Chapter 2 SAS® Programs

0.1 Introduction.....Error! Bookmark not defined.

0.2 A Section Title.....Error! Bookmark not defined.

Demonstration: <Type title of demo here.>.....**Error! Bookmark not defined.**

Exercises .....**Error! Bookmark not defined.**

0.3 Chapter Summary.....Error! Bookmark not defined.

0.4 Solutions.....Error! Bookmark not defined.

Solutions to Exercises .....**Error! Bookmark not defined.**

Solutions to Student Activities (Polls/Quizzes) .....**Error! Bookmark not defined.**

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.1 Introduction to SAS Programs

Objectives

- List the components of a SAS program.

3

SAS Programs

A *SAS program* is a sequence of one or more steps.

- *DATA steps* typically create SAS data sets.
- *PROC steps* typically process SAS data sets to generate reports and graphs, and to manage data.

4

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Program Steps

A *step* is a sequence of SAS statements. This program has a DATA step and a PROC step.

```
data work.newemps;
   infile "&path\newemps.csv" dlm=',';
   input First $ Last $ Title $ Salary;
run;

proc print data=work.newemps;
run;
```

5

Step Boundaries

SAS steps begin with either of the following:

- a DATA statement
- a PROC statement

SAS detects the end of a step when it encounters one of the following:

- a RUN statement (for most steps)
- a QUIT statement (for some procedures)
- the beginning of another step (DATA statement or PROC statement)

6

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.01 Quiz

How many steps are in this program?

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

7 p102d01

SAS Program Example

This DATA step creates a temporary SAS data set named **work.newsalesemps** by reading four fields from a file.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

9 p102d01

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Program Example

This PROC PRINT step lists the **work.newsalesemps** data set.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

10 p102d01

SAS Program Example

This PROC MEANS step summarizes the **Salary** variable in the **work.newsalesemps** data set.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

11 p102d01

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.02 Quiz

How does SAS detect the end of each step in this program?

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;

proc means data=work.newsalesemps;
   var Salary;
```

12

2.2 Submitting a SAS Program

Objectives

- Use SAS Enterprise Guide to open and submit a SAS program and browse the results.
- Use the SAS windowing environment to open and submit a SAS program and browse the results.

16

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Submitting a SAS Program: SAS Enterprise Guide

p102d01

- Start SAS Enterprise Guide.
- Open and submit a SAS program.
- Examine the results.
- Select output formats.

This demonstration is based on SAS Enterprise Guide 5.1.

Starting SAS Enterprise Guide

1. Start SAS Enterprise Guide. The method that you use varies by customizations in effect at your site.
2. Close the Welcome to SAS Enterprise Guide window by clicking .

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Opening and Submitting a SAS Program

1. Open **p102d01**. To open a SAS program in Enterprise Guide, select **File** ⇨ **Open** ⇨ **Program** or click and then select **Program**. Navigate to your data folder, select the file that you want to include, and click **Open**. The program is displayed on the Program tab of the workspace.

You can use the Program tab to access and edit existing SAS programs, write new programs, submit programs, and save programs to a file. The program is color-coded in the Program Editor.

2. To submit the program for execution, select **Program** ⇨ **Run** *program-name* **on** <*server*> or click **Run** on the Program tab. The F3 or F8 keys can also be used to submit a program.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Examining the Log

You should always check the log to make sure that the program ran successfully, even when output is generated. The log echoes the programming statements that were submitted as well as statements added by SAS Enterprise Guide. The log also contains notes about the program execution and results, as well as error and warning messages when appropriate. In this example, there are no warning or error messages.

The statements added by Enterprise Guide are used during processing and are referred to as *wrapper code*. To suppress the display of the wrapper code in the log, select **Tools** ⇨ **Options** ⇨ **Results General** and clear **Show generated wrapper code in SAS log**.

To scroll horizontally in the log, use the horizontal scroll bar. To scroll vertically, use the vertical scroll bar or use the PAGE UP or PAGE DOWN keys on the keyboard.

Examining the Results

The Results tab displays the program output. It becomes the active tab each time it receives output.

Obs	First_Name	Last_Name	Job_Title	Salary
1	Satyakam	Denny	Sales Rep. II	26780
2	Monica	Kletschkus	Sales Rep. IV	30890
3	Kevin	Lyon	Sales Rep. I	26955
4	Petrea	Soltau	Sales Rep. II	27440
5	Marina	Iyengar	Sales Rep. III	29715

To scroll vertically in the Results window, use the vertical scroll bar or use the PAGE UP or PAGE DOWN keys on the keyboard.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Selecting Output Formats

SAS Enterprise Guide 5.1 creates SAS Report output by default. Other formats, including HTML, PDF, RTF, or Text output, can be requested instead of, or in addition to, the default output.

1. To change the result format, select **Tools** ⇨ **Options** ⇨ **Results** ⇨ **Results General**. The changes affect all further output, until another change is made.

2. Select the desired result formats: **SAS Report** and **Text output**. You can also set an alternate default format. Click **OK**.

3. Click the **Program** tab to display the program and then run the program again.

4. Click **Yes** to replace the results from the previous run.

5. Click the **Results** tabs to view the different result formats.

6. Reset the output formats to generate SAS Report or text output (or both) as desired.

Text output is shown throughout this book.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Submitting a SAS Program: SAS Windowing Environment

p102d01

- Start a SAS session.
- Open and submit a SAS program.
- Examine the results.
- Select output formats.

This demonstration is based on SAS 9.3.

Starting a SAS Session

1. Start SAS. The method that you use varies by customizations in effect at your site.
2. Close the Change Notice/Getting Started with SAS window by clicking or **Close**.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Opening and Submitting a SAS Program

1. Open **p102d01**. To open a SAS program, select **File** ⇨ **Open Program**, or with the Editor window active, click . Use the Open window to navigate to the folder that contains your SAS programs. Select the desired file in the Open Program window and click **Open**. The program is displayed in an Editor window.

You use the editor to access and edit existing SAS programs, write new programs, submit programs, and save programs to a file. Programs are color-coded in the Editor window.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Alternatively, you can issue the INCLUDE command to open or include a program in your session. With the Editor window active, type **include** on the command bar followed by the name of the program file enclosed in quotation marks. SAS looks for the file in the current or active folder.

To change the current folder, double-click the path indicator S:\workshop on the SAS taskbar. A Change Folder dialog box is displayed. Navigate to the appropriate folder and click **OK**.

Type **include 'p102d01.sas'** and press ENTER. The program is displayed in the Editor window.

2. To submit the program for execution, with the Editor window active, click , or select **Run** ⇨ **Submit**. Alternatively, you can type **submit** on the command bar or press F3 or F8.

Examining the Log

The Log window is one of the three primary windows and is open by default. The Log window provides an audit trail of your SAS session. It echoes the programming statements that were submitted and includes error or warning messages, as well as any notes about the program execution.

You should always check the log for errors or warnings, even if the program generates output.

1. To access the Log window, click the **Log** tab near the bottom of the SAS window. You can also select **View** ⇨ **Log** or submit the LOG command using the command bar.

   With the Log window active, use the vertical scroll bar or the Page Up/Page Down keys to scroll up to the first line of the program that you just submitted. The contents of the Log window are cumulative, with the most recent information added to the bottom, so be sure you are looking at the most recent information.

   In this example, the Log window contains no warning or error messages.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2. To clear the contents of the Log window, make sure that the window is active and then click . You can also issue the CLEAR command or select **Edit** ⇨ **Clear All**.

Examining the Results

Results Viewer

The SAS System

Obs	First_Name	Last_Name	Job_Title	Salary
1	Satyakam	Denny	Sales Rep. II	26780
2	Monica	Kletschkus	Sales Rep. IV	30890
3	Kevin	Lyon	Sales Rep. I	26955
4	Petrea	Soltau	Sales Rep. II	27440
5	Marina	Iyengar	Sales Rep. III	29715

The MEANS Procedure

Analysis Variable : Salary				
N	Mean	Std Dev	Minimum	Maximum
71	27287.18	1764.74	25020.00	31865.00

SAS HTML output, the default output in SAS 9.3, is displayed automatically in the Results Viewer.

Use the vertical scroll bar or the PAGE UP and PAGE DOWN keys on the keyboard to scroll from top to bottom. You can also enter the TOP and BOTTOM commands on the command bar to scroll vertically. Use the horizontal scroll bar, if displayed, to scroll side to side.

Selecting Output Formats

SAS 9.3 creates HTML output by default. LISTING output can be requested in addition to or instead of the default HTML output.

1. Select **Tools** ⇨ **Options** ⇨ **Preferences** and click the **Results** tab.

2. Select or clear the **Create Listing** and **Create HTML** boxes to generate both LISTING and HTML output.

LISTING output is shown throughout this book.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

3. Return to the Editor window for **p101d01** and resubmit the program. The output is added to the Results window, so it now contains two copies of the report. The log also contains information about the previous and current programs, unless you cleared it before resubmitting the program.

Output Window

1. LISTING output, produced by request only, is displayed in the Output window. Click the **Output** tab to view the Output window.

```
                                  The SAS System      14:12 Monday, September

Obs     First_Name      Last_Name                Job_Title          Salary

  1     Satyakam        Denny                 Sales Rep. II          26780
  2     Monica          Kletschkus            Sales Rep. IV          30890
  3     Kevin           Lyon                  Sales Rep. I           26955
  4     Petrea          Soltau                Sales Rep. II          27440
  5     Marina          Iyengar               Sales Rep. III         29715
```

   - In the SAS windowing environment, the content displayed in the Output window is the last page of output generated by the most recent program. Scroll up to see the top of the report.
   - To scroll vertically in the Output window, use the PAGE UP or PAGE DOWN keys on the keyboard, use the vertical scroll bar, or submit the FORWARD and BACKWARD commands.
   - You can also use the TOP and BOTTOM commands to scroll vertically in the Output window.
   - Use the horizontal scroll bar or issue the RIGHT and LEFT commands to scroll horizontally.

2. To clear the contents of the Output window, make sure that window is active and then click or issue the CLEAR command, or select **Edit** ⇨ **Clear All**.

Navigating with the Results Window

You can use the Results window to navigate through your output. It contains bookmarks that can be expanded and collapsed.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Click a plus sign to expand or collapse individual bookmarks, or right-click the word **Results** and select **Expand All/Collapse All**. Each bookmark contains an icon indicating the file type: HTML or LISTING. Double-click a bookmark to view the corresponding report in the appropriate viewer.

Accessing SAS Product Documentation on the Web

1. In SAS Enterprise Guide, select **Help** ⇨ **SAS on the Web** ⇨ **SAS Product Documentation**. The SAS Product Documentation web page is displayed.

2. In the SAS windowing environment, select **Help** ⇨ **SAS on the Web** ⇨ **Customer Support Center**. Select **Documentation** on the Customer Support web page. The SAS Product Documentation web page is displayed.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.03 Multiple Choice Poll

Which environment (or environments) will you use during this course?

a. SAS windowing environment
b. SAS Enterprise Guide
c. both the SAS windowing environment and SAS Enterprise Guide

21

Exercises

You can use SAS Enterprise Guide or the SAS windowing environment to complete your exercises. Select the type of output that you prefer. LISTING output is shown in the exercises.

Level 1

1. Submitting a Program

a. With the appropriate Editor window active, open the SAS program **p102e01**.

To open the program in SAS Enterprise Guide:

Select **Program** ⇨ **Open Program**, select **p102e01**, and click **Open**.

To open the program in SAS:

Windows or UNIX	Select **File** ⇨ **Open Program** and select **p102e01**.
z/OS (OS/390)	Issue the command: **include '.workshop.sascode(p102e01)'**.

b. Submit the program for execution. How many rows and columns are in the report?

rows: _____ columns: _____

c. Examine the Log window. Based on the log notes following the DATA step, how many observations and variables are in the **work.country** data set?

observations: _____ variables: _____

d. Clear the Log and Output windows (SAS windowing environment only).

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Level 2

2. Exploring Your Environment: SAS Enterprise Guide

a. Customize the appearance and functionality of the Editor by selecting **Tools** ⇨ **Options** ⇨ **SAS Programs**. For example, click **Editor Options** and click the **Appearance** tab to modify the font and font size.

b. Customize the type of output produced by selecting **Tools** ⇨ **Options** ⇨ **Results General**. Select or deselect **SAS Report**, **HTML**, **PDF**, **RTF**, and **Text Output**. The default format is SAS Report. Text output is shown in the course notes.

3. Exploring Your SAS Environment: Windows

a. Customize the appearance and functionality of the Enhanced Editor by selecting **Tools** ⇨ **Options** ⇨ **Enhanced Editor**. For example, click the **Appearance** tab to modify the font and font size.

b. Customize the type of output produced by selecting **Tools** ⇨ **Options** ⇨ **Preferences**. Click the **Results** tab and select or clear **Create listing** and **Create HTML** as appropriate. The default format is HTML. LISTING output is shown in the course notes.

Challenge

4. Enabling and Disabling the Project Log (SAS Enterprise Guide Only)

a. Use the Index tab in SAS Enterprise Guide Help to find information about the project log.

b. Enable and turn on the project log. Run a few programs and view the log to see its contents.

5. Setting Up a Function Key to Clear the Log and Output Windows (SAS Windowing Environment Only)

a. Issue the KEYS command or select **Tools** ⇨ **Options** ⇨ **Keys** to access the KEYS window. The KEYS window is a secondary window used to browse or change function key definitions.

b. Type the following commands in the Definition column for the F12 key:

```
clear log; clear output
```

c. Which key is programmed to submit a KEYS command? _____

d. Close the KEYS window. This saves the key definition in your user profile.

e. Press the F12 key and confirm that the Log and Output windows are cleared.

It is not necessary to clear the Log and Output windows in SAS Enterprise Guide because this is done automatically each time a program is submitted.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.3 SAS Program Syntax

Objectives

- Identify the characteristics of SAS statements.
- Define SAS syntax rules.
- Document a program using comments.
- Diagnose and correct a program with errors.
- Save the corrected program.

24

Business Scenario

Well-formatted, clearly documented SAS programs are an industry best practice.

25

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Syntax Rules: Statements

SAS statements

- usually begin with an ***identifying keyword***
- always end with a ***semicolon***.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

26 p102d01

2.04 Quiz

How many statements make up this DATA step?

a. one
b. three
c. five
d. seven

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;
```

27

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Program Structure

SAS code is free format.

```
data work.newsalesemps;
length First_Name $ 12
Last_Name $ 18 Job_Title $ 25;
infile "&path\newemps.csv" dlm=',';
input First_Name $ Last_Name $
Job_Title $ Salary;run;
proc print data=work.newsalesemps; run;
   proc means data  =work.newsalesemps;
var Salary;run;
```

This program is syntactically correct but difficult to read.

p102d02

29

SAS Program Structure

Rules for SAS Statements

- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can appear on the same line.
- Unquoted values can be lowercase, uppercase, or mixed case.

```
data work.newsalesEmps;
length First_Name $ 12
Last_Name $ 18 Job_Title $ 25;
infile "&path\newemps.csv" dlm=',';
input First_Name $ Last_Name $
Job_Title $ Salary;run;
proc print data=work.newsalesemps; run;
   proc means data  =work.newsalesemps;
var Salary;run;
```

unconventional formatting

30

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Recommended Formatting

- Begin each statement on a new line.
- Use white space to separate words and steps.
- Indent statements within a step.
- Indent continued lines in multi-line statements.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

conventional formatting

31

White space can be blanks, tabs, and new lines. Add them to increase the readability of the code.

Automatic Formatting with SAS Enterprise Guide

p102d02

SAS Enterprise Guide has an auto-formatting tool that applies conventional formatting style to a SAS program by adding line breaks and indenting each line correctly according to its nesting level.

```
data work.newsalesemps;
length First_Name $ 12
Last_Name $ 18 Job_Title $ 25;
 infile "&path\newemps.csv" dlm=',';
input First_Name $ Last_Name $
Job_Title $ Salary;run;
proc print data=work.newsalesemps; run;
   proc means data =work.newsalesemps;
var Salary;run;
```

1. Open and submit **p102d02**. This is the program seen earlier, but with unconventional formatting.
2. View the log and output to confirm that the program ran successfully.
3. Click the **Program** tab and select **Edit** ⇨ **Format Code**. The program is formatted automatically.

   You can undo the automatic formatting by selecting **Edit** ⇨ **Undo** or pressing the CTRL and Z keys simultaneously.

4. Select **File** ⇨ **Save p102d02** to save the modified program file.

   To customize the formatting, select **Program** ⇨ **Editor Option** and click the **Indenter** tab.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Program Documentation

You can embed comments in a program as explanatory text.

```
  /* create a temporary data set, newsalesemps */
  /* from the text file newemps.csv            */

data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   *read a comma delimited file;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;
```

/ comment */*

** comment statement ;*

SAS ignores comments during processing but writes them to the SAS log.

33

*** comment;** These comments must be written as separate statements and cannot contain internal semicolons.

/* comment */ These comments can be any length and can contain semicolons. They cannot be nested.

SAS Comments

This program contains four comments.

```
*----------------------------------------*
|   This program creates and uses the    |  ❶
|   data set called work.newsalesemps.   |
*----------------------------------------*;
data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary /*numeric*/;❷
run;
  /*
proc print data=work.newsalesemps; ❸
run;
  */
proc means data=work.newsalesemps;
   *var Salary; ❹
run;
```

p102d03

34

In SAS Enterprise Guide and the Enhanced Editor in SAS, to comment out a block of code using the /* */ technique, you can highlight the code and then press the CTRL key and the / (forward slash) key simultaneously. To uncomment a block of code, highlight the block and press the CTRL, SHIFT, and / keys simultaneously.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.05 Quiz

Open and examine **p102a01**. Based on the comments, which steps do you think will execute and what output will be generated?

Submit the program. Which steps are executed?

35

Business Scenario

Orion Star programmers must be able to identify and correct syntax errors in a SAS program.

38

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Syntax Errors

A *syntax error* is an error in the spelling or grammar of a SAS statement. SAS finds syntax errors as it compiles each SAS statement, before execution begins.

