Introduction to SAS

STAT2005

Chapter 8

What is SAS?

SAS (Statistical Analysis System) is a set of solutions for enterprise-wide business users as well as a powerful programming language for performing tasks such as

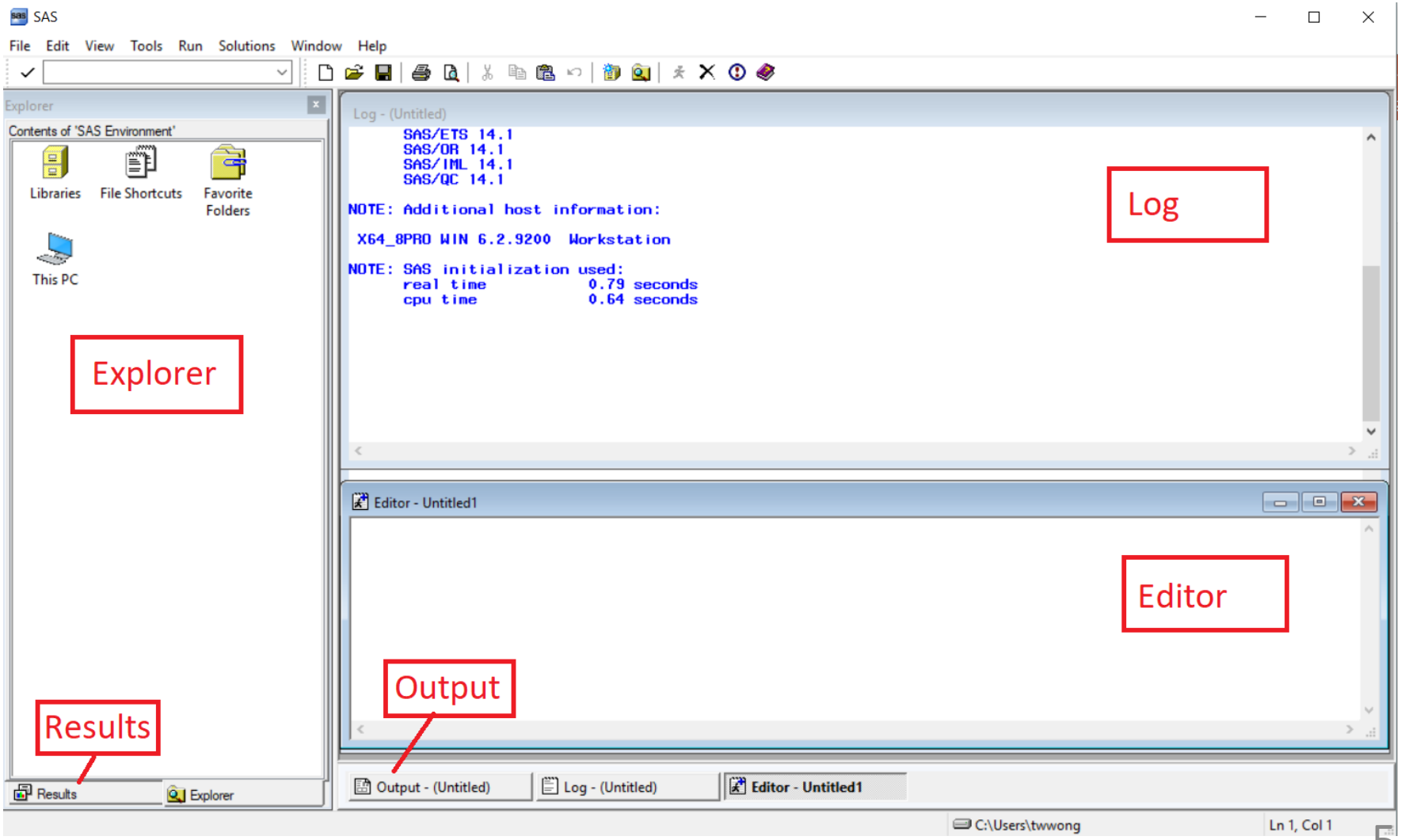
- data entry, retrieval, and management
- report writing and graphics
- statistical and mathematical analysis
- business planning, forecasting, and decision support
- operations research and project management
- quality improvement
- applications development

- The Base SAS software is the heart of the SAS System. Apart from the Base SAS software, there are many optional SAS add-on products, whose installation requires the Base SAS software.
- An important add-on product is SAS/STAT which is the software for statistical analysis, and is a product heavily used by statisticians.
- Apart from SAS/STAT, there are some other special purpose add-on products that are related to statistics, for example
  - SAS/ETS is for econometric forecasting and modelling,
  - SAS/GRAPH is for statistical graphics,
  - SAS/INSIGHT is for interactive data analyses, and
  - SAS/QC is for quality control.

