SAS Certification Prep Guide: Base Programming for SAS 9, Third Edition

by SAS Institute

Chapter 5: Creating SAS Data Sets from External Files

Overview

Introduction

In order to create reports with SAS procedures, your data must be in the form of a SAS data set. If your data is not stored in the form of a SAS data set, you need to create a SAS data set by entering data, by reading raw data, or by accessing files that were created by other software.

This chapter shows you how to design and write DATA step programs to create SAS data sets from raw data that is stored in an external file and from data stored in Microsoft Excel worksheets. It also shows you how to read data from a SAS data set and write observations out to these destinations.

Regardless of the input data source — raw files or Excel worksheets — you use the DATA step to read in the data and create the SAS data set.

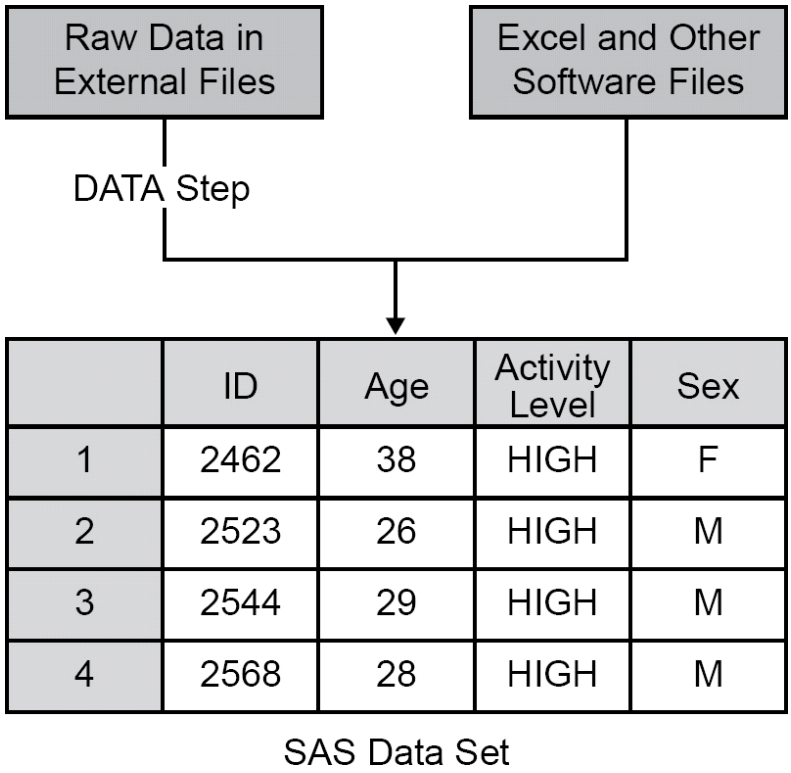


Figure 5.1: Using the DATA Step to Create SAS Data Sets

Objectives

In this chapter, you learn to

- reference a SAS library
- reference a raw data file
- name a SAS data set to be created
- specify a raw data file to be read
- read standard character and numeric values in fixed fields
- create new variables and assign values
- select observations based on conditions

- read instream data
- submit and verify a DATA step program
- read a SAS data set and write the observations out to a raw data file.
- use the DATA step to create a SAS data set from an Excel worksheet
- use the SAS/ACCESS LIBNAME statement to read from an Excel worksheet
- create an Excel worksheet from a SAS data set
- use the IMPORT procedure to read external files

Raw Data Files

A raw data file is an external text file whose records contain data values that are organized in fields. Raw data files are non-proprietary and can be read by a variety of software programs. The sample raw data files in this chapter are shown with a ruler to help you identify where individual fields begin and end. The ruler is not part of the raw data file.

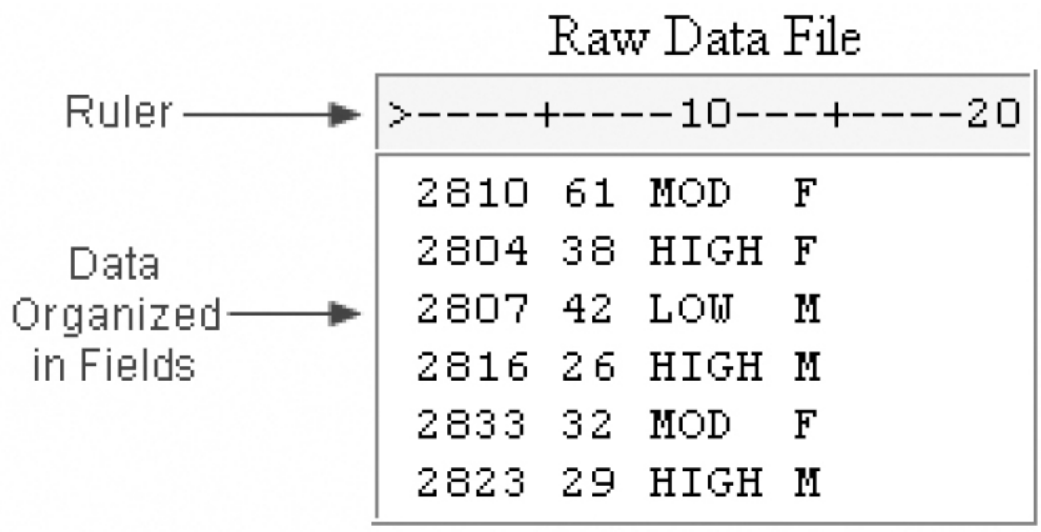


Figure 5.2: Raw Data File

The table below describes the record layout for a raw data file that contains readings from exercise stress tests that have been performed on patients at a health clinic. Exercise physiologists in the clinic use the test results to prescribe various exercise therapies. The file contains fixed fields. That is, values for each variable are in the same location in all records.

Table 5.1: Record Layout for Raw Data

Field Name	Starting Column	Ending Column	Description of Field
ID	1	4	patient ID number
Name	6	25	patient name
RestHR	27	29	resting heart rate
MaxHR	31	33	maximum heart rate during test
RecHR	35	37	recovery heart rate after test
TimeMin	39	40	time, complete minutes
TimeSec	42	43	time, seconds
Tolerance	45	45	comparison of stress test tolerance between this test and the last test (I=increased, D=decreased, S=same, N=no previous test)

Steps to Create a SAS Data Set from a Raw Data File

Let's look at the steps for creating a SAS data set from a raw data file. In the first part of this chapter, you will follow these steps to create a SAS data set from a raw data file that contains fixed fields.

Before reading raw data from a file, you might need to reference the SAS library in which you will store the data set. Then you can write a DATA step program to read the raw data file and create a SAS data set.

To read the raw data file, the DATA step must provide the following instructions to SAS:

- the location or name of the external text file
- a name for the new SAS data set
- a reference that identifies the external file
- a description of the data values to be read.

After using the DATA step to read the raw data, you can use a PROC PRINT step to produce a report that displays the data values that are in the new data set.

The table below outlines the basic statements that are used in a program that reads raw data in fixed fields. Throughout this chapter, you'll see similar tables that show sample SAS statements.

Table 5.2: Basic Statements for Reading Raw Data

To do this…	Use this SAS statement…
Reference SAS data library	LIBNAME statement
Reference external file	FILENAME statement
Name SAS data set	DATA statement
Identify external file	INFILE statement
Describe data	INPUT statement
Execute DATA step	RUN statement
Print the data set	PROC PRINT statement
Execute final program step	RUN statement

You can also use additional SAS statements to perform tasks that customize your data for your needs. For example, you may want to create new variables from the values of existing variables.

Referencing a SAS Library

Using a LIBNAME Statement

As you begin to write the program, remember that you might need to use a LIBNAME statement to reference the permanent SAS library in which the data set will be stored.

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname taxes 'c: \users\name \sasuser';`

For example, the LIBNAME statement below assigns the libref Taxes to the SAS library in the folder **`C:\Acct\Qtr1 \Report`** in the Windows environment.

```
libname taxes 'c:\acct\qtrl\report';
```

You do not need to use a LIBNAME statement in all situations. For example, if you are storing the data set in a temporary SAS data set or if SAS has automatically assigned the libref for the permanent library that you are using.

Additional Note Many of the examples in this chapter use the libref Sasuser, which SAS automatically assigns.

Referencing a Raw Data File

Using a FILENAME Statement

When reading raw data, you can use the FILENAME statement to point to the location of the external file that contains the data. Just as you assign a libref by using a LIBNAME statement, you assign a fileref by using a FILENAME statement.

Table 5.3: Referencing a Raw Data File

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname libref 'SAS-data-library';`
Reference an external file	FILENAME statement	`filename tests 'c: \users\tmill.dat';`

Filerefs perform the same function as librefs: they temporarily point to a storage location for data. However, librefs reference SAS data libraries, whereas filerefs reference external files.

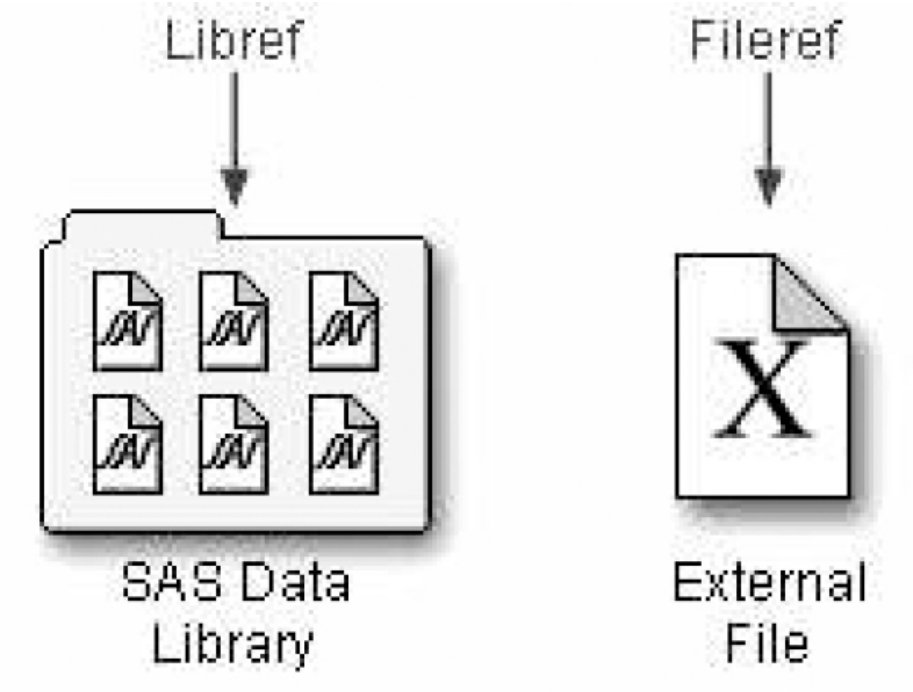


Figure 5.3: Filerefs and Librefs

General form, FILENAME statement:
`FILENAME` *`fileref'filename';`*

where

- *fileref* is a name that you associate with an external file. The name must be 1 to 8 characters long, begin with a letter or underscore, and contain only letters, numerals, or underscores.
- *'filename'* is the fully qualified name or location of the file.

Defining a Fully Qualified Filename

The following FILENAME statement temporarily associates the fileref Tests with the external file that contains the data from the exercise stress tests. The complete filename is specified as **`C:\Users\Tmill.dat`** in the Windows environment.

```
filename tests 'c:\users\tmill.dat';
```

Raw Data File Tests

```
1---+----10---+----20---+----30---+----40---+--
2458 Murray, W            72   185 128 12 38 D
2462 Almers, C            68   171 133 10 5  I
2501 Bonaventure, T       78   177 139 11 13 I
2523 Johnson, R           69   162 114 9  42 S
2539 LaMance, K           75   168 141 11 46 D
2552 Reberson, P          69   158 139 15 41 D
```

Figure 5.4: File Location

Defining an Aggregate Storage Location

You can also use a FILENAME statement to associate a fileref with an aggregate storage location, such as a directory that contains multiple external files.

