

The George Washington University

Department of Statistics

STAT 4197/STAT 6197

Week 1: SAS System, Interfaces, and Concepts

1. SAS System and its Products
2. SAS Interfaces
3. SAS Statements, Variable Names, Data Types, Missing Values, and SAS System Options
4. DATA Step and PROC Step
5. SAS Files and SAS Libraries

Statistical Analysis System® or SAS® - An Overview



What Is SAS?

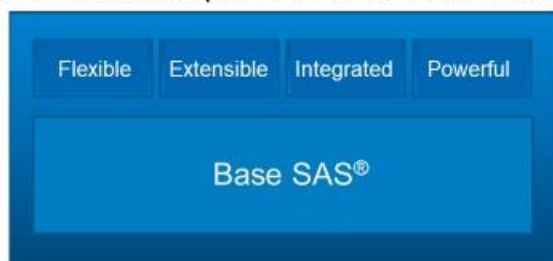
SAS is a suite of business solutions and technologies to help organizations solve business problems.



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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

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
- 6 ■ a graphical user interface for administering SAS tasks

History

1966: SAS was created by Anthony Barr (a graduate student at UNC).

1976: SAS Institute, Inc. was incorporated by Barr, Goodnight, Sall, and Helwig.

2022: SAS continues to record over \$3 billion in annual sales.

The SAS software suite includes more than 200 components (examples below).

- Base SAS (reading data into SAS, manipulating, aggregating, reporting, macro facility)
- SAS/STAT (Statistical analysis and modeling)
- SAS/GRAPH (Graphs, Charts, and Plots)
- SAS/ETS (Econometrics and Time Series Analysis)
- SAS/PH (Clinical Trial Analysis)
- SAS/IML (SAS Interactive matrix language)
- Enterprise Miner (Data Mining)
- Enterprise Guide (GUI-based code editor and project manager)

To find the version and SAS products on your computer, run the following SAS code:

PROC SETINIT; run;

To see what products have been installed on your computer, run the following 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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

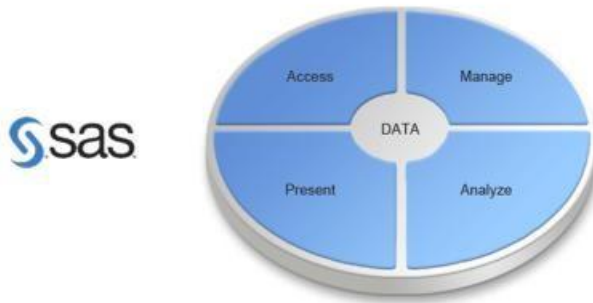
code: `PROC PRODUCT_STATUS; RUN;`



What Can You Do with SAS?

SAS software enables you to do the following:

- access data across multiple sources
- manage data
- perform sophisticated analyses
- deliver information across your organization



5

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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.


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

Standard SAS interfaces for running SAS programs in an interactive environment.

SAS Licensed Software

- Windowing Environment (by default)
- JupyterLab
- Enterprise Guide
- SAS Studio (Single-User)

SAS On Demand for Academics (ODA) - Free Software

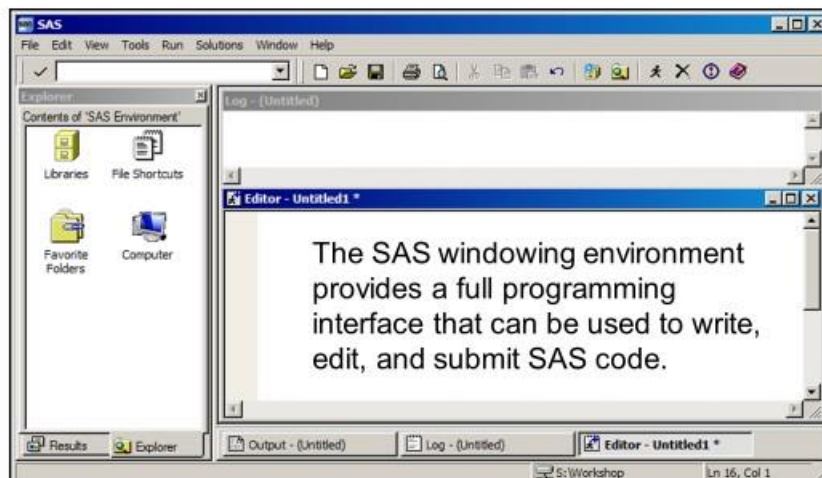
- SAS Studio
- JupyterLab

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.



