

### **3. INTRODUCTION TO SAS**

**Reading Assignment: SELECTED SAS DOCUMENTATION FOR BIOS111  
Part 1: Introduction to SAS Software**

**REVISED FALL 2000**



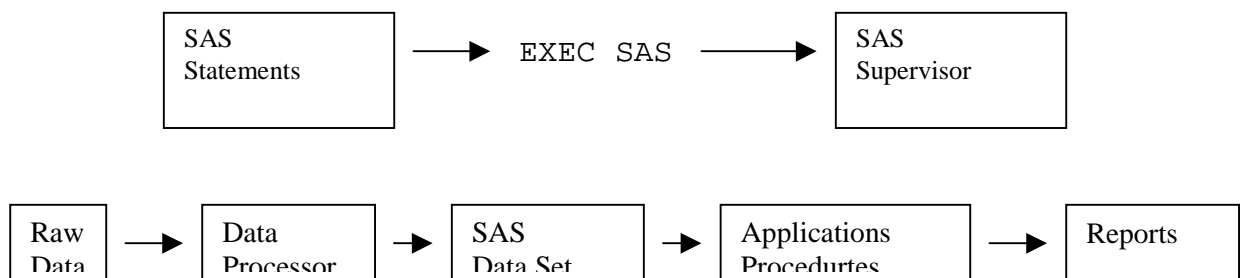
# STATISTICAL ANALYSIS SYSTEM

## What is SAS?

SAS is a computer software system that provides all the tools needed for data analysis:

- \* reading data: flexible input techniques
- \* transformations: programming language with statistical and mathematical functions
- \* manipulation: sorting, subsetting, concatenating, and merging
- \* maintenance: storing, documenting, updating, and editing
- \* report writing: printing information using pre-written procedures or customized programs
- \* graphics: charts, plots, maps and slides
- \* data reduction and summarization: descriptive statistics
- \* statistical analysis: from simple crosstabulations to complex multivariate techniques

SAS consists of a data-handling language and a library of procedures that work together as a system. A supervisor program controls the execution of you SAS job.

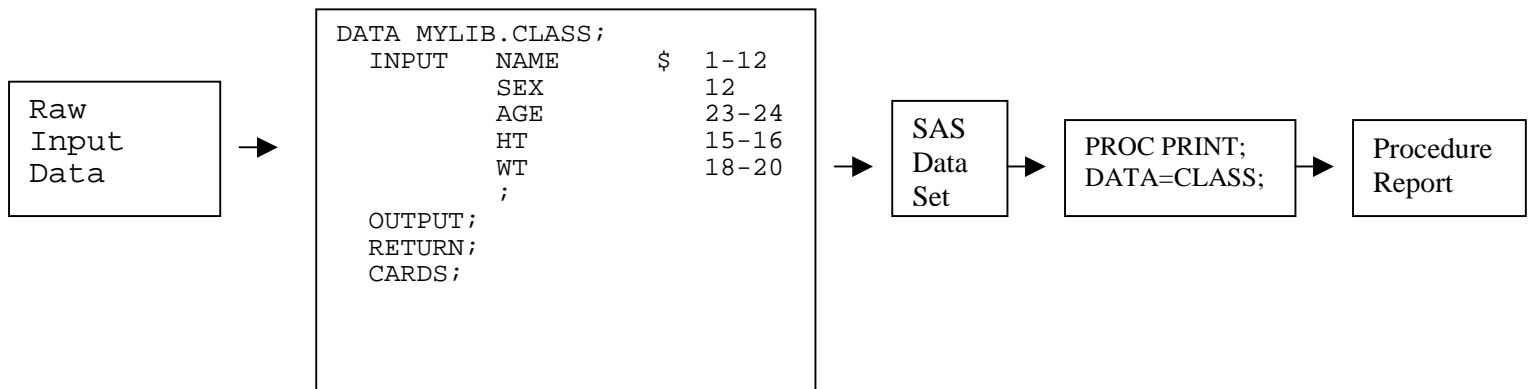
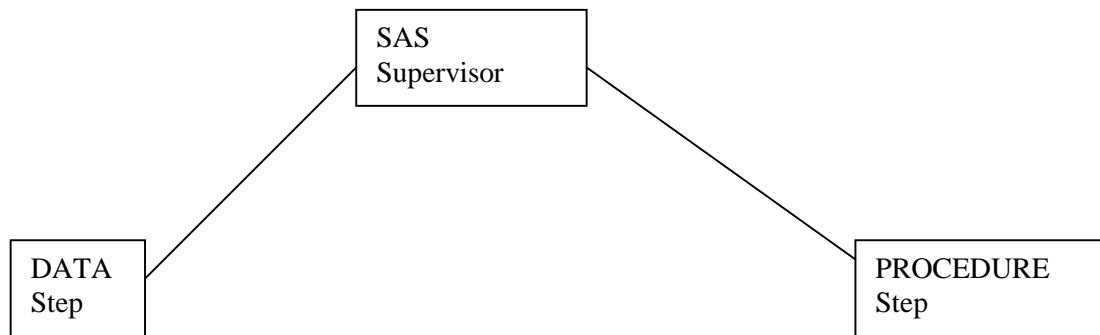