Examples of syntax errors:

- misspelled keywords
- unmatched quotation marks
- missing semicolons
- invalid options

39

2.06 Quiz

This program includes three syntax errors. One is an invalid option. What are the other two syntax errors?

```
daat work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps
run;

proc means data=work.newsalesemps average min;
   var Salary;
run;
```

invalid option

40

p102d04

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Syntax Errors

The Enhanced Editor in SAS and the Program Editor in SAS Enterprise Guide use the color red to indicate a potential error in your SAS code.

```
daat work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps
run;

proc means data=work.newsalesemps average min;
  var Salary;
run;
```

42

Syntax Errors

When SAS encounters a syntax error, it writes a warning or error message to the log.

```
ERROR 22-322: Syntax error, expecting one of the following:
              a name, a quoted string, (, /, ;, _DATA_, _LAST_,
              _NULL_.
```

```
WARNING: Data set WORK.TEST was not replaced because this step was
stopped.
```

You should always check the log to make sure that the program ran successfully, even if output is generated.

43

Diagnosing and Correcting Syntax Errors

p102d04

- Submit a SAS program with errors.
- Diagnose and correct the errors.
- Save the corrected program.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Submitting a SAS Program with Errors

```
daat work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=','
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps
run;

proc means data=work.newsalesemps average max;
   var Salary;
run;
```

1. Open and submit **p102d04**.
2. Use the vertical scroll bar to view error messages and warnings in the log.

In SAS Enterprise Guide, you can use the Up and Down arrows in the Log menu to find the previous or next warning or error in the log.

Partial SAS Log

```
77   daat work.newsalesemps;
     ----
     14
WARNING 14-169: Assuming the symbol DATA was misspelled as daat.
78        length First_Name $ 12
79               Last_Name $ 18 Job_Title $ 25;
80         infile "&path\newemps.csv" dlm=',';
81         input First_Name $ Last_Name $
82               Job_Title $ Salary;
83   run;
NOTE: The infile "s:\workshop\newemps.csv" is:
      Filename=s:\workshop\newemps.csv,
NOTE: 71 records were read from the infile "s:\workshop\newemps.csv".
NOTE: The data set WORK.NEWSALESEMPS has 71 observations and 4 variables.
84
85   proc print data=work.newsalesemps
86   run;
     ---
     22
     202
ERROR 22-322: Syntax error, expecting one of the following: ;, (, BLANKLINE, DATA, DOUBLE,
              HEADING, LABEL, N, NOOBS, OBS, ROUND, ROWS, SPLIT, STYLE, SUMLABEL, UNIFORM,
              WIDTH.
```

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
ERROR 202-322: The option or parameter is not recognized and will be ignored.
88
NOTE: The SAS System stopped processing this step because of errors.
89   proc means data=work.newsalesemps average max;
                                       -------
                                       22
                                       202
ERROR 22-322: Syntax error, expecting one of the following: ;, (, ALPHA, CHARTYPE, CLASSDATA,
              CLM, COMPLETETYPES, CSS, CV, DATA, DESCEND, DESCENDING, DESCENDTYPES, EXCLNPWGT,
              EXCLNPWGTS, EXCLUSIVE, FW, IDMIN, KURTOSIS, LCLM, MAX, MAXDEC, MEAN, MEDIAN, MIN,
              MISSING, MODE, N, NDEC, NMISS, NOLABELS, NONOBS, NOPRINT, NOTHREADS, NOTRAP,
              NWAY, ORDER, P1, P10, P20, P25, P30, P40, P5, P50, P60, P70, P75, P80, P90, P95,
              P99, PCTLDEF, PRINT, PRINTALL, PRINTALLTYPES, PRINTIDS, PRINTIDVARS, PROBT, Q1,
              Q3, QMARKERS, QMETHOD, QNTLDEF, QRANGE, RANGE, SKEWNESS, STACKODS,
              STACKODSOUTPUT, STDDEV, STDERR, SUM, SUMSIZE, SUMWGT, T, THREADS, UCLM, USS, VAR,
              VARDEF.
ERROR 202-322: The option or parameter is not recognized and will be ignored.
90      var Salary;
91   run;
NOTE: The SAS System stopped processing this step because of errors.
```

Diagnosing and Correcting the Errors

The log indicates that SAS

- assumed that the keyword DATA was misspelled, issued a warning, and executed the DATA step
- interpreted the word RUN as an option in the PROC PRINT statement (because of a missing semicolon), so the PROC PRINT step did not execute
- did not recognize the word AVERAGE as a valid option in the PROC MEANS statement, so the PROC MEANS step did not execute.

If you are using the Enterprise Guide Editor or the SAS Enhanced Editor, the program remains in the Editor window. However, if you use the SAS Program Editor, the code disappears with each submission. Use the RECALL command to recall the program. You can also select **Run** ⇨ **Recall Last Submit**, or press F4. The program is redisplayed in the Program Editor.

1. Edit the program.

   a. Correct the spelling of DATA.

   b. Put a semicolon at the end of the PROC PRINT statement.

   c. Change the word AVERAGE to MEAN in the PROC MEANS statement.

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;
```

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
proc means data=work.newsalesemps mean max;
   var Salary;
run;
```

2. Submit the program. Verify that it executes without warnings or errors and produces output.

When you modify a program, an asterisk (*) is displayed after the program name in the Editor window. When you save the program, the asterisk is removed.

Saving the Corrected Program: SAS Enterprise Guide

With the modified program displayed in the Program window, click Save or select **File** ⇨ **Save** *program* or **File** ⇨ **Save** *program* **As** to save your program to a new file.

Saving the Corrected Program: SAS Windowing Environment

With the modified program displayed in the Editor window, click to save your program to a file. You can also select **File** ⇨ **Save** or **File** ⇨ **Save As**, or use the FILE command (**FILE 'p102d04.sas';**).

If the code is not in the Program Editor, recall it before saving the program.

2.07 Quiz

What is the syntax error in this program?

```
data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

p102d05

45

Correcting Unbalanced Quotation Marks: SAS Windowing Environment

p102d05

- Submit a SAS program that contains unbalanced quotation marks.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

- Diagnose the error.
- Correct and resubmit the program.

Submitting a SAS Program That Contains Unbalanced Quotation Marks

In program **p102d05**, the closing quotation mark for the DLM= option in the INFILE statement is missing. The color-coding in the editor indicates a very long character string.

Submit the program.

```
data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',;
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

Diagnosing the Error

View the log. There are no notes in the log because all of the statements after the DLM= option became part of the quoted string. SAS waits for the closing quotation mark to end the quoted string.

SAS Log

```
93   data work.newsalesemps;
94      length First_Name $ 12 Last_Name $ 18
95             Job_Title $ 25;
96      infile "&path\newemps.csv" dlm=',;
97      input First_Name $ Last_Name $
98            Job_Title $ Salary;
99  run;
100
101  proc print data=work.newsalesemps;
102  run;
103
105  proc means data=work.newsalesemps;
106     var Salary;
107  run;
```

There are no messages of any kind following each step in the log because the steps did not execute. The absence of messages from SAS typically indicates unbalanced quotation marks. Another indication is the “DATA STEP running” message in the banner of the Editor window. This is because the RUN statement was viewed as part of the character literal and not as a step boundary.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Stopping a Program

You can stop an executing program in an interactive session.

1. To stop a program in the Windows environment, click (the break icon) or press the CTRL and Break keys.
2. Select **1. Cancel Submitted Statements** in the Tasking Manager window and click **OK**.

3. Select **Y to cancel submitted statements**, and click **OK**. The program stops executing.

Programmatic Approach to Balancing Quotation Marks

You can balance quotation marks programmatically and eliminate this problem by submitting the following series of statements:

```
*';*";run;
```

SAS maintains a quotation mark counter, and expects an even number of quotation marks. When there is an odd number, the appropriate quotation mark in the code above serves as the matching quotation mark, making the count even. Both single quotation marks and double quotation marks are used in this series of statements, which can fix unbalanced double quotation marks or unbalanced single quotation marks.

Resubmitting a Program

1. Add a closing quotation mark to the DLM= option in the INFILE statement to correct the program.
2. You might choose to insert the statements to balance quotations marks programmatically before your program code. This is not necessary if you have already stopped the DATA step, but some SAS programmers include this as the first line of every program that they write. Alternatively, some programmers add these statements at the end of every program.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
*';*";run;

data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';

   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

3. Resubmit the program and verify that the program ran successfully.

Correcting Unbalanced Quotation Marks: SAS Enterprise Guide

p102d05

- Submit a SAS program that contains unbalanced quotation marks in SAS Enterprise Guide.
- Diagnose the error.
- Correct and resubmit the program.

1. Open and submit **p102d05** in SAS Enterprise Guide.

```
data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',;
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

2. View the log. It contains SAS Enterprise Guide wrapper code in addition to the code that you submitted. Notice the first line of wrapper code.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

It is very similar to the statement that is entered in SAS to correct the unbalanced quotation marks, but it also contains a single semicolon, a */ to close a comment, and a QUIT statement. SAS Enterprise Guide is attempting to fix any potential unbalanced situation from a previously submitted program.

The program did not execute successfully, but this time there is a warning and an error in the log, and the DATA step stopped. (Your log might contain different warnings or errors than those shown here.)

```
 WARNING: The quoted string currently being processed has become more than 262 characters long.
         You might have unbalanced quotation marks.

36          %LET _CLIENTPROJECTNAME=;
37          %LET _SASPROGRAMFILE=;
38
39          ;*';*";*/;quit;run;
                      ____
                      180
ERROR 180-322: Statement is not valid or it is used out of proper order.

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.NEWSALESEMPS may be incomplete.  When this step was stopped there
         were 0 observations and 3 variables.
```

The first warning above says that a quoted string has exceeded 262 characters. This is not a syntax error, but the message is an indication that the program might have unbalanced quotation marks.

The error is caused by the QUIT statement. Although it is included to prevent errors, in this case it caused an error, but the program did not hang as it did in the SAS session.

3. Add a closing quotation mark to the DLM= option in the INFILE statement to correct the program.

4. Resubmit the program, replacing the previous results. Verify that the program runs successfully.

Stopping a Program

SAS Enterprise Guide has built-in features to prevent SAS programs from hanging, but occasionally you might need to stop or interrupt a program while it is executing. There are two methods:

Method 1: Click the **Stop** button next to the Run button. It is normally dimmed, but it is red and selectable while a program is executing.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Method 2: Click **View** ⇨ **Task Status** to access the Task Status window.

The Task Status window shows the status of all running tasks and programs.

Right-click the task and then select **Stop** or **End SAS Process**.

- **Stop** – ends the current task while maintaining your connection to the server. If the server is in the middle of processing a step, it might take a few minutes for the server to stop running the task.
- **End SAS Process** – immediately ends the current task and terminates the connection with the server. Because the connection with the server is terminated, any temporary data is lost.

Exercises

Level 1

6. Diagnosing and Correcting Syntax Errors

a. With the appropriate Editor window active, open the SAS program **p102e06**.

b. Submit the program and use the notes in the SAS log to identify the error.

c. Correct the error and resubmit the program.

d. Save the corrected program.

Level 2

7. Diagnosing and Correcting Syntax Errors

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

a. With the appropriate Editor window active, open the SAS program **p102e07**.

b. Submit the program and use the notes in the SAS log to identify the error.

c. Correct the error and resubmit the program.

d. Save the corrected program.

Challenge

8. Identifying SAS Components

Explore the Internet to find how to list SAS software products licensed at your site.

2.4 Solutions

Solutions to Exercises

1. Submitting a Program

a. Submit the program.

b. The report has **238 rows** and **3 columns**.

c. The Log window indicates **238 observations** and **2 variables**.

d. SAS windowing environment only: Click in the Log and Output windows, or select **Edit** ⇨ **Clear All** in both windows.

2. Exploring Your Environment: SAS Enterprise Guide

a. Select **Tools** ⇨ **Options** ⇨ **Enhanced Editor**. Click the **Appearance** tab. Select the desired font size using the Size box in the Font section. Click **OK**.

b. Select **Tools** ⇨ **Options** ⇨ **Preferences**. Click the **Results** tab. Select the **Create listing** box in the Listing section. Select or clear the **Create HTML** box in the HTML section.

3. Exploring Your SAS Environment: Windows

a. Select **Tools** ⇨ **Options** ⇨ **SAS Programs**. Click **Editor Options**. Click the **Appearance** tab. Select the desired font size using the Size box in the Font section. Click **OK**.

b. Select **Tools** ⇨ **Options** ⇨ **Results - Results General**. Select or clear the result formats: **SAS Report**, **HTML**, **PDF,** and **RTF**. If **SAS Report** and **HTML** are both selected, you can set one as the default using the Default box.

4. Enabling and Disabling the Project Log (SAS Enterprise Guide Only)

a. Click **Help** ⇨ **SAS Enterprise Guide Help**. Click the **Index** tab and type **Project Log** in the text box. Double-click **project log** from the list of topics.

b. To enable the Project Log, click the **Project Log** tab in the workspace, or select **View** ⇨ **Project Log**. When the Project Log is enabled, you can turn it on and off and clear it using the buttons on the Project Log toolbar.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

5. **Setting Up a Function Key to Clear the Log and Output Windows (SAS Windowing Environment Only)**

   a. Submit the Keys command or select **Tools** ⇨ **Options** ⇨ **Keys**.

   b. Add the commands to the Definition column for the F12 key:

   c. Which key is programmed to submit a KEYS command? **the F9 key**

   d. Close the Keys window.

   e. Press the F12 key to clear the Log and Output windows.

   Only the first three characters are needed for each command, so the following are alternate ways to define the F12 key: **clear log; clear out** or **cle log; cle out**

6. **Diagnosing and Correcting Syntax Errors**

   The program contains a misspelled word. Change **prnt** to **print** and resubmit.

7. **Diagnosing and Correcting Syntax Errors**

   The VAR statement is missing a semicolon, so SAS thinks RUN is a variable in the VAR statement. The error indicates that the variable RUN does not exist. The PROC PRINT step might continue to run, depending on the environment. If so, terminate the program, type the semicolon, and resubmit.

```
proc print data=work.country2;
   var Country_Code Country_Name;
run;
```

8. **Identifying SAS Components**

   PROC SETINIT writes the names and expiration dates of all licensed SAS products to the log.

```
proc setinit;
run;
```

   Partial SAS Log

```
                                Product expiration dates:
---Base Product                                                  31DEC2012
---SAS/STAT                                                      31DEC2012
```

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
---SAS/GRAPH                                                    31DEC2012
```

Solutions to Student Activities (Polls/Quizzes)

2.01 Quiz – Correct Answer

How many steps are in this program? **three**

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

DATA Step

PROC Step

PROC Step

8 p102d01

2.02 Quiz – Correct Answer

How does SAS detect the end of each step in this program?

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;

proc means data=work.newsalesemps;
   var Salary;
```

The DATA step ends at the RUN statement. The PROC PRINT step ends at the PROC MEANS statement. The end of the PROC MEANS step might or might not be detected. A RUN statement is recommended.

13

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.04 Quiz – Correct Answer

How many statements make up this DATA step?

a. one
b. three
c. five
d. seven

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;
```

28

2.05 Quiz – Correct Answer

Open and examine **p102a01**. Based on the comments, which steps do you think will execute and what output will be generated?

Submit the program. Which steps are executed?

- **The DATA step executes and creates an output data set.**
- **The PROC PRINT step executes and produces a report.**
- **The PROC MEANS step is "commented out" and therefore does not execute.**

36

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

2.06 Quiz – Correct Answer

This program includes three syntax errors. One is an invalid option. What are the other two syntax errors?

```
daat work                    mps;
   length                    e $ 12
           Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps
run;

proc means data=work.newsalesemps average min;
   var Salary;
run;
```

misspelled keyword

missing semicolon

invalid option

41

p102d04

2.07 Quiz – Correct Answer

What is the syntax error in this program?

```
data work.newsalesemps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',;
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

The program contains unbalanced quotation marks in the DLM= option in the INFILE statement.

46

p102d05

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 3: Accessing Data

3.1 Examining SAS Data Sets

3.2 Accessing SAS Libraries

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 3: Accessing Data

3.1 Examining SAS Data Sets

3.2 Accessing SAS Libraries

Copyright © 2012, SAS Institute Inc. All rights reserved.

Objectives

- Define the components of a SAS data set.
- Browse the descriptor portion of a SAS data set using the CONTENTS procedure.
- Browse the data portion of a SAS data set using the PRINT procedure.
- Define a SAS variable.
- Define a missing value.
- Define a SAS date value.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Many SAS data sets related to the Orion Star project already exist. The programmers need to know how to display the structure and contents of the data sets.

Copyright © 2012, SAS Institute Inc. All rights reserved.

What Is a SAS Data Set?

A *SAS data set* is a specially structured data file that SAS creates and that only SAS can read. A SAS data set is a table that contains observations and variables.

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Data Set Terminology

SAS Terminology		Database Terminology
SAS Data Set	⟷	Table
Observation	⟷	Row
Variable	⟷	Column

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Data Set Terminology

A SAS data set contains a descriptor portion and a data portion.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Descriptor Portion

The *descriptor portion* contains the following metadata:

- general properties (such as data set name and number of observations)
- variable properties (such as name, type, and length)

Partial **work.newsalesemps**

Data Set Name	**WORK.NEWSALESEMPS**
Engine	**V9**
Created	**Mon, Feb 27, 2012 01:28 PM**
Observations	**71**
Variables	**4**
...	