SAS Windowing Environment

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

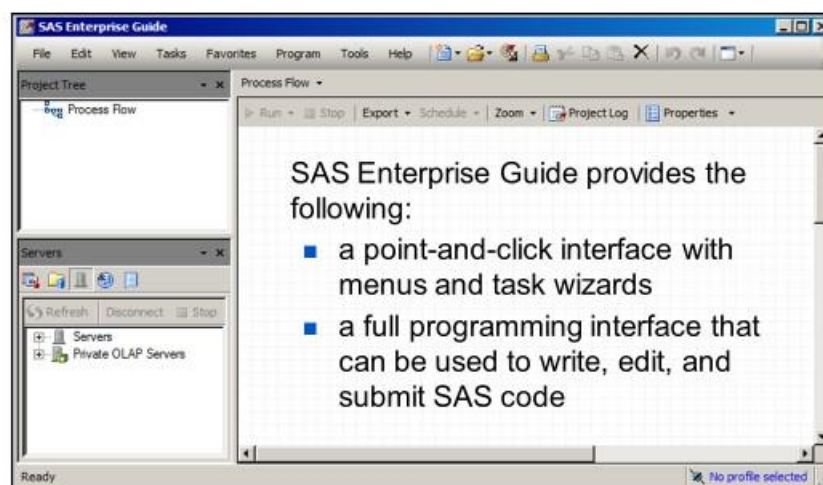


16



SAS Enterprise Guide

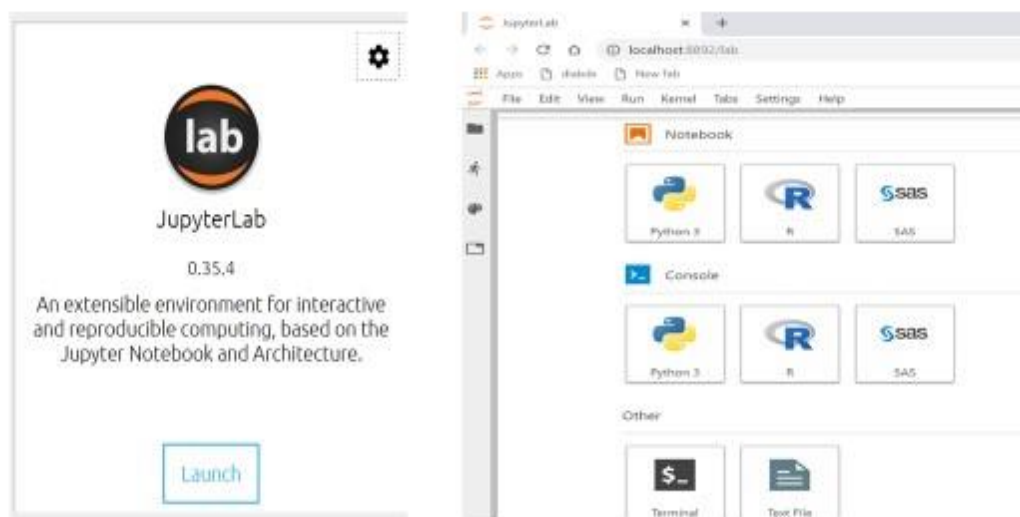
SAS Enterprise Guide is a client application that is accessed from the Windows operating environment.



17

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

JupyterLab – an web-based user interface for programming in many languages including Julia, Python, R, and SAS.



See Hemedinger (2016)

<https://blogs.sas.com/content/sasdummy/2016/04/24/how-to-run-sas-programs-in-jupyternotebook/>

Within Jupyter, the `sas_kernel` provides multiple ways to access SAS programming methods. For example, within Jupyter, you can create a new SAS notebook, available from the **New** menu in the Jupyter Home window and the **File** menu in an active notebook. From a SAS notebook, you can write your SAS code in a cell and run it directly from that cell. Another way that you can run SAS code is by using the Jupyter magic command (e.g., `%%SAS`) supported by the `sas_kernel`. Thus, you can write your SAS program code and get your results out within a Python language notebook. This magic command allows you to run Python and SAS in a single environment (Hemedinger, 2016)

SAS Windowing Environment

Enhanced Editor provides color-coding to identify SAS program elements. To run the code, you click the **Submit** button (“running person” icon) in the toolbar 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.