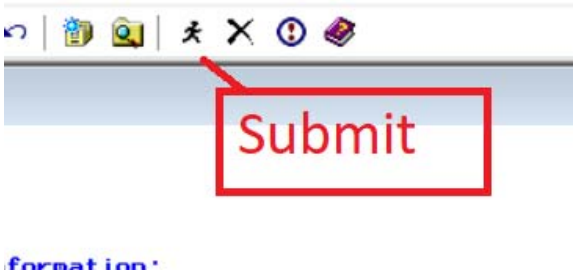
SAS user interface

- There are five basic SAS windows:
  - the Results and Explorer windows, and
  - three programming windows: Editor, Log, and Output.
- It is possible to bring up SAS without all these windows, and sometimes the windows are not immediately visible (for example, in the Windows operating environment, the Output window comes up behind the Editor and Log windows), but all these windows do exist in your SAS session.

The following figure shows the default view for a Microsoft Windows SAS session, with pointers to the five main SAS windows.



- When you are ready to execute a SAS program that you prepared in the Program Editor window, save it to your disk first, then from the RUN pull-down menu choose SUBMIT.
- Alternatively, you can click on the submit button.



- As the program is executed, a transcript of the session--with any error messages--will be written to the Log window.
- If the program runs correctly, the Output window will display the results.

- Always review the Log window after running your program--even if it appears that your program ran correctly. Some serious errors, such as misread or incorrectly aligned data, may go unnoticed otherwise. Always check for error messages and warnings in the Log window.
- If you are using the Enhanced Editor (Windows operating environment), after you submit your program, the program remains in the Enhanced Editor window and the results of your program go into the Log and Output windows.

SAS data set

Below is the typical form of a SAS data set.

Variables (Also Called Columns)

Observations (Also Called Rows)

	Id	Name	Height	Weight
1	53	Susie	42	41
2	54	Charlie	46	55
3	55	Calvin	40	35
4	56	Lucy	46	52
5	57	Dennis	44	.
6	58		43	50

Data types

- In SAS there are just two data types: numeric (real number) and character.
- Numeric value can be positive or negative, and can contain plus sign (+), minus sign (`-`), decimal point (`.`), or `e` for scientific notation.
- Examples of valid numeric constants: `10`, `-10.`, `10.0`, `1e3`, `-2e-1` (`1e3` means 10^3 and `-2e-1` means -2×10^{-1}). Numeric constants are restricted in size to about plus or minus `72e15`, the same as numeric variables.

- Character constant can contain numerals, letters or special characters (such as $ # !) and can be from 1 to 200 columns long.
- Character constants must be enclosed by a pair of single quotes (') or quotation marks ("), e.g. `"ABC Co."` and `'Block 5'`. Quotes themselves can be included in a character constant by enclosing the constant in quotes of the other type, e.g. `"Family's income"`.
- If a variable contains letters or special characters, it must be character data. However, if it contains only numbers, then it may be numeric or character.
- You should base your decision on how you will use the variable. ZIP codes, for example, are made up of numbers, but it just doesn't make sense to add, subtract, multiply, or divide ZIP codes. Such numbers make more sense as character data.

Missing value

- Missing value is a value that indicates that no data value is stored for the variable in the current observation. There are two kinds of missing values:
  - Numeric
  - Character
- By default, SAS prints
  - a missing numeric value as a single period (**.**) and
  - a missing character value as a blank space.