First_Name	Last_Name	Job_Title	Salary
$ 12	$ 18	$ 25	N 8

general properties

variable properties

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing the Descriptor Portion

Use *PROC CONTENTS* to display the descriptor portion of a SAS data set.

```
proc contents data=work.newsalesemps;
run;
```

PROC CONTENTS DATA=*SAS-data-set***;**
RUN;

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC CONTENTS Output

```
                          The CONTENTS Procedure

Data Set Name    WORK.NEWSALESEMPS                Observations          71
Member Type      DATA                             Variables             4
Engine           V9                               Indexes               0
Created          Mon, Feb 27, 2012 01:28:51 PM    Observation Length    64
Last Modified    Mon, Feb 27, 2012 01:28:51 PM    Deleted Observations  0
Protection                                        Compressed            NO
Data Set Type                                     Sorted                NO


                     Engine/Host Dependent Information
                                   ...


                 Alphabetic List of Variables and Attributes

                       #    Variable       Type    Len

                       1    First_Name     Char     12
                       3    Job_Title      Char     25
                       2    Last_Name      Char     18
                       4    Salary         Num       8
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.01 Quiz

How many observations are in the data set **work.donations**?

- Retrieve program **p103a01**.
- After the DATA step, add a PROC CONTENTS step to view the descriptor portion of **work.donations**.
- Submit the program and review the results.

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.01 Quiz – Correct Answer

How many observations are in the data set **work.donations**? **124 observations**

```
data work.donations;
   infile "&path\donation.dat";
   input Employee_ID Qtr1 Qtr2 Qtr3 Qtr4;
   Total=sum(Qtr1,Qtr2,Qtr3,Qtr4);
run;

proc contents data=work.donations;
run;
```


p103a01s

Copyright © 2012, SAS Institute Inc. All rights reserved.

Data Portion

The *data portion* of a SAS data set contains the data values, which are either character or numeric.

Partial **work.newsalesemps**

First_Name	Last_Name	Job_Title	Salary
Satyakam	Denny	Sales Rep. II	26780
Monica	Kletschkus	Sales Rep. IV	30890
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau	Sales Rep. II	27440

variable names

data values

character values

numeric values

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing the Data Portion

Use *PROC PRINT* to display the data portion of a SAS data set.

```
proc print data=work.newsalesemps;
run;
```

PROC PRINT DATA=*SAS-data-set***;**
RUN;

p103d02

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Job_Title	Salary
1	Satyakam	Denny	Sales Rep. II	26780
2	Monica	Kletschkus	Sales Rep. IV	30890
3	Kevin	Lyon	Sales Rep. I	26955
4	Petrea	Soltau	Sales Rep. II	27440
5	Marina	Iyengar	Sales Rep. III	29715

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Variable Names

SAS variable names

- can be 1 to 32 characters long.
- must start with a letter or underscore. Subsequent characters can be letters, underscores, or numerals.
- can be uppercase, lowercase, or mixed case.
- are not case sensitive.

Salary

score2

cust_ID

month1

FirstName

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.02 Multiple Answer Poll

Which variable names are invalid?

a. data5mon
b. 5monthsdata
c. data#5
d. five months data
e. five_months_data
f. FiveMonthsData
g. fivemonthsdata

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.02 Multiple Answer Poll – Correct Answer

Which variable names are invalid?

a. data5mon
b. 5monthsdata
c. data#5
d. five months data
e. five_months_data
f. FiveMonthsData
g. fivemonthsdata

Copyright © 2012, SAS Institute Inc. All rights reserved.

Data Types

A SAS data set supports two types of variables.

Character variables

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

Numeric variables

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

Copyright © 2012, SAS Institute Inc. All rights reserved.

Missing Data Values

Missing values are valid values in a SAS data set.

Partial **work.newsalesemps**

First_Name	Last_Name	Job_Title	Salary
Monica	Kletschkus	Sales Rep. IV	.
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau		27440

A blank represents a missing character value.

A period represents a missing numeric value.

- A value must exist for every variable in every observation.

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Date Values

SAS stores calendar dates as numeric values.

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

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.03 Quiz

What is the numeric value for today's date?

- Submit program **p103a02**.
- View the output to retrieve the current date as a SAS date value (that is, a numeric value referencing January 1, 1960).

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.03 Quiz – Correct Answer

What is the numeric value for today's date?

The answer depends on the current date.

Example: If the current date is February 27, 2012, the numeric value is 19050.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exercise

This exercise reinforces the concepts discussed previously.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 3: Accessing Data

3.1 Examining SAS Data Sets

3.2 Accessing SAS Libraries

Copyright © 2012, SAS Institute Inc. All rights reserved.

Objectives

- Explain the concept of a SAS library.
- State the difference between a temporary library and a permanent library.
- Assign a library reference name to a SAS library using a LIBNAME statement.
- Investigate a SAS library programmatically and interactively.
- Access a data set in a user-created permanent library.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Orion Star programmers need to access existing SAS data sets, so they need to understand how the data sets are stored in SAS.

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Libraries

SAS data sets are stored in *SAS libraries*. A SAS library is a collection of SAS files that are referenced and stored as a unit.

Each file is a member of the library.

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Libraries

You can think of a SAS library as a drawer in a filing cabinet and a SAS data set as one of the files in the drawer.

Copyright © 2012, SAS Institute Inc. All rights reserved.

How SAS Libraries Are Defined

When a SAS session starts, SAS automatically creates one temporary and at least one permanent SAS library that you can access. These libraries are open and ready to be used.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Assigning a Libref

Regardless of the operating system that you use, you refer to a SAS library by a logical name called a library reference name, or *libref*.

A libref is a shortcut to the library.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Temporary Library

Work is a temporary library where you can store and access SAS data sets for the duration of the SAS session. It is the default library.

SAS deletes the **work** library and its contents when the session terminates.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Permanent Libraries

Sashelp is a permanent library that contains sample SAS data sets you can access during your SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Permanent Libraries

Sasuser is a permanent library that you can use to store and access SAS data sets in any SAS session.

SAS data sets in permanent libraries are saved after your SAS session terminates.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Accessing SAS Data Sets

All SAS data sets have a two-level name that consists of the libref and the data set name, separated by a period.

libref.data-set-name

When a data set is in the temporary **work** library, you can use a one-level name (for example, **newsalesemps)**.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Orion Star programmers need to access and view SAS data sets that are stored in a permanent user-defined library.

Copyright © 2012, SAS Institute Inc. All rights reserved.

User-Defined Libraries

Users can create their own SAS libraries. A user-defined library

- is permanent. Data sets are stored until the user deletes them.
- is implemented within the operating environment's file system.
- is not automatically available in a SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

User-Defined Libraries

Operating Environment	A SAS library is…	Example
Microsoft Windows	A folder	s:\workshop
UNIX	A directory	~/workshop
z/OS (OS/390)	A sequential file	userid.workshop.sasdata

The user must assign a libref to the user-defined library to make it available in a SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Accessing a Permanent Library

Step 1 Identify the location of the library.

In this example, **s:\workshop**, a Microsoft Windows folder, is used as the SAS library.

Identify the location of ***your*** course data.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Accessing a Permanent Library

Step 2	Use a SAS LIBNAME statement to associate the libref with the physical location of the library.

Associate the libref **orion** with the Windows folder so that it is available to your SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

LIBNAME Statement

The SAS LIBNAME statement is a *global* SAS statement.

```
libname orion "s:\workshop";
```

LIBNAME *libref* "*SAS-library*" <*options*>**;**

- It is not required to be in a DATA step or PROC step.
- It does not require a RUN statement.
- It executes immediately.
- It remains in effect until changed or canceled, or until the session ends.

Use the location of ***your*** course data in your LIBNAME statement.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Partial SAS Log

```
47   libname orion "s:\workshop";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:        V9
      Physical Name: s:\workshop
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.04 Multiple Choice Poll

Which of the following correctly assigns the libref **myfiles** to a SAS library in the **c:\mysasfiles** folder?

a. libname orion myfiles "c:\mysasfiles";

b. libname myfiles "c:\mysasfiles";

c. libref orion myfiles "c:\mysasfiles";

d. libref myfiles "c:\mysasfiles";

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.04 Multiple Choice Poll – Correct Answer

Which of the following correctly assigns the libref **myfiles** to a SAS library in the **c:\mysasfiles** folder?

a. libname orion myfiles "c:\mysasfiles";

b. libname myfiles "c:\mysasfiles";

c. libref orion myfiles "c:\mysasfiles";

d. libref myfiles "c:\mysasfiles";

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing a Library

Step 3 You can browse a library interactively in a SAS or SAS Enterprise Guide session, or programmatically using the CONTENTS procedure.

PROC CONTENTS

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing a Library Programmatically

Use PROC CONTENTS with the _ALL_ keyword to generate a list of all SAS files in a library.

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

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

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

p103d03

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC CONTENTS Output

```
                    The CONTENTS Procedure

                           Directory

                    Libref          ORION
                    Engine          V9
                    Physical Name   S:\workshop
                    Filename        S:\workshop


                       Member      File
#   Name               Type        Size   Last Modified

1   CHARITIES          DATA        9216   23Aug12:15:58:39
2   CONSULTANTS        DATA        5120   23Aug12:15:58:39
3   COUNTRY            DATA       17408   13Oct10:19:04:39
    COUNTRY            INDEX      17408   13Oct10:19:04:39
4   CUSTOMER           DATA       33792   04Nov11:09:52:27
5   CUSTOMER_DIM       DATA       33792   04Nov11:09:52:27
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Accessing a Permanent Data Set

Step 4 After the libref is assigned, you can access SAS files in the library.

```
proc print data=orion.country;
run;
```

PROC PRINT Output

Obs	Country	Country_Name	Population	Country_ ID	Continent_ ID	Country_Former Name
1	AU	Australia	20,000,000	160	96	
2	CA	Canada	.	260	91	
3	DE	Germany	80,000,000	394	93	East/West Germany
4	IL	Israel	5,000,000	475	95	
5	TR	Turkey	70,000,000	905	95	
6	US	United States	280,000,000	926	91	
7	ZA	South Africa	43,000,000	801	94	

p103d04

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Partial SAS Log

```
25   proc print data=orion.country;
26   run;

NOTE: There were 7 observations read from the data set
ORION.COUNTRY.
```

The libref **orion** remains in effect until you change or cancel it, or until you end your SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing or Canceling a Libref

To change a libref, submit a LIBNAME statement with the same libref but a different path.

```
libname orion "c:\myfiles";
```

To cancel a libref, submit a LIBNAME statement with the CLEAR option.

```
libname orion clear;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing SAS Libraries: SAS Enterprise Guide

This demonstration illustrates defining and accessing a SAS library using SAS Enterprise Guide.

libname
p103d03

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.05 Poll

The library display in the Server List updates immediately when a libref is assigned or cleared using SAS Enterprise Guide.

- ❍ True
- ❍ False

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.05 Poll – Correct Answer

The library display in the Server List updates immediately when a libref is assigned or cleared using SAS Enterprise Guide.

- ❍ True
- ❍ False

The library display might or might not update immediately. Click Libraries ⇨ Refresh to update the list.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Browsing SAS Libraries: SAS Windowing Environment

This demonstration illustrates defining and accessing a SAS library using the SAS windowing environment.

libname
p103d03

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.06 Multiple Answer Poll

If you end your SAS or Enterprise Guide session and then need to access an **orion** data set again, which of the following must you do?

a. start a new SAS or Enterprise Guide session

b. use Windows Explorer to locate the files

c. create a new folder to be used as a SAS library

d. submit a LIBNAME statement to define a libref

e. use a two-part name to access the data set

Copyright © 2012, SAS Institute Inc. All rights reserved.

3.06 Multiple Answer Poll – Correct Answers

If you end your SAS or Enterprise Guide session and then need to access an **orion** data set again, which of the following must you do?

a. start a new SAS or Enterprise Guide session
b. use Windows Explorer to locate the files
c. create a new folder to be used as a SAS library
d. submit a LIBNAME statement to define a libref
e. use a two-part name to access the data set

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exercise

This exercise reinforces the concepts discussed previously.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter Review

Copyright © 2012, SAS Institute Inc. All rights reserved.

1. In which portion of a SAS data set are the following found?
   - the name of the data set
   - the type of the variable **Salary**
   - the creation date of the data set

   a. descriptor portion

   b. data portion

Copyright © 2012, SAS Institute Inc. All rights reserved.

1. In which portion of a SAS data set are the following found?
   - the name of the data set
   - the type of the variable **Salary**
   - the creation date of the data set

   a. descriptor portion

   b. data portion

Copyright © 2012, SAS Institute Inc. All rights reserved.

2. In this PROC CONTENTS output, what is the default length of the variable **Month**?

a. 2 bytes

b. 8 bytes

c. 16 or 17 bytes

d. 32,767 bytes

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	Month	Num	?

Copyright © 2012, SAS Institute Inc. All rights reserved.

2. In this PROC CONTENTS output, what is the default length of the variable **Month**?

a. 2 bytes

b. 8 bytes

c. 16 or 17 bytes

d. 32,767 bytes

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	Month	Num	?

Copyright © 2012, SAS Institute Inc. All rights reserved.

3. Which LIBNAME statement has the correct syntax?

   a. libname reports 's:\workshop';
   b. libname orion s:\workshop;
   c. libname 3456a 's:\workshop';

Copyright © 2012, SAS Institute Inc. All rights reserved.

3. Which LIBNAME statement has the correct syntax?

a. libname reports 's:\workshop';

b. libname orion s:\workshop;

c. libname 3456a 's:\workshop';

Copyright © 2012, SAS Institute Inc. All rights reserved.

4. Which PROC step successfully prints a list of all data sets in the **orion** library without printing descriptor portions for the individual data sets?

   a. proc contents data=orion.nods _all_;
      run;
   b. proc contents data=orion._all_ nods;
      run;
   c. proc print data=orion._all_ noobs;
      run;
   d. proc print data=orion._all_ nods;
      run;

Copyright © 2012, SAS Institute Inc. All rights reserved.

4. Which PROC step successfully prints a list of all data sets in the **orion** library without printing descriptor portions for the individual data sets?

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

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

c. proc print data=orion._all_ noobs;
run;

d. proc print data=orion._all_ nods;
run;

Copyright © 2012, SAS Institute Inc. All rights reserved.

5. In this data set, what type of variable is **Employee_ID**?

   a. character
   b. numeric
   c. temporary
   d. missing

Obs	Employee_ID	Last	Salary
1	.	Ralston	29250
2	120101	Lu	163040
3	120104	Billington	46230
4	120105	Povey	27110
5	120106	Hornsey	.

Copyright © 2012, SAS Institute Inc. All rights reserved.

5. In this data set, what type of variable is **Employee_ID**?

   a. character
   b. numeric
   c. temporary
   d. missing

Obs	Employee_ID	Last	Salary
1	.	Ralston	29250
2	120101	Lu	163040
3	120104	Billington	46230
4	120105	Povey	27110
5	120106	Hornsey	.

Copyright © 2012, SAS Institute Inc. All rights reserved.

6. What type of data set is the input data set in this PROC PRINT step?

```
proc print data=order_fact;
run;
```

a. temporary
b. permanent
c. There is not enough information to determine the type.

Copyright © 2012, SAS Institute Inc. All rights reserved.

6. What type of data set is the input data set in this PROC PRINT step?

```
proc print data=order_fact;
run;
```

a. temporary
b. permanent
c. There is not enough information to determine the type.

Copyright © 2012, SAS Institute Inc. All rights reserved.

7. A numeric variable can store numeric values with a maximum of eight digits.

   a. True
   b. False

Copyright © 2012, SAS Institute Inc. All rights reserved.

7. A numeric variable can store numeric values with a maximum of eight digits.

a. True
b. False

Copyright © 2012, SAS Institute Inc. All rights reserved.

8. Which of the following is not true of SAS date values?

   a. They are numeric.
   b. They can be positive or negative values.
   c. They represent the number of days between the day being stored and a base date.
   d. The base date is January 1, 1900.

Copyright © 2012, SAS Institute Inc. All rights reserved.

8. Which of the following is not true of SAS date values?

   a. They are numeric.
   b. They can be positive or negative values.
   c. They represent the number of days between the day being stored and a base date.
   d. The base date is January 1, 1900.

Copyright © 2012, SAS Institute Inc. All rights reserved.

9. Which statement about SAS libraries is true?

   a. You refer to a SAS library by a logical name called a libname.
   b. A SAS library is a collection of one or more SAS files that are referenced and stored as a unit.
   c. A single SAS library can contain files that are stored in different physical locations.
   d. At the end of each session, SAS deletes the contents of all SAS libraries.

Copyright © 2012, SAS Institute Inc. All rights reserved.

9. Which statement about SAS libraries is true?

   a. You refer to a SAS library by a logical name called a libname.
   b. A SAS library is a collection of one or more SAS files that are referenced and stored as a unit.
   c. A single SAS library can contain files that are stored in different physical locations.
   d. At the end of each session, SAS deletes the contents of all SAS libraries.

Copyright © 2012, SAS Institute Inc. All rights reserved.

10. Which of the following librefs is valid?

a. orionstar

b. orion/01

c. or_01

d. 1_or_a

Copyright © 2012, SAS Institute Inc. All rights reserved.

10. Which of the following librefs is valid?

a. orionstar

b. orion/01

c. or_01

d. 1_or_a

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 4: Producing Detail Reports

4.1 Subsetting Report Data

4.2 Sorting and Grouping Report Data

4.3 Enhancing Reports

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 4: Producing Detail Reports

4.1 Subsetting Report Data

4.2 Sorting and Grouping Report Data

4.3 Enhancing Reports

Copyright © 2012, SAS Institute Inc. All rights reserved.

Objectives

- Create a default PROC PRINT report.
- Select variables with a VAR statement.
- Calculate totals with a SUM statement.
- Select observations with a WHERE statement.
- Define a date constant.
- Identify observations with an ID statement.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Orion Star management wants a report that displays the names, salaries, and a salary total for all sales employees.

orion.sales

PROC PRINT

Obs	Last_Name	First_Name	Salary
1	xxxxxxx	xxxxxxxxxx	99999
2	xxxxxxx	xxxxxxxxxx	99999
3	xxxxxxx	xxxxxxxxxx	99999

			99999

Copyright © 2012, SAS Institute Inc. All rights reserved.

PRINT Procedure

By default, PROC PRINT displays all observations, all variables, and an Obs column on the left side.

```
proc print data=orion.sales;
run;
```

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_ Date
1	120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
3	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114
4	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-1984	6756
5	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	1732	9405

Statements and options can be added to the PRINT procedure to modify the default behavior.

p104d01

Copyright © 2012, SAS Institute Inc. All rights reserved.

VAR Statement

The VAR statement selects variables to include in the report and specifies their order.

```
proc print data=orion.sales;
   var Last_Name First_Name Salary;
run;
```

VAR *variable(s)*;

Partial PROC PRINT Output

Obs	Last_Name	First_ Name	Salary
1	Zhou	Tom	108255
2	Dawes	Wilson	87975
3	Elvish	Irenie	26600
4	Ngan	Christina	27475
5	Hotstone	Kimiko	26190

p104d01

Copyright © 2012, SAS Institute Inc. All rights reserved.