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone’s use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

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 filename.mht). The default method is changeable as follows: From the menu bar, Select **Tool >> Option >> Preference >>**

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.

SAS Windowing Environment: The SAS Menus

File: Allows you to open and save SAS programs, import and export data, and print files. Select File >> Save As from the menu bar and specify your program's name 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.

Some Common Data Formats

A typical SAS program generally consists of DATA step(s) and PROC steps with or without global statements (e.g., OPTIONS statement, LIBNAME statement).



SAS Statements

A SAS statement is a series of items that might include keywords, SAS names, special characters, and operators.

The two types of SAS statements are as follows:

- those that are used in DATA and PROC steps
- those that are global in scope and can be used anywhere in a SAS program

All SAS statements end with a semicolon.

42

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.

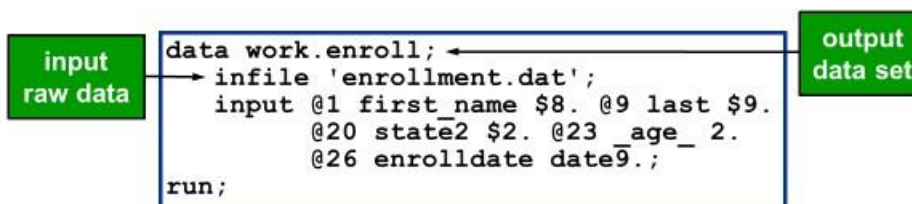
Two types of SAS statements are used in a data step:

- 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, IFTHEN/ELSE, INPUT, INFILE, GO TO, LEAVE, LINK, LIST, LOSTCARD, MERGE, MODIFY, OUTPUT, PUT, REDIRECT, REMOVE, REPLACE, RETURN, MERGE, RETURN, SELECT, SET, STOP, and UPDATE)

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.



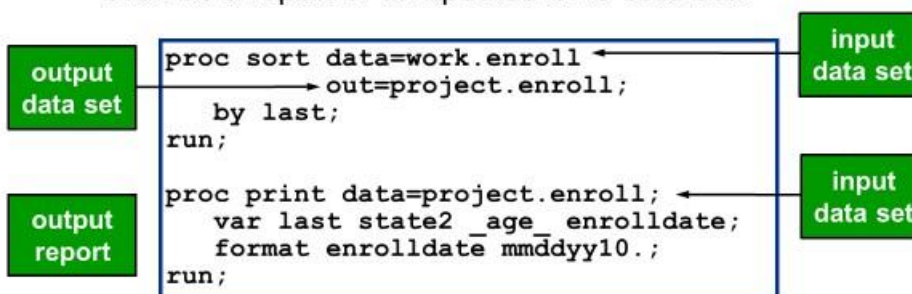
38

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.

A
may



SAS Step Boundary

step
boundary
begin with

- a DATA statement
- a PROC statement

41

A
boundary may end with

- a DATA statement
- a PROC statement
- a RUN statement (for DATA steps and most PROCs)

step

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

- 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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Raw Data Files

The file consists of *records*.

survey.dat

JM	M	GA	17	F
TM	M	CT	5	E
GS	F	FL	2	A
DW	F	AL	10	D
BW	F	NC	12	B
.
.

File

Record

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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

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

Definition of a SAS Name

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 [\(Link\)](#)

• 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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

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

Some Rules for Reading Numeric and Character Data (SAS Documentation)

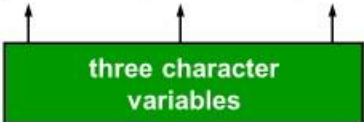
Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Variable Types

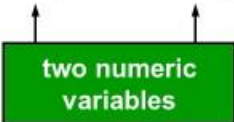
The two types of SAS variables are listed below:

- 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



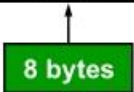
two numeric variables

62

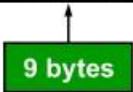
Variable Types: Character

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

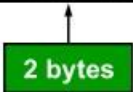
VIEWTABLE: Project.Enroll					
	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287



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	Nonstandard Numeric Data
58	(23)
67.23 -23	\$67.23
5.67E5 00.99	5,823 01/12/2010
1.2E-2	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.