Naming SAS variables

Name in SAS follows the following rules:

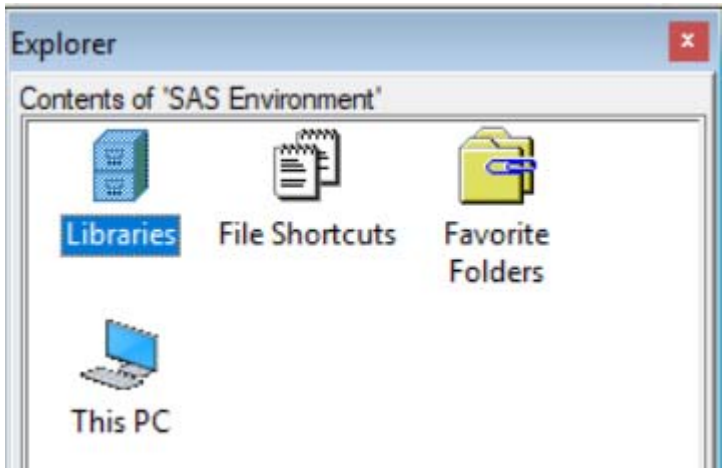
- It is safe to have names with length not exceeding 8 characters (many SAS versions allow length up to 32).
- Names must start with an alphabetic (a through z) character or an underscore ("_"; not the minus sign,"-").
- Names can contain only alphabetic characters (a through z), underscores, and numbers (0 through 9).
- Names can contain upper- and lowercase letters.

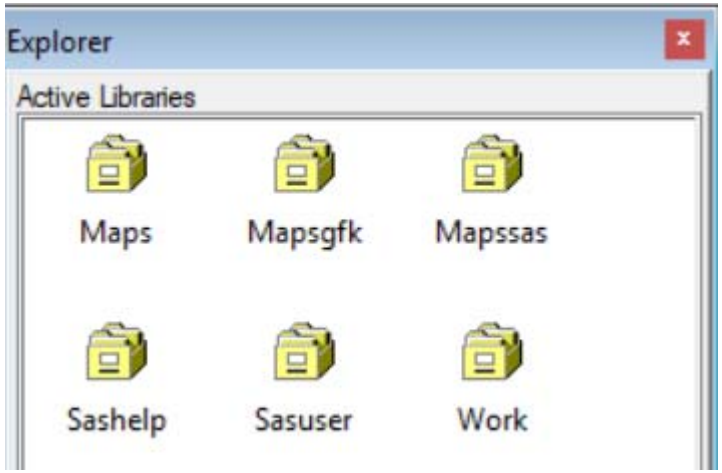
- Although SAS is case insensitive, there is one difference for variable names. SAS remembers the case of the first occurrence of each variable name and uses that case when printing results.
- It is not suggested to use keywords of the SAS system (such as **DATA** , INPUT) as user-supplied variable name. SAS reserves a few names for automatic variables and variable lists, SAS data sets, and librefs (SAS libraries).
- To be safe, it is not recommended to name a variable or data set with a beginning and ending underscore (SAS reserves a few names of this form for special purpose, such as _ALL_, _NULL_, _N_).

SAS Data Libraries

- Before you can use a SAS data set, you have to tell SAS where to find it.
- You do that by setting up a SAS library.
- A SAS library is simply a location where SAS data sets (as well as other types of SAS files) are stored.
- Depending on your operating environment, a SAS library might be a folder or directory on your computer, or it might be a physical location like a hard drive, USB storage, or CD-ROM.

- To set up a library, all you have to do is to make up a name for your library and tell SAS where it is.
- There are several ways to do this including using the LIBNAME statement (to be studied in the next chapter) or using the New Library window in the SAS Windowing Environment.
- If you double-click on the Libraries icon, Explorer will open the Active Libraries window showing all the libraries that are currently defined.





The Active Libraries window

- When you open the Active Libraries window, you will see at least three libraries: `Sashelp`, `Sasuser`, and `Work`.
- You may have other libraries for specific SAS products (such as the `Maps` library for SAS/GRAPH software), or libraries that have been set up by you or someone you work with.
- The `Work` library is a temporary storage location for SAS data sets. It is also the default library. If you create a SAS data set without specifying a library, SAS will put it in the `Work` library, and then delete it when you end your session.
- In Example 2, the `fitness` data set is in the `Work` library.

Creating a new library

- You can create new SAS libraries using the `New Library` window.
- To open this window, either left click in the `Active Libraries` window (to make it active) and choose `New` from the File menu, or right click in the `Active Libraries` window and choose `New` from the pop-up menu.
- You can then input a name (say `ABC`) for the new library and input a Path (say `C:\`). In that way, the directory `C:\` will be represented as a library called `ABC` in SAS.