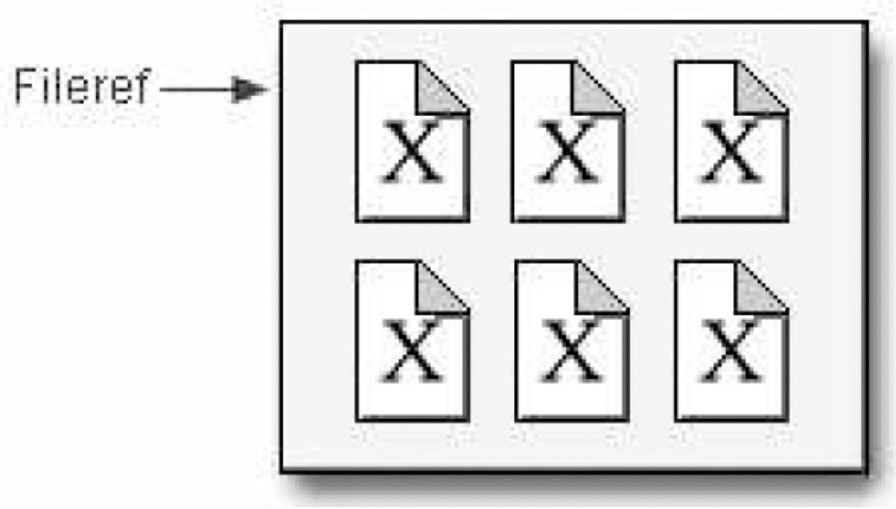


Figure 5.5: Aggregate Storage Location

This FILENAME statement temporarily associates the fileref Finance with the aggregate storage directory **C:\Users\Personal\Finances:**

```
filename finance 'c:\users\personal\finances';
```

Viewing Active Filerefs

Like librefs, the filerefs currently defined for your SAS session are listed in the SAS Explorer window.

To view details about a referenced file, double-click **File Shortcuts** (or select **File Shortcuts** and then **Open** from the pop-up menu). Then select **View** ⇨ **Details**. Information for each file (name, size, type, and host path name) is listed.

Additional Note Both the LIBNAME and FILENAME statements are global. In other words, they remain in effect until you change them, cancel them, or end your SAS session.

Referencing a Fully Qualified Filename

When you associate a fileref with an individual external file, you specify the fileref in subsequent SAS statements and commands.

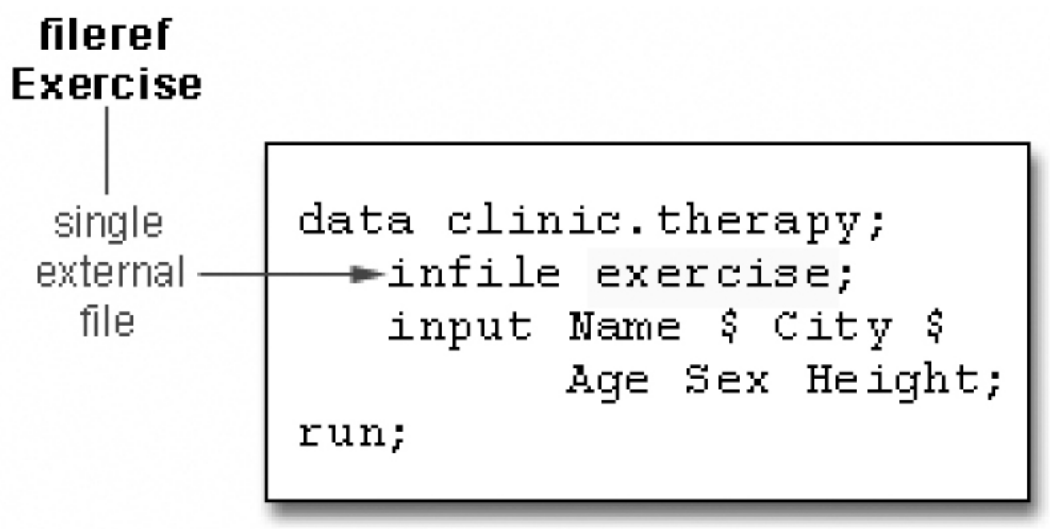


Figure 5.6: Referencing a Fully Qualified Filename

Referencing a File in an Aggregate Storage Location

To reference an external file with a fileref that points to an aggregate storage location, you specify the fileref followed by the individual filename in parentheses:

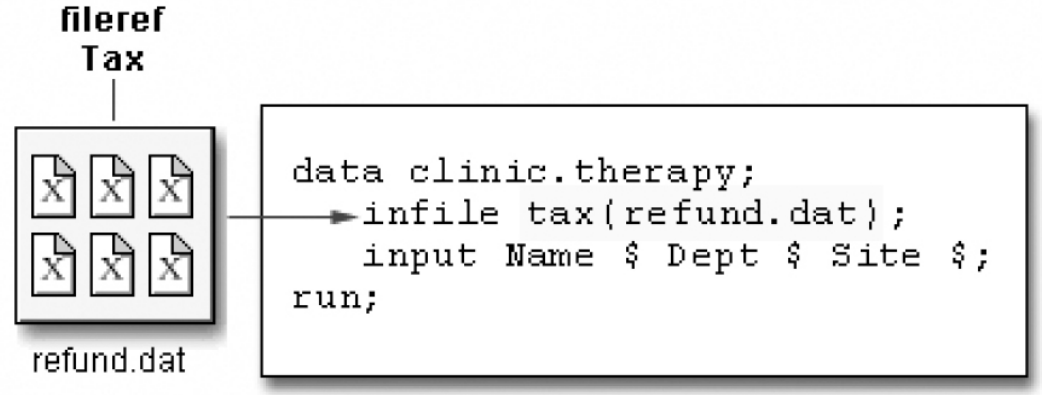


Figure 5.7: Referencing a File in an Aggregate Storage Location

Additional Note In the Windows environment, you can omit the filename extension but you will need to add quotation marks when referencing an external file, as in

```
infile tax('refund');
```

For details on referencing external files stored in aggregate storage locations, see the SAS documentation for your operating environment.

Writing a DATA Step Program

Naming the Data Set

The DATA statement indicates the beginning of the DATA step and names the SAS data set to be created.

Table 5.4: Naming the Data Set

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname libref 'SAS-data-library';`
Reference an external file	FILENAME statement	`filename tests 'c: \users\tmill.dat';`
Name a SAS data set	DATA statement	`data clinic.stress;`

General form, basic DATA statement:
DATA *SAS-data-set-1 <…SAS-data-set-n>;*

where *SAS-data-set* names (in the format *libref.flename)* the data set or data sets to be created.

Remember that the SAS data set name is a two-level name. For example, the two-level name Clinic.Admit specifies that the data set Admit is stored in the permanent SAS library to which the libref Clinic has been assigned.

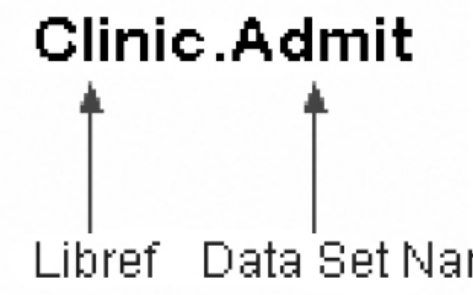


Figure 5.8: Two-Level Names

Specifying the Raw Data File

When reading raw data, use the INFILE statement to indicate which file the data is in.

Table 5.5: Specifying the Raw Data File

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname libref 'SAS-data-library';`
Reference an external file	FILENAME statement	`filename tests 'c: \users\tmill.dat';`
Name a SAS data set	DATA statement	`data clinic.stress;`
Identify an external file	INFILE statement	`infile tests;`

General form, INFILE statement:
INFILE *file-specification <options>;*

where

- *file-specification* can take the form *fileref* to name a previously defined file reference or *'filename'* to point to the actual name and location of the file
- *options* describe the input file's characteristics and specify how it is to be read with the INFILE statement.

To read the raw data file to which the fileref Tests has been assigned, you write the following INFILE statement:

```
infile tests;
```

Additional Note Instead of using a FILENAME statement, you can choose to identify the raw data file by specifying the entire filename and location in the INFILE statement. For example, the following statement points directly to the **C:\lrs\Personal\Refund.dat** file:

```
infile 'c:\irs\personal\refund.dat';
```

Column Input

Column input specifies actual column locations for values. However, column input is appropriate only in certain situations. When you use column input, your data *must* be

- standard character or numeric values
- in fixed fields.

Standard and Nonstandard Numeric Data

Standard numeric data values can contain only

- numerals
- decimal points
- numbers in scientific or E-notation (2.3E4, for example)
- plus or minus signs.

Nonstandard numeric data includes

- values that contain special characters, such as percent signs (%), dollar signs ($), and commas (,)
- date and time values
- data in fraction, integer binary, real binary, and hexadecimal forms.

The external file that is referenced by the fileref Staff contains the personnel information for a technical writing department of a small computer manufacturer. The fields contain values for each employee's last name, first name, job code, and annual salary.

Notice that the values for Salary contain commas. So, the values for Salary are considered to be nonstandard numeric values. You cannot use column input to read these values.

Raw Data File Staff

```
1---+----10---+----20---+---
EVANS    DONNY 112 29,996.63
HELMS    LISA  105 18,567.23
HIGGINS  JOHN  111 25,309.00
LARSON   AMY   113 32,696.78
MOORE    MARY  112 28,945.89
```

Figure 5.9: Raw Data File

Fixed-Field Data

Raw data can be organized in several different ways.

The following external delimited file contains data that is not arranged in columns, meaning data that is not arranged in columns. Notice that the values for a particular field do not begin and end in the same columns. You cannot use column input to read this file.

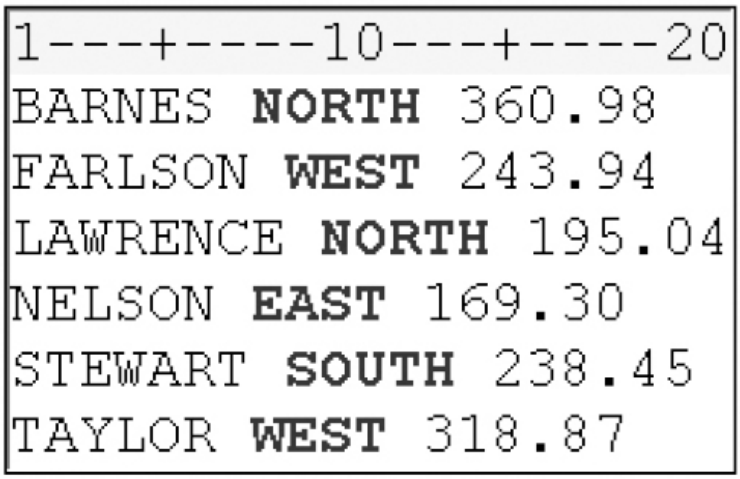


Figure 5.10: Fixed Field Data

The following external file contains data that is arranged in columns or fixed fields. You can specify a beginning and ending column for each field. Let's look at how column input can be used to read this data.

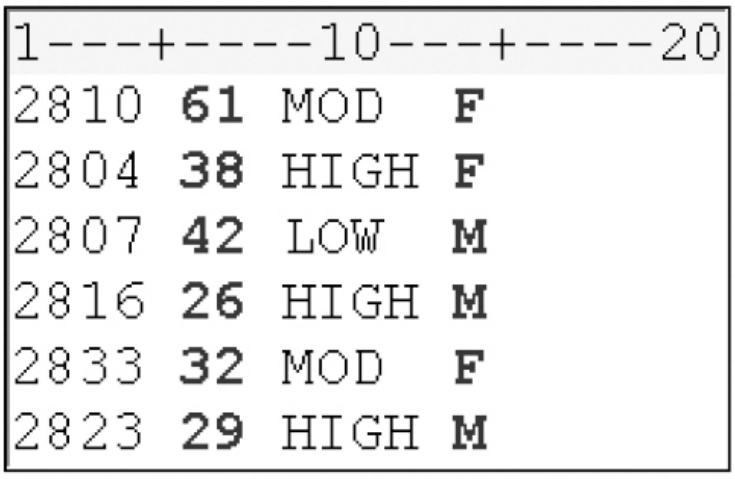


Figure 5.11: External File with Columns

Additional Note If you are not familiar with the content and structure of your raw data files, you can use PROC FSLIST to view them.

Describing the Data

The INPUT statement describes the fields of raw data to be read and placed into the SAS data set.

Table 5.6: Describing the Data

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname libref 'SAS-data-library';`
Reference an external file	FILENAME statement	`filename tests 'c: \users\tmill.dat';`
Name a SAS data set	DATA statement	`data clinic.stress;`
Identify an external file	INFILE statement	`infile tests;`
Describe data	INPUT statement	`input ID $ 1-4 Name $ 6-25 …;`
Execute the DATA step	RUN statement	`run;`

General form, INPUT statement using column input:
INPUT *variable* <$>*startcol-endcol…*;

where

- *variable* is the SAS name that you assign to the field
- the dollar sign ($) identifies the variable type as character (if the variable is numeric, then nothing appears here)
- *startcol* represents the starting column for this variable
- *endcol* represents the ending column for this variable.