SUM Statement

The *SUM statement* calculates and displays report totals for the requested ***numeric*** variables.

```
proc print data=orion.sales;
   var Last_Name First_Name Salary;
   sum Salary;
run;
```

SUM *variable(s)*;

Partial PROC PRINT Output

```
                          First_
Obs     Last_Name         Name          Salary

  1     Zhou              Tom           108255
  2     Dawes             Wilson         87975
  3     Elvish            Irenie         26600
        ...
164     Capachietti       Renee          83505
165     Lansberry         Dennis         84260
                                       =======
                                       5141420
```

p104d01

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Partial SAS Log

```
84   proc print data=orion.sales;
85      var Last_Name First_Name Salary;
86      sum salary;
87   run;

NOTE: There were 165 observations read from the data set ORION.SALES.
```

- The order of statements in a SAS procedure is usually not important.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Orion Star management wants a report that displays the names and salaries of the sales employees earning less than $25,500. Suppress the Obs column.

Copyright © 2012, SAS Institute Inc. All rights reserved.

WHERE Statement

The *WHERE statement* selects observations that meet the criteria specified in the WHERE expression.

```
proc print data=orion.sales;
   var Last_Name First_Name Salary;
   where Salary<25500;
run;
```

WHERE *WHERE-expression*;

p104d02

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Only 7 of the 165 observations from **orion.sales** were selected by the WHERE statement.

```
295  proc print data=orion.sales;
296     var Last_Name First_Name Salary;
297     where Salary<25500;
298  run;

NOTE: There were 7 observations read from the data set ORION.SALES.
      WHERE Salary<25500;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

PROC PRINT Output

Obs	Last_ Name	First_ Name	Salary
49	Tilley	Kimiko	25185
50	Barcoe	Selina	25275
85	Anstey	David	25285
104	Voron	Tachaun	25125
111	Polky	Asishana	25110
131	Ould	Tulsidas	22710
148	Buckner	Burnetta	25390

original observation numbers

Copyright © 2012, SAS Institute Inc. All rights reserved.

Suppressing the Obs Column

Use the NOOBS option in the PROC PRINT statement to suppress the Obs column.

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Salary;
   where Salary<25500;
run;
```

PROC PRINT DATA=*SAS-data-set* **NOOBS;**

PROC PRINT Output

Last_ Name	First_ Name	Salary
Tilley	Kimiko	25185
Barcoe	Selina	25275
Anstey	David	25285
Voron	Tachaun	25125
Polky	Asishana	25110
Ould	Tulsidas	22710
Buckner	Burnetta	25390

p104d02

Copyright © 2012, SAS Institute Inc. All rights reserved.

WHERE Statement

The WHERE expression defines the condition (or conditions) for selecting observations.

WHERE *WHERE-expression*;

Operands

- character constants
- numeric constants
- date constants
- character variables
- numeric variables

Operators

- symbols that represent a comparison, calculation, or logical operation

- SAS functions
- special WHERE operators

Copyright © 2012, SAS Institute Inc. All rights reserved.

Operands

Constants are fixed values.

- Character values are enclosed in quotation marks and are case sensitive.
- Numeric values do not use quotation marks or special characters.

Variables must exist in the input data set.

```
where Gender='M';
```

variable constant

```
where Salary>50000;
```

variable constant

Copyright © 2012, SAS Institute Inc. All rights reserved.

SAS Date Constant

A *SAS date constant* is a date written in the following form: **'ddmmm<yy>yy'd**

Examples
'01JAN2000'd
'31Dec11'D
'1jan04'd
'06NOV2000'D

SAS automatically converts a date constant to a SAS date value.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Comparison Operators

Comparison operators compare a variable with a value or with another variable.

Symbol	Mnemonic	Definition
=	EQ	Equal to
^= ¬= ~=	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal
<=	LE	Less than or equal
	IN	Equal to one of a list

Copyright © 2012, SAS Institute Inc. All rights reserved.

Comparison Operators

Examples
`where Gender eq ' ';`
`where Salary ne .;`
`where Salary>=50000;`
`where Hire_Date<'01Jan2000'd;`
`where Country in ('AU','US');`
`where Country in ('AU' 'US');`
`where Order_Type in (1,2,3);`

The value list in the IN operator must be enclosed in parentheses and separated by either commas or blanks. Character values must be enclosed in quotation marks.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Setup for the Poll

Program **p104a01** contains two WHERE statements. Open and submit the program.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.01 Multiple Choice Poll

Which of the following is true?

a. The program executes, applying both WHERE conditions successfully.

b. The program fails and an error message is written to the log.

c. The program executes, but only the first WHERE condition is applied.

d. The program executes, but only the second WHERE condition is applied.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.01 Multiple Choice Poll – Correct Answer

Which of the following is true?

a. The program executes, applying both WHERE conditions successfully.

b. The program fails and an error message is written to the log.

c. The program executes, but only the first WHERE condition is applied.

d. The program executes, but only the second WHERE condition is applied.

```
182  proc print data=orion.sales;
183     where Country='AU';
184     where Salary<30000;
NOTE: WHERE clause has been replaced.
185  run;

NOTE: There were 134 observations read from the data set ORION.SALES.
      WHERE Salary<30000;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Logical Operators

Logical operators combine or modify WHERE expressions.

```
proc print data=orion.sales;
   where Country='AU' and
         Salary<30000;
run;
```

WHERE *WHERE-expression-1* AND | OR *WHERE-expression-n*;

p104d03

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Partial SAS Log

```
67   proc print data=orion.sales;
68      where Country='AU' and
69            Salary<30000;
70   run;

NOTE: There were 51 observations read from the data set ORION.SALES.
WHERE (Country='AU') and (Salary<30000);
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Logical Operator Priority

The operators can be written as symbols or mnemonics, and parentheses can be added to modify the order of evaluation.

Symbol	Mnemonic	Priority
^ ¬ ~	NOT	I
&	AND	II
|	OR	III

The NOT operator modifies a condition by finding the complement of the specified criteria.

```
where City not in ('London','Rome','Paris');
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Logical Operators

Examples
`where Country ne 'AU' and Salary>=50000;`
`where Gender eq 'M' or Salary ge 50000;`
`where Country='AU' or Country='US';`
`where Country in ('AU','US');`
`where Country not in ('AU','US');`

equivalent expressions

You should use only one WHERE statement in a step.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.02 Quiz

Which WHERE statement correctly subsets the numeric values for May, June, or July and missing character names?

a.
```
where Month in (5-7)
      and Names=.;
```

b.
```
where Month in (5,6,7)
      and Names=' ';
```

c.
```
where Month in ('5','6','7')
      and Names='.';
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.02 Quiz – Correct Answer

Which WHERE statement correctly subsets the numeric values for May, June, or July and missing character names?

a.
```
where Month in (5-7)
      and Names=.;
```

b.
```
where Month in (5,6,7)
      and Names=' ';
```

c.
```
where Month in ('5','6','7')
      and Names='.';
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Orion Star management wants a report that lists only the Australian sales representatives.

orion.sales

Last_Name	First_ Name	Country	Job_Title
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxxx

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exploring the Data

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Country
       Job_Title;
run;
```

Partial PROC PRINT Output

```
Plested        Billy       AU      Sales Rep. II
Wills          Matsuoka    AU      Sales Rep. III
George         Vino        AU      Sales Rep. II
Body           Meera       AU      Sales Rep. III
Highpoint      Harry       US      Chief Sales Officer
Magolan        Julienne    US      Sales Rep. II
Desanctis      Scott       US      Sales Rep. IV
Ridley         Cherda      US      Sales Rep. IV
```

p104d04

Copyright © 2012, SAS Institute Inc. All rights reserved.

Subsetting in a PROC PRINT Step

Include a WHERE statement to subset by **Country** and **Job_Title**.

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Country
       Job_Title;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

CONTAINS is a special WHERE operator.

p104d04

Copyright © 2012, SAS Institute Inc. All rights reserved.

CONTAINS Operator

The *CONTAINS operator* selects observations that include the specified substring.

Equivalent Statements
`where Job_Title contains 'Rep';`
`where Job_Title ? 'Rep';`

- ? can be used instead of the mnemonic.
- The position of the substring within the variable's values is not important.
- Comparisons made with the CONTAINS operator are case sensitive.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

Last_Name	First_ Name	Country	Job_Title
Elvish	Irenie	AU	Sales Rep. II
Ngan	Christina	AU	Sales Rep. II
Hotstone	Kimiko	AU	Sales Rep. I
Daymond	Lucian	AU	Sales Rep. I
Hofmeister	Fong	AU	Sales Rep. IV

Copyright © 2012, SAS Institute Inc. All rights reserved.

Special WHERE Operators

Special WHERE operators are operators that can be used only in WHERE expressions.

Operator	Definition	Char	Num
CONTAINS	Includes a substring	x	
BETWEEN-AND	An inclusive range	x	x
WHERE SAME AND	Augment a WHERE expression	x	x
IS NULL	A missing value	x	x
IS MISSING	A missing value	x	x
LIKE	Matches a pattern	x	

Copyright © 2012, SAS Institute Inc. All rights reserved.

BETWEEN-AND Operator

The *BETWEEN-AND operator* selects observations in which the value of a variable falls within an inclusive range of values.

Examples
`where salary between 50000 and 100000;`
`where salary not between 50000 and 100000;`
`where Last_Name between 'A' and 'L';`
`where Last_Name between 'Baker' and 'Gomez';`

Copyright © 2012, SAS Institute Inc. All rights reserved.

BETWEEN-AND Operator

Equivalent Statements
`where salary between 50000 and 100000;`
`where salary>=50000 and salary<=100000;`
`where 50000<=salary<=100000;`

Copyright © 2012, SAS Institute Inc. All rights reserved.

WHERE SAME AND Operator

Use the *WHERE SAME AND operator* to add more conditions to an existing WHERE expression.

```
proc print data=orion.sales;
   where Country='AU' and Salary <30000;
   where same and Gender='F';
   var First_Name Last_Name Gender
       Salary Country;
run;
```

The WHERE SAME AND condition ***augments*** the original condition.

p104d05

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

Partial SAS Log

```
22   proc print data=orion.sales;
23       where Country='AU'  and Salary<30000;
24       where also Gender='F';
NOTE: WHERE clause has been augmented.
25       var First_Name Last_Name Gender Salary Country;
26   run;

NOTE: There were 23 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and (Gender='F') and (Salary<30000);
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Gender	Salary	Country
3	Irenie	Elvish	F	26600	AU
4	Christina	Ngan	F	27475	AU
5	Kimiko	Hotstone	F	26190	AU
9	Sharryn	Clarkson	F	28100	AU
14	Fancine	Kaiser	F	28525	AU
15	Petrea	Soltau	F	27440	AU
19	Marina	Iyengar	F	29715	AU
20	Shani	Duckett	F	25795	AU
21	Fang	Wilson	F	26810	AU
23	Amanda	Liebman	F	27465	AU

Country='AU'
Gender='F'
Salary<30000

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.03 Quiz

1. Open **p104a01b**.
2. Change WHERE SAME AND to WHERE ALSO.
3. Submit the program and view the log.

What message is written to the log?

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.03 Quiz – Correct Answer

WHERE ALSO results in the same message:

WHERE clause has been augmented.

```
27   proc print data=orion.sales;
28       where Country='AU'  and Salary<30000;
29       where also Gender='F';
NOTE: WHERE clause has been augmented.
30       var First_Name Last_Name Gender Salary Country;
31   run;

NOTE: There were 23 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and (Gender='F') and (Salary<30000);
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

IS NULL Operator

The *IS NULL operator* selects observations in which a variable has a missing value.

Examples
`where Employee_ID is null;`
`where Employee_ID is not null;`

IS NULL can be used for both character and numeric variables, and is equivalent to the following statements:

```
where employee_ID=' ';
```

```
where employee_ID=.;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

IS MISSING Operator

The *IS MISSING operator* selects observations in which a variable has a missing value.

Examples
`where Employee_ID is missing;`
`where Employee_ID is not missing;`

IS MISSING can be used for both character and numeric variables, and is equivalent to the following statements:

```
where employee_ID=' ';
```

```
where employee_ID=.;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

LIKE Operator

The *LIKE operator* selects observations by comparing character values to specified patterns. Two special characters are used to define a pattern:

- A percent sign (%) specifies that ***any number*** of characters can occupy that position.
- An underscore (_) specifies that ***exactly one*** character can occupy that position.

Examples
`where Name like '%N';`
`where Name like 'T_m';`
`where Name like 'T_m%';`

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.04 Quiz

Which WHERE statement returns all the observations that have a first name starting with the letter M for the given values?

a. `where Name like '_, M_';`

b. `where Name like '%, M%';`

c. `where Name like '_, M%';`

d. `where Name like '%, M_';`

Name
Elvish, Irenie
Ngan, Christina
Hotstone, Kimiko
Daymond, Lucian
Hofmeister, Fong
Denny, Satyakam
Clarkson, Sharryn
Kletschkus, Monica

last name, first name

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.04 Quiz – Correct Answer

Which WHERE statement returns all the observations that have a first name starting with the letter M for the given values?

a. `where Name like '_, M_';`

b. `where Name like '%, M%';`

c. `where Name like '_, M%';`

d. `where Name like '%, M_';`

Name
Elvish, Irenie
Ngan, Christina
Hotstone, Kimiko
Daymond, Lucian
Hofmeister, Fong
Denny, Satyakam
Clarkson, Sharryn
Kletschkus, Monica

last name, first name

Copyright © 2012, SAS Institute Inc. All rights reserved.

Questions ???

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

The Sales Manager wants a report that includes only customers who are 21 years old.

orion.customer_dim

Obs	Customer_ID	Customer_Name	Customer_ Gender	Customer_ Country	Customer_Group	Customer_ Age_Group	Customer_ Type
1	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX
2	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX
3	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX

Copyright © 2012, SAS Institute Inc. All rights reserved.

Subsetting the Data Set

Display the required rows and variables.

```
proc print data=orion.customer_dim;
   where Customer_Age=21;
   var Customer_ID Customer_Name
       Customer_Gender Customer_Country
       Customer_Group Customer_Age_Group
       Customer_Type;
run;
```

- The subsetting variable does not need to be included in the report.

p104d06

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

In this output, two lines are used for each observation.

PROC PRINT Output

```
                                          Customer_    Customer_
Obs      Customer_ID    Customer_Name       Gender      Country          Customer_Group

 37               79    Najma Hicks            F          US         Orion Club members
 58            11171    Bill Cuddy             M          CA         Orion Club Gold members
 66            46966    Lauren Krasowski       F          CA         Orion Club members
 ...
 76            70210    Alex Santinello        M          CA         Orion Club members

         Customer_
Obs      Age_Group                  Customer_Type

 37     15-30 years    Orion  Club members medium activity
 58     15-30 years    Orion Club Gold members low activity
 66     15-30 years    Orion  Club members high activity
 ...
 76     15-30 years    Orion  Club members medium activity
```

The Obs column helps identify observations in a report that span multiple lines.

Copyright © 2012, SAS Institute Inc. All rights reserved.

ID Statement

The *ID statement* specifies the variable or variables to print at the beginning of each row instead of an observation number.

```
proc print data=orion.customer_dim;
   where Customer_Age=21;
   id Customer_ID;
   var Customer_Name Customer_Gender
       Customer_Country Customer_Group
       Customer_Age_Group Customer_Type;

run;
```

Choose ID variables that uniquely identify observations.

p104d07

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

PROC PRINT Output

Customer_ID	Customer_Name	Customer_ Gender	Customer_ Country	Customer_Group
79	Najma Hicks	F	US	Orion Club members
11171	Bill Cuddy	M	CA	Orion Club Gold members
46966	Lauren Krasowski	F	CA	Orion Club members
70079	Lera Knott	F	CA	Orion Club members
70187	Soberina Berent	F	CA	Orion Club members
70210	Alex Santinello	M	CA	Orion Club members

Customer_ID	Customer_ Age_Group	Customer_Type
79	15-30 years	Orion Club members medium activity
11171	15-30 years	Orion Club Gold members low activity
46966	15-30 years	Orion Club members high activity
70079	15-30 years	Orion Club members medium activity
70187	15-30 years	Orion Club members medium activity
70210	15-30 years	Orion Club members medium activity

Copyright © 2012, SAS Institute Inc. All rights reserved.

Questions ???

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exercise

This exercise reinforces the concepts discussed previously.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 4: Producing Detail Reports

4.1 Subsetting Report Data

4.2 Sorting and Grouping Report Data

4.3 Enhancing Reports

Copyright © 2012, SAS Institute Inc. All rights reserved.

Objectives

- Sort the observations in a SAS data set based on the values of one or more variables.
- Display the sorted observations.
- Display a data set with report totals and subtotals for each BY group.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Display observations from **orion.sales** in ascending order by the variable **Salary**.

Employee_ID	Last_Name	Salary
999999	xxxxxxxxxx	99999
999999	xxxxxxxxxx	99999
999999	xxxxxxxxxx	99999

Copyright © 2012, SAS Institute Inc. All rights reserved.

Creating a Sorted Report

Step 1	Use the SORT procedure to create a new data set, **work.sales**, ordering the observations by the value of **Salary**.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Creating a Sorted Report

Step 2	Use the PRINT procedure to display the sorted data set, **work.sales**.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Step 1: SORT Procedure

The *SORT procedure* rearranges the observations in the input data set based on the values of the variable or variables listed in the BY statement.

```
proc sort data=orion.sales
          out=work.sales;
   by Salary;
run;
```

PROC SORT DATA=*input-SAS-data-set*
<**OUT=***output-SAS-data-set*>**;**
BY <**DESCENDING**> *variable*<*s*>**;**
RUN;

The BY statement in a PROC SORT step specifies the sort variables and, optionally, the sort order.

p104d08

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Log

The SORT procedure does not produce a report. Check the log for errors or warnings.