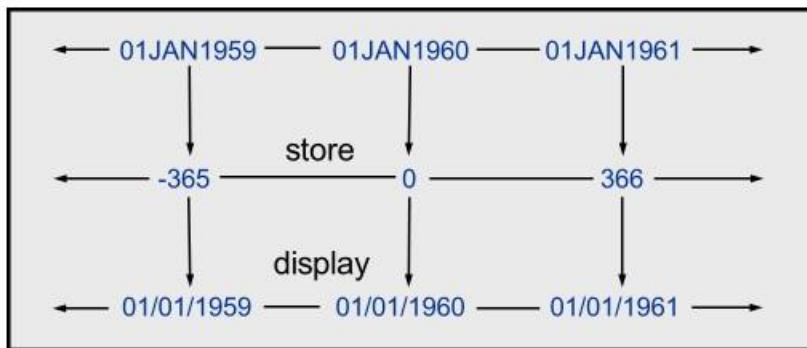
Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.



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 can read and perform calculations on dates starting from A.D. 1582 through A. D. 19,900.

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 a data step session, in which a temporary SAS data set is created, even if WORK is not specified. In the above example, 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: ROUND() 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.

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.

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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

•

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



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.

```

CONTENTS misspelled

DATA= missing

80

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

```

1  *Ex1B_Color_coding.sas;
2  PROC FORMAT;
3  value $sexfmt 'M' = 'Male'
4               'F' = 'Female';
5  run;
6  Title "Table from SASHELP Data Set";
7  PROC FREQ DATA=SASHELP.CLASS;
8  Tables Sex;
9  format Sex $sexfmt.;
10 run;
11
12 Title "One-way Table;
13 PROC FREQ DATA=SASHELP.CLASS;
14 Tables Sex;
15 format Sex $sexfmt.;
16 run;

```

Lines 2-10

Keywords: DATA, value, Tables, format [Blue]

Format Name: \$SEXFMT. [Green]

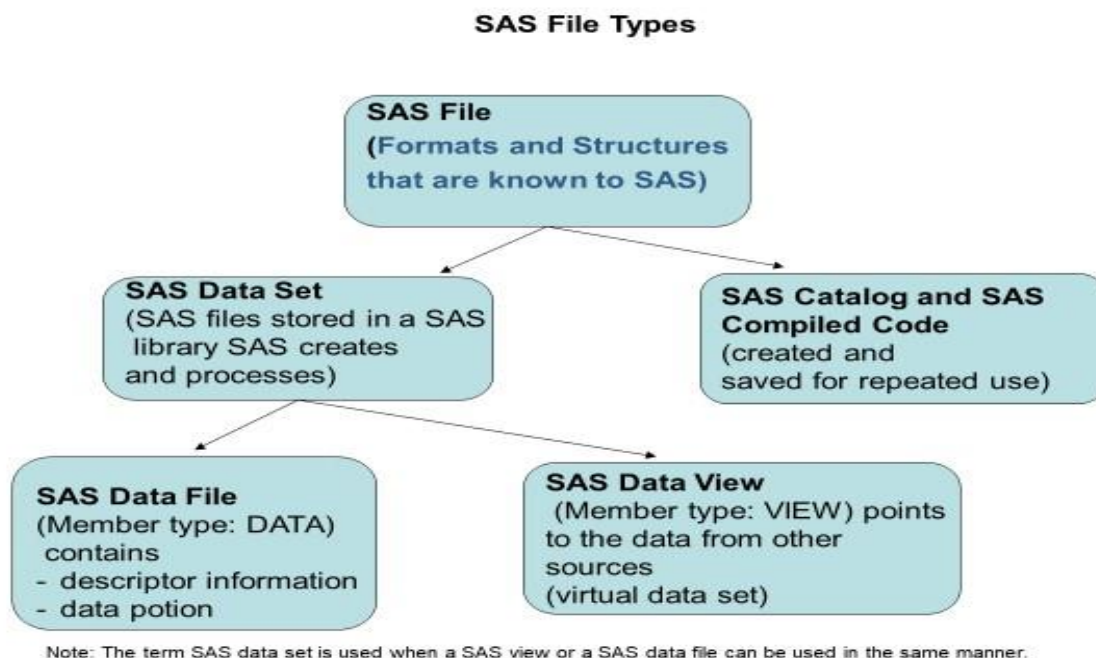
Text within single or double quotes: e.g., 'M' [Red] Variable
or Data Set Name SASHELP.CLASS [Black]

PROC Name: e.g., PROC FORMAT or PROC FREQ [Dark Blue]

Lines 12-16

Note that we are missing an end quote in Line 12 and that all of the syntax is red.

SAS File Concepts



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

Definition of a SAS Data File



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

Consider creating a permanent SAS data set if you think you would access it frequently for further data manipulation or analysis.

SAS Libraries

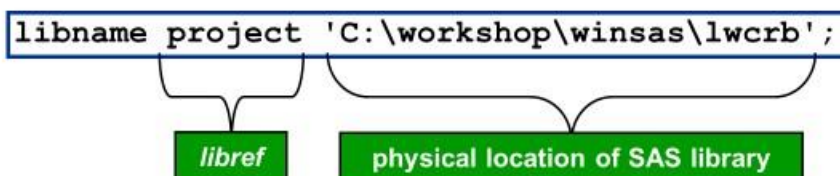
A SAS library is a collection of one or more SAS files, including SAS data sets, that are referenced and stored as a unit.

- 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.



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

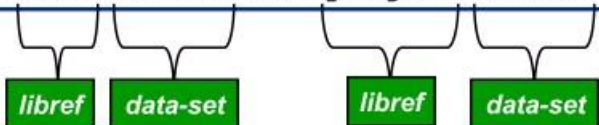
Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

“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.

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



- *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

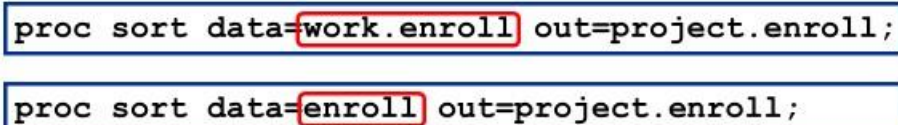
One-Level SAS Data Set Names

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

Built-in SAS Libraries

SASHELP and SASUSER are built-in libraries and will always show up when SAS is invoked.

- The SASHELP library (predefined by SAS) 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. This is a read-only library. Try the following SAS code to see the folder locations of the SASHELP library.