Look at the small data file shown below. For each field of raw data that you want to read into your SAS data set, you must specify the following information in the INPUT statement:

- a valid SAS variable name
- a type (character or numeric)
- a range (starting column and ending column).

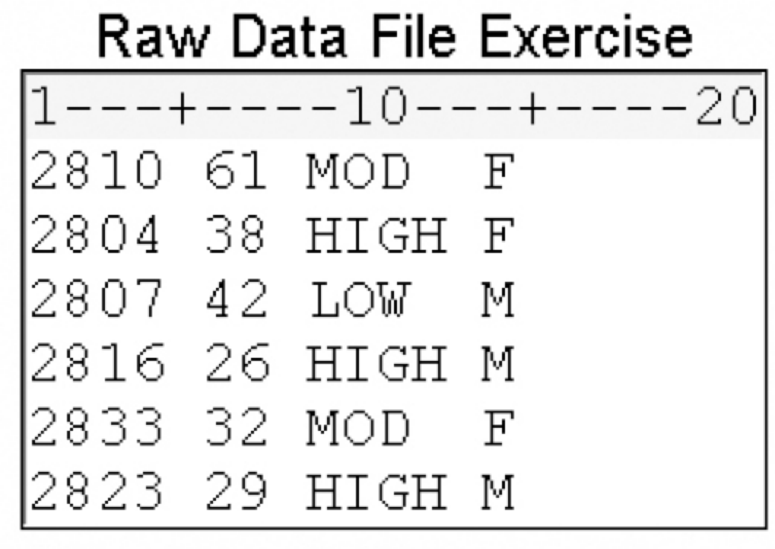


Figure 5.12: Raw Data File

The INPUT statement below assigns the character variable ID to the data in columns 1-4, the numeric variable Age to the data in columns 6-7, the character variable ActLevel to the data in columns 9-12, and the character variable Sex to the data in column 14.

```
filename exer 'c:\users\exer.dat';
data exercise;
   infile exer;
   input ID $ 1-4 Age 6-7 ActLevel $ 9-12 Sex $ 14;
run;
```

SAS Data Set Work. Exercise

Obs	ID	Age	ActLevel	Sex
1	2810	61	MOD	F
2	2804	38	HIGH	F
3	2807	42	LOW	M
4	2816	26	HIGH	M
5	2833	32	MOD	F

6	2823	29	HIGH	M

Figure 5.13: Assigning Column Ranges to Variables

When you use column input, you can

- read any or all fields from the raw data file
- read the fields in any order
- specify only the starting column for values that occupy only one column.

```
input ActLevel $ 9-12 Sex $ 14 Age 6-7;
```

Additional Note Remember that when you name a new variable, you must specify the name in the exact case that you want it stored, for example NewBalance. Thereafter, you can specify the name in uppercase, lowercase, or mixed case.

Specifying Variable Names

Each variable has a name that conforms to SAS naming conventions. Variable names

- must be 1 to 32 characters in length
- must begin with a letter (A-Z) or an underscore (_)
- can continue with any combination of numerals, letters, or underscores.

Let's look at an INPUT statement that uses column input to read the three data fields in the raw data file below.

Raw Data File Admit

```
1---+----10---+----20
58MOD M
29LOW F
34LOW M
41HIGHF
30MOD F
22HIGHM
```

Figure 5.14: Raw Data File

The values for the variable Age are located in columns 1-2. Because Age is a numeric variable, you do not specify a dollar sign ($) after the variable name.

```
input Age 1-2
```

The values for the variable ActLevel are located in columns 3-6. You specify a $ to indicate that ActLevel is a character variable.

```
input Age 1-2 ActLevel $ 3-6
```

The values for the character variable Sex are located in column 7. Notice that you specify only a single column.

```
input Age 1-2 ActLevel $ 3-6 Sex $ 7;
```

Submitting the DATA Step Program

Verifying the Data

To verify your data, it is a good idea to use the OBS= option in the INFILE statement. Adding OBS=*n* to the INFILE statement enables you to process only records 1 through n, so you can verify that the correct fields are read before reading the entire data file.

The program below reads the first ten records in the raw data file referenced by the fileref Tests. The data is stored in a permanent SAS data set, named Sasuser.Stress. Don't forget a RUN statement, which tells SAS to execute the previous SAS statements.

```
data sasuser.stress;
   infile tests obs=10;
   input ID $ 1-4 Name $ 6-25
         RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40
         TimeSec 42-43 Tolerance $ 45;
run;
```

ID	Name	RestHR	MaxHR	RecHR	TimeMin	TimeSec	Tolerance
2458	Murray, W	72	185	128	12	38	D
2462	Almers, C	68	171	133	10	5	I
2501	Bonaventure, T	78	177	139	11	13	I
2523	Johnson, R	69	162	114	9	42	S
2539	LaMance, K	75	168	141	11	46	D
2544	Jones, M	79	187	136	12	26	N
2552	Reberson, P	69	158	139	15	41	D
2555	King, E	70	167	122	13	13	I
2563	Pitts, D	71	159	116	10	22	S
2568	Eberhardt, S	72	182	122	16	49	N

Figure 5.15: SAS Data Set sasuser.stress

Checking DATA Step Processing

If you submit the DATA step below it will run successfully.

```
data sasuser.stress;
   infile tests obs=10;
   input ID $ 1-4 Name $ 6-25
         RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40
         TimeSec 42-43 Tolerance $ 45;
run;
```

Messages in the log verified that the raw data file was read correctly. The notes in the log indicate that

- 10 records were read from the raw data file
- the SAS data set Sasuser.Stress was created with 10 observations and 8 variables.

SAS Log

```
NOTE: The infile TESTS is:
      File Name=C:\My SAS Files\tests.dat,
      RECFM=V,LRECL=256

NOTE: 10 records were read from the infile TESTS.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: The data set SASUSER.STRESS has 10 observations
      and 8 variables.
NOTE: DATA statement used 0.07 seconds
```

Figure 5.16: SAS Log

Printing the Data Set

The messages in the log seem to indicate that the DATA step program correctly accessed the raw data file. But it is a good idea to look at the ten observations in the new data set before reading the entire raw data file. You can submit a PROC PRINT step to view the data.

Table 5.7: Printing the Data Set

To do this…	Use this SAS statement…	Example
Reference a SAS library	LIBNAME statement	`libname libref 'SAS-data-library';`
Reference an external file	FILENAME statement	`filename tests 'c: \users\tmill.dat';`
Name a SAS data set	DATA statement	`data clinic.stress;`
Identify an external file	INFILE statement	`infile tests obs=10;`
Describe data	INPUT statement	`input ID $ 1-4 Name $ 6-25 …;`
Execute the DATA step	RUN statement	`run;`
Print the data set	PROC PRINT statement	`proc print data=clinic.stress;`
Execute the final program step	RUN statement	`run;`

The following PROC PRINT step prints the Sasuser.Stress data set.

```
proc print data=sasuser.stress;
run;
```

The PROC PRINT output indicates that the variables in the Sasuser.Stress data set were read correctly for the first ten records.

Obs	ID	Name	RestHR	MaxHR	RecHR	TimelVlin	TlmeSec	Tolerance
1	2458	Murray, W	72	186	128	12	38	D
2	2462	Almers, C	68	171	133	10	5	I
3	2501	Bonaventura, T	78	177	139	11	13	I
4	2523	Johnson, R	69	162	114	9	42	S
5	2539	LaMance, K	75	168	141	11	46	D
6	2544	Jones, M	79	187	136	12	26	I
7	2552	Reberson, P	69	158	139	15	41	D
8	2555	King, E	70	167	122	13	13	I
9	2563	Pitts, D	71	159	116	10	22	S
10	2568	Eberhardt, S	72	182	122	16	49	N

Figure 5.17: PROC Print Output

Reading the Entire Raw Data File

Now that you've checked the log and verified your data, you can modify the DATA step to read the entire raw data file. To do so, remove the OBS= option from the INFILE statement and re-submit the program.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25
         RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40
         TimeSec 42-43 Tolerance $ 45;
run;
```

Invalid Data

When you submit the revised DATA step and check the log, you see a note indicating that invalid data appears for the variable RecHR in line 14 of the raw data file, columns 35-37.

This note is followed by a column ruler and the actual data line that contains the invalid value for RecHR.

SAS Log

```
NOTE: Invalid data for RecHR in line 14 35-37.
RULE:     ----+----1----+----2----+----3----+----4----+----5---
14        2575 Quigley, M           74  152 Q13 11 26 I 45
ID=2575 Name=Quigley, M RestHR=74 MaxHR=152 RecHR=. TimeMin=11
TimeSec=26 Tolerance=I _ERROR_=1
_N_=14
NOTE: 21 records were read from the infile TESTS.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: The data set SASUSER.STRESS has 21 observations
      and 8 variables.
NOTE: DATA statement used 0.13 seconds
```

Figure 5.18: SAS Log

The value **Q13** is a data-entry error. It was entered incorrectly for the variable RecHR.

RecHR is a numeric variable, but **Q13** is not a valid number. So RecHR is assigned a missing value, as indicated in the log. Because RecHR is numeric, the missing value is represented with a period.

Notice, though, that the DATA step does not fail as a result of the invalid data but continues to execute. Unlike syntax errors, invalid data errors do not cause SAS to stop processing a program.

Assuming that you have a way to edit the file and can justify a correction, you can correct the invalid value and rerun the DATA step. If you did this, the log would then show that the data set Sasuser.Stress was created with 21 observations, 8 variables, and no messages about invalid data.

SAS Log

```
NOTE: The infile TESTS2 is:
      File Name=C:\My SAS Files\tests2.dat,
      RECFM=V,LRECL=256

NOTE: 21 records were read from the infile TESTS2.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: The data set SASUSER.STRESS has 21 observations
      and 8 variables.
NOTE: DATA statement used 0.14 seconds
```

Figure 5.19: SAS Log

After correcting the raw data file, you can print the data again to verify that it is correct.

```
proc print data=sasuser.stress;
run;
```

Obs	ID	Name	RestHR	MaxHR	RecHR	TimeMin	TimeSec	Tolerance
1	2458	Murray, W	72	185	128	12	38	D
2	2462	Almers, C	68	171	133	10	5	I
3	2501	Bonaventura, T	78	177	139	11	13	I
4	2523	Johnson, R	69	162	114	9	42	S
5	2539	LaMance, K	75	168	141	11	46	D
6	2544	Jones, M	79	187	136	12	26	I
7	2552	Reberson, P	69	158	139	15	41	D
8	2555	King, E	70	167	122	13	13	I
9	2563	Pitts, D	71	159	116	10	22	S
10	2568	Eberhardt, S	72	182	122	16	49	I
11	2571	Nunnelly, A	65	181	141	15	2	I
12	2572	Oberon, M	74	177	138	12	11	D
13	2574	Peterson, V	80	164	137	14	9	D
14	2575	Quigley, M	74	152	113	11	26	I
15	2578	Cameron, L	75	158	108	14	27	I
16	2579	Underwood, K	72	165	127	13	19	S
17	2584	Takahashi, Y	76	163	135	16	7	D
18	2586	Derber, B	68	176	119	17	35	I
19	2588	Ivan, H	70	182	126	15	41	I
20	2589	Wilcox, E	78	189	138	14	57	I
21	2595	Warren, C	77	170	136	12	10	S

Figure 5.20: PROC Print Output

Whenever you use the DATA step to read raw data, remember the steps that you followed in this chapter, which help ensure that you don't waste resources when accessing data:

- write the DATA step using the OBS= option in the INFILE statement
- submit the DATA step

- check the log for messages
- view the resulting data set
- remove the OBS= option and resubmit the DATA step
- check the log again
- view the resulting data set again.

Creating and Modifying Variables

Overview

So far in this chapter, you've read existing data. But sometimes existing data doesn't provide the information you need. To modify existing values or to create new variables, you can use an assignment statement in any DATA step.

General form, assignment statement:

variable=expression;

where

- *variable* names a new or existing variable
- *expression* is any valid SAS expression

Additional Note The assignment statement is one of the few SAS statements that doesn't begin with a keyword.

For example, here is an assignment statement that assigns the character value **`Toby Witherspoon`** to the variable Name:

```
Name='Toby Witherspoon';
```

SAS Expressions

You use SAS expressions in assignment statements and many other SAS programming statements to

- transform variables
- create new variables
- conditionally process variables
- calculate new values
- assign new values.

An expression is a sequence of operands and operators that form a set of instructions. The instructions are performed to produce a new value:

- Operands are variable names or constants. They can be numeric, character, or both.
- Operators are special-character operators, grouping parentheses, or functions. You can learn about functions in "Transforming Data with SAS Functions" on page 404 .

Using Operators in SAS Expressions

To perform a calculation, you use arithmetic operators. The table below lists arithmetic operators.

Table 5.8: Using Operators in SAS Expressions

Operator	Action	Example	Priority
-	negative prefix	**`negative=-x;`**	I

**	exponentiation	**raise=x**y;**	I
*	multiplication	**mult=x*y;**	II
/	division	**divide=x/y;**	II
+	addition	**sum=x+y;**	III
-	subtraction	**diff=x-y;**	III

When you use more than one arithmetic operator in an expression,

- operations of priority I are performed before operations of priority II, and so on
- consecutive operations that have the same priority are performed
  - from right to left within priority I
  - from left to right within priority II and III
- you can use parentheses to control the order of operations.

Caution When a value that is used with an arithmetic operator is missing, the result of the expression is missing. The assignment statement assigns a missing value to a variable if the result of the expression is missing.

You use the following comparison operators to express a condition.

Table 5.9: Comparison Operators

Operator	Meaning	Example
= or eq	equal to	**name='Jones, C.'**
^= or ne	not equal to	**temp ne 212**
>or gt	greater than	**income>20000**
<or lt	less than	**partno lt "BG05"**
>= or ge	greater than or equal to	**id>='1543'**
<= or le	less than or equal to	**pulse le 85**

To link a sequence of expressions into compound expressions, you use logical operators, including the following:

Table 5.10: Logical Operators

Operator	Meaning
AND or &	and, both. If both expressions are true, then the compound expression is true.
OR or |	or, either. If either expression is true, then the compound expression is true.

More Examples of Assignment Statements

The assignment statement in the DATA step below creates a new variable, TotalTime, by multiplying the values of TimeMin by 60 and then adding the values of TimeSec.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
    TotalTime=(timemin*60)+timesec;
run;
```

SAS Data Set Sasuser Stress (Partial Listing)__

ID	Name	RestHR	MaxHR	RecHR	TimeMin	TimeSec	Tolerance	TotalTime
2458	Murray, W	72	185	128	12	38	D	**758**

2462	Almers, C	68	171	133	10	5	I	**605**
2501	Bonaventure, T	78	177	139	11	13	I	**573**
2523	Johnson, R	69	162	114	9	42	S	**582**
2539	LaMance, K	75	168	141	11	46	D	**705**

Figure 5.21: Assignment Statement Output

The expression can also contain the variable name that is on the left side of the equal sign, as the following assignment statement shows. This statement re-defines the values of the variable RestHR as 10 percent higher.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   resthr=resthr+(resthr*.10);
run;
```

When a variable name appears on both sides of the equal sign, the original value on the right side is used to evaluate the expression. The result is assigned to the variable on the left side of the equal sign.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   resthr=resthr+(resthr*.10);
run;   ^      ^
 result      original value
```

Date Constants

You can assign date values to variables in assignment statements by using date constants. To represent a constant in SAS date form, specify the date as *'ddmmmyy'* or *'ddmmmyyyy'*, immediately followed by a D.

General form, date constant:

'ddmmmyy'd

or

"ddmmmyy"d

where

- *dd* is a one-or two-digit value for the day
- mmm is a three-letter abbreviation for the month (JAN, FEB, and so on)
- *yy* or*yyyy* is a two-or four-digit value for the year, respectively.

Additional Note Be sure to enclose the date in quotation marks.

Example

In the following program, the second assignment statement assigns a date value to the variable TestDate.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   TotalTime=(timemin*60)+timesec;
```

```
   TestDate='01jan20 0 0'd;
run;
```

Additional Note You can also use SAS time constants and SAS datetime constants in assignment statements.

```
Time='9:25't;

DateTime='18jan20 05:9:27:05'dt;
```

Subsetting Data

As you read your data, you can subset it by processing only those observations that meet a specified condition. To do this, you can use a subsetting IF statement in any DATA step.

Using a Subsetting IF Statement

The subsetting IF statement causes the DATA step to continue processing only those observations that meet the condition of the expression specified in the IF statement. The resulting SAS data set or data sets contain a subset of the original external file or SAS data set.

General form, subsetting IF statement:

IF *expression;*

where *expression* is any valid SAS expression.

- If the expression is *true*, the DATA step continues to process that observation.
- If the expression is *false*, no further statements are processed for that observation, and control returns to the top of the DATA step.

Example

The subsetting IF statement below selects only observations whose values for Tolerance are **D**. It is positioned in the DATA step so that other statements do not need to process unwanted observations.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   if tolerance='D';
   TotalTime=(timemin*60)+timesec;
run;
```

Because Tolerance is a character variable, the value **D** must be enclosed in quotation marks, and it must be the same case as in the data set.

Additional Note See the SAS documentation for a comparison of the WHERE and subsetting IF statements when they are used in the DATA step.

Reading Instream Data

Overview

Throughout this chapter, our program has contained an INFILE statement that identifies an external file to read.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
if tolerance='D';
   TotalTime=(timemin*60)+timesec;
```

```
run;
```

However, you can also read instream data lines that you enter directly in your SAS program, rather than data that is stored in an external file. Reading instream data is extremely helpful if you want to create data and test your programming statements on a few observations that you can specify according to your needs.

To read instream data, you use

- a DATALINES statement as the last statement in the DATA step and immediately preceding the data lines
- a null statement (a single semicolon) to indicate the end of the input data.

```
data sasuser.stress;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   datalines;
   .
   .
   .
data lines go here
   .
   .
   .
;
```

General form, DATALINES statement:

DATALINES;

- **Additional Note** You can use only one DATALINES statement in a DATA step. Use separate DATA steps to enter multiple sets of data.
- **Additional Note** You can also use LINES;or CARDS;as the last statement in a DATA step and immediately preceding the data lines. Both LINES and CARDS are aliases for the DATALINES statement.
- **Additional Note** If your data contains semicolons, use the DATALINES4 statement plus a null statement that consists of four semicolons (;;;;).

Example

To read the data for the treadmill stress tests as instream data, you can submit the following program:

```
data sasuser.stress;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   if tolerance='D';
   TotalTime=(timemin*60)+timesec;
   datalines;
2458 Murray, W            72   185 128 12 38 D
2462 Almers, C            68   171 133 10 5 I
2501 Bonaventure, T       78   177 139 11 13 I
2523 Johnson, R           69   162 114 9 42 S
2539 LaMance, K           75   168 141 11 46 D
2544 Jones, M             79   187 136 12 26 N
2552 Reberson, P          69   158 139 15 41 D
2555 King, E              70   167 122 13 13 I
2563 Pitts, D             71   159 116 10 22 S
2568 Eberhardt, S         72   182 122 16 49 N
2571 Nunnelly, A          65   181 141 15 2 I
2572 Oberon, M            74   177 138 12 11 D
2574 Peterson, V          80   164 137 14 9 D
2575 Quigley, M           74   152 113 11 26 I
2578 Cameron, L           75   158 108 14 27 I
2579 Underwood, K         72   165 127 13 19 S
```

```
2584 Takahashi, Y          76   163 135 16 7 D
2586 Derber, B             68   176 119 17 35 N
2588 Ivan, H               70   182 126 15 41 N
2589 Wilcox, E             78   189 138 14 57 I
2595 Warren, C             77   170 136 12 10 S;
```

Caution Notice that you do not need a RUN statement following the null statement (the semicolon after the data lines). The DATALINES statement functions as a step boundary, so the DATA step is executed as soon as SAS encounters it.

Creating a Raw Data File

Overview

Look at the SAS program and SAS data set that you created earlier in this chapter.

```
data sasuser.stress;
   infile tests;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   if tolerance='D';
   TotalTime=(timemin*60)+timesec;
run;
```

SAS Data Set Sasuser.Stress

ID	Name	RestHR	MaxHR	Ree HR	TimeMin	TimeSec	Tolerance	Tota lTime
2458	Murray, W	72	185	128	12	38	D	758
2539	LaMance, K	75	168	141	11	46	D	706
2552	Reberson, P	69	158	139	15	41	D	941
2572	Oberon, M	74	177	138	12	11	D	731
2574	Peterson, V	80	164	137	14	9	D	849
2584	Takahashi, Y	76	163	135	16	7	D	967

Figure 5.22: SAS Data Set sasuser.stress Output

As you can see, the data set has been modified with SAS statements. If you wanted to write the new observations to a raw data file, you could reverse the process that you've been following and write out the observations from a SAS data set as records or lines to a new raw data file.

Using the _NULL_ Keyword

Because the goal of your SAS program is to create a raw data file and not a SAS data set, it is inefficient to print a data set name in the DATA statement. Instead, use the keyword _NULL_, which enables you to use the DATA step without actually creating a SAS data set. A SET statement specifies the SAS data set that you want to read from.

```
data _null_;
    set sasuser.stress;
```

The next step is to specify the output file.

Specifying the Raw Data File

You use the FILE and PUT statements to write the observations from a SAS data set to a raw data file, just as you used the INFILE and INPUT statements to create a SAS data set. These two sets of statements work almost identically.

When writing observations to a raw data file, use the FILE statement to specify the output file.

General form, FILE statement:

FILE *file-specification <options><operating-environment-options>;*

where

- *file-specification* can take the form *fileref* to name a previously defined file reference or *'filename'* to point to the actual name and location of the file
- *options* names options that are used in creating the output file
- *operating-environment-options* names options that are specific to an operating environment (for more information, see the SAS documentation for your operating environment).

For example, if you want to read the Sasuser.Stress data set and write it to a raw data file that is referenced by the fileref Newdat, you would begin your program with the following SAS statements.

```
data _null_;
   set sasuser.stress;
   file newdat;
```

Instead of identifying the raw data file with a SAS fileref, you can choose to specify the entire filename and location in the FILE statement. For example, the following FILE statement points directly to the **`C:\Clinic\Patients\Stress.dat`** file. Note that the path specifying the filename and location must be enclosed in quotation marks.

```
data _null_;
   set sasuser.stress;
   file 'c:\clinic\patients\stress.dat';
```

Describing the Data

Whereas the FILE statement specifies the output raw data file, the PUT statement describes the lines to write to the raw data file.

General form, PUT statement using column output:

PUT *variable startcol-endcol…;*

where

- *variable* is the name of the variable whose value is written
- *startcol* indicates where in the line to begin writing the value
- *endcol* indicates where in the line to end the value.

In general, the PUT statement mirrors the capabilities of the INPUT statement. In this case you are working with column output. Therefore, you need to specify the variable name, starting column, and ending column for each field that you want to create. Because you are creating raw data, you don't need to follow character variable names with a dollar sign ($).

```
data _null_;
   set sasuser.stress;
   file 'c:\clinic\patients\stress.dat';
   put id $ 1-4 name 6-25 resthr 27-29 maxhr 31-33
        rechr 35-37 timemin 39-40 timesec 42-43
        tolerance 45 totaltime 47-49;
run;
```

SAS Data Set Sasuser.Stress

ID	Name	RestHR	MaxHR	Ree HR	TimeMin	TimeSec	Tolerance	Tota lTime
2458	Murray, W	72	185	128	12	38	D	758
2539	LaMance, K	75	168	141	11	46	D	706
2552	Reberson, P	69	158	139	15	41	D	941
2572	Oberon, M	74	177	138	12	11	D	731

2574	Peterson, V	80	164	137	14	9	D	849
2584	Takahashi, Y	76	163	135	16	7	D	967

Figure 5.23: SAS Data Set sasuser.stress Output with PUT Statement

The resulting raw data file would look like this.

Raw Data File Stress.Dat