The SAS system is comprised of numerous SAS products. This course will focus on base SAS software. Other SAS products that will be introduced include SAS/FSP, SAS/GRAPH, SAS/ASSIST, SAS/ACCESS, SAS/STAT, and SAS/IML.

## SAS Processing

All SAS jobs are a sequence of SAS steps. There are only two kinds of SAS steps:

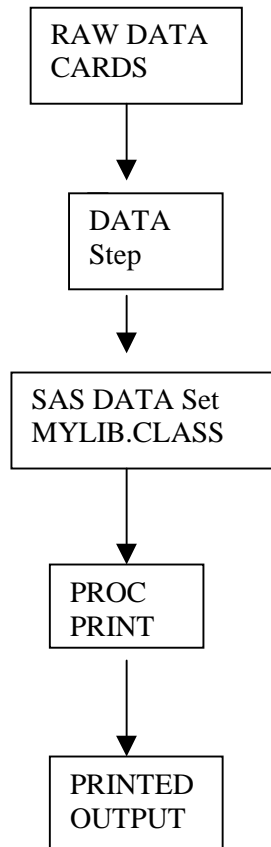
- \* DATA steps are usually used to create SAS data sets, but can be used to produce reports
- \* PROCEDURE steps analyze or process SAS data sets (generate reports and graphs, edit data, sort data) and, in some cases to create SAS data sets.



## Combining SAS Steps into a Job:

A SAS job consists of one or more DATA and/or PROC steps, along with the necessary system commands, which are submitted to the computer together.

- \* SAS steps are processed sequentially; the first step listed is executed first.
- \* A job may have more than one DATA step and create more than one SAS data set.
- \* A SAS data set may be used in turn by more than one PROC.
- \* A PROC can use a SAS data set created in a previous job.



## The PROC STEP

In what is called a PROC step, a large library of prewritten procedures enables end users to produce reports easily. You can use PROC steps in base SAS software for:

- ❖ List and tabular reports
- ❖ Graphics
- ❖ Statistical analysis
- ❖ Data management
- ❖ Ad hoc queries
- ❖ Accessing other software files

## The DATA STEP

In what is called the DATA step, a powerful programming language gives programmers great flexibility in designing applications. DATA step capabilities include:

- ❖ Sophisticated record i/o
- ❖ Conditional logic
- ❖ Iterative do loops
- ❖ Array processing
- ❖ Structured programming logic
- ❖ A wide range of functions
- ❖ Producing customized reports

## The SAS Data Set

Data must be in the form of a SAS data set to be processed by most SAS procedures and some DATA statements. SAS data sets consist of a descriptor portion that contains information about the data and a data portion that contains the data values. The data values in the data portion are arranged in a rectangular table.