SAS programs

- With SAS, we use statements to write a series of instructions called a SAS program. The program communicates what we want to do and is written using the SAS language.
- A SAS program is a sequence of statements (a series of instructions) executed in order.
- These instructions typically start by some definitions detailing where to find data files, what format the data files are written in, what to call the files, and how to input the data.

- After this, normally one would write instructions to read the data into a SAS data set (a form that SAS can use while it's running).
- As the data are read into a SAS data set, variables may be recoded and/or modified, other variables can be created and added into the data set.
- A series of data manipulations, including very complicated restructuring, can be done as data are input into a SAS data set.
- After the instructions to create a SAS data set, the subsequent instructions to tell SAS which statistical procedures to perform and with what specifications.

DATA and PROC steps

- SAS programs are constructed from two basic building blocks: DATA steps and PROC steps.
- A typical program starts with a DATA step to create a SAS data set and then passes the data to some PROC steps for processing.

Example 1: A simple program

```
DATA distance;
     Miles = 26.22;
     Kilometers = 1.61 * Miles;

PROC PRINT DATA = distance;
RUN;
```

DATA Step

- The DATA Step has a number of different functions. A major function is to convert raw data into a SAS data set.
- In addition, DATA Step provides programming functionality similar to other programming languages.
- For example, the control flow statements such as `IF-THEN-ELSE` statement, `DO` loop statement and etc. can be used to alter the sequence of actions within a data set.
- Therefore, the DATA Step can range from the very simple to very complex.
- We shall focus on DATA step in this course.

PROC Step

- SAS PROCs are pre-written programs that work with SAS data sets.
- Using SAS PROC is like calling a function with the data as input.
- For example, `PROC MEANS` could be applied to compute sample mean. `PROC PRINT` is for printing results.
- Basically, no programming functionality is provided within a PROC.
- However, a SAS PROC can communicate with other SAS DATA Steps or PROCs by outputting data sets.

Syntax of SAS statements

There are some general basic rules of writing up a SAS program:

- Every statement ends with a semicolon.
- SAS statements can be in UPPERCASE or lowercase.
- Statements can continue on the next line (as long as you don’t split a word into two).
- Multiple SAS statements can appear on a single line.
- Statements can start in any column.

- Words in SAS statements are separated by blanks or by special characters (e.g. =, +, *).
- A blank is not treated as a character in a SAS statement unless it is enclosed in quotation marks as a literal or part of a literal. Therefore, you can put multiple blanks on any place in a SAS statement where you can put a single blank, with no effect on the syntax.
- It is a good programming practice to enter **RUN ;** statement at the end of a unit of work. It informs SAS that a unit ends here and asks SAS to execute that unit.

Comments

- Comment statements allow you to document your program without affecting processing.
- There are two kinds of comments in SAS:
- One starts with an asterisk (*) and ends with a semicolon (;).
- The other style starts with a slash asterisk (/*) and ends with an asterisk slash (*/).
- The following SAS program shows the use of both of these style comments (words in green are comments):

A sample program

```
* Read animals' weights from file;
DATA animals;
    INFILE 'C:\MyData\Zoo.dat';
    INPUT Lions Tigers;
RUN;
PROC PRINT DATA = animals;
/* Print the results */
RUN;
```

Creating variables

In a DATA Step, there are two ways to create variables.

- The first way is to assign a value to a variable, such as

```
Miles = 26.22;
```

In this way, the variable `Miles` will only get one single value.

- The second way is to use an `INPUT` statement to read in-stream data or external data files, as shown in the next example

Note: If there are multiple observations of `Miles`, and we take

```
Kilometers = 1.61 * Miles;
```

then `Kilometers` is a variable that depends on `Miles` for each observations.

