DIXON	$120,330
4	E02659	CLIFTON G.	WILDER	$53,630
5	E01642	NANCY A.	MCELROY	$78,260

in08d06

43

FORMAT Statement

Formats associated with variables in a PROC step remain in effect only for that step.

```
proc print data=pilotdata;
   format Salary dollar8.;
run;

proc print data=pilotdata;
run;
```

Salary is formatted for this output.

Salary is ***not*** formatted for this output.

in08d07

44

Example of the PRINT Procedure

```
options nodate nonumber ps=30 ls=64;

proc print data=sashelp.shoes noobs split='*';
   var subsidiary product inventory sales;
   where product='Boot' or product='Sandal';
   sum inventory sales;
   by region;
   pageby region;
   label inventory='Total*Inventory'
         sales='Total*Sales';
   format inventory sales dollar14.2;
   title 'Boot and Sandal Report';
   footnote 'Created by Tony Smith';
   footnote2 'Chicago, IL';
run;
```

7

NOOBS Option

The *NOOBS option* suppresses the column in the output that identifies each observation by number.

```
proc print data=sashelp.shoes noobs split='*';
```

By default, the PRINT procedure gives an observation column.

11

NOOBS Option

Without NOOBS

Boot and Sandal Report

Region=Asia

Obs	Subsidiary	Product	Total Inventory	Total Sales
57	Bangkok	Boot	$9,576.00	$1,996.00
59	Bangkok	Sandal	$15,087.00	$3,230.00
62	Seoul	Boot	$160,589.00	$60,712.00
65	Seoul	Sandal	$21,483.00	$4,978.00
Region				

Partial Output

With NOOBS

Boot and Sandal Report

Region=Asia

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

Partial Output

12

VAR Statement

The *VAR statement* selects variables that appear in the report and determines the order of the variables.

```
var subsidiary product inventory sales;
```

Boot and Sandal Report

Region=Asia

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

By default, the PRINT procedure displays all variables in the order that the variables are stored in the data set.

13

WHERE Statement

The *WHERE statement* subsets the input data set by specifying certain conditions that each observation must meet before it is available for the report.

```
where product='Boot' or product='Sandal';
```

- The WHERE statement does not alter the original data set.
- Use only one WHERE statement in a step unless you use a WHERE SAME AND or WHERE ALSO statement with a WHERE statement.
- Character values are case sensitive.

16

WHERE Statement

Examples:

```
where sales > 100000;
where sales eq .;
where name = 'Smith';
where name = ' ';
where sales ge 100000 and name = 'Smith';
where sales ge 100000 or name = 'Smith';
where revenue >= 150 and revenue <= 999;
where revenue between 150 and 999;
where revenue not between 150 and 999;
where month contains 'uary';
where birthdate > '11JUL1968'd;
```

19

SUM Statement

The *SUM statement* totals values of numeric variables.

```
sum inventory sales;
```

Partial Output

Boot and Sandal Report

Region=Asia

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

The SUM statement always gives grand totals and gives subtotals if used with a BY statement.

22

BY Statement

The *BY statement* produces a separate section of the report for each BY group.

```
by region;
```