	VARIABLES					
	NAME	SEX	AGE	HEIGHT	WEIGHT	
observation 1	JOHN	M	12	59.0	99.5	
observation 2	JAMES	M	12	57.0	83.0	
observation 3	ALFRED	M	12	69.0	112.5	
.						
observation 19	ALICE	F	13	56.5	84.0	

The columns in the table are called variables.

- ❖ Variables correspond to fields of data.
- ❖ Each variable is given a name.

❖ Rules for names:

1-8 characters (through version 6.12)

1-32 characters (beginning with version 7.0)

start with the letter A-Z or the underscore character (\_\_\_)

continue with letters, numbers, and underscores

choose names that describe the fields

❖ There are two kinds of variables.

Character variables: ASCII representation,  
1-200 bytes long (through version 6.12)  
1-32,767 (beginning with version 7.0)  
(e.g., NAME, SEX)

Numeric variables: Floating point representation, 3-8 bytes long  
(e.g., AGE, HEIGHT, WEIGHT)

❖ There is no limit to the number of variables that can be stored in a SAS data set.

❖ The rows in the table are called observations (or records). There is no limit to the number of observations.

## Missing Values

Most collections of data contain missing values. The rectangular structure of a SAS data set implies that a value must exist for every variable for every observation.

❖ In SAS data sets, missing values are represented by:

-- a period (".") for a numeric variable

-- a blank (" ") for a character variable

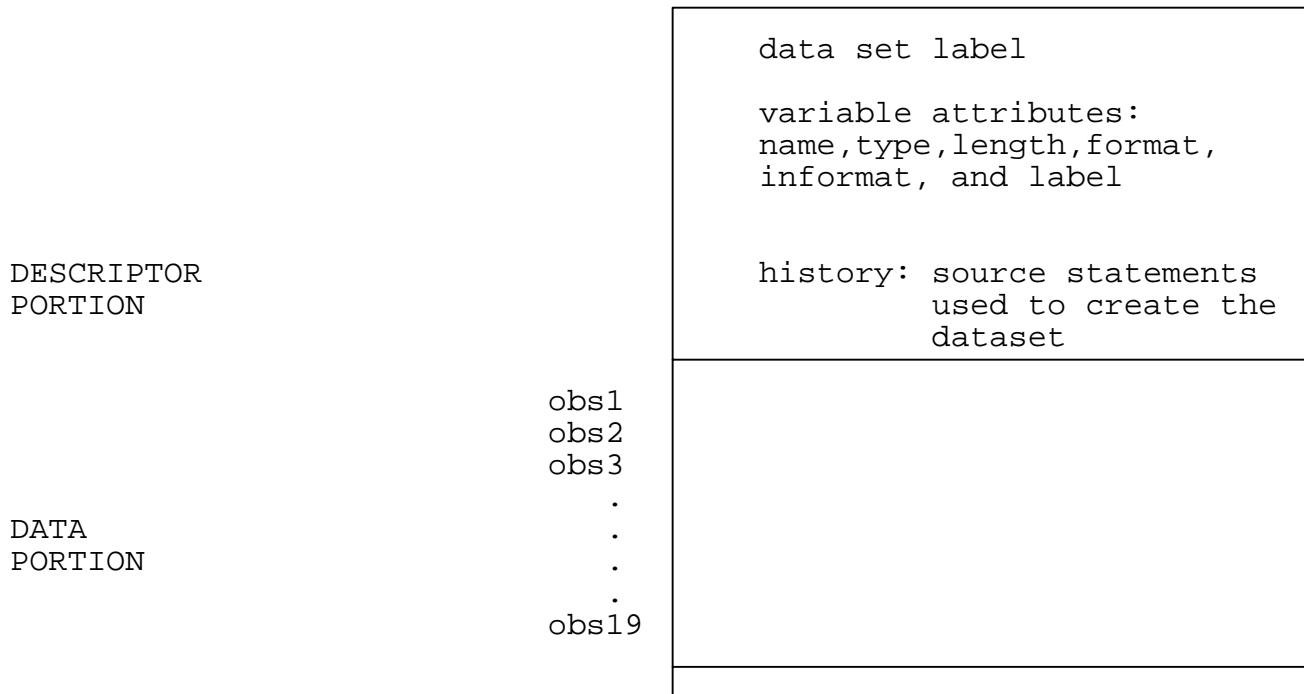
❖ Missing numeric variables are not zero. They are excluded from arithmetic and statistical computations.