Partial SAS Log

```
34   proc sort data=orion.sales
35             out=work.sales;
36      by Salary;
37   run;

NOTE: There were 165 observations read from the data set
ORION.SALES.
NOTE: The data set WORK.SALES has 165 observations and 9
variables.
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Step 2: Viewing the Output

```
proc print data=work.sales noobs;
   var Employee_ID Last_Name Salary;
run;
```

Partial PROC PRINT Output

Employee_ID	Last_Name	Salary
121084	Ould	22710
121064	Polky	25110
121057	Voron	25125
...		
121143	Favaron	95090
120102	Zhou	108255
120261	Highpoint	243190

p104d08

Copyright © 2012, SAS Institute Inc. All rights reserved.

SORT Procedure

The SORT procedure

- replaces the original data set or creates a new one
- can sort on multiple variables
- sorts in ascending (default) or descending order
- does not generate printed output.

The input data set is overwritten unless the OUT= option is used to specify an output data set.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.05 Quiz

Which step sorts the observations in a SAS data set and overwrites the same data set?

a.
```
proc sort data=work.EmpsAU
          out=work.sorted;
   by First;
run;
```

b.
```
proc sort data=orion.EmpsAU
          out=EmpsAU;
   by First;
run;
```

c.
```
proc sort data=work.EmpsAU;
   by First;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.05 Quiz – Correct Answer

Which step sorts the observations in a SAS data set and overwrites the same data set?

a.
```
proc sort data=work.EmpsAU
          out=work.sorted;
   by First;
run;
```

b.
```
proc sort data=orion.EmpsAU
          out=EmpsAU;
   by First;
run;
```

c.
```
proc sort data=work.EmpsAU;
   by First;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Produce a report that lists sales employees grouped by **Country**, in descending **Salary** order within country.

```
-------------------------------------Country=AU-----------------------------------------

                 First_   Last_                                    Birth_   Hire_
  Employee_ID     Name     Name    Gender   Salary    Job_Title     Date     Date

     9999         xxxx     xxxxx     x      99999     xxxxxx        9999     9999
     9999         xxxx     xxxxx     x      99999     xxxxxx        9999     9999


-------------------------------------Country=US-----------------------------------------

                 First_   Last_                                    Birth_   Hire_
  Employee_ID     Name     Name    Gender   Salary    Job_Title     Date     Date

     9999         xxxx     xxxxx     x      99999     xxxxxx        9999     9999
     9999         xxxx     xxxxx     x      99999     xxxxxx        9999     9999
     9999         xxxx     xxxxx     x      99999     xxxxxx        9999     9999
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Creating a Grouped Report

Step 1	Use the SORT procedure to group data in a data set. This scenario requires two variables to be sorted: ■ **Country** ■ descending **Salary** within **Country**
Step 2	Use a BY statement in PROC PRINT to display the sorted observations grouped by **Country**.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Step 1: Sort the Data

Sort the data set to group the observations.

```
proc sort data=orion.sales
          out=work.sales;
   by Country descending Salary;
run;
```

BY <**DESCENDING**> *variable(s)*;

p104d09

Copyright © 2012, SAS Institute Inc. All rights reserved.

Specifying Sort Order

The *DESCENDING option* reverses the sort order for the variable that immediately follows it. The observations are sorted from the largest value to the smallest value.

Examples:

```
by descending Last First;
```

```
by Last descending First;
```

```
by descending Last descending First;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Specifying Multiple BY Variables

- PROC SORT first arranges the data set by the values of the first BY variable.

- PROC SORT then arranges any observations that have the same value of the first BY variable by the values of the second BY variable.

- This sorting continues for every specified BY variable.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Step 2: Specify Report Groupings

The BY statement in a PROC PRINT step specifies the variable or variables to use to form *BY groups*.

```
proc print data=work.sales noobs;
   by Country;
run;
```

BY <**DESCENDING**> *variable(s)*>**;**

- The variables in the BY statement are called *BY variables*.
- The observations in the data set ***must*** be in order by the BY variable (or variables).

p104d09

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

```
---------------------------- Country=AU -------------------------

                                                          Hire_
Employee_ID   First_Name   Last_Name  Gender  Salary        Date

     120102   Tom          Zhou          M    108255       12205
     120103   Wilson       Dawes         M     87975  ...    6575
     120168   Selina       Barcoe        M     36605       18567

---------------------------- Country=US -------------------------

                                                          Hire_
Employee_ID   First_Name   Last_Name   Gender  Salary        Date

     120261   Harry        Highpoint     M     243190      11535
     121143   Louis        Favaron       M      95090  ...  15157
     121064   Asishana     Polky         M      84260      13027
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.06 Quiz

Open and submit **p104a02**. View the log.

Why did the program fail?

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.06 Quiz – Correct Answer

Open and submit **p104a02**. View the log.

Why did the program fail?

The input data set was not sorted by Gender.

```
188  proc sort data=orion.sales
189            out=work.sorted;
190       by Country Gender;
191  run;

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.SORTED has 165 observations and 9 variables.

192
193  proc print data=work.sorted;
194       by Gender;
195  run;

ERROR: Data set WORK.SORTED is not sorted in ascending sequence. The current
       BY group has Gender = M and the next BY group has Gender = F.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 64 observations read from the data set WORK.SORTED.
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Modify the previous report to display selected variables, the salary subtotal for each country, and the salary grand total.

```
-------------------- Country=AU ------------------

First_Name   Last_Name     Gender    Salary

XXXX         XXXXXXX        X        99999
XXXX         XXXXXXX        X        99999
--------                             ---------
Country                              999999

-------------------- Country=US ------------------

First_Name   Last_Name     Gender    Salary

XXXXXXX      XXXXXXX        X        99999
XXXXXXX      XXXXXXX        X        99999
--------                             ---------
Country                               999999
                                     =========
                                      9999999
```

subtotals

grand total

Copyright © 2012, SAS Institute Inc. All rights reserved.

Generating Subtotals

Use a BY statement and a SUM statement in a PROC PRINT step.

```
proc sort data=orion.sales
          out=work.sales;
   by Country descending Salary;
run;

proc print data=work.sales noobs;
   by Country;
   sum Salary;
   var First_Name Last_Name Gender Salary;
run;
```

p104d10

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

```
----------------- Country=AU -----------------

First_Name     Last_Name        Gender     Salary

Tom            Zhou               M        108255
Wilson         Dawes              M         87975
Daniel         Pilgrim            M         36605
...
Kimiko         Tilley             F         25185
--------                                   -------
Country                                    1900015

------------------ Country=US ------------------

First_Name     Last_Name          Gender     Salary

Harry          Highpoint            M        243190
Louis          Favaron              M         95090
Dennis         Lansberry            M         84260
...
Tulsidas       Ould                 M         22710
------------                                -------
Country                                     3241405
                                            =======
                                            5141420
```

subtotal for AU

subtotal for US

grand total

Copyright © 2012, SAS Institute Inc. All rights reserved.

Setup for the Poll

Modify the previous report to display only employees earning less than 25,500. Which WHERE statement (or statements) will result in the most efficient processing?

```
proc sort data=orion.sales
          out=work.sales;
   /* where Salary<25500;*/
   by Country descending Salary;
run;
proc print data=work.sales noobs;
   by Country;
   sum Salary;
   /* where Salary<25500;*/
   var First_Name Last_Name Gender Salary;
run;
```

p104a03

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.07 Multiple Choice Poll

Which WHERE statement (or statements) will result in the most efficient processing?

a. The WHERE statement in the PROC SORT step.

b. The WHERE statement in the PROC PRINT step.

c. Both WHERE statements are needed.

d. The WHERE statements are equally efficient.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.07 Multiple Choice Poll – Correct Answer

Which WHERE statement (statements) will result in the most efficient processing?

a. The WHERE statement in the PROC SORT step.
b. The WHERE statement in the PROC PRINT step.
c. Both WHERE statements are needed.
d. The WHERE statements are equally efficient.

Subsetting in the PROC SORT is more efficient. It selects and sorts only the required observations.

Be sure to use the OUT= option when subsetting in a PROC SORT or you will overwrite your original data set with the subset.

p104a03s

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exercise

This exercise reinforces the concepts discussed previously.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter 4: Producing Detail Reports

4.1 Subsetting Report Data

4.2 Sorting and Grouping Report Data

4.3 Enhancing Reports

Copyright © 2012, SAS Institute Inc. All rights reserved.

Objectives

- Include titles and footnotes in a report.
- Define descriptive column headings using the LABEL statement.
- Control the use of column headings with the LABEL and SPLIT= options.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Business Scenario

Enhance the payroll report by adding titles, footnotes, and descriptive column headings.

Obs	Employee_ID	Last_ Name	Salary
1	9999	xxxxxxxxxx	99999
2	9999	xxxxxxxxxx	99999
3	99		

Orion Star Sales Staff
Salary Report

Obs	Employee ID	Last Name	Annual Salary
1	9999	xxxxxxxxxx	99999
2	9999	xxxxxxxxxx	99999
3	9999	xxxxxxxxxx	99999

Confidential

Copyright © 2012, SAS Institute Inc. All rights reserved.

Displaying Titles and Footnotes

Use TITLE and FOOTNOTE statements to enhance the report.

TITLE*n* '*text* ';

FOOTNOTE*n* '*text* ';

```
title1 'Orion Star Sales Staff';
title2 'Salary Report';

footnote1 'Confidential';

proc print data=orion.sales;
   var Employee_ID Last_Name Salary;
run;

title;
footnote;
```

p104d11

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

```
                Orion Star Sales Staff
                    Salary Report

Obs      Employee_ID     Last_Name        Salary

  1          120102     Zhou             108255
  2          120103     Dawes             87975
  3          120121     Elvish            26600
...
164          121144     Capachietti       83505
165          121145     Lansberry         84260


                    Confidential
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

TITLE Statement

The global *TITLE statement* specifies title lines for SAS output.

TITLE*n* '*text* '**;**

- Titles appear at the top of the page.
- The default title is **The SAS System**.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **TITLE** is equivalent to **TITLE1**.
- Titles remain in effect until they are changed or canceled, or you end your SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

FOOTNOTE Statement

The global *FOOTNOTE statement* specifies footnote lines for SAS output.

FOOTNOTE*n* '*text* '**;**

- Footnotes appear at the bottom of the page.
- No footnote is printed unless one is specified.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **FOOTNOTE** is equivalent to **FOOTNOTE1**.
- Footnotes remain in effect until they are changed or canceled, or you end your SAS session.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing Titles and Footnotes

To change a title line, submit a TITLE statement with the same number but different text.

- replaces a previous title with the same number
- cancels all titles with higher numbers

```
title1 'ABC Company';
title2 'Sales Division';
title3 'Salary Report';
```

```
title1 'Salary Report';
```

This statement changes title 1 and cancels titles 2 and 3.

Footnotes are changed the same way.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Canceling All Titles and Footnotes

- The null TITLE statement cancels all titles.

```
title;
```

- The null FOOTNOTE statement cancels all footnotes.

```
footnote;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`	

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`	

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`	

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`	

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`	

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`	

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line
	
	The Third Line

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line
	
	The Third Line

`title;`
`proc print data=orion.sales;`
`run;`	

...

Copyright © 2012, SAS Institute Inc. All rights reserved.

Changing and Canceling Titles and Footnotes

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Second Line

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`
	The First Line
	The Next Line

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`
	The Top Line
	
	The Third Line

`title;`
`proc print data=orion.sales;`
`run;`	

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.08 Quiz

Which footnote or footnotes appear in the second procedure output?

a. Non Sales Employees

b. Orion Star
Non Sales Employees

c. Non Sales Employees
Confidential

d. Orion Star
Non Sales Employees
Confidential

```
footnote1 'Orion Star';
footnote2 'Sales Employees';
footnote3 'Confidential';
proc print data=orion.sales;
run;

footnote2 'Non Sales Employees';
proc print data=orion.nonsales;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

4.08 Quiz – Correct Answer

Which footnote or footnotes appear in the second procedure output?

a. Non Sales Employees

b. Orion Star
Non Sales Employees

c. Non Sales Employees
Confidential

d. Orion Star
Non Sales Employees
Confidential

```
footnote1 'Orion Star';
footnote2 'Sales Employees';
footnote3 'Confidential';
proc print data=orion.sales;
run;

footnote2 'Non Sales Employees';
proc print data=orion.nonsales;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Idea Exchange

Which of the following programs do you prefer and why?

a.
```
title 'Orion Star Employees';
proc print data=orion.staff;
   where Gender='F';
   var Employee_ID Salary;
run;
```

b.
```
title 'Orion Star Female Employees';
proc print data=orion.staff;
   where Gender='F';
   var Employee_ID Salary;
run;
```

c.
```
title 'Orion Star Employees';
proc print data=orion.staff;
   where Gender='F';
   var Employee_ID Gender Salary;
run;
```

d.
```
title 'Orion Star Female Employees';
proc print data=orion.staff;
   where Gender='F';
   var Employee_ID Gender Salary;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

LABEL Statement and Option

Use a LABEL statement and the LABEL option to display descriptive column headings instead of variable names.

```
title1 'Orion Star Sales Staff';
title2 'Salary Report';
footnote1 'Confidential';

proc print data=orion.sales label;
   var Employee_ID Last_Name Salary;
   label Employee_ID='Sales ID'
         Last_Name='Last Name'
         Salary='Annual Salary';
run;

title;
footnote;
```

LABEL *variable-1*=**'***label***'**
...
variable-n=**'***label***';**

p104d12

Copyright © 2012, SAS Institute Inc. All rights reserved.

LABEL Statement

The LABEL statement assigns descriptive labels to variables.

- A label can be up to 256 characters and include any characters, including blanks.
- Labels are used automatically by many procedures.
- The PRINT procedure uses labels only when the LABEL or SPLIT= option is specified.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Orion Star Sales Staff
Salary Report

Obs	Sales ID	Last Name	Annual Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
...			
164	121144	Capachietti	83505
165	121145	Lansberry	84260

Confidential

Copyright © 2012, SAS Institute Inc. All rights reserved.

SPLIT= Option

The SPLIT= option in PROC PRINT specifies a split character to control line breaks in column headings.

```
proc print data=orion.sales split='*';
   var Employee_ID Last_Name Salary;
   label Employee_ID='Sales ID'
         Last_Name='Last*Name'
         Salary='Annual*Salary';
run;
```

SPLIT='*split-character***'**

The SPLIT= option can be used instead of the LABEL option in a PROC PRINT step.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Viewing the Output

Partial PROC PRINT Output

Orion Star Sales Staff
Salary Report

Obs	Sales ID	Last Name	Annual Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
...			
164	121144	Capachietti	83505
165	121145	Lansberry	84260

Confidential

Copyright © 2012, SAS Institute Inc. All rights reserved.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Exercise

This exercise reinforces the concepts discussed previously.

Copyright © 2012, SAS Institute Inc. All rights reserved.

Chapter Review

Copyright © 2012, SAS Institute Inc. All rights reserved.

1. Which observation or observations will be selected by the following WHERE statement?

```
where Job_Title contains 'I';
```

Obs	Last_Name	First_Name	Country	Job_Title
1	Wu	Christine	AU	Sales Rep. I
2	Stone	Kimiko	AU	Sales Manager
3	Hofmann	Fred	AU	Insurance Sales

a. observation 1
b. observation 2
c. observation 3
d. observations 1 and 3
e. all observations

Copyright © 2012, SAS Institute Inc. All rights reserved.

1. Which observation or observations will be selected by the following WHERE statement?

```
where Job_Title contains 'I';
```

Obs	Last_Name	First_Name	Country	Job_Title
1	Wu	Christine	AU	Sales Rep. I
2	Stone	Kimiko	AU	Sales Manager
3	Hofmann	Fred	AU	Insurance Sales

a. observation 1
b. observation 2
c. observation 3
d. observations 1 and 3
e. all observations

Copyright © 2012, SAS Institute Inc. All rights reserved.

2. Which statement in a PROC SORT step prepares data to be displayed as shown in this output?

Postal_Code	Employee_ID
92129	121074
92129	121001
92128	121128
92128	120755
92128	120730

a. `by Postal_Code Employee_ID;`

b. `by descending Postal_Code Employee_ID;`

c. `by Postal_Code descending Employee_ID;`

d. `by descending Postal_Code`
`descending Employee_ID;`

Copyright © 2012, SAS Institute Inc. All rights reserved.

2. Which statement in a PROC SORT step prepares data to be displayed as shown in this output?

Postal_Code	Employee_ID
92129	121074
92129	121001
92128	121128
92128	120755
92128	120730

a. `by Postal_Code Employee_ID;`

b. `by descending Postal_Code Employee_ID;`

c. `by Postal_Code descending Employee_ID;`

d. `by descending Postal_Code descending Employee_ID;`

Copyright © 2012, SAS Institute Inc. All rights reserved.

3. Which statement about this PROC SORT step is true?

```
proc sort data=orion.staff;
          out=work.staff;
   by descending Salary
      Manager_ID;
run;
```

a. The sorted data set overwrites the input data set.
b. The observations are sorted by **Salary** in descending order, and then by **Manager_ID** in descending order.
c. A semicolon should not appear after the input data set name.
d. The sorted data set contains only the variables specified in the BY statement.

Copyright © 2012, SAS Institute Inc. All rights reserved.

3. Which statement about this PROC SORT step is true?

```
proc sort data=orion.staff;
          out=work.staff;
   by descending Salary
      Manager_ID;
run;
```

a. The sorted data set overwrites the input data set.
b. The observations are sorted by **Salary** in descending order, and then by **Manager_ID** in descending order.
c. A semicolon should not appear after the input data set name.
d. The sorted data set contains only the variables specified in the BY statement.

Copyright © 2012, SAS Institute Inc. All rights reserved.

4. Which of the following statements selects from a data set only those observations for which the value of the variable **Style** is *RANCH*, *SPLIT*, or *TWOSTORY*?

a. `where Style='RANCH' or 'SPLIT' or 'TWOSTORY';`

b. `where Style in 'RANCH' or 'SPLIT' or 'TWOSTORY';`

c. `where Style in (RANCH, SPLIT, TWOSTORY);`

d. `where Style in ('RANCH','SPLIT','TWOSTORY');`

Copyright © 2012, SAS Institute Inc. All rights reserved.

4. Which of the following statements selects from a data set only those observations for which the value of the variable **Style** is *RANCH*, *SPLIT*, or *TWOSTORY*?

a. `where Style='RANCH' or 'SPLIT' or 'TWOSTORY';`

b. `where Style in 'RANCH' or 'SPLIT' or 'TWOSTORY';`

c. `where Style in (RANCH, SPLIT, TWOSTORY);`

d. `where Style in ('RANCH','SPLIT','TWOSTORY');`

Copyright © 2012, SAS Institute Inc. All rights reserved.