```
1---+----10---+----20---+----30---+----40---+----50---+
2458 Murray, W            72  185 128 12 38 D 758
2539 LaMance, K           75  168 141 11 46 D 706
2552 Reberson, P          69  158 139 15 41 D 941
2572 Oberon, M            74  177 138 12 11 D 731
2574 Peterson, V          80  164 137 14 9  D 849
2584 Takahashi, Y         76  163 135 16 7  D 967
```

Figure 5.24: Creating a Raw Data File

In later chapters you'll learn how to use INPUT and PUT statements to read and write raw data in other forms and record types.

Additional Note If you do not execute a FILE statement before a PUT statement in the current iteration of the DATA step, SAS writes the lines to the SAS log. If you specify the PRINT fileref in the FILE statement, before the PUT statement, SAS writes the lines to the procedure output file.

Additional Features

In this section, you learned to read raw data by using an INPUT statement that uses column input. You also learned how to write to a raw data file by using the FILE statement with column input. However, column input is appropriate only in certain situations. When you use column input, your data must be

- standard character and numeric values. If the raw data file contains nonstandard values, then you need to use formatted input, another style of input. To learn about formatted input, see "Reading SAS Data Sets" on page 334 .
- in fixed fields. That is, values for a particular variable must be in the same location in all records. If your raw data file contains values that are not in fixed fields, you need to use list input. To learn about list input, see "Reading Free-Format Data" on page 536 .

Other forms of the INPUT statement enable you to read

- nonstandard data values such as hexadecimal, packed decimal, dates, and monetary values that contain dollar signs and commas
- free-format data (data that is not in fixed fields)
- implied decimal points
- variable-length data values
- variable-length records
- different record types.

Reading Microsoft Excel Data

Overview

In addition to reading raw data files, SAS can also read Microsoft Excel data. Whether the input data source is a SAS data set, a raw data file, or a file from another application, you use the DATA step to create a SAS data set. The difference between reading these various types of input is in how you reference the data. To read in Excel data you use one of the following methods:

- SAS/ACCESS LIBNAME statement
- Import Wizard

Remember, the Base SAS LIBNAME statement associates a SAS name (libref) with a SAS DATA library by pointing to its physical location. But, the SAS/ACCESS

LIBNAME statement associates a SAS name with an Excel workbook file by pointing to its location.

In doing so, the Excel workbook becomes a new library in SAS, and the worksheets in the workbook become the individual SAS data sets in that library.

The figure below illustrates the difference between how the two LIBNAME statements treat the data.

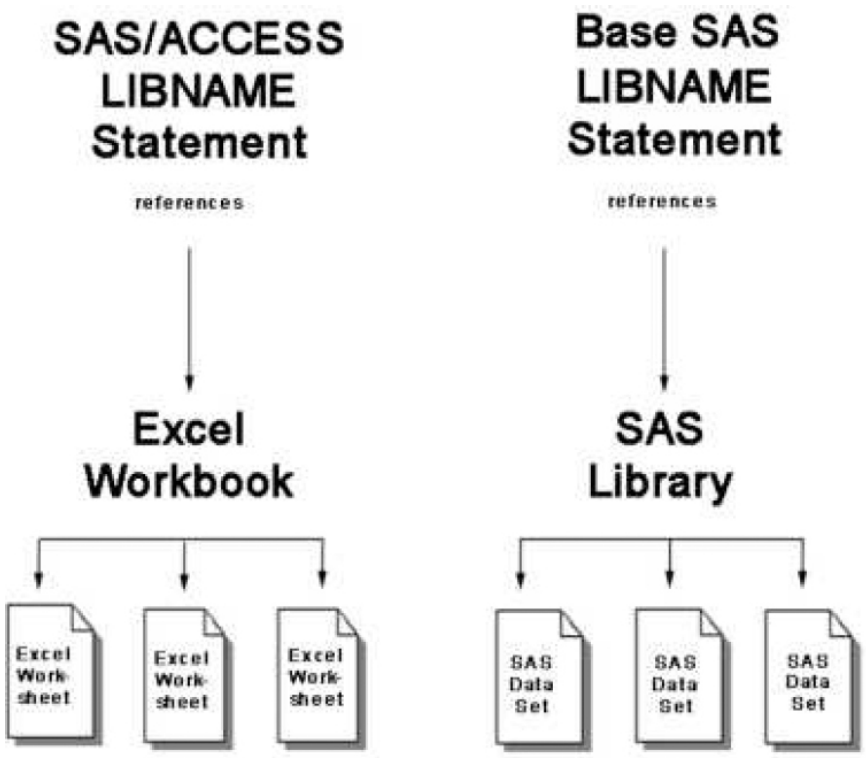


Figure 5.25: Comparing Libname Statements

The next figure shows how the DATA step is used with three types of input data.

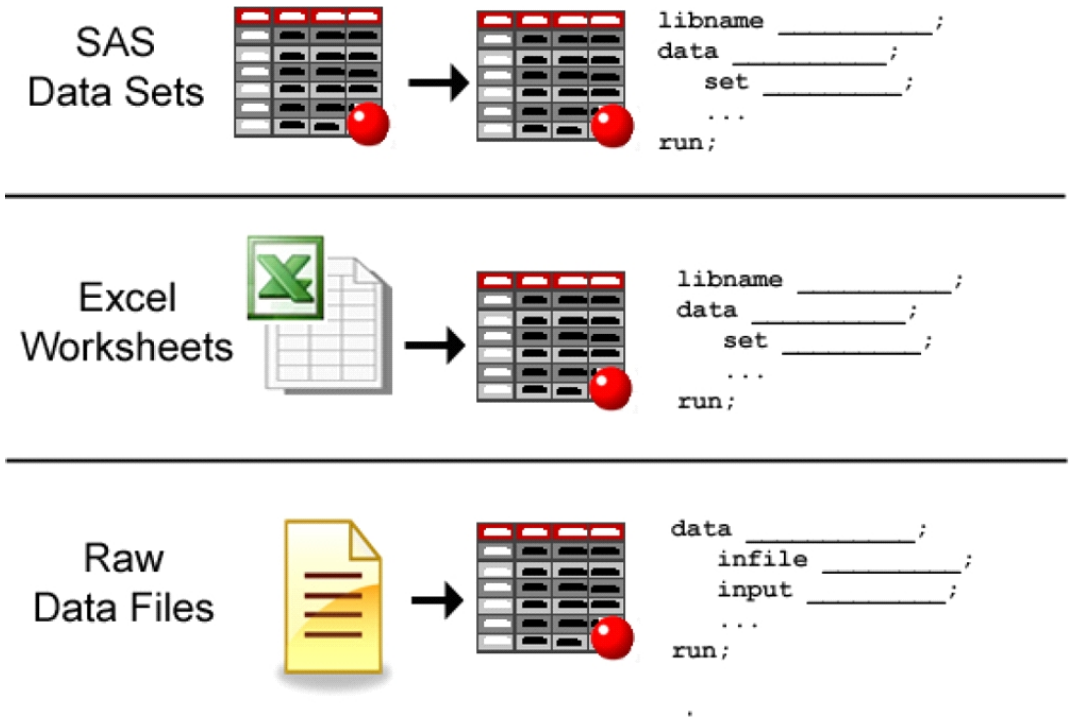


Figure 5.26: Using the DATA step with Different Types of Output

Notice how the INFILE and INPUT statements are used in the DATA step for reading raw data, but the SET statement is used in the DATA step for reading in the Excel worksheets.

Running SAS with Microsoft Excel

- You must have licensed SAS/ACCESS Interface to PC Files to use a SAS/ACCESS LIBNAME statement that references an Excel workbook.
- If you are running SAS version 9.1 or earlier and want to read in Microsoft Excel data, you must use Microsoft Excel 2003 or earlier.
- To read Microsoft Excel 2007 data you must be running SAS version 9.2 or later.
- The examples in this section are based on SAS version 9.2 running with Microsoft Excel 2007.

Steps for Reading Excel Data

Let's look at the steps for reading in an Excel workbook file.

To read the Excel workbook file, the DATA step must provide the following instructions to SAS:

- a libref to reference the Excel workbook to be read
- the name and location (using another libref) of the new SAS data set
- the name of the Excel worksheet that is to be read

The table below outlines the basic statements that are used in a program that reads Excel data and creates a SAS data set from an Excel worksheet. The PROC CONTENTS and PROC PRINT statements are not requirements for reading in Excel data and creating a SAS data set. However, these statements are useful for confirming that your Excel data has successfully been read into SAS.

Table 5.11: Basic Steps for Reading Excel Data into a SAS Data Set

To do this…	Use this SAS statement…	Example
Reference an Excel workbook file	SAS/ACCESS LIBNAME statement	libname results `'c: \users \exercise.xlsx';`

Output the contents of the SAS Library	PROC CONTENTS	proc contents data=results._all_;
Execute the PROC CONTENTS statement	RUN statement	run;
Name and create a new SAS data set	DATA statement	data work.stress;
Read in an Excel worksheet (as the input data for the new SAS data set)	SET statement	set results.'ActLevel$'n;
Execute the DATA step	RUN statement	run;
View the contents of a particular data set	PROC PRINT	proc print data=stress;
Execute the PROC PRINT statement	RUN statement	run;

The SAS/ACCESS LIBNAME Statement

The general form of the SAS/ACCESS LIBNAME statement is as follows:

General form, SAS/ACCESS LIBNAME statement:

LIBNAME *libref 'location-of-Excel-workbook '<options>;*

where

- *libref* is a name that you associate with an Excel workbook.
- *'location-of-Excel-workbook'* is the physical location of the Excel workbook.

`Example:`

libname results **`'c:\users\exercise.xlsx';`**

Referencing an Excel Workbook

Overview

This example uses data similar to the scenario used for the raw data in the previous section. The data shows the readings from exercise stress tests that have been performed on patients at a health clinic.

The stress test data is located in an Excel workbook named exercise.xlsx (shown below), which is stored in the location **`c:\users.`**

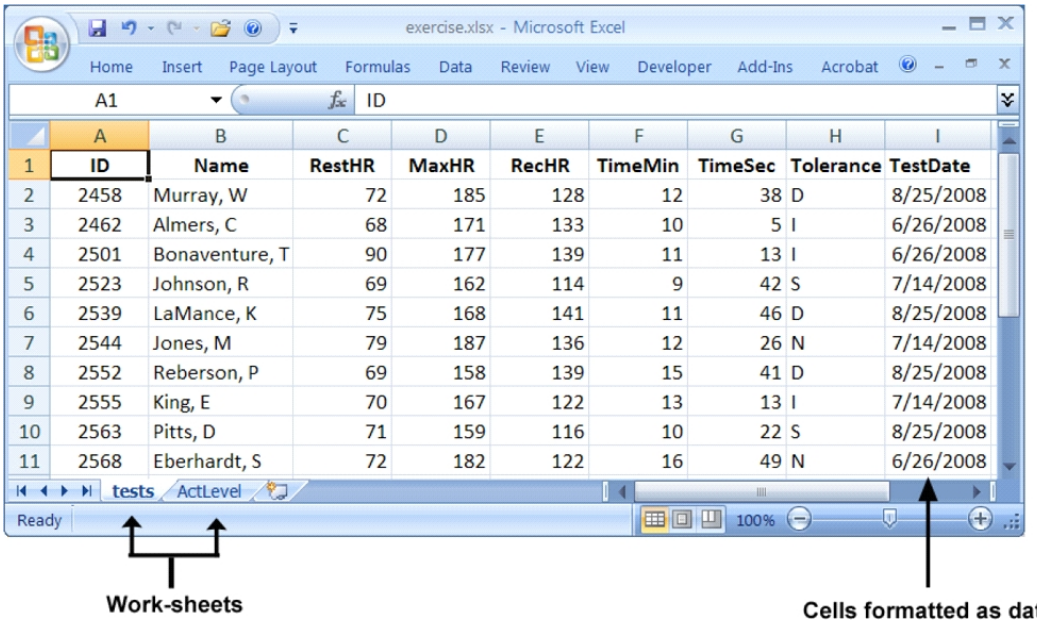


Figure 5.1: Excel Workbook

Notice in the sample worksheet above that the date column is defined in Excel as dates. That is, if you right-click on the cells and select **Format Cells** (in Excel), the cells have a category of Date. SAS reads this data just as it is stored in Excel. If the date had been stored as text in Excel, then SAS would have read it as a character string.

To read in this workbook, you must first create a libref to point to the workbook's location:

```
libname results 'c:\users\exercise.xlsx';
```

The LIBNAME statement creates the libref **`results`**, which points to the Excel workbook exercise.xlsx. The workbook contains two worksheets, **`tests`** and **`ActLevel`**, which are now available in the new SAS library (results) as data sets.

After submitting the LIBNAME statement, you can look in the SAS Explorer window to see how SAS handles your Excel workbook. The Explorer window enables you to manage your files in the SAS windowing environment.

Figure 5.2: SAS Explorer Window

Name Literals

In the figure above, notice how the LIBNAME statement created a permanent library, **`results`**, which is the SAS name (libref) we gave to the workbook file and its location. The new library contains two SAS data sets, which accesses the data from the Excel worksheets. From this window you can browse the list of SAS libraries or display the descriptor portion of a SAS data set.

Notice that the Excel worksheet names have the special character ($) at the end. All Excel worksheets are designated this way. But remember, special characters such as these are not allowed in SAS data set names by default. So, in order for SAS to allow this character to be included in the data set name, you must assign a name literal to the data set name. A SAS name literal is a name token that is expressed as a string within quotation marks, followed by the uppercase or lowercase letter *n*. The name literal tells SAS to allow the special character ($) in the data set name.

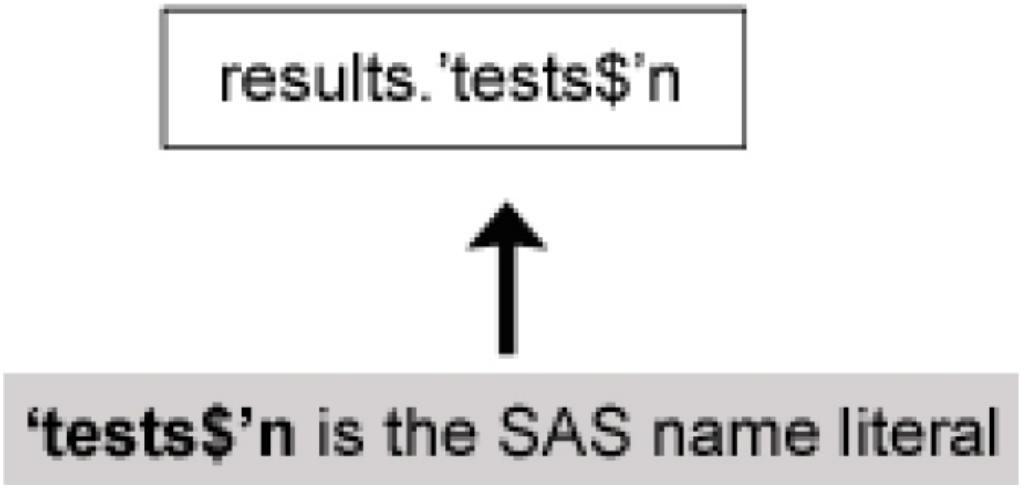


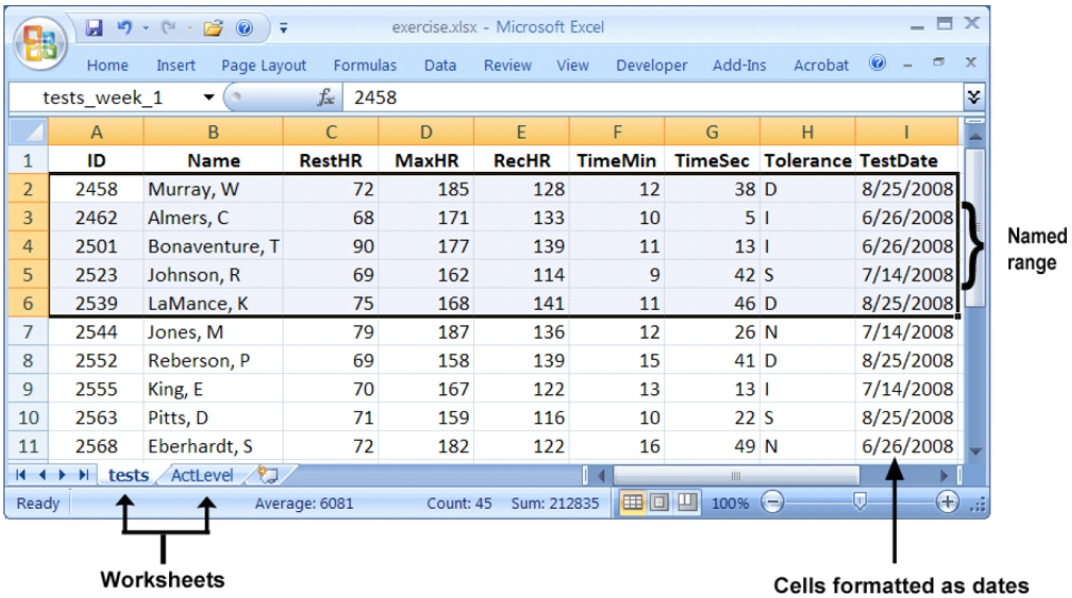
Figure 5.27: Name Literal

Named Ranges

A named range is a range of cells within a worksheet that you define in Excel and assign a name to. In the example below, the worksheet contains a named range, tests_week_1, which SAS recognizes as a data set.

The named range, tests_week_1, and its parent worksheet, tests, will appear in the SAS Explorer window as separate data sets, except that the data set created from the named range will have no dollar sign ($) appended to its name.

For more information on named ranges, see your Microsoft Excel documentation.



	A	B	C	D	E	F	G	H	I
1	ID	Name	RestHR	MaxHR	RecHR	TimeMin	TimeSec	Tolerance	TestDate
2	2458	Murray, W	72	185	128	12	38	D	8/25/2008
3	2462	Almers, C	68	171	133	10	5	I	6/26/2008
4	2501	Bonaventure, T	90	177	139	11	13	I	6/26/2008
5	2523	Johnson, R	69	162	114	9	42	S	7/14/2008
6	2539	LaMance, K	75	168	141	11	46	D	8/25/2008
7	2544	Jones, M	79	187	136	12	26	N	7/14/2008
8	2552	Reberson, P	69	158	139	15	41	D	8/25/2008
9	2555	King, E	70	167	122	13	13	I	7/14/2008
10	2563	Pitts, D	71	159	116	10	22	S	8/25/2008
11	2568	Eberhardt, S	72	182	122	16	49	N	6/26/2008

Figure 5.28: Named Range

Using PROC CONTENTS

In addition to using the SAS explorer window to view library data, you can also use the CONTENTS procedure with the _ALL_ keyword to produce information about a data library and its contents. In the example below, PROC CONTENTS outputs summary information for the SAS data set tests, including data set name, variables, data types, and other summary information. This statement is useful for making sure that SAS successfully read in your Excel data before moving on to the DATA step.

```
proc contents data=results._all_;
run;
```

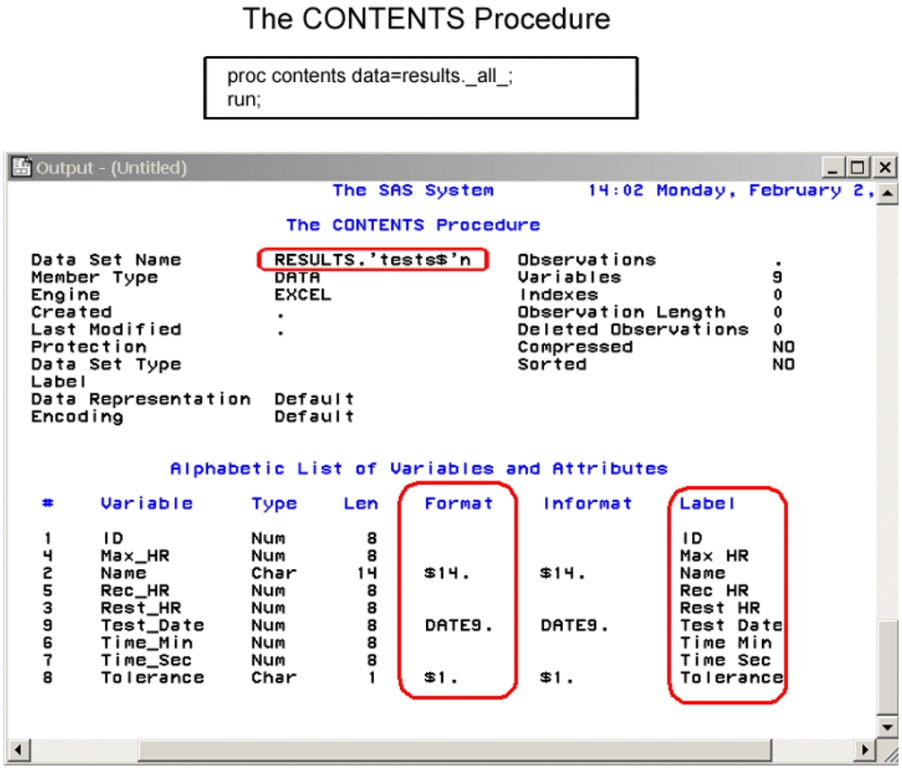


Figure 5.29: CONTENTS Procedure Output

About the sample output above:

- The variables in the data set are pulled from the Excel column headings. SAS uses underscores to replace the spaces.
- The Excel dates in the Format column are converted to SAS dates with the default DATE9. format.
- This is partial output. The ACTLEVEL data set would also be included in the PROC CONTENTS report.

Creating the DATA Step

You use the DATA statement to indicate the beginning of the DATA step and name the SAS data set to be created. Remember that the SAS data set name is a two-level name. For example, the two-level name results.Admit specifies that the data set Admit is stored in the permanent SAS library to which the libref results has been assigned.

When reading Excel data, use the SET statement to indicate which worksheet in the Excel file that you want to read. To read in the Excel file you write the DATA and SET statements as follows:

```
data work.stress;
   set results.'ActLevel$'n;
run;
```

In this example, the DATA statement tells SAS to name the new data set, stress, and store it in the temporary library WORK. The SET statement in the DATA step specifies the libref (reference to the Excel file) and the worksheet name as the input data.

You can use several statements in the DATA step to subset your data as needed. Here, the WHERE statement is used with a variable to include only those participants whose activity level is HIGH.

```
data work.stress;
   set results.'ActLevel$'n;
   where ActLevel='HIGH';
run;
```

The figure below shows the partial output for this DATA step in table format.

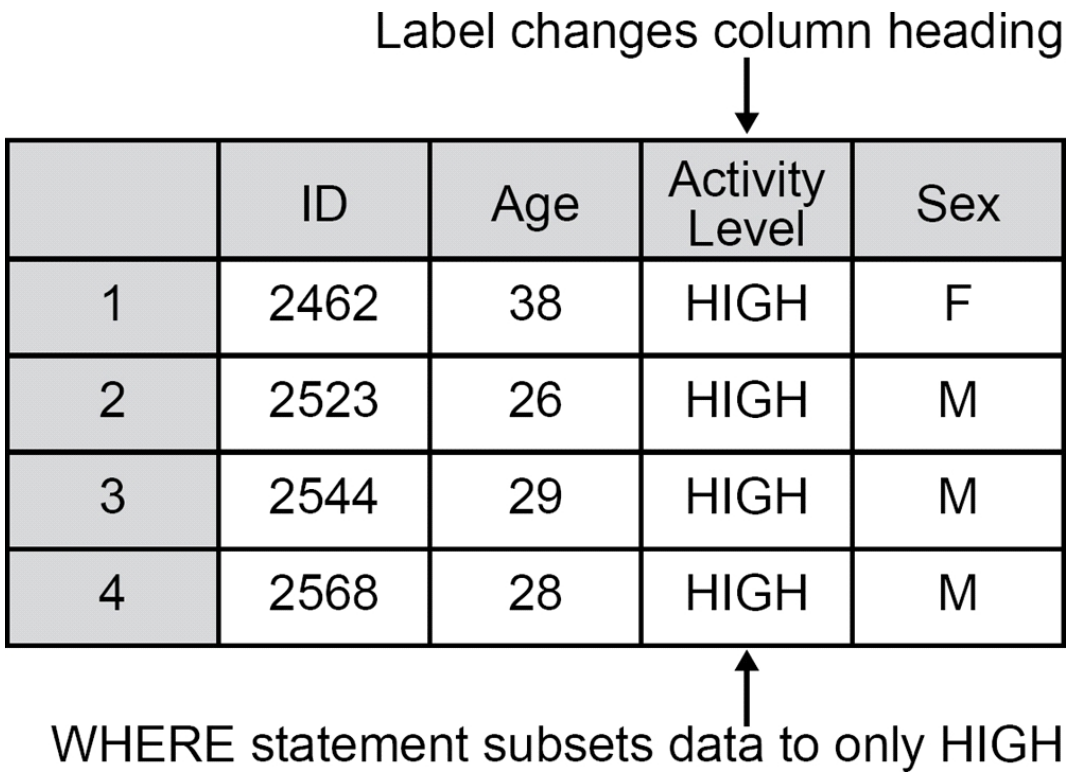


Figure 5.30: DATA Step Output

Using PROC PRINT

After using the DATA step to read in the Excel data and create the SAS data set, you can use PROC PRINT to produce a report that displays the data set values.

You can also use the PRINT procedure to refer to a specific worksheet. Remember to use the name literal when referring to a specific Excel worksheet. In the example below, the first PRINT statement displays the data values for the new data set that was created in the DATA step. The second PRINT statement displays the contents of the Excel worksheet that was referenced by the LIBNAME statement.

```
proc print data=work.stress;
run;
proc print data=results.'ActLevel$'n;
run;
```

Disassociating a Libref

If SAS has a libref assigned to an Excel workbook, the workbook cannot be opened in Excel. To disassociate a libref, use a LIBNAME statement, specifying the libref and the CLEAR option.