❖ Each SAS PROC checks variables for missing values and takes appropriate action.

❖ See the individual PROC descriptions in the User's Guide for details.

## Documenting SAS Data Sets

A SAS data set contains, in addition to the data values, descriptors with names and attributes of the variables.



If you store your data as a SAS data set, you can forget all the original physical details. The descriptor portion of a SAS data set contains detailed information about the data set. This information includes:

- ❖ The name of the data set and its member type
- ❖ The date and time the data set was created
- ❖ The number of observations
- ❖ The number of variables
- ❖ The engine type

Below is a partial listing of the descriptor portion of a SAS data set.

Data Set Name: SC.CLASS	Observations: 6
Member Type: DATA	Variables: 5
Engine: V612	Indexes: 0
Created: 13:22 Thursday, July 30, 1998	Observation Length: 37
Last Modified: 10:26 Monday, January 25, 1999	Deleted Observations: 0
Protection:	Compressed: NO
Data Set Type:	Sorted: NO
Label:	

SAS data set names should be descriptive of the information stored in the data set. They also must follow the rules for all SAS names

- ❖ Be 1-8 characters in length ( through version 6.12)
- ❖ Be 1-32 characters in length (starting with version 7.0)

- ❖ Begin with a letter (A-Z) or an underscore ( \_ )
- ❖ Continue with any combination of letters, numbers, or underscores

The descriptor portion of a SAS data sets also contains information about the attributes of all variables in the data set. The attribute information includes the variable's name, type, length, position, format, informat, and label.

## **SAS Data Libraries**

- ❖ Every SAS data set is stored in a SAS data library.
- ❖ A SAS data library is a collection of SAS files recognized as a unit by the SAS system.
- ❖ In directory based operating systems, such as Windows, OS/2, or UNIX, a SAS data library is a collection of SAS files of the same engine type stored in a specific directory .
- ❖ Other types of files can also reside in the same directory
- ❖ SAS automatically assigns extensions to SAS files in a directory based operating system. The extension varies across operating system (.SD2, .SSD, SD7 for windows & OS/2, .SSD01 or SD7 for UNIX).
- ❖ Any number of data sets can be stored in a single library ( as long as the library is big enough to hold them).
- ❖ Any number of SAS data sets from any number of libraries can be used, created, and deleted in a single job.
- ❖ Every SAS file has a two level name. The first level determines whether the file is temporary or permanent.
- ❖ The general form of a SAS filename is:

*libref.SAS-filename*

*libref* is a name specified in a LIBNAME statement that is associated with a directory

SAS-filename refers to a specific SAS file in the library

- ❖ If you do not specify a libref (first level name):
  - the default is WORK
  - the data set is temporary (exists only for the duration of the current SAS session)
- ❖ The **LIBNAME** statement is used to point to or reference a SAS data library. The LIBNAME statement is used to associate a libref (short for SAS data library reference) with a directory.

SAS knows which directory a libref references because you associate the libref with a directory using the LIBNAME statement. Once defined a libref can be used repeatedly throughout a program. You can think of librefs as temporary nicknames that you use to identify SAS data libraries during a SAS session.