5. Which of the following statements selects rows in which **Amount** is less than or equal to $5,000 or **Rate** equals 0.095?

   a. `where Amount<=5000 or Rate=0.095;`
   b. `where Amount le 5000 or Rate=0.095;`
   c. `where Amount<=5000 or Rate eq 0.095;`
   d. all of the above

Copyright © 2012, SAS Institute Inc. All rights reserved.

5. Which of the following statements selects rows in which **Amount** is less than or equal to $5,000 or **Rate** equals 0.095?

a. `where Amount<=5000 or Rate=0.095;`

b. `where Amount le 5000 or Rate=0.095;`

c. `where Amount<=5000 or Rate eq 0.095;`

d. all of the above

Copyright © 2012, SAS Institute Inc. All rights reserved.

6. When you run this code, which title or titles appear in the last PROC PRINT output?

a. The Top Line

b. The Top Line
   The Next Line

c. The Top Line
   The First Line
   The Next Line

```
title1 'The First Line';
title2 'The Second Line';
proc print data=orion.sales;
run;

title2 'The Next Line';
proc print data=orion.sales;
run;

title 'The Top Line';
proc print data=orion.sales;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

6. When you run this code, which title or titles appear in the last PROC PRINT output?

a. The Top Line

b. The Top Line
The Next Line

c. The Top Line
The First Line
The Next Line

```
title1 'The First Line';
title2 'The Second Line';
proc print data=orion.sales;
run;

title2 'The Next Line';
proc print data=orion.sales;
run;

title 'The Top Line';
proc print data=orion.sales;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

7. Which program creates the output shown here?

a.
```
proc print data=orion.staff;
   var Employee_ID Emp_Hire_Date;
   label Employee_ID='Emp ID'
         'Employee_Hire Date';
run;
```

b.
```
proc print data=orion.staff split='+';
   var Employee_ID Emp_Hire_Date;
   label Employee_ID='Emp ID'
         Emp_Hire_Date='Employee+Hire Date';
run;
```

Partial PROC PRINT Output

Obs	Emp ID	Employee Hire Date
1	120101	01JUL2003
2	120102	01JUN1989
3	120103	01JAN1974
4	120104	01JAN1981
5	120105	01MAY1999
6	120106	01JAN1974

Copyright © 2012, SAS Institute Inc. All rights reserved.

7. Which program creates the output shown here?

a.
```
proc print data=orion.staff;
   var Employee_ID Emp_Hire_Date;
   label Employee_ID='Emp ID'
         'Employee_Hire Date';
run;
```

b.
```
proc print data=orion.staff split='+';
   var Employee_ID Emp_Hire_Date;
   label Employee_ID='Emp ID'
         Emp_Hire_Date='Employee+Hire Date';
run;
```

Partial PROC PRINT Output

Obs	Emp ID	Employee Hire Date
1	120101	01JUL2003
2	120102	01JUN1989
3	120103	01JAN1974
4	120104	01JAN1981
5	120105	01MAY1999
6	120106	01JAN1974

Copyright © 2012, SAS Institute Inc. All rights reserved.

8. Which BY statement is valid for this PROC PRINT step?

```
proc sort data=orion.staff
          out=work.staffsort;
   by Gender Start_Date;
run;

proc print data=work.staffsort label;
   by ____________________________;
   label Start_Date='Start';
run;
```

a. `by Start_Date Gender;`

b. `by Start;`

c. `by descending Gender;`

d. `by Gender;`

Copyright © 2012, SAS Institute Inc. All rights reserved.

8. Which BY statement is valid for this PROC PRINT step?

```
proc sort data=orion.staff
          out=work.staffsort;
   by Gender Start_Date;
run;

proc print data=work.staffsort label;
   by ____________________________;
   label Start_Date='Start';
run;
```

a. `by Start_Date Gender;`

b. `by Start;`

c. `by descending Gender;`

d. `by Gender;`

Copyright © 2012, SAS Institute Inc. All rights reserved.

9. Suppose you already ran the first program, which created a one-page report. Next, you want to run the second program. What will appear at the top of the second report?

```
title1 'RADIX Company';
title3 'DVD Sales';
proc print data=radix.sales;
   where UnitSold>=30;
run;
```

```
title2 'Best Sales';
title;
proc print data=radix.staff;
   where Sales>25000;
run;
```

a. no titles

b. RADIX Company
   Best Sales
   DVD Sales

c. RADIX Company
   Best Sales

d. RADIX Company

Copyright © 2012, SAS Institute Inc. All rights reserved.

9. Suppose you already ran the first program, which created a one-page report. Next, you want to run the second program. What will appear at the top of the second report?

a. no titles

b. RADIX Company
Best Sales
DVD Sales

c. RADIX Company
Best Sales

d. RADIX Company

```
title1 'RADIX Company';
title3 'DVD Sales';
proc print data=radix.sales;
   where UnitSold>=30;
run;
```

```
title2 'Best Sales';
title;
proc print data=radix.staff;
   where Sales>25000;
run;
```

Copyright © 2012, SAS Institute Inc. All rights reserved.

10. Which statement about this program is true?

```
proc print data=orion.sales;
   var Employee_ID Salary;
   where Country='AU';
   by Gender;
   label Salary='Annual Salary';
run;
```

a. This program will run correctly only if **orion.sales** is sorted in ascending order by **Country**.

b. The PROC PRINT report displays only the observations in which the value of **Country** is *AU*.

c. Annual Salary will be displayed at the top of the **Salary** column.

Copyright © 2012, SAS Institute Inc. All rights reserved.

10. Which statement about this program is true?

```
proc print data=orion.sales;
   var Employee_ID Salary;
   where Country='AU';
   by Gender;
   label Salary='Annual Salary';
run;
```

a. This program will run correctly only if **orion.sales** is sorted in ascending order by **Country**.

b. The PROC PRINT report displays only the observations in which the value of **Country** is *AU*.

c. *Annual Salary* will be displayed at the top of the **Salary** column.

Chapter 5 Formatting Data Values

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

5.1 Using SAS Formats

Objectives

- Describe SAS formats.
- Apply SAS formats with the FORMAT statement.

3

Business Scenario

Enhance the appearance of variable values in reports.

Last_Name	First_ Name	Country	Job_Title	Salary	Hire_ Date
Zhou	Tom	AU	Sales Manager	108255	12205
Dawes	Wilson	AU	Sales Manager	87975	6575
Elvish	Irenie	AU	Sales Rep. II	26600	6575

Last_Name	First_ Name	Country	Job_Title	Salary	Hire_Date
Zhou	Tom	AU	Sales Manager	$108,255	06/01/1993
Dawes	Wilson	AU	Sales Manager	$87,975	01/01/1978
Elvish	Irenie	AU	Sales Rep. II	$26,600	01/01/1978

4

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Formats

SAS formats can be used in a PROC step to change how values are displayed in a report.

5

FORMAT Statement

The *FORMAT statement* associates a format with a variable.

```
proc print data=orion.sales noobs;
   format Salary dollar8. Hire_Date mmddyy10.;
   var Last_Name First_Name Country
       Job_Title Salary Hire_Date;
run;
```

FORMAT *variable(s) format …*;

6

p105d01

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Viewing the Output

Partial PROC PRINT Output

Last_Name	First_ Name	Country	Job_Title	Salary	Hire_Date
Zhou	Tom	AU	Sales Manager	$108,255	06/01/1993
Dawes	Wilson	AU	Sales Manager	$87,975	01/01/1978
Elvish	Irenie	AU	Sales Rep. II	$26,600	01/01/1978
Ngan	Christina	AU	Sales Rep. II	$27,475	07/01/1982
Hotstone	Kimiko	AU	Sales Rep. I	$26,190	10/01/1989

DOLLAR8. MMDDYY10.

7

What Is a Format?

A *format* is an instruction to write data values.

- A format changes the appearance of a variable's value in a report.
- The values stored in the data set are ***not*** changed.

8

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Formats

SAS formats have the following form:

<$>format<w>.<d>

$	Indicates a character format.
format	Names the SAS format.
w	Specifies the total format width, including decimal places and special characters.
.	Is required syntax. Formats always contain a period (.) as part of the name.
d	Specifies the number of decimal places to display in numeric formats.

9

SAS Formats

Selected SAS formats:

Format	Definition
$*w*.	Writes standard character data.
w.d	Writes standard numeric data.
COMMA*w.d*	Writes numeric values with a comma that separates every three digits and a period that separates the decimal fraction.
DOLLAR*w.d*	Writes numeric values with a leading dollar sign, a comma that separates every three digits, and a period that separates the decimal fraction.
COMMAX*w.d*	Writes numeric values with a period that separates every three digits and a comma that separates the decimal fraction.
EUROX*w.d*	Writes numeric values with a leading euro symbol (€), a period that separates every three digits, and a comma that separates the decimal fraction.

10

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Format Examples

Selected SAS formats:

Format	Stored Value	Displayed Value
$4.	Programming	Prog
12.	27134.5864	27135
12.2	27134.5864	27134.59
COMMA12.2	27134.5864	27,134.59
DOLLAR12.2	27134.5864	$27,134.59
COMMAX12.2	27134.5864	27.134,59
EUROX12.2	27134.5864	€27.134,59

11

SAS Format Examples

If the format width is not large enough to accommodate a numeric value, the displayed value is automatically adjusted to fit the width.

Format	Stored Value	Displayed Value
DOLLAR12.2	27134.5864	$27,134.59
DOLLAR9.2	27134.5864	$27134.59
DOLLAR8.2	27134.5864	27134.59
DOLLAR5.2	27134.5864	27135
DOLLAR4.2	27134.5864	27E3

12

One aspect of the adjustment is rounding.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

5.01 Quiz

Use SAS documentation or the SAS Help Facility to explore the *Zw.d* numeric format. What is it used for?

Hint: Search for *Zw.d* or explore “Formats by Category.”

13

SAS Date Format Examples

SAS date formats display SAS date values in standard date forms.

Format	Stored Value	Displayed Value
MMDDYY10.	0	01/01/1960
MMDDYY8.	0	01/01/60
MMDDYY6.	0	010160
DDMMYY10.	365	31/12/1960
DDMMYY8.	365	31/12/60
DDMMYY6.	365	311260

15

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

SAS Date Format Examples

Additional date formats:

Format	Stored Value	Displayed Value
DATE7.	-1	31DEC59
DATE9.	-1	31DEC1959
WORDDATE.	0	January 1, 1960
WEEKDATE.	0	Friday, January 1, 1960
MONYY7.	0	JAN1960
YEAR4.	0	1960

16

5.02 Quiz

Which FORMAT statement creates the output shown below?

a.
```
format Birth_Date Hire_Date mmddyy10.
       Term_Date monyy7.;
```

b.
```
format Birth_Date Hire_Date ddmmyyyy.
       Term_Date mmmyyyy.;
```

c.
```
format Birth_Date Hire_Date ddmmyy10.
       Term_Date monyy7.;
```

output

Birth_Date	Hire_Date	Term_Date
21/05/1969	15/10/1992	MAR2007

17

National Language Support (NLS) enables a software product to function properly in every global market for which the product is targeted. SAS contains NLS features to ensure that SAS applications conform to local language conventions.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Format	Locale	Example
NLDATEw.	English_UnitedStates	January 01, 1960
	German_Germany	01. Januar 1960
NLDATEMNw.	English_UnitedStates	January
	German_Germany	Januar
NLDATEWw.	English_UnitedStates	Fri, Jan 01, 60
	German_Germany	Fr, 01. Jan 60
NLDATEWNw.	English_UnitedStates	Friday
	German_Germany	Freitag

NLS date formats convert SAS date values to a locale-sensitive date string. The LOCALE= system option is used to specify the locale, which reflects the local conventions, language, and culture of a geographical region. For example, a locale value of *English_Canada* represents the country of Canada with a language of English. A locale value of *French_Canada* represents the country of Canada with a language of French.

The LOCALE= system option can be specified in a configuration file, at SAS invocation, or in the OPTIONS statement. For more information, refer to *SAS® 9.3 National Language Support Reference Guide* in the SAS documentation.

Exercises

If you restarted your SAS session since the last exercise, open and submit the **libname.sas** program found in the data folder.

Level 1

1. Displaying Formatted Values in a Detail Report

a. Open **p105e01** and submit. Review the output.

b. Modify the PROC PRINT step to display only **Employee_ID**, **Salary**, **Birth_Date**, and **Employee_Hire_Date**.

c. Add a FORMAT statement to display **Salary** in a dollar format, **Birth_Date** in 01/31/2012 date style, and **Employee_Hire_Date** in the 01JAN2012 date style, as shown in the report below.

```
                                                Employee_
Obs     Employee_ID        Salary    Birth_Date   Hire_Date

  1          120101   $163,040.00    08/18/1980   01JUL2007
  2          120102   $108,255.00    08/11/1973   01JUN1993
  3          120103    $87,975.00    01/22/1953   01JAN1978
...
423          121147    $29,145.00    05/28/1973   01SEP1991
```

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
424          121148    $52,930.00    01/01/1973    01JAN2002
```

Level 2

2. Displaying Formatted Values in a Detail Report

a. Write a PROC PRINT step to display the report below using **orion.sales** as input. Subset the observations and variables to produce the report shown below. Include titles, labels, and formats. The results contain 13 observations.

```
                          US Sales Employees
                        Earning Under $26,000

               First                                                  Date
Employee_ID    Name        Last Name       Title            Salary    Hired

     121036    Teresa      Mesley       Sales Rep. I       $25,965   OCT2007
     121038    David       Anstey       Sales Rep. I       $25,285   AUG2010
     121044    Ray         Abbott       Sales Rep. I       $25,660   AUG1979
      ...
     121106    James       Hilburger    Sales Rep. I       $25,880   FEB2000
     121108    Libby       Levi         Sales Rep. I       $25,930   NOV2010
```

Challenge

3. Exploring Formats by Category

a. Display **orion.sales** as shown in the report below. Refer to SAS Help or product documentation to explore the Dictionary of Formats and investigate “SAS Formats by Category.” Identify and use the character format that displays values in uppercase and a format that displays a character value in quotation marks. The results contain 165 observations.

```
               First_
Employee_ID    Name         Last_Name       Job_Title

     120102    TOM          ZHOU            "Sales Manager"
     120103    WILSON       DAWES           "Sales Manager"
     120121    IRENIE       ELVISH          "Sales Rep. II"
     ...
     121144    RENEE        CAPACHIETTI     "Sales Manager"
     121145    DENNIS       LANSBERRY       "Sales Manager"
```

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

5.2 User-Defined Formats

Objectives

- Create user-defined formats using the FORMAT procedure.
- Apply user-defined formats using a FORMAT statement in a report.
- Use formats to recode data values.
- Use formats to collapse or aggregate data.

22

Business Scenario

Display country names instead of country codes in a report.

Current Report (partial output)

Obs	Employee_ID	Salary	Country	Birth_ Date	Hire_ Date
1	120102	$108,255	AU	AUG1973	JUN1993
2	120103	$87,975	AU	JAN1953	JAN1978
3	120121	$26,600	AU	AUG1948	JAN1978

Desired Report (partial output)

Obs	Employee_ID	Salary	Country	Birth_ Date	Hire_ Date
1	120102	$108,255	Australia	AUG1973	JUN1993
2	120103	$87,975	Australia	JAN1953	JAN1978
3	120121	$26,600	Australia	AUG1948	JAN1978

p105d02

23

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

User-Defined Formats: Part 1

Use PROC FORMAT to create a user-defined format.

```
proc format;
   value $ctryfmt  'AU'='Australia'
                   'US'='United States'
                  other='Miscoded';
run;
```

PROC FORMAT;
VALUE *format-name range1* **=** '*label*'
range2 **=** '*label*'
. . . **;**
RUN;

24 p105d03

User-Defined Formats: Part 2

Use a FORMAT statement in the PROC PRINT step to apply the format to a specific variable.

```
proc print data=orion.sales;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   format Salary dollar10.
          Birth_Date Hire_Date monyy7.
          Country $ctryfmt.;
run;
```

25 p105d03

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Viewing the Output

Partial PROC PRINT Output

Obs	Employee_ID	Salary	Country	Birth_ Date	Hire_ Date
1	120102	$108,255	Australia	AUG1973	JUN1993
2	120103	$87,975	Australia	JAN1953	JAN1978
3	120121	$26,600	Australia	AUG1948	JAN1978
4	120122	$27,475	Australia	JUL1958	JUL1982
5	120123	$26,190	Australia	SEP1968	OCT1989

26

VALUE Statement

VALUE *format-name range1='label '*
range2='label '
. . . ;

A format name

- can be up to 32 characters in length
- for character formats, must begin with a dollar sign ($), followed by a letter or underscore
- for numeric formats, must begin with a letter or underscore
- cannot end in a number
- cannot be given the name of a SAS format
- cannot include a period in the VALUE statement.

27

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

VALUE Statement

VALUE *format-name range1='label '*
range2='label '
. . . ;

Each range can be
- a single value
- a range of values
- a list of values.

Labels
- can be up to 32,767 characters in length
- are enclosed in quotation marks.

28

Enclosing labels in quotation marks is a best practice, and it is required if a label contains internal blanks.

5.03 Multiple Answer Poll

Which names are invalid for user-defined formats?

a. $stfmt
b. $3levels
c. _4years
d. salranges
e. dollar

29

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Defining a Character Format

character format name

discrete character values

```
proc format;
   value $ctryfmt 'AU'='Australia'
                  'US'='United States'
                 other='Miscoded';
run;
```

keyword

labels

The OTHER keyword includes all values that do not match any other value or range.

31 p105d03

Applying a Format

User-defined and SAS formats can be applied in a single FORMAT statement.