Partial Output

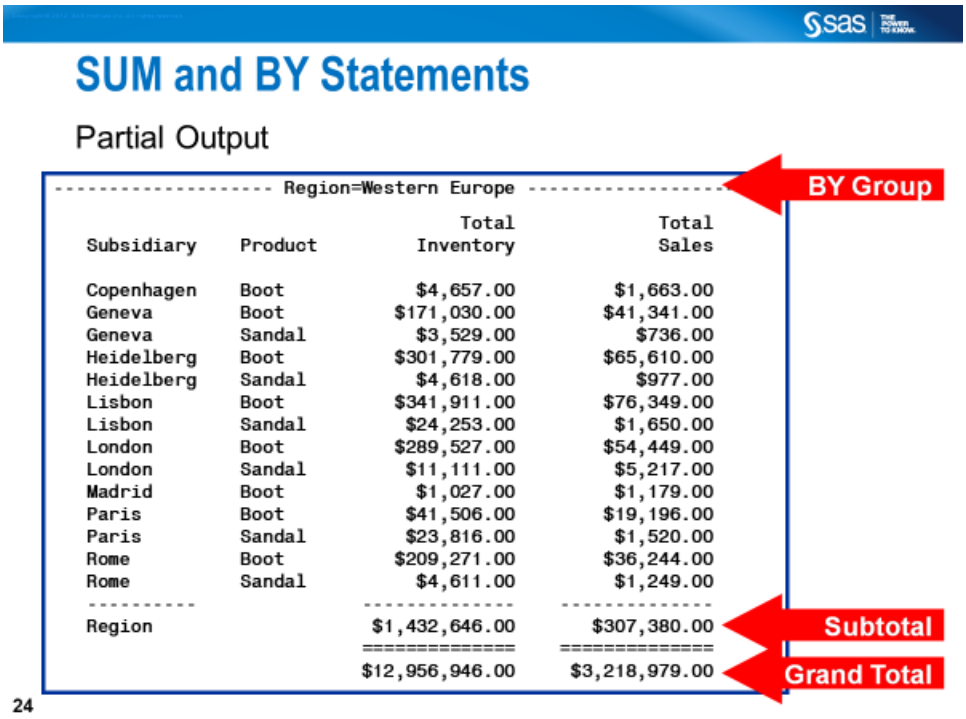
Boot and Sandal Report

Region=Asia

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

Data must be indexed or sorted to use a BY statement.

23



PAGEBY Statement

The *PAGEBY statement* puts each separate section of a BY group on separate pages.

```
pageby region;
```

The PAGEBY statement must name a variable that appears in the BY statement.

25

PAGEBY Statement

Partial Output

Boot and Sandal Report

Region=Asia

Boot and Sandal Report

Region=Eastern Europe

Boot and Sandal Report

Region=Middle East

Subsidiary	Product	Total Inventory	Total Sales
Al-Khobar	Boot	$44,658.00	$15,062.00
Al-Khobar	Sandal	$13,343.00	$1,380.00
Dubai	Boot	$403,259.00	$90,972.00
Dubai	Sandal	$59,985.00	$17,492.00
Tel Aviv	Boot	$222,165.00	$65,248.00
Tel Aviv	Sandal	$71,094.00	$16,314.00
Region		$814,504.00	$206,468.00

26

ID Statement

The *ID statement* specifies the variable(s) to print at the beginning of each row instead of an observation number.

```
id region;
by region;
```

When used with a BY statement, the ID statement eliminates the BY line and suppresses repetitious printing of the BY variable(s).

Partial Output

Region	Subsidiary	Product	Total Inventory	Total Sales
Western Europe	Copenhagen	Boot	$4,657.00	$1,663.00
	Geneva	Boot	$171,030.00	$41,341.00
	Geneva	Sandal	$3,529.00	$736.00
	Heidelberg	Boot	$301,779.00	$65,610.00
	Heidelberg	Sandal	$4,618.00	$977.00

29

LABEL Statement

The *LABEL statement* assigns descriptive labels to variable names.

```
label inventory='Total*Inventory'
      sales='Total*Sales';
```

Partial Output

Boot and Sandal Report

Region=Asia

Obs	Subsidiary	Product	Total Inventory	Total Sales
57	Bangkok	Boot	$9,576.00	$1,996.00
59	Bangkok	Sandal	$15,087.00	$3,230.00
62	Seoul	Boot	$160,589.00	$60,712.00
65	Seoul	Sandal	$21,483.00	$4,978.00
Region			$206,735.00	$70,916.00

30

- A label can be up to 256 characters.

LABEL Statement

The LABEL option enforces variables' labels as column headings.

```
proc print data=sashelp.shoes noobs label;
```

The SPLIT= option specifies the split character, which controls line breaks in column headers and implies the use of labels.

```
proc print data=sashelp.shoes noobs split='*';
```

33

FORMAT Statement

The *FORMAT statement* associates formats to variable values.

```
  format inventory sales dollar14.2;
```

Partial Output

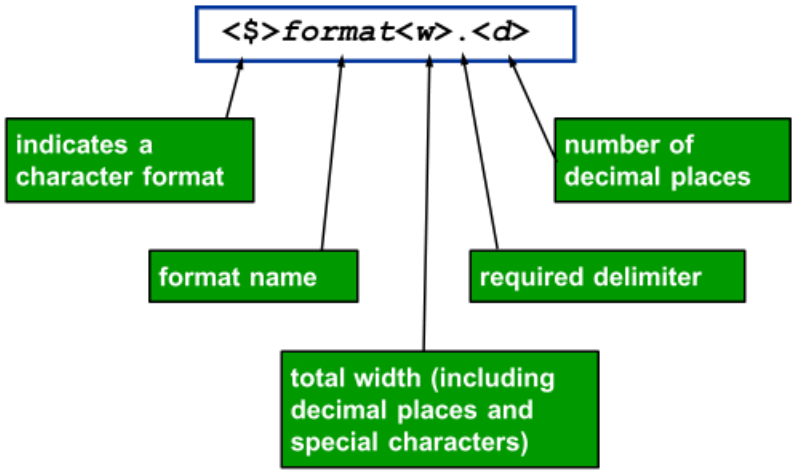
```
------------------------ Region=Asia -------------------------

                                          Total          Total
  Obs   Subsidiary   Product            Inventory          Sales

   57   Bangkok     Boot            $9,576.00       $1,996.00
   59   Bangkok     Sandal         $15,087.00       $3,230.00
   62   Seoul       Boot          $160,589.00      $60,712.00
   65   Seoul       Sandal         $21,483.00       $4,978.00
------                            -------------   -------------
Region                           $206,735.00      $70,916.00
```

A *format* is an instruction that SAS uses to write data values.