General form of the LIBNAME statement is:

```
LIBNAME libref < engine-name > 'SAS-data-library' options ;
```

*libref* any valid SAS name

*SAS-data-library* a directory

*engine-name* an optional parameter specifying one of the library engines supported by a given operating system

V7 accesses 7.x Release SAS data sets

V8 accesses 8.x Release SAS data sets

V612 accesses 6.12 Release SAS data sets

V604 accesses 6.03 and 6.04 Release SAS data sets

XPORT accesses transport format files

Example:

```
LIBNAME CLASSLIB V612 'C:\BIOS111\SASDATA';  
PROC PRINT DATA=CLASSLIB.LIPIDS ;  
RUN;
```

In the example above, the LIBNAME statement associates the libref CLASSLIB with the directory C:\BIOS111\SASDATA. Next, the SAS procedure PRINT is used to print the SAS data set LIPIDS, which resides in the directory C:\BIOS111\SASDATA .

- ❖ If a directory contains SAS files created by several different engines, only SAS files created with the engine assigned to a given libref can be accessed using that libref.
- ❖ You can assign multiple librefs with different engines to the same directory. For Example:

```
LIBNAME one V604 'C:\BIOS111' ;  
LIBNAME two V612 'C:\BIOS111' ;
```

- ❖ If you do not specify an engine name in a LIBNAME statement, SAS attempts to determine the engine that should be assigned to the data library. The SAS system looks at file extension in the given directory and uses the following rules to determine the engine:

- ❖ If the directory contains SAS data sets from only one of the supported native library engines (not including XPORT), the libref is assigned to that engine.
- ❖ If there are no SAS data sets in the given directory, then the default engine is assigned. In version 6.12, the default engine is V612. In version 7.0, the default engine is V7. In version 8.0, the default engine is V8.
- ❖ If the directory contains SAS data sets from more than one engine, the default engine is assigned.
- ❖ SAS Versions 6.10, 6.11, and 6.12 data sets are given the file extension .SD2 for Windows and OS/2.
- ❖ SAS Versions 6.10, 6.11, and 6.12 data sets are given the file extension .SSD01 for UNIX.
- ❖ SAS Version 7.0 data sets are given the file extension .SD7 for all operating systems.
- ❖ SAS Versions 6.03 and 6.04 data sets are given the file extension .SSD.

## **Temporary/Permanent SAS Data Libraries**

- ❖ You can store SAS data sets in a temporary SAS data library by omitting the libref or by using the libref WORK. For example:

```
DATA ONE ;
  INFILE XYZ ;
  INPUT A B C ;
RUN;
```

- ❖ You can permanently store SAS data sets by using a libref other than WORK. The directory where you want to store your data sets must exist. For example:

```
LIBNAME BIOS 'Y:\BIOS111';
DATA BIOS.ONE ;
  INFILE XYZ ;
  INPUT A B C ;
RUN;
```

## **SAS FILES**

- ❖ The individual files in the library are considered members of the library. Member types include DATA, VIEW, CATALOG, ACCESS, and PROGRAM.
- ❖ SAS data sets can have one of two member types, DATA or VIEW, depending on the kind of information they contain.
- ❖ A SAS catalog (member type catalog) is a specially formatted type of SAS file that contains a directory and entries. There are several types of SAS catalogs. We will study these later in the course.



## Rules for Writing SAS Statements

- ❖ Each step in a SAS program is comprised of one or more SAS statements.
- ❖ Some SAS statements can be used
  - ❖ only in DATA steps
  - ❖ only in PROC steps
  - ❖ anywhere
- ❖ Most SAS statements begin with an identifying keyword.
- ❖ All SAS statements end with a semicolon.
- ❖ SAS statements can be upper or lower case.
- ❖ SAS statements are free format.
  - ❖ They can begin anywhere on a line and end anywhere on a line.
  - ❖ One statement can continue over several lines.
  - ❖ Several statements can be on one line.
  - ❖ Blanks--as many as you want--separate fields. Special characters also separate fields.
- ❖ Recommended style:
  - ❖ Start each statement on a new line.
  - ❖ Start DATA and PROC statements in column 1; indent the other statements within a DATA step or a PROC step to indicate the logical structure of the step.
- ❖ A SIMPLE SAS JOB;  

```
DATA WORK.CLASS;  
  SET CLASSLIB.CLASS;  
  HTMET = HT*2.54;  
  LABEL HT = 'HEIGHT IN INCHES'  
        HTMET = 'HEIGHT IN CM';  
RUN;  
  
PROC PRINT DATA=WORK.CLASS;  
RUN;  
  
PROC PLOT DATA=WORK.CLASS;  
  PLOT WT*HT;  
RUN;
```
- ❖ The SAS supervisor detects the end of a step when a DATA, PROC, or RUN statement is encountered.