```
libname results 'c:\users\exercise.xlsx';
   proc print data=results.'tests$'n;
run;

libname results clear;
```

SAS disconnects from the data source and closes any resources that are associated with that libref's connection.

LIBNAME Statement Options

There are several options that you can use with the LIBNAME statement to control how SAS interacts with the Excel data. The general form of the SAS/ACCESS LIBNAME statement (with options) is as follows:

```
libname libref 'location-of-Excel-workbook' <options>;
```

Example:

```
libname doctors 'c:\clinicNotes\addresses.xlsx' mixed=yes;
```

DBMAX_TEXT=n

indicates the length of the longest character string where n is any integer between 256 and 32,767 inclusive. Any character string with a length greater than this value is truncated. The default is 1024.

GETNAMES=YES|NO

determines whether SAS will use the first row of data in an Excel worksheet or range as column names.

YES specifies to use the first row of data in an Excel worksheet or range as column names.

NO specifies not to use the first row of data in an Excel worksheet or range as column names. SAS generates and uses the variable names F1, F2, F3, and so on.

The default is YES.

MIXED=YES|NO

Specifies whether to import data with both character and numeric values and convert all data to character.

YES specifies that all data values will be converted to character.

NO specifies that numeric data will be missing when a character type is assigned. Character data will be missing when a numeric data type is assigned.

The default is NO.

SCANTEXT=YES|NO

specifies whether to read the entire data column and use the length of the longest string found as the SAS column width.

YES scans the entire data column and uses the longest string value to determine the SAS column width.

NO does not scan the column and defaults to a width of 255.

The default is YES.

SCANTIME=YES|NO

specifies whether to scan all row values in a date/time column and automatically determine the TIME. format if only time values exist.

YES specifies that a column with only time values be assigned the TIME8. format.

NO specifies that a column with only time values be assigned the DATE9. format.

The default is NO.

USEDATE=YES|NO

specifies whether to use the DATE9. format for date/time values in Excel workbooks.

YES specifies that date/time values be assigned the DATE9. format.

NO specifies that date/time values be assigned the DATETIME. format.

The default is YES.

Creating Excel Worksheets

In addition to being able to read Excel data, SAS can also create Excel worksheets from SAS data sets.

To do this, you use the SAS/ACCESS LIBNAME statement. For example, to create a new worksheet named **high_stress** from the temporary SAS data set **work.high_stress** and save this worksheet in the new Excel file **newExcel.xlsx**, you would submit the following LIBNAME statement and DATA step:

```
libname clinic 'c:\Users\mylaptop\admitxl.xlsx' mixed=yes;
data clinic.admit;
   set work.admit;
run;
```

The IMPORT Wizard

Importing Data

As an alternative to using programming statements, you can use the Import Wizard to guide you through the process of creating a SAS data set from both raw data and from Excel worksheets. The Import Wizard enables you to create a SAS data set from different types of external files, such as

- dBase files (*.dbf)
- Excel 2007 (or earlier version) workbooks (*.xls, *.xlsx, *.xlsb, or *.xlsm)
- Microsoft Access tables (*.mdb, *.accdb)
- Delimited files (*.*)
- Comma-separated values (*.csv).

Additional Note The data sources that are available to you depend on which SAS/ACCESS products you have licensed. If you do not have any SAS/ACCESS products licensed, the only type of data source files available to you are CSV files, TXT files, and delimited files.

To access the Import Wizard, select **File** ⇨ **Import Data** from the menu bar. The Import Wizard opens with the **Select import type** screen.

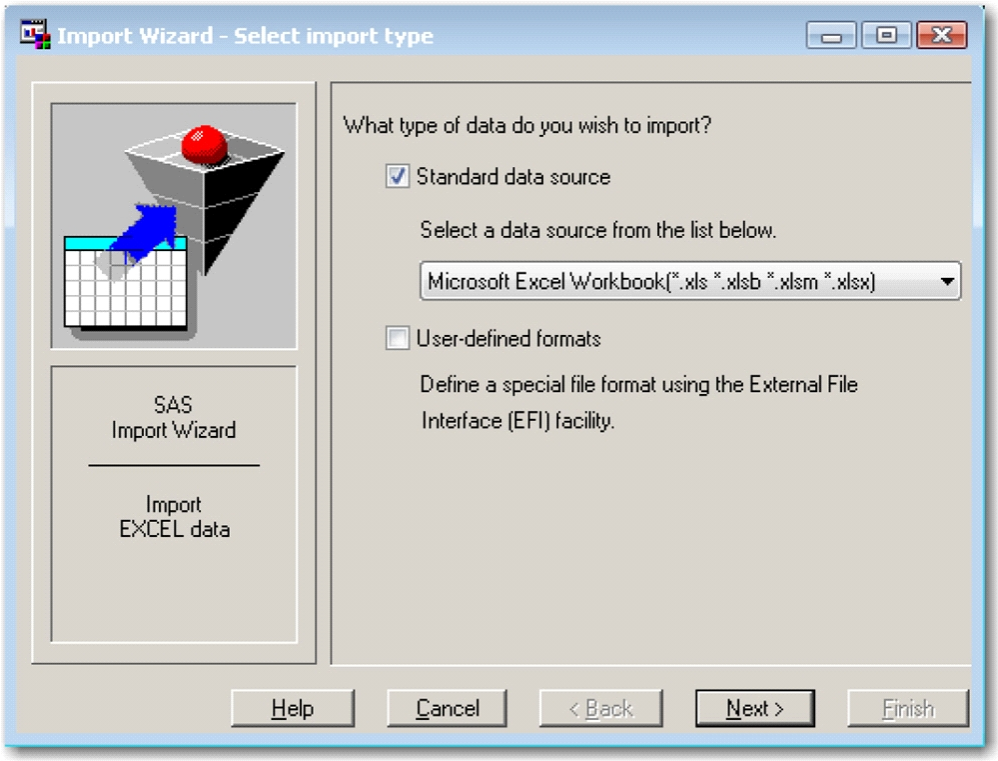


Figure 5.31: Import Wizard

Follow the instructions on each screen of the Import Wizard to read in your data. If you need additional information, select the **Help** button at the bottom of each screen in the wizard.

Just as you can create a SAS data set from raw data by using the Import Wizard, you can use the Export Wizard to read data from a SAS data set and to write the data to an external data source. To access the Export Wizard, select **File** ⇨ **Export Data** from the menu bar.

Chapter Summary

Text Summary

Raw Data Files

A raw data file is an external file whose records contain data values that are organized in fields. The raw data files in this chapter contain fixed fields.

Steps to Create a SAS Data Set

You need to follow several steps to create a SAS data set using raw data. You need to

- reference the raw data file to be read
- name the SAS data set
- identify the location of the raw data
- describe the data values to be read.

Referencing a SAS Library

To begin your program, you might need to use a LIBNAME statement to reference the SAS library in which your data set will be stored.

Writing a DATA Step Program

The DATA statement indicates the beginning of the DATA step and names the SAS data set(s) to be created.

Next, you specify the raw data file by using the INFILE statement. The OBS= option in the INFILE statement enables you to process a specified number of observations.

This chapter teaches column input, the most simple input style. Column input specifies actual column locations for data values. The INPUT statement describes the raw data to be read and placed into the SAS data set.

Submitting the Program

When you submit the program, you can use the OBS= option with the INFILE statement to verify that the correct data is being read before reading the entire data file.

After you submit the program, view the log to check the DATA step processing. You can then print the data set by using the PROC PRINT procedure.

Once you've checked the log and verified your data, you can modify the DATA step to read the entire raw data file by removing the OBS= option from the INFILE statement.

If you are working with a raw data file that contains invalid data, the DATA step continues to execute. Unlike syntax errors, invalid data errors do not cause SAS to stop processing a program. If you have a way to edit the invalid data, it's best to correct the problem and rerun the DATA step.

Creating and Modifying Variables

To modify existing values or to create new variables, you can use an assignment statement in any DATA step. Within assignment statements, you can specify any SAS expression.

You can use date constants to assign dates in assignment statements. You can also use SAS time constants and SAS datetime constants in assignment statements.

Subsetting Data

To process only observations that meet a specified condition, use a subsetting IF statement in the DATA step.

Reading Instream Data

To read instream data lines instead of an external file, use a DATALINES statement, a CARDS statement, or a LINES statement and enter data directly in your SAS program. Omit the RUN at the end of the DATA step.

Creating a Raw Data File

When the goal of your SAS program is to create a raw data file and not a SAS data set, it is inefficient to list a data set name in the DATA statement. Instead use the keyword _NULL_, which allows the power of the DATA step without actually creating a SAS data set. A SET statement specifies the SAS data set that you want to read from.

You can use the FILE and PUT statements to write out the observations from a SAS data set to a raw data file just as you used the INFILE and INPUT statements to create a SAS data set. These two sets of statements work almost identically.

Microsoft Excel Files

You can read Excel worksheets by using the SAS/ACCESS LIBNAME statement.

Steps to Create a SAS Data Set from Excel Data

You need to follow several steps to create a SAS data set using Excel. You need to

- provide a name for the new SAS data set
- provide the location or name of the libref and Excel worksheet

Referencing an Excel Workbook

To begin your program, you need to use a LIBNAME statement to reference the Excel workbook.

Writing a DATA Step Program

The DATA statement indicates the beginning of the DATA step and names the SAS data set(s) to be created.

Next, you specify the Excel worksheet to be read by using the SET statement. You must use a SAS name literal since SAS uses the special character ($) to name Excel worksheets.

Submitting the Program

When you submit the program, you can use the CONTENTS procedure to explore the new library and contents.

After you submit the program, view the log to check the DATA step processing. You can then print the data sets created from the Excel worksheets by using the PROC PRINT procedure.

Once you've checked the log and verified your data, you can modify the DATA step along with the WHERE statement to subset parts of the data as needed.

Syntax

```
Reading Data from a Raw File or Reading Instream Data
LIBNAME libref'SAS-data-library';
FILENAME fileref'filename';
DATA SAS-data-set;
                              INFILE file-specification<OBS=n>;
                              INPUT variable <$>startcol-endcol…;
                              IF expression;
                              variable=expression;
DATALINES;
               instream data goes here if used
;
RUN; /* not used with the DATALINES statement */
PROC PRINT DATA= SAS-data set;
RUN;

Creating a Raw Data File
LIBNAME libref'SAS-data-library';
DATA _NULL_;
SET SAS-data-set;
                     FILE file-specification;
                     PUT variable startcol-endcol…;
RUN;

            Reading Data from an Excel Workbook
LIBNAME libref'<location-of-Excel-workbook>';
```

```
PROC CONTENTS DATA=libref._ALL_;
DATA SAS-data-set;
          SET libref'worksheet_name$'n;
                     WHERE where-expression;
RUN;
PROC PRINT DATA= SAS-data set;
RUN;
```

Sample Programs

Reading Data from an External File

```
libname clinic 'c:\bethesda\patients\admit';
filename admit 'c:\clinic\patients\admit.dat';
data clinic.admittan;
   infile admit obs=5;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   if tolerance='D';
   TotalTime=(timemin*60)+timesec;
run;
proc print data=clinic.admittan;
run;
```

Reading Instream Data

```
libname clinic 'c:\bethesda\patients\admit';
data clinic.group1;
   input ID $ 1-4 Name $ 6-25 RestHR 27-29 MaxHR 31-33
         RecHR 35-37 TimeMin 39-40 TimeSec 42-43
         Tolerance $ 45;
   if tolerance='D';
   TotalTime=(timemin*60)+timesec;
datalines;
2458 Murray, W            72   185 128 12 38 D
2462 Almers, C            68   171 133 10  5 I
2501 Bonaventure, T       78   177 139 11 13 I
2523 Johnson, R           69   162 114 9 42 S
2539 LaMance, K           75   168 141 11 46 D
2544 Jones, M             79   187 136 12 26 N
2595 Warren, C            77   170 136 12 10 S
;
proc print data=clinic.group1;
run;
```

Reading Excel Data

```
libname sasuser 'c:\users\admit.xlsx' mixed=yes;
proc contents data=sasuser._all_;
run;
proc print data=sasuser.'worksheet1$'n;
run;
```

Creating an Excel Worksheet