34

FORMAT Statement

Formats have the following form:



35

FORMAT Statement

Stored Value	Format	Displayed Value
Washington	$4.	Wash
1234.4567	8.0	1234
1234.4567	8.2	1234.46
1234.4567	comma8.2	1,234.46
1234.4567	dollar9.2	$1,234.46

36

FORMAT Statement

Partial Output

What minimum widths are needed to complete the FORMAT statement for this desired output?

```
-------------------- Region=Western Europe --------------------

                                        Total           Total
Subsidiary      Product              Inventory           Sales

Copenhagen      Boot                    $4,657.00       $1,663.00
Geneva          Boot                  $171,030.00      $41,341.00
                dal                     $3,529.00         $736.00
                t                     $301,779.00      $65,610.00
                dal                     $4,618.00         $977.00
                t                     $341,911.00      $76,349.00
                dal                    $24,253.00       $1,650.00
                t                     $289,527.00      $54,449.00
                dal                    $11,111.00       $5,217.00
Madrid          Boot                    $1,027.00       $1,179.00
Paris           Boot                   $41,506.00      $19,196.00
Paris           Sandal                 $23,816.00       $1,520.00
Rome            Boot                  $209,271.00      $36,244.00
Rome            Sandal                  $4,611.00       $1,249.00
----------                          --------------  --------------
Region                              $1,432,646.00     $307,380.00
                                    ==============  ==============
                                   $12,956,946.00    $3,218,979.00
```

```
format inventory dollar14.2 sales dollar13.2;
```

FORMAT Statement

Stored Value	Format	Displayed Value
17332	mmddyy6.	061507
17332	mmddyy8.	06/15/07
17332	mmddyy10.	06/15/2007
17332	date7.	15JUN07
17332	date9.	15JUN2007
17332	ddmmyy8.	15/06/07
17332	worddate.	June 15, 2007
17332	weekdate.	Friday, June 15, 2007
17332	monyy7.	JUN2007

41

LABEL and FORMAT Statements

LABEL and FORMAT statements assigned in a PROC step are considered ***temporary*** attributes (apply only for the duration of the step).

LABEL and FORMAT statements assigned in a DATA step are considered ***permanent*** attributes (stored in the descriptor portion).

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
6	Inventory	Num	8	DOLLAR12.	DOLLAR12.	Total Inventory
2	Product	Char	14			
1	Region	Char	25			
7	Returns	Num	8	DOLLAR12.	DOLLAR12.	Total Returns
5	Sales	Num	8	DOLLAR12.	DOLLAR12.	Total Sales
4	Stores	Num	8			Number of Stores
3	Subsidiary	Char	12			

44

TITLE Statement

The *TITLE statement* specifies up to 10 lines of text at the top of output.

```
title 'Boot and Sandal Report';
```

Partial Output

Boot and Sandal Report

Region=Asia

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

TITLE is the same as TITLE1.

45

FOOTNOTE Statement

The *FOOTNOTE statement* specifies up to 10 lines of text at the bottom of output.

```
footnote 'Created by Tony Smith';
footnote2 'Chicago, IL';
```

Partial Output

Subsidiary	Product	Total Inventory	Total Sales
Bangkok	Boot	$9,576.00	$1,996.00
Bangkok	Sandal	$15,087.00	$3,230.00
Seoul	Boot	$160,589.00	$60,712.00
Seoul	Sandal	$21,483.00	$4,978.00
Region		$206,735.00	$70,916.00

Created by Tony Smith
Chicago, IL

FOOTNOTE is the same as FOOTNOTE1.

46

TITLE and FOOTNOTE Statements

The TITLE and FOOTNOTE statements are global statements, which means that the statements stay in effect until they are canceled or changed, or you end your SAS session.

The code **`title;`** cancels all titles.

The code **`footnote;`** cancels all footnotes.

TITLE*n* or FOOTNOTE*n*

- replaces a previous title or footnote with the same number
- cancels all titles or footnotes with higher numbers.

47

TITLE and FOOTNOTE Statements

A

The following SAS program is submitted:

```
proc print data=shoes1;
   title1 'Shoe Store';
   title2 'Report One';
   title3 'Accounting';
run;
proc print data=shoes2;
   title2 'Report Two';
run;
```

What titles appear in the second procedure output?

A.
```
Report Two
```

C.
```
Report Two
Accounting
```

B.
```
Shoe Store
Report Two
```

D.
```
Shoe Store
Report Two
Accounting
```

49

Run this program: **Ex11_proc_print.sas**.