## COMMENTS IN SAS CODE

There are 2 ways to insert comments in SAS:

```
* message ;
```

or

```
/* message */
```

- ❖ comments can be used anywhere in a SAS job for documentation purposes
- ❖ SAS ignores comments during processing
- ❖ \* message ; must be written as a separate statement and can not contain internal semicolons
- ❖ /\* message \*/ can be written within statements or anywhere a blank can appear. These comments can contain semicolons
- ❖ Examples:

```
* this is a comment ;
```

```
*----- *  
* this is a comment *  
*----- *;
```

```
/* this is a comment */
```

```
proc print /* comment*/ data=one ;  
run;
```

## A SAS Program

- ❖ SAS program consists of a series of DATA steps and PROC steps.
- ❖ When you execute a SAS program, the output generated by SAS is divided into two parts: SAS LOG and OUTPUT.
- ❖ SAS LOG
  - ❖ Contains information about the processing of the SAS program
  - ❖ Prints the statements you entered
  - ❖ Prints errors and warning messages
  - ❖ Prints NOTES relating to each step:
    - ❖ --- For the DATA step, documents the creation of the data set
    - ❖ --- For the PROC step, indicates the page numbers of the output
- ❖ OUTPUT contains the results of the PROC steps.
- ❖ Example:

The Job, Log, and Procedure output from a simple SAS job.

```
TITLE1 'CHAPTER 3, EXAMPLE 1: A SIMPLE SAS JOB';
```

```
LIBNAME CLASSLIB 'C:\BIOS111\SASDATA';
```

```
DATA WORK.CLASS;  
  SET CLASSLIB.CLASS;  
  HTMET=HT*2.54;  
  WTMET=WT/2.2;  
  LABEL HTMET='HEIGHT IN CENTIMETERS'  
        WTMET='WEIGHT IN KILOGRAMS';
```

```
RUN;
```

```
PROC PRINT DATA=WORK.CLASS;  
  TITLE2 'LISTING OF DATA PORTION OF SAS DATASET WORK.CLASS';  
RUN;
```

```
PROC CONTENTS DATA=WORK.CLASS;  
  TITLE2 'LISTING OF DESCRIPTOR PORTION OF SAS DATASET WORK.CLASS';  
RUN;
```



## SAS LOG

NOTE: Copyright(c) 1985,86,87 SAS Institute Inc., Cary, NC 27512-8000, U.S.A.

NOTE: SAS (r) Proprietary Software Release 6.04

Licensed to UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL, Site 03944001.

NOTE: AUTOEXEC processing completed.

```
1  TITLE1 'CHAPTER 3, EXAMPLE : A SIMPLE SAS JOB' ;
2  LIBNAME CLASSLIB 'C:\BIOS111\SASDATA' ;
3
4  DATA WORK.CLASS;
5      SET CLASSLIB.CLASS ;
6      HTMET=HT*2.54 ;
7      WTMET=WT/2.2 ;
8      LABEL HTMET='HEIGHT IN CENTIMETERS'
9            WTMET='WEIGHT IN KILOGRAMS'
10             ;
11  RUN;
```

NOTE: Missing values were generated as a result of performing an operation on missing values.

Each place is given by: (Number of times) at (Line):(Column).

1 at 7:10

NOTE: The data set WORK.CLASS has 6 observations and 7 variables.