```
libname clinic 'c:\Users\mylaptop\admitxl.xlsx' mixed=yes;
data clinic.admit;
   set work.admit;
run;
```

Points to Remember

- LIBNAME and FILENAME statements are global. Librefs and filerefs remain in effect until you change them, cancel them, or end your SAS session.
- For each field of raw data that you read into your SAS data set, you *must* specify the following in the INPUT statement: a valid SAS variable name, a type (character or numeric), a starting column, and if necessary, an ending column.
- When you use column input, you can read any or all fields from the raw data file, read the fields in any order, and

specify only the starting column for variables whose values occupy only one column.

- Column input is appropriate only in some situations. When you use column input, your data *must* be standard character and numeric values, and these values *must* be in fixed fields. That is, values for a particular variable must be in the same location in all records.

Chapter Quiz

Select the best answer for each question. After completing the quiz, you can check your answers using the answer key in the appendix.

1. Which SAS statement associates the fileref Crime with the raw data file **`C:\States \Data\crime.dat?`** ?

a. `filename crime 'c:\states\data\crime.dat';`

b. `filename crime c:\states\data\crime.dat;`

c. `fileref crime 'c:\states\data\crime.dat';`

d. `filename 'c:\states\data\crime' crime.dat;`

2. Filerefs remain in effect until … ?

a. you change them.

b. you cancel them.

c. you end your SAS session.

d. all of the above

3. Which statement identifies the name of a raw data file to be read with the fileref Products and specifies that the DATA step read-only records 1-15? ?

a. `infile products obs 15;`

b. `infile products obs=15;`

c. `input products obs=15;`

d. `input products 1-15;`

4. Which of the following programs correctly writes the observations from the data set below to a raw data file? ?

SAS data set Work.Patients

ID	Sex	Age	Height	Weight	Pulse
2304	F	16	61	102	100
1128	M	43	71	218	76
4425	F	48	66	162	80
1387	F	57	64	142	70
9012	F	39	63	157	68
6312	M	52	72	240	77
5438	F	42	62	168	83
3788	M	38	73	234	71
9125	F	56	64	159	70
3438	M	15	66	140	67

Figure 5.32: Data Set work.patients

a.
```
data _null_;
    set work.patients;
```

```
   infile 'c:\clinic\patients\referals.dat';
   input id $ 1-4 sex 6 $ age 8-9 height 11-12
         weight 14-16 pulse 18-20;
run;
```

b.
```
data referals.dat;
   set work.patients;
   input id $ 1-4 sex $ 6 age 8-9 height 11-12
         weight 14-16 pulse 18-20;
run;
```

c.
```
data _null_;
   set work.patients;
   file c:\clinic\patients\referals.dat;
   put id $ 1-4 sex 6 $ age 8-9 height 11-12
       weight 14-16 pulse 18-20;
run;
```

d.
```
data _null_;
   set work.patients;
   file 'c:\clinic\patients\referals.dat';
   put id $ 1-4 sex 6 $ age 8-9 height 11-12
       weight 14-16 pulse 18-20;
run;
```

5. Which raw data file can be read using column input? ?

a.
```
1---+----10---+----20---+
Henderson CA 26 ADM
Josephs SC 33 SALES
Williams MN 40 HRD
Rogan NY RECRTN
```

Figure 5.33: Raw Data File A

b.
```
1---+----10---+----20---+----30
2803 Deborah Campos  173.97
2912 Bill    Marin   205.14
3015 Helen   Stinson 194.08
3122 Nicole  Terry   187.65
```

Figure 5.34: Raw Data File B

```
1---+----10---+----20---+
Avery John          $601.23
Davison Sherrill    $723.15
Holbrook Grace      $489.76
Jansen Mike         $638.42
```

c.

Figure 5.35: Raw Data File C

d. all of the above.

6. Which program creates the output shown below? ?

```
1---+----10---+----20---+----30
3427 Chen     Steve Raleigh
1436 Davis    Lee   Atlanta
2812 King     Vicky Memphis
1653 Sanchez  Jack  Atlanta
```

Obs	ID	LastName	FirstName	City
1	3427	Chen	Steve	Raleigh
2	1436	Davis	Lee	Atlanta
3	2812	King	Vicky	Memphis
4	1653	Sanchez	Jack	Atlanta

Figure 5.36: Raw Data and SAS Output Data Set

a.
```
data work.salesrep;
   infile empdata;
   input ID $ 1-4 LastName $ 6-12
         FirstName $ 14-18 City $ 20-29;
run;
proc print data=work.salesrep;
run;
```

b.
```
data work.salesrep;
   infile empdata;
   input ID $ 1-4 Name $ 6-12
         FirstName $ 14-18 City $ 20-29;
run;
proc print data=work.salesrep;
run;
```

c.
```
data work.salesrep;
   infile empdata;
   input ID $ 1-4 name1 $ 6-12
         name2 $ 14-18 City $ 20-29;
run;
proc print data=work.salesrep;
run;
```

d. all of the above.

7. Which statement correctly reads the fields in the following order: StockNumber, Price, Item, Finish, Style? ?

Field Name	Start Column	End Column	Data Type
StockNumber	1	3	character
Finish	5	9	character
Style	11	18	character
Item	20	24	character
Price	27	32	numeric

```
1---+----10---+----20---+----30---+
310 oak   pedestal table  329.99
311 maple pedestal table  369.99
312 brass floor    lamp   79.99
313 glass table    lamp   59.99
313 oak   rocking  chair  153.99
```

Figure 5.37: Raw Data

a.
```
input StockNumber $ 1-3 Finish $ 5-9 Style $ 11-18
      Item $ 20-24 Price 27-32;
```

b.
```
input StockNumber $ 1-3 Price 27-32
      Item $ 20-24 Finish $ 5-9 Style $ 11-18;
```

c.
```
input $ StockNumber 1-3 Price 27-32 $
     Item 20-24 $ Finish 5-9 $ Style 11-18;
```

d.
```
input StockNumber $ 1-3 Price $ 27-32
      Item $ 20-24 Finish $ 5-9 Style $ 11-18;
```

8. Which statement correctly re-defines the values of the variable Income as 100 percent higher? ?

a.
```
income=income*1.00;
```

b.
```
income=income+(income*2.00);
```

c.
```
income=income*2;
```

d.
```
income=*2;
```

9. Which program correctly reads instream data? ?

a.
```
data finance.newloan;
   input datalines;
   if country='JAPAN';
```

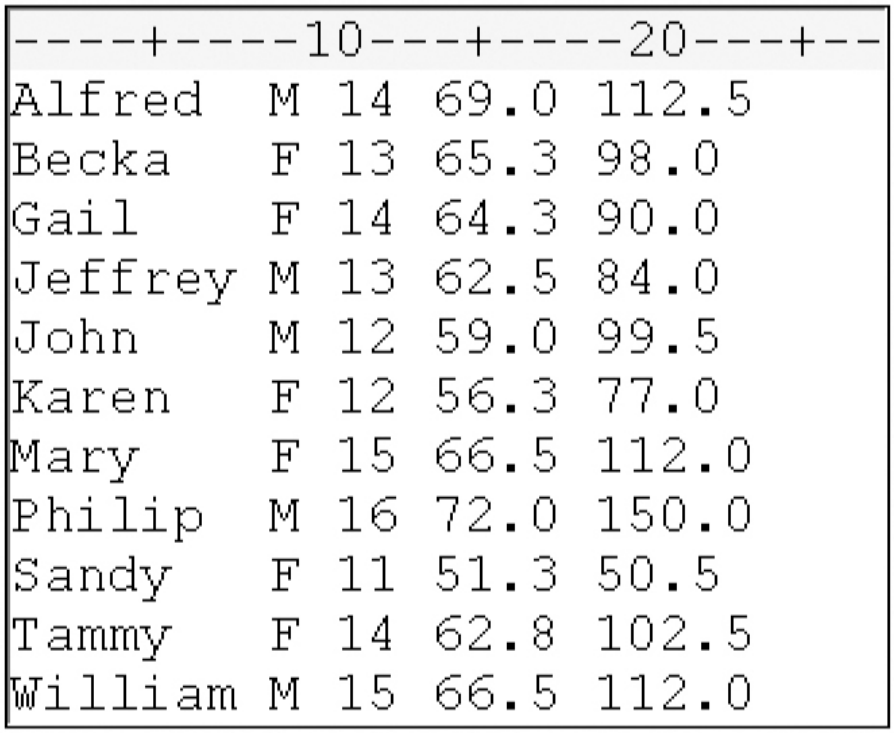
```
   MonthAvg=amount/12;
1998 US     CARS   194324.12
1998 US     TRUCKS 142290.30
1998 CANADA CARS    10483.44
1998 CANADA TRUCKS  93543.64
1998 MEXICO CARS    22500.57
1998 MEXICO TRUCKS  10098.88
1998 JAPAN  CARS    15066.43
1998 JAPAN  TRUCKS  40700.34
;
```

b.
```
data finance.newloan;
   input Year 1-4 Country $ 6-11
         Vehicle $ 13-18 Amount 20-28;
   if country='JAPAN';
   MonthAvg=amount/12;
   datalines;
run;
```

c.
```
data finance.newloan;
   input Year 1-4 Country 6-11
         Vehicle 13-18 Amount 20-28;
   if country='JAPAN';
   MonthAvg=amount/12;
   datalines;
1998 US     CARS   194324.12
1998 US     TRUCKS 142290.30
1998 CANADA CARS   10483.44
1998 CANADA TRUCKS  93543.64
1998 MEXICO CARS    22500.57
1998 MEXICO TRUCKS  10098.88
1998 JAPAN  CARS    15066.43
1998 JAPAN  TRUCKS  40700.34
;
```

d.
```
data finance.newloan;
   input Year 1-4 Country $ 6-11
         Vehicle $ 13-18 Amount 20-28;
   if country='JAPAN';
   MonthAvg=amount/12;
   datalines;
1998 US     CARS   194324.12
1998 US     TRUCKS 142290.30
1998 CANADA CARS    10483.44
1998 CANADA TRUCKS  93543.64
1998 MEXICO CARS    22500.57
1998 MEXICO TRUCKS  10098.88
1998 JAPAN  CARS    15066.43
1998 JAPAN  TRUCKS  40700.34
;
```

10. Which SAS statement subsets the raw data shown below so that only the observations in which Sex (in the second field) has a value of **F** are processed? ?



```
----+----10---+----20---+--
Alfred  M 14 69.0 112.5
Becka   F 13 65.3 98.0
Gail    F 14 64.3 90.0
Jeffrey M 13 62.5 84.0
John    M 12 59.0 99.5
Karen   F 12 56.3 77.0
Mary    F 15 66.5 112.0
Philip  M 16 72.0 150.0
Sandy   F 11 51.3 50.5
Tammy   F 14 62.8 102.5
William M 15 66.5 112.0
```

Figure 5.38: Raw Data

a. `if sex=f;`

b. `if sex=F;`

c. `if sex='F';`

d. a or b

Answers

1. Correct answer: a

You assign a fileref by using a FILENAME statement in the same way that you assign a libref by using a LIBNAME statement.

2. Correct answer: d

Like LIBNAME statements, FILENAME statements are global; they remain in effect until you change them, cancel them, or end your SAS session.

3. Correct answer: b

You use an INFILE statement to specify the raw data file to be read. You can specify a fileref or an actual filename (in quotation marks). The OBS= option in the INFILE statement enables you to process only records 1 through *n*.

4. Correct answer: d

The keyword _NULL_ in the DATA statement enables you to use the power of the DATA step without actually creating a SAS data set. You use FILE and PUT statements to write observations from a SAS data set to a raw data file. The FILE statement specifies the raw data file and the PUT statement describes the lines to write to the raw data file. The filename and location specified in the FILE statement must be enclosed in quotation marks.

5. Correct answer: b

Column input is appropriate only in some situations. When you use column input, your data must be standard character or numeric values, and they must be in fixed fields. That is, values for a particular variable must be in the same location in all records.

6. Correct answer: a

The INPUT statement creates a variable using the name that you assign to each field. Therefore, when you write an INPUT statement, you need to specify the variable names exactly as you want them to appear in the SAS data set.

7. Correct answer: b

You can use column input to read fields in any order. You must specify the variable name, identify character variables with a $, and specify the correct starting and ending column for each field.

8. Correct answer: c

To re-define the values of the variable Income in an assignment statement, you specify the variable name on the left side of the equal sign and an appropriate expression including the variable name on the right side of the equal sign.

9. Correct answer: d

To read instream data, you specify a DATALINES statement and data lines, followed by a null statement (single semicolon) to indicate the end of the input data. Program a contains no DATALINES statement, and the INPUT statement doesn't specify the fields to read. Program b contains no data lines, and the INPUT statement in program c doesn't specify the necessary dollar signs for the character variables Country and Vehicle.

10. Correct answer: c

To subset data, you can use a subsetting IF statement in any DATA step to process only those observations that meet a specified condition. Because Sex is a character variable, the value F must be enclosed in quotation marks and must be in the same case as in the data set.