```
%put %sysfunc(pathname(SASHELP)) ;
```

- SASUSER is a permanent library (predefined by SAS) that contains SAS files in the profile catalog that stores your personal settings. This is also a convenient place where users can store their own SAS files. Try the following SAS code to see the folder locations of the SASUSER library.

```
%put %sysfunc(pathname(SASUSER)) ;
```

Exploring the SAS Data Library

The CONTENTS procedure with the `_ALL_` keyword generates a list of all SAS files in a library.

```
proc contents data=orion._all_ nods;
run;
```

```
PROC CONTENTS DATA=libref._ALL_ NODS;
RUN;
```

- `_ALL_` requests all the files in the library.
- The NODS option suppresses the individual data set descriptor information.
- NODS can be used only with the keyword `_ALL_`.

18

psm02d03

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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

SAS Data Sets



- The *descriptor portion* contains attribute information about the data in a SAS data set.
- The *data portion* contains the data values in the form of a rectangular table that consists of observations and variables.

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

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Generating Contents of a SAS Data Set (Various Ways)

```
options nocenter nodate nonumber;

proc contents data=sashelp.heart;
run;

proc contents data=sashelp.heart varnum;
run;
proc contents data=sashelp.iris position;
ods select variables;
run;
proc sql;
select memname, nobs format =comma9. ,
       nvar format=comma9.
from dictionary.tables
where libname='SASHELP' and memname like 'IRIS';
quit;
```

Comparable R and Python Code

```
/******
```

```
# R Code
```

```
data(iris)
```

```
str(iris)
```

```
*****/
```

```
/******
```

```
# Python Code
```

```
import seaborn as sns
```

```
import seaborn as sns
```

```
iris = sns.load_dataset('iris')
```

```
iris.info()
```

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. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.