NOTE: The DATA statement used 4.00 seconds.

```
12
13  PROC PRINT DATA=WORK.CLASS ;
14      TITLE2 'LISTING OF DATA PORTION OF SAS DATASET WORK.CLASS' ;
15  RUN ;
```

NOTE: The PROCEDURE PRINT used 3.00 seconds.

```
16
17  PROC CONTENTS DATA=WORK.CLASS ;
18      TITLE2 'LISTING OF DESCRIPTOR PORTION OF SAS DATASET WORK.CLASS' ;
19  RUN ;
```

NOTE: The PROCEDURE CONTENTS used 2.00 seconds.

## **Syntax Errors**

If you misspell a SAS keyword, forget a semicolon, or make similar mistakes, SAS prints the word ERROR or WARNING followed by an error number. A message explaining the error is printed at the end of the step.

Example: Log output from a SAS job with several syntax errors.

```
TITLE1 'CHAPTER 3, EXAMPLE2: A SAS JOB WITH SYNTAX ERRORS';

LIBNAME CLASSLIB 'C:\BIOS111\SASDATA';

DATA WORK.CLASS;
  SET CLASSLIB.CLASS;
  HTMET=HT*2.54;
  WTINKILOS=WT/2.2;
  LABEL HTMET='HEIGHT IN CENTIMERS'
        WTMET='WEIGHT IN KILOGRAMS'
        ;
RUN;
PROC FREQ DATA=WORK.CLASS; TABLES SEX;
  TITLE1 'FREQUENCY BY SEX';

PROC PRINT DATA=WORK.CLASS;
  TITLE2 'LISTING OF DATA PORTION OF SAS DATASET WORK.CLASS';

PROC CONTENTS DATA=WORK.CLASS;
  TITLE2 'LISTING OF DESCRIPTOR PORTION OF SAS DATASET WORK.CLASS';

PROC MEANS DATA=WORK.CLASS; VAR HTH WT;
  TITLE2 'SIMPLE STATISTICS FOR HEIGHT AND WEIGHT';

PROC RPINT DATA=WORK.CLASS;
  TITLE2 'LISTING OF DATA PROCTION OF SAS DATASET WORK.CLASS';

PROC CONTENTS DATA=WORK.CLASS;
  TITLE2 'LISTING OF DESCRIPTOR PORTION OF SAS DATASET WORK.CLASS';
```

NOTE: Copyright(c) 1985,86,87 SAS Institute Inc., Cary, NC 27512-8000, U.S.A.  
NOTE: SAS (r) Proprietary Software Release 6.04  
Licensed to UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL, Site 03944001.

NOTE: AUTOEXEC processing completed.

```
1  TITLE1 'CHAPTER 3, EXAMPLE 2: A SAS JOB WITH SYNTAX ERRORS';  
2  
3  LIBNAME CLASSLIB 'C:\BIOS111\SASDATA';  
4  
5  DATA WORK.CLASS ;  
6    SET CLASSLIB.CLASS ;  
7    HTMET=HT*2.54 ;  
8    WTINKILOS=WT/2.2 ;
```

**ERROR: The variable named WTINKILOS contains more than 8 characters.**

```
9  LABEL HTMET='HEIGHT IN CENTIMETERS'  
10     WTMET='WEIGHT IN KILOGRAMS'  
11 ;  
12 RUN;
```

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

NOTE: The data set WORK.CLASS has 0 observations and 7 variables.

NOTE: The DATA statement used 1.00 seconds.

```
13  
14 PROC FREQ DATA=WORK.CLASS ; TABLES SEX;  
15 TITLE2 'FREQUENCT BY SEX';  
16 RUN ;
```

WARNING: No observations in data set WORK.CLASS.