Example 2

```
* Create a data set called "fitness";

DATA fitness;
   INPUT name $ weight waist pulse
chins situps jumps;
   CARDS;
   Hodges     191   36   50    5   162    60
   Kerr       189   37   52    2   110    60
   Putnam     193   38   58   12   101   101
   Roberts    162   35   62   12   105    37
   Blake      189   35   46   13   155    58
RUN;
```

```
/* Sort the data set "fitness" by name: */

PROC SORT;
  BY name;
RUN;

/* Print out the sorted data set: */

PROC PRINT data=fitness;
  title 'Fitness data';
RUN;

/* Get some descriptive statistics: */

PROC MEANS maxdec=1 data=fitness;
RUN;
```

Example 3

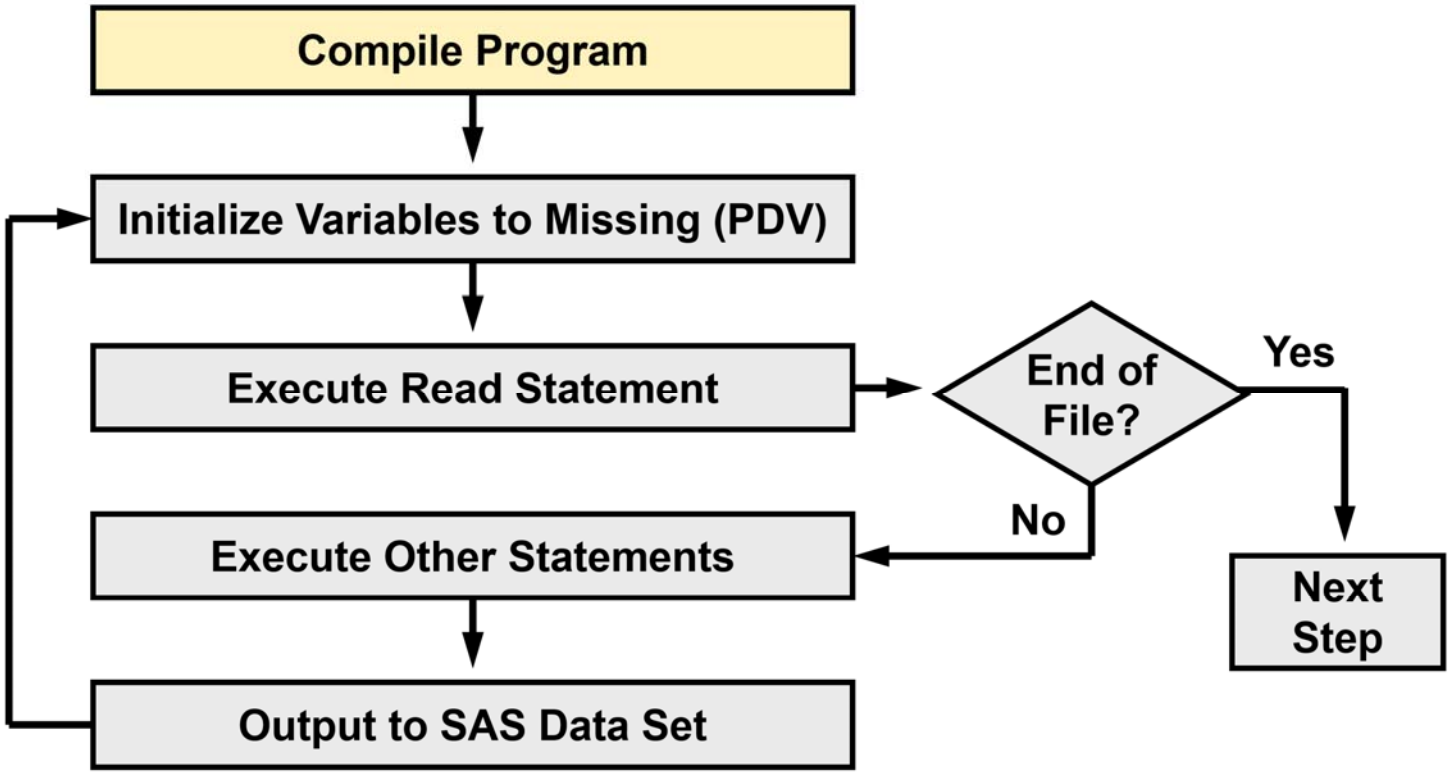
We can also mix the two variable creating methods as shown below.

```
DATA example3;
     INPUT Z;
     X = 5;
     Y = 2 * Z;
     CARDS;
     1
     2
RUN;
PROC PRINT data=example3;
RUN;
```

Obs	Z	X	Y
1	1	5	2
2	2	5	4

DATA Step execution details

When a DATA step is submitted, it is first compiled and then executed.