```
proc print data=orion.sales label;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   format Salary dollar10.
          Birth_Date Hire_Date monyy7.
          Country $ctryfmt.;
run;
```

- A period (for example, at the end of the $CTRYFMT format) is required when user-defined formats are used in a FORMAT statement.

32 p105d03

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Idea Exchange

The formatting examples shown in this section are sometimes referred to as *translating values*.

Can you give an example of where this type of application might be useful?

33

Business Scenario

An Orion Star manager wants a report showing employee salaries collapsed into three user-defined groups or tiers.

Current Report

Obs	Employee_ID	Last_Name	Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
4	120122	Ngan	27475

Desired Report

Obs	Employee_ID	Last_Name	Salary
1	120102	Zhou	Tier 3
2	120103	Dawes	Tier 2
3	120121	Elvish	Tier 1
4	120122	Ngan	Tier 1

35

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Specifying Ranges of Values

Use PROC FORMAT to specify the salary range for each tier.

Salary	Value
20,000 to 49,999	Tier1
50,000 to 99,999	Tier2
100,000 to 250,000	Tier3

36

Defining a Numeric Format

numeric ranges

```
proc format;
   value tiers      0-49999='Tier 1'
               50000-99999='Tier 2'
             100000-250000='Tier 3';
run;
```

numeric format name

labels

37 p105d04

Defining and Using a Numeric Format

p105d04

This demonstration illustrates the use of a user-defined numeric format.

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

```
proc format;
   value tiers 20000-49999 ='Tier 1'
               50000-99999 ='Tier 2'
              100000-250000='Tier 3';
run;

data work.salaries;
   input Name $ Salary;
   Original_Salary=Salary;
   datalines;
Abi 50000
Mike 65000
Jose 50000.00
Joe 37000.50
Ursula 142000
Lu 49999.99
;

proc print data=work.salaries;
   format Salary tiers.;
run;
```

1. Program **p105d04** includes a PROC FORMAT step to create the TIERS format. The DATA step reads data lines within the program to create a data set containing names and salaries. The PROC PRINT step displays the new data set, applying the TIERS format. Notice in the program that **Salary** is assigned to **Original_Salary** so that both can be included in the report. The assignment statement is covered in a later chapter.

2. Look at the data values and predict what label will be displayed when the TIERS format is applied to **Salary**. Lu's salary falls within a "gap."

Name	Salary	Coded Salary
Abi	50000	
Mike	65000	
Jose	50000.00	
Joe	37000.50	
Ursula	142000	
Lu	49999.99	

3. Submit the program. What **Salary** value is displayed for Lu? _____

   When a value does not match any of the ranges, PROC PRINT attempts to display the actual value. In this case, the column width was determined by the width of the formatted values, which is 6. As mentioned earlier, if the format width is not large enough to accommodate a numeric value, the displayed value is automatically adjusted to fit in the width. What can you do to correct this?

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

4. Specify a width of 8 on the TIERS format in the FORMAT statement and resubmit the program.

```
proc print data=work.salaries;
   format Salary tiers8.;
run;
```

Now what **Salary** value is displayed for Lu? _____

Defining a Continuous Range

The less than (<) symbol excludes the endpoint from a range, allowing a continuous range.

- Put < after the starting value in a range to exclude it.
- Put < before the ending value in a range to exclude it.

Range	Starting Value	Ending Value
50000 - 100000	Includes 50000	Includes 100000
50000 - < 100000	Includes 50000	Excludes 100000
50000 < - 100000	Excludes 50000	Includes 100000
50000 < - < 100000	Excludes 50000	Excludes 100000

39

The < symbol is used to define an exclusive range. The > symbol is not permitted in a VALUE statement.

5.04 Quiz

How will a value of *50000* be displayed if the TIERS format below is applied to the value?

a. Tier 1
b. Tier 2
c. 50000
d. a missing value

```
proc format;
   value tiers  20000-<50000 ='Tier 1'
                50000-<100000='Tier 2'
                100000-250000='Tier 3';
run;
```

p105d05

40

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

LOW and HIGH Keywords

the lowest possible value

```
proc format;
   value tiers     low-<50000 ='Tier 1'
                 50000-<100000='Tier 2'
                100000-high   ='Tier 3';
run;
```

the highest possible value

The LOW keyword

- includes missing values for character variables
- does not include missing values for numeric variables.

42 p105d06

Applying a Numeric Format

Part 1

```
proc format;
   value tiers     low-<50000 ='Tier 1'
                 50000-<100000='Tier 2'
                 100000-high  ='Tier 3';
run;
```

Part 2

```
proc print data=orion.sales;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   format Birth_Date Hire_Date monyy7.
          Salary tiers.;
run;
```

43 p105d06

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Viewing the Output

Partial PROC PRINT Output

Obs	Employee_ID	Job_Title	Salary	Country	Birth_ Date	Hire_ Date
1	120102	Sales Manager	Tier 3	AU	AUG1973	JUN1993
2	120103	Sales Manager	Tier 2	AU	JAN1953	JAN1978
3	120121	Sales Rep. II	Tier 1	AU	AUG1948	JAN1978
4	120122	Sales Rep. II	Tier 1	AU	JUL1958	JUL1982
5	120123	Sales Rep. I	Tier 1	AU	SEP1968	OCT1989

44

User-Defined Format Example

Ranges can be specified using lists, ranges, discrete values, and keywords.

```
proc format;
   value mnthfmt 1,2,3='Qtr 1'
                   4-6='Qtr 2'
                   7-9='Qtr 3'
                 10-12='Qtr 4'
                     .='missing'
                 other='unknown';
run;
```

45

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Multiple User-Defined Formats

Multiple VALUE statements can be included in a single PROC FORMAT step.

```
proc format;
   value $ctryfmt  'AU'='Australia'
                   'US'='United States'
                  other='Miscoded';

   value tiers     low-<50000 ='Tier 1'
                 50000-<100000='Tier 2'
                 100000-high  ='Tier 3';
run;
```

46 p105d07

Viewing the Output

```
proc print data=orion.sales;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   format Birth_Date Hire_Date monyy7.
          Country $ctryfmt.
          Salary tiers.;
run;
```

Partial PROC PRINT Output

Obs	Employee_ID	Job_Title	Salary	Country	Birth_ Date	Hire_ Date
1	120102	Sales Manager	Tier 3	Australia	AUG1973	JUN1993
2	120103	Sales Manager	Tier 2	Australia	JAN1953	JAN1978
3	120121	Sales Rep. II	Tier 1	Australia	AUG1948	JAN1978
4	120122	Sales Rep. II	Tier 1	Australia	JUL1958	JUL1982
5	120123	Sales Rep. I	Tier 1	Australia	SEP1968	OCT1989

47 p105d07

Copyright © 2013, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Chapter 6: Reading SAS® Data Sets

6.1 Reading a SAS Data Set

6.2 Customizing a SAS Data Set

Chapter 6: Reading SAS® Data Sets

6.1 Reading a SAS Data Set

6.2 Customizing a SAS Data Set

Objectives

- Define the business scenario that will be used when reading from a data source to create a SAS data set.
- Use a DATA step to create a SAS data set from an existing SAS data set.
- Subset observations with a WHERE statement.
- Create a new variable with an assignment statement.

Business Scenario

Information about Orion Star sales employees resides in several input sources.

Considerations

Management wants a series of reports for Australian sales employees. You will read data from various input sources to create a SAS data set that can be analyzed and presented.

6.01 Multiple Answer Poll

Which types of files will you read into SAS?

a. SAS data sets
b. Excel worksheets
c. database tables
d. raw data files
e. other
f. not sure

Business Scenario: Part 1

Read an existing SAS data set to create a new data set. The new data set should include only the observations for the Australian sales representatives.

Using a SAS Data Set as Input

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

DATA *output-SAS-data-set*;
    SET *input-SAS-data-set*;
    WHERE *WHERE-expression*;
RUN;

p106d01

DATA Statement

The *DATA statement* begins a DATA step and provides the name of the SAS data set to create.

DATA *output-SAS-data-set*;

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

A DATA step can create temporary or permanent data sets.

✎ The rules for SAS variable names also apply to data set names.

p106d01

SET Statement

The *SET statement* reads observations from an existing SAS data set for further processing in the DATA step.

SET *input-SAS-data-set*;

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

- The SET statement reads all observations and all variables from the input data set.
- Observations are read sequentially, one at a time.
- The SET statement can read temporary or permanent data sets.

p106d01

WHERE Statement

The *WHERE statement* selects observations from a SAS data set that meet a particular condition.

WHERE *WHERE-expression*;

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

The variables named in the WHERE expression must exist in the input SAS data set.

p106d01

Viewing the Log

Partial SAS Log

```
42   data work.subset1;
43      set orion.sales;
44      where Country='AU' and
45            Job_Title contains 'Rep';
46   run;

NOTE: There were 61 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep';
NOTE: The data set WORK.SUBSET1 has 61 observations and 9 variables.
```

SAS read 61 of the 165 observations.

Viewing the Output

```
proc print data=work.subset1 noobs;
run;
```

Partial PROC PRINT Output

Employee_ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_ Date
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-4169	6575
120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-523	8217
120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	3193	10866
120124	Lucian	Daymond	M	26480	Sales Rep. I	AU	1228	8460
120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU	-391	8460

p106d01

Setup for the Poll

Consider the DATA step below.

```
data us;
   set orion.sales;
   where Country='US';
run;
```

p106a01

6.02 Multiple Choice Poll

Considering this DATA step, which statement is true?

a. It reads a temporary data set and creates a permanent data set.
b. It reads a permanent data set and creates a temporary data set.
c. It contains a syntax error and will not execute.
d. It will not execute because you cannot work with permanent and temporary data sets in the same step.

6.02 Multiple Choice Poll – Correct Answer

Considering this DATA step, which statement is true?

a. It reads a temporary data set and creates a permanent data set.

b. It reads a permanent data set and creates a temporary data set.

c. It contains a syntax error and will not execute.

d. It will not execute because you cannot work with permanent and temporary data sets in the same step.

```
data us;                /* Create a temporary data set */
   set orion.sales;   /* Read a permanent data set */
   where Country='US';
run;
```

Questions ???

Business Scenario: Part 2

Orion Star management wants to give a 10% bonus to each Australian Sales representative hired before January 1, 2000.

Considerations

Subsetting is based on **Hire_Date**, which contains a SAS date value. How can you compare a SAS date value to a calendar date?

Date Constant

A date constant can be used in any SAS expression, including a WHERE expression.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep' and
         Hire_Date<'01jan2000'd;
run;
```

- A SAS date constant is a date written in the form **'ddmmm<yy>yy'd**.

p106d02

Considerations

Create a data set that includes the new variable, **Bonus**, which represents a 10% bonus.

Assignment Statement

The *assignment statement* evaluates an expression and assigns the result to a new or existing variable.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep' and
         Hire_Date<'01jan2000'd;
   Bonus=Salary*.10;
run;
```

***variable=expression*;**

p106d02a

Assignment Statement

The *expression* consists of operands and operators.

***variable=expression*;**

Operands

- character constants
- numeric constants
- date constants
- character variables
- numeric variables

Operators

- symbols that represent a calculation or manipulation

- SAS functions

Sample Assignment Statements

Example	Type
`Salary=26960;`	Numeric constant
`Gender='F';`	Character constant
`Hire_Date='21JAN1995'd;`	Date constant
`BonusMonth=month(Hire_Date);`	SAS function
`Bonus=Salary*.10;`	Arithmetic expression

Arithmetic Operators

If any operand in an arithmetic expression has a missing value, the result is a missing value.

Symbol	Definition	Priority
**	Exponentiation	I
*	Multiplication	II
/	Division	II
+	Addition	III
-	Subtraction	III

Parentheses can be used to clarify or alter the order of operations in an arithmetic expression.

Viewing the Log

Partial SAS Log

```
214  data work.subset1;
215     set orion.sales;
216     where Country='AU' and
217           Job_Title contains 'Rep' and
218           Hire_Date<'01jan2000'd;
219     Bonus=Salary*.10;
220  run;

NOTE: There were 29 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep' and
      (Hire_Date<'01JAN2000'D);
NOTE: The data set WORK.SUBSET1 has 29 observations and 10 variables.
```

The input data set has 9 variables, and the new data set has 10 variables.

Viewing the Output

```
proc print data=work.subset1 noobs;
   var First_Name Last_Name Salary
       Job_Title Bonus Hire_Date;
   format Hire_Date date9.;
run;
```

Partial PROC PRINT Output

First_Name	Last_Name	Salary	Job_Title	Bonus	Hire_Date
Irenie	Elvish	26600	Sales Rep. II	2660.0	01JAN1978
Christina	Ngan	27475	Sales Rep. II	2747.5	01JUL1982
Kimiko	Hotstone	26190	Sales Rep. I	2619.0	01OCT1989
Lucian	Daymond	26480	Sales Rep. I	2648.0	01MAR1983
Fong	Hofmeister	32040	Sales Rep. IV	3204.0	01MAR1983

p106d02a

6.03 Quiz

Evaluate the assignment statements below given the values shown in the PDV.

x	y	z
.	4	10

a. `num=y+z/2;`

---

b. `num=x+z/2;`

6.03 Quiz – Correct Answer

Evaluate the assignment statements below given the values shown in the PDV.

x	y	z
.	4	10

a. `num=y+z/2;` → 4+10/2 → 4+5 → 9

`num=(y+z)/2;` → 14/2 → 7

---

b. `num=x+z/2;` → .+10/2 → .+5 → .

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 6: Reading SAS® Data Sets

6.1 Reading a SAS Data Set

6.2 Customizing a SAS Data Set

Objectives

- Subset variables by using the DROP and KEEP statements.
- Explore the compilation and execution phases of the DATA step.
- Store labels and formats in the descriptor portion of a SAS data set.

Business Scenario: Part 3

All Australian sales representatives will get a bonus, regardless of hire date. The new data set should contain a subset of the variables from the input data set.

orion.sales

work.subset1

DROP Statement

The DROP statement specifies the variables to ***exclude*** from the output data set.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

DROP *variable-list*;

Partial SAS Log

```
NOTE: There were 61 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep';
NOTE: The data set WORK.SUBSET1 has 61 observations and 6 variables.
```

p106d03

Viewing the Output

```
proc print data=work.subset1;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Job_Title	Hire_ Date	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d03

KEEP Statement

The KEEP statement specifies all variables to ***include*** in the output data set.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   keep First_Name Last_Name Salary
        Job_Title Hire_Date Bonus;
run;
```

KEEP *variable-list*;

- If a KEEP statement is used, it must include ***every*** variable to be written, including any new variables.

p106d04

Viewing the Log

Partial SAS Log

```
NOTE: There were 61 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep';
NOTE: The data set WORK.SUBSET1 has 61 observations and 6 variables.
```

Viewing the Output

```
proc print data=work.subset1;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Job_Title	Hire_ Date	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d04

Questions ???

Business Scenario: Behind the Scenes

Orion Star programmers need to understand the internal processing that occurs when a DATA step is submitted.

DATA Step Processing

SAS processes the DATA step in two phases:

Compilation Phase

Execution Phase

Compilation Phase

Scans the program for syntax errors; translates the program into machine language.

PDV

Name	Salary

Creates the *program data vector* (*PDV*) to hold one observation.

Creates the descriptor portion of the output data set.

Compilation

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

p106d03

...

Compilation

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	Last_Name $ 18	Gender $ 1	Salary N 8	Job_Title $ 25

Country $ 2	Birth_Date N 8	Hire_Date N 8

...

Compilation

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	Last_Name $ 18	Gender $ 1	Salary N 8	Job_Title $ 25

Country $ 2	Birth_Date N 8	Hire_Date N 8	Bonus N 8

...

Compilation

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
D N 8	$ 12	$ 18	D $ 1	N 8	$ 25

Country	Birth_Date	Hire_Date	Bonus
D $ 2	D N 8	N 8	N 8

...

Compilation

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
D N 8	$ 12	$ 18	D $ 1	N 8	$ 25

Country	Birth_Date	Hire_Date	Bonus
D $ 2	D N 8	N 8	N 8

Descriptor Portion of **work.subset1**

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus

Execution Phase

Execution

Partial **orion.sales**

Employee _ID		Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

Initialize PDV

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

D Employee _ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
.		.			.	.	.

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus

...

Execution

Partial **orion.sales**

Employee_ID	...	Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

D Employee_ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120121		26600		AU	-4169	6575	.

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus

...

Execution

Partial **orion.sales**

Employee_ID	...	Hire_Date
120121		6575
120122		8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

D Employee_ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120121		26600		AU	-4169	6575	2660

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus

...

Execution

Partial **orion.sales**

Employee _ID	...	Hire_Date
120121		6575
120122		8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

D Employee _ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120121		26600		AU	-4169	6575	2660

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660

...

Execution

Partial **orion.sales**

Employee_ID	...	Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

Reinitialize PDV

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

New variables are reinitialized.

PDV

D Employee_ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120121		26600		AU	-4169	6575	.

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660

...

Execution

Partial **orion.sales**

Employee_ID	...	Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

D Employee_ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120122		27475		AU	-523	8217	.

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660

...

Execution

Partial **orion.sales**

Employee_ID	...	Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

PDV

D Employee_ID	...	Salary	...	D Country	D Birth_Date	Hire_Date	Bonus
120122		27475		AU	-523	8217	274.75

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660

...

Execution

Partial **orion.sales**

Employee _ID	...	Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
         Birth_D
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

Employee _ID (D)	...	Salary	...	Country (D)	Birth_Date (D)	Hire_Date	Bonus
120122		27475		AU	-523	8217	274.75

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660.00
Christina	Ngan	27475	Sales Rep II.	8217	2747.50

...

Execution

Partial **orion.sales**

Employee _ID		Hire_Date
120121		6575
120122	...	8217
120123		10866
120124		8460

```
data work.subset1;
   set orion.sale
   where Country=
         Job_Titl
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

Continue until EOF

PDV

Employee _ID (D)	...	Salary	...	Country (D)	Birth_Date (D)	Hire_Date	Bonus
120122		27475		AU	-523	8217	.

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep II.	6575	2660.00
Christina	Ngan	27475	Sales Rep II.	8217	2747.50

Viewing the Output

```
proc print data=work.subset1;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Job_Title	Hire_ Date	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d03

Questions ???

Business Scenario: Part 4

Create a data set that contains all Australian employees whose **Bonus** is at least $3000.

Selecting Observations

Subsetting is based on the new variable, **Bonus**, that is created with an assignment statement.

```
data work.auemps;
   set orion.sales;
   where Country='AU';
   Bonus=Salary*.10;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

A WHERE statement is used to subset observations when the selected variables exist in the ***input*** data set.

p106d03

6.04 Quiz

Open and submit **p106a03**. Is the output data set created successfully?

```
data work.usemps;
   set orion.sales;
   Bonus=Salary*.10;
   where Country='US' and Bonus>=3000;
run;
```

p106a03

6.04 Quiz – Correct Answer

Open and submit **p106a03**. Is the output data set created successfully?