NOTE: The PROCEDURE FREQ used 1.00 seconds.

```
17  
18 PLCO PRINT DATA=WORK.CLASS ;  
PLCO PRINT DATA=WORK.CLASS ;
```

----  
**ERROR: Statement is not valid or it is used out of proper order.**

```
19 TITLE2 'LISTING OF DATA PORTION OF WORK.CLASS';  
20 RUN ;  
21  
22 PROC CONTENTS DATA=WORK.CLASS;
```

**ERROR: Data set WORK.CLASSS not found.**

```
23 TITLE2 'LISTING OF DESCRIPTOR PORTION OF WORK.CLASS';  
24 RUN ;
```

**NOTE: Statements not processed because of errors noted above.**

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

NOTE: The PROCEDURE CONTENTS used 1.00 seconds.

```
25  
26 PROC MEANS DATA=WORK.CLASS;  
27   VAR HTH WT ;
```

**ERROR: Variable HTH not found.**

```
28 TITLE2 'DESCRIPTIVE STATISTICS FOR HEIGHT AND WEIGHT';  
29 RUN ;
```

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

NOTE: The PROCEDURE MEANS used 0.00 seconds.

```
30  
31 PROC RPINT DATA=WORK.CLASS ;
```

**ERROR: PROCEDURE RPINT not found.**

```
32 TITLE2 'LISTING OF DATA PORTION OF WORK.CLASS';  
33 RUN ;
```

```
34  
35 PROC CONTENTS DATA=WORK.CLASS  
36 TITLE2 'LISTING OF DESCRIPTOR PORTION OF WORK.CLASS';  
TITLE2 'LISTING OF DESCRIPTOR PORTION OF WORK.CLASS';
```

-----  
**ERROR: Syntax error detected.**

NOTE: Expecting one of the following:

(

37 RUN ;

NOTE: Statements not processed because of errors noted above.

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

NOTE: The PROCEDURE CONTENTS used 1.00 seconds.

## Starting and Running SAS Programs

You can start a SAS session or program by entering the **SAS** command. The SAS command, as any other operating system command, is entered after the system's prompt. The syntax of the SAS command is:

**SAS** [*file\_spec options*]

where

*file\_spec* is the name of the file containing a SAS program. The *file\_spec* is only used when executing SAS in non-interactive mode(explained below).

*options* are SAS global options. These will be explained later in the course.

There are three modes of execution or environments you can use to run SAS programs:

- \* interactive display manager mode
- \* interactive line mode
- \* noninteractive or batch mode

We will only discuss interactive display manager mode and noninteractive mode in the course. These are the two most common modes of execution.

### Interactive Display Manager Mode

In this mode, SAS programs are executed using the SAS Display Manager System, which is a full screen facility with windows for editing and executing programming statements, displaying the SAS log, displaying procedure output, and more. Interactive Display Manager is invoked by double clicking on the SAS icon or by entering the SAS command at an operating system's command prompt:

**SAS** [*options*]

Where *options* are global options. For example:

SAS -nodate

We will study Interactive Display Manager more thoroughly in the next section.

### Noninteractive or Batch Mode

In noninteractive mode, SAS programming statements are stored in an external file and the statements in the file execute when you issue a SAS command referencing the file. In this mode, the SAS log and output are written to external files. They are not displayed on the screen.

Noninteractive SAS is invoked with the SAS command:

**SAS** [*file\_spec options*]

where *file\_spec* is the name of the external file containing the SAS program you want to execute. The

file\_spec should also include the path if the external file is not in the current directory. Again *options* are the global options. For example, suppose that you have a SAS program in a file called \BIOS111\PROG1.SAS . You can execute the program with the command:

```
SAS \BIOS111\PROG1.SAS
```

If PROG1.SAS is in the current directory then the path can be omitted.

### Log and Output Files

The SAS log from a SAS program run in noninteractive mode is written to a file with the same filename as the source file and .LOG as an extension. The output is written to a file with the same filename as the source file and .LST as the extension. By default, the .LOG and .LST files are written to the current directory.

For example, suppose the current directory is BIOS111 and the name of the source file is PROG1.SAS . The log and output files from executing this file in batch mode are BIOS111\PROG1.LOG and BIOS111\PROG1.LST .

If an error occurs in a SAS program run in batch mode a .LST file may not be created.