Compilation phase

During the compilation phase, SAS does the following:

- Checks the syntax of the SAS statements
- Translates the statements into machine code
- Identifies the name, type, and length of each variable

The following three items are potentially created:

- Input Buffer
- Program Data Vector (PDV)
- Descriptor Information

Input buffer

The input buffer is a logical area in memory into which SAS reads each record of a raw data file when SAS executes an INPUT statement.

- This buffer is created only when the DATA step reads raw data.
- When the DATA step reads a SAS data set, SAS reads the data directly into the program data vector.

Program Data Vector (PDV)

The PDV is a logical area in memory where SAS builds a data set, one observation at a time. Along with data set variables and computed variables, the PDV contains the following two automatic variables:

- The **_N_** variable, which counts the number of times the DATA step begins to iterate
- The **_ERROR_** variable, which signals the occurrence of an error caused by the data during execution

  The value of **_ERROR_** is either 0 (indicating no errors exist) or 1 (indicating that one or more errors occurred).

Descriptor information

The descriptor portion is information that SAS creates and maintains about each SAS data set, including data set attributes and variable attributes.

Examples of descriptor information include the following:

- The name of the data set
- The date and time that the data set was created
- The names, data types (character or numeric), and lengths of the variables

Execution phase

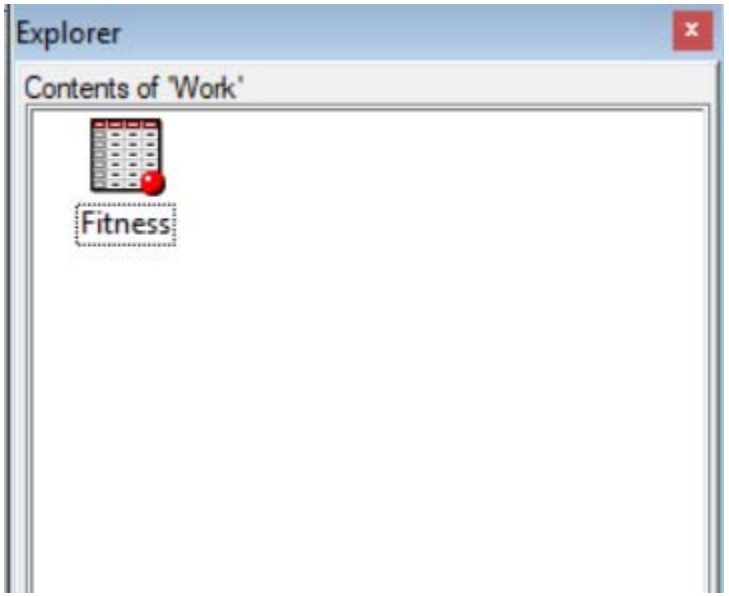
During the execution phase, SAS does the following:

- Initializes the PDV to missing and sets the initial values of **_N_** and **_ERROR_**
- Reads data values into the PDV
- Executes any subsequent programming statements
- Outputs the observation to a SAS data set
- Returns to the top of the DATA step
- Resets the PDV to missing for any variables not read directly from a data set and increments **_N_** by 1
- Repeats the process until the end-of-file is detected

This is the default flow of the DATA *step execution phase. Options and the placement of statements can alter this flow.*

Viewing data sets with SAS Explorer

- In addition to listing your current libraries and creating new libraries, you can also use SAS Explorer to open SAS data sets for viewing and editing, or to list information about their contents such as the date of the data set being created and the names of variables.
- If you double-click on a library, SAS will open a Contents window showing you all the files (including SAS data sets) and folders in that particular library.
- This window shows the contents of a library. If you double-click on a data set, SAS will open a Viewtable window showing that data set. This window allows you to create, browse, and edit data sets.



VIEWTABLE: Work.Fitness

	name	weight	waist	pulse	chins	situps	jumps
1	Blake	189	35	46	13	155	58
2	Hodges	191	36	50	5	162	60
3	Kerr	189	37	52	2	110	60
4	Putnam	193	38	58	12	101	101
5	Roberts	162	35	62	12	105	37

Changing column headings

- By default, Viewtable uses variable labels for column headings, or, if a variable does not have a label, the variable name is displayed.
- Sometimes you may want to see the actual variable names instead of the labels.
- To do this, click the Viewtable window to make it active, then select `View > Column Names` from the menu bar.