```
260  data work.usemps;
261     set orion.sales;
262     Bonus=Salary*.10;
263     where Country='US' and Bonus>=3000;
ERROR: Variable Bonus is not on file ORION.SALES.
264  run;

NOTE: The SAS System stopped processing this step because of
errors.
WARNING: The data set WORK.USEMPS may be incomplete.  When
this step was stopped there were 0 observations and 10
variables.
```

No. Bonus cannot be used in a WHERE statement because it is not in the input data set. It is a new variable created in this DATA step.

Subsetting IF

The *subsetting IF* statement tests a condition to determine whether the DATA step should continue processing the current observation.

```
data work.auemps;
   set orion.sales;
   where Country='AU';
   Bonus=Salary*.10;
   if Bonus>=3000;
run;
```

IF *condition*;

In this program, processing will reach the bottom of the DATA step and output an observation only if the condition is true.

p106d05

Viewing the Log

Partial SAS Log

```
11   data work.auemps;
12      set orion.sales;
13      where Country='AU';
14      Bonus=Salary*.10;
15      if Bonus>=3000;
16   run;

NOTE: There were 63 observations read from the data set ORION.SALES.
      WHERE Country='AU';
NOTE: The data set WORK.AUEMPS has 12 observations and 10 variables.
```

Of the 165 observations in **orion.sales**, 63 were read into the PDV for processing, and only 12 were written to the new data set.

Viewing the Output

```
proc print data=work.auemps;
   var First_Name Last_Name Salary Bonus;
run;
```

PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Bonus
1	Tom	Zhou	108255	10825.5
2	Wilson	Dawes	87975	8797.5
3	Fong	Hofmeister	32040	3204.0
4	Monica	Kletschkus	30890	3089.0
5	Alvin	Roebuck	30070	3007.0
6	Alexei	Platts	32490	3249.0
7	Viney	Barbis	30265	3026.5
8	Caterina	Hayawardhana	30490	3049.0
9	Daniel	Pilgrim	36605	3660.5
10	Lynelle	Phoumirath	30765	3076.5
11	Rosette	Martines	30785	3078.5
12	Fadi	Nowd	30660	3066.0

Processing the Subsetting IF Statement

A subsetting IF statement is valid only in a DATA step.

Idea Exchange

File **p106a04** contains two versions of the previous program. Submit both programs and compare the output and number of observations read. What do you notice about the results?

```
data work.auemps;
    set orion.sales;
    Bonus=Salary*.10;
    if Country='AU' and Bonus>=3000;
run;
```

WHERE versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
SET statement	Yes	Yes
assignment statement	No	Yes

Questions ???

Business Scenario: Part 5

Define permanent labels and formats for some of the variables in the new data set.

LABEL Statement

The LABEL statement assigns descriptive labels to variables.

Sales Title **Date Hired**

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus

Defining Permanent Labels

Use a LABEL statement in a DATA step to permanently assign labels to variables. The labels are stored in the descriptor portion of the data set.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

LABEL *variable='label '*
<variable='label '…>;

p106d06

Viewing the Output

```
proc contents data=work.subset1;
run;
```

Partial PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes

#     Variable        Type     Len     Label

6     Bonus           Num        8
1     First_Name      Char      12
5     Hire_Date       Num        8     Date Hired
4     Job_Title       Char      25     Sales Title
2     Last_Name       Char      18
3     Salary          Num        8
```

p106d06

Viewing the Output: Displaying Labels

To use labels in the PRINT procedure, use the LABEL option in the PROC PRINT statement.

```
proc print data=work.subset1 label;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d06

Viewing the Output: Splitting Labels

Use the PROC PRINT SPLIT= option to split labels across lines based on a split character.

```
proc print data=work.subset1 split=' ';
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d06

6.05 Quiz

What column heading will be displayed for **Job_Title** in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
   Bonus=Salary*.10;
   label Job_Title='Sales Title';
   drop Employee_ID Gender Country
        Birth_Date;
run;
proc print data=work.subset1 label;
   label Job_Title='Title';
run;
```

p106a05

6.05 Quiz – Correct Answer

What column heading will be displayed for **Job_Title** in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
   Bonus=Salary*.10;
   label Job_Title='Sales Title';
   drop Employee_ID Gender Country
        Birth_Date;
run;
proc print data=work.subset1 label;
   label Job_Title='Title';
run;
```

The column heading will be Title. Labels and formats in PROC steps override permanent labels and formats.

FORMAT Statement

The FORMAT statement associates formats with variables.

FORMAT statement → DATA Step

commax8. ddmmyy10. commax8.2

work.subset1

First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Irenie	Elvish	26600	Sales Rep. II	6575	2660.0

Defining Permanent Formats

Use a FORMAT statement in a DATA step to permanently associate formats with variables.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   Bonus=Salary*.10;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary commax8. Bonus commax8.2
          Hire_Date ddmmyy10.;
   drop Employee_ID Gender Country
        Birth_Date;
run;
```

FORMAT *variable(s) format …*;

p106d07

Viewing the Output

```
proc contents data=work.subset1;
run;
```

Partial PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes

#    Variable      Type    Len    Format       Label

6    Bonus         Num       8    COMMAX8.2
1    First_Name    Char     12
5    Hire_Date     Num       8    DDMMYY10.    Date Hired
4    Job_Title     Char     25                 Sales Title
2    Last_Name     Char     18
3    Salary        Num       8    COMMAX8.
```


p106d07

Viewing the Output

```
proc print data=work.subset1 label;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired	Bonus
1	Irenie	Elvish	26.600	Sales Rep. II	01/01/1978	2.660,00
2	Christina	Ngan	27.475	Sales Rep. II	01/07/1982	2.747,50
3	Kimiko	Hotstone	26.190	Sales Rep. I	01/10/1989	2.619,00
4	Lucian	Daymond	26.480	Sales Rep. I	01/03/1983	2.648,00
5	Fong	Hofmeister	32.040	Sales Rep. IV	01/03/1983	3.204,00

p106d07

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter
Review

1. Which statement is used to read a SAS data set in a DATA step?

   a. DATA statement
   b. WHERE statement
   c. SET statement
   d. assignment statement

1. Which statement is used to read a SAS data set in a DATA step?

   a. DATA statement
   b. WHERE statement
   c. SET statement
   d. assignment statement

2. What is the name of the input data set in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
run;
```

a. work.us
b. orion.sales
c. Country
d. sales

2. What is the name of the input data set in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
run;
```

a. work.us
b. orion.sales
c. Country
d. sales

3. What is the name of the output data set in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
run;
```

a. work.us
b. orion.sales
c. Country
d. sales

3. What is the name of the output data set in the program below?

```
data work.us;
   set orion.sales;
   where Country='US';
run;
```

a. work.us
b. orion.sales
c. Country
d. sales

4. Which of the following DATA steps correctly reads the permanent data set **salesinfo** from the **sporting** library and creates a new data set named **salesinfo2** in the same library?

a.
```
data sporting.salesinfo2;
    set salesinfo;
run;
```

b.
```
data salesinfo2;
    set sporting.salesinfo;
run;
```

c.
```
data sporting.salesinfo2;
    set sporting.salesinfo;
run;
```

4. Which of the following DATA steps correctly reads the permanent data set **salesinfo** from the **sporting** library and creates a new data set named **salesinfo2** in the same library?

a.
```
data sporting.salesinfo2;
    set salesinfo;
run;
```

b.
```
data salesinfo2;
    set sporting.salesinfo;
run;
```

c.
```
data sporting.salesinfo2;
    set sporting.salesinfo;
run;
```

5. Which of the following is ***not*** created during the compilation phase?

   a. the descriptor portion of the output data set
   b. the first observation
   c. the program data vector (PDV)

5. Which of the following is ***not*** created during the compilation phase?

   a. the descriptor portion of the output data set
   b. the first observation
   c. the program data vector (PDV)

6. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   drop Gender Salary Birth_Date;
run;
```

a. 6
b. 7
c. 9
d. None. This program contains a logic error.

6. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   drop Gender Salary Birth_Date;
run;
```

a. 6
b. 7
c. 9
d. None. This program contains a logic error.

7. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   keep Employee_ID Gender Job_Title Salary;
run;
```

a. 4

b. 9

c. 13

d. None. This program contains a logic error.

7. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   keep Employee_ID Gender Job_Title Salary;
run;
```

a. 4
b. 9
c. 13
d. None. This program contains a logic error.

8. A KEEP statement in a DATA step omits all variables except **Name**, **Color**, and **Price** from the output data set. Are the omitted variables included in the PDV?

- ❍ Yes
- ❍ No

8. A KEEP statement in a DATA step omits all variables except **Name**, **Color**, and **Price** from the output data set. Are the omitted variables included in the PDV?

- ❍ Yes
- ❍ No

9. What value will be assigned to **Units**?

```
data work.comp;
   set work.sales;
   Units=Total+Bonus/Quantity;
run;
```

Partial PDV

Total	Quantity	Bonus	Units
140	10	50	.

a. 19

b. 145

c. 3

d. missing

9. What value will be assigned to **Units**?

```
data work.comp;
   set work.sales;
   Units=Total+Bonus/Quantity;
run;
```

Partial PDV

Total	Quantity	Bonus	Units
140	10	50	.

a. 19
b. 145
c. 3
d. missing

10. Which procedure can be used to view the permanent labels and formats stored in a data set?

a. PROC CONTENTS
b. PROC PRINT
c. PROC FORMAT
d. PROC UNIVARIATE

10. Which procedure can be used to view the permanent labels and formats stored in a data set?

a. PROC CONTENTS
b. PROC PRINT
c. PROC FORMAT
d. PROC UNIVARIATE

Chapter 7: Reading Spreadsheet and Database Data

7.1 Reading Spreadsheet Data

7.2 Reading Database Data

Chapter 7: Reading Spreadsheet and Database Data

7.1 Reading Spreadsheet Data

7.2 Reading Database Data

Objectives

- Assign a libref to a Microsoft Excel workbook using a SAS/ACCESS LIBNAME statement.
- Access an Excel worksheet using a SAS two-level name.
- Create a SAS data set using a subset of worksheet data.

Business Scenario

The Sales Manager has requested a report about Orion Star sales employees from Australia and the United States.

The input data is in an Excel workbook.

Business Scenario

Use SAS/ACCESS Interface to PC Files to read the worksheets within the **sales.xls** workbook as if they were SAS data sets.

Examine the Workbook

Partial **sales.xls**

	A	B	C	D	E	F	G	H	I
1	Employee	First Name	Last Name	Gender	Salary	Job Title	Country	Birth Date	Hire Date
2	120102	Tom	Zhou	M	108255	Sales Manager	AU	11-Aug-73	6/1/1993
3	120103	Wilson	Dawes	M	87975	Sales Manager	AU	22-Jan-53	1/1/1978
4	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	2-Aug-48	1/1/1978
5	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	27-Jul-58	7/1/1982
6	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	28-Sep-68	10/1/1989
7	120124	Lucian	Daymond	M	26480	Sales Rep. I	AU	13-May-63	3/1/1983
8	120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU	6-Dec-58	3/1/1983
9	120126	Satyakam	Denny	M	26780	Sales Rep. II	AU	20-Sep-92	8/1/2010

two worksheets

cells formatted as dates

SAS/ACCESS LIBNAME Statement

When the bit count of SAS and Microsoft Office are the same (both are 32-bit or both are 64-bit), you can use the default SAS/ACCESS Excel engine.

libref

engine name

```
libname orionx excel "&path\sales.xls";
```

LIBNAME *libref* <*engine*> **"***workbook-name***"** <*options*>**;**

When the bit counts differ, use the PC Files Server.

libref

engine name

```
libname orionx pcfiles path="&path\sales.xls";
```

LIBNAME *libref* <*engine*> <**PATH=**>**"***workbook-name***"** <*options*>**;**

SAS Explorer Window

SAS treats the workbook as a library, and each worksheet as a SAS data set.

- A named range might exist for each worksheet.
- Worksheet names end with a dollar sign.
- Named ranges do ***not*** end with a dollar sign.

CONTENTS Procedure

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

```
        The CONTENTS Procedure

                Directory

 Libref          ORIONX
 Engine          PCFILES
 Physical Name   s:\workshop\sales.xls
 Schema/Owner    .


                     Member   DBMS Member
#   Name             Type     Type

1   Australia        DATA     TABLE
2   Australia$       DATA     SYSTEM TABLE
3   UnitedStates     DATA     TABLE
4   UnitedStates$    DATA     SYSTEM TABLE
```

p107d01

The CONTENTS Procedure

Data Set Name	ORIONX.'Australia$'n	Observations	.
Member Type	DATA	Variables	9
Engine	PCFILES	Indexes	0
Created	.	Observation Length	0
Last Modified	.	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	Default		
Encoding	Default		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
8	Birth_Date	Num	8	DATE9.	DATE9.	Birth Date
7	Country	Char	2	$2.	$2.	Country
1	Employee_ID	Num	8			Employee ID
2	First_Name	Char	10	$10.	$10.	First Name
4	Gender	Char	1	$1.	$1.	Gender
9	Hire_Date	Num	8	DATE9.	DATE9.	Hire Date
6	Job_Title	Char	14	$14.	$14.	Job Title
3	Last_Name	Char	12	$12.	$12.	Last Name
5	Salary	Num	8			Salary

The CONTENTS Procedure

Data Set Name	ORIONX.‘UnitedStates$'n	Observations	.
Member Type	DATA	Variables	9
Engine	PCFILES	Indexes	0
Created	.	Observation Length	0
Last Modified	.	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	Default		
Encoding	Default		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
8	Birth_Date	Num	8	DATE9.	DATE9.	Birth Date
7	Country	Char	2	$2.	$2.	Country
1	Employee_ID	Num	8			Employee ID
2	First_Name	Char	10	$10.	$10.	First Name
4	Gender	Char	1	$1.	$1.	Gender
9	Hire_Date	Num	8	DATE9.	DATE9.	Hire Date
6	Job_Title	Char	14	$14.	$14.	Job Title
3	Last_Name	Char	12	$12.	$12.	Last Name
5	Salary	Num	8			Salary

SAS Name Literals

A *SAS name literal* is a string within quotation marks, followed by the letter n.

```
orionx.'Australia$'n
```

SAS name literal

SAS name literals permit special characters in data set names.

Printing an Excel Worksheet

```
libname orionx pcfiles path="&path\sales.xls";

proc print data=orionx.'Australia$'n;
run;
```

Partial PROC PRINT Output

Obs	Employee_ ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_Date
1	120102	Tom	Zhou	M	108255	Sales Manager	AU	11AUG1973	01JUN1993
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU	22JAN1953	01JAN1978
3	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	02AUG1948	01JAN1978
4	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	27JUL1958	01JUL1982
5	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	28SEP1968	01OCT1989

p107d01

Subsetting Worksheet Data

You can select a subset of the worksheet data.

```
libname orionx pcfiles path="&path\sales.xls";

proc print data=orionx.'Australia$'n noobs;
   where Job_Title ? 'IV';
   var Employee_ID Last_Name Job_Title Salary;
run;
```

p107d01

Viewing the Output

PROC PRINT Output

```
Employee_
    ID        Last_Name        Job_Title          Salary

  120125      Hofmeister       Sales Rep. IV       32040
  120128      Kletschkus       Sales Rep. IV       30890
  120135      Platts           Sales Rep. IV       32490
  120159      Phoumirath       Sales Rep. IV       30765
  120166      Nowd             Sales Rep. IV       30660
```

Disassociating a Libref

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

```
libname orionx pcfiles path="&path\sales.xls";

  /* program to access the worksheets */

libname orionx clear;
```

SAS disconnects from the data source and closes any resources associated with the connection.

p107d01

7.01 Quiz

Which PROC PRINT step displays the worksheet containing employees from the United States?

a.
```
proc print data=orionx.'UnitedStates';
run;
```

b.
```
proc print data=orionx.'UnitedStates$';
run;
```

c.
```
proc print data=orionx.'UnitedStates'n;
run;
```

d.
```
proc print data=orionx.'UnitedStates$'n;
run;
```

7.01 Quiz – Correct Answer

Which PROC PRINT step displays the worksheet containing employees from the United States?

a.
```
proc print data=orionx.'UnitedStates';
run;
```

b.
```
proc print data=orionx.'UnitedStates$';
run;
```

c.
```
proc print data=orionx.'UnitedStates'n;
run;
```

d.
```
proc print data=orionx.'UnitedStates$'n;
run;
```

Reading Excel Data

This demonstration illustrates reading from an Excel workbook using SAS Enterprise Guide and the SAS windowing environment.

p107d01

Business Scenario

Create a SAS data set using a Microsoft Excel workbook as input.

Considerations

Use a SAS/ACCESS LIBNAME statement to read the **Australia$** worksheet and create a temporary data set.

The new data set should include the following:

- only the employees with **Rep** in their job title
- a **Bonus** variable that is 10% of **Salary**
- permanent labels and formats

7.02 Poll

A DROP or KEEP statement can be used to control which worksheet columns are written to the new data set.

- ❍ True
- ❍ False

7.02 Poll – Correct Answer

A DROP or KEEP statement can be used to control which worksheet columns are written to the new data set.

- ❍ True
- ❍ False

Creating a SAS Data Set

This demonstration illustrates reading from a Microsoft Excel workbook to generate a report and create a SAS data set.

p107d02

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 7: Reading Spreadsheet and Database Data

7.1 Reading Spreadsheet Data

7.2 Reading Database Data

Objectives

- Assign a libref to an Oracle database using a SAS/ACCESS LIBNAME statement.
- Access an Oracle table using a SAS two-level name.
- Create a SAS data set that contains a subset of an Oracle table.

Business Scenario

The Northeast Sales Manager requested a report listing supervisors from New York and New Jersey. The input data is in an Oracle database.

Business Scenario

Use SAS/ACCESS to read the tables within the database as if they were SAS data sets.

SAS/ACCESS LIBNAME Statement

The SAS/ACCESS LIBNAME statement assigns a libref to a relational database.

libref *engine name* *options*

```
libname oralib oracle user=edu001 pw=edu001
                      path=dbmssrv schema=educ;
```

LIBNAME *libref engine* <*SAS/ACCESS options*>**;**

This example uses the LIBNAME statement supported by the SAS/ACCESS Interface to Oracle.

Accessing an Oracle Database

The Oracle database is treated like a SAS library.

Any table in this Oracle database can be referenced using a SAS two-level name.

7.03 Multiple Choice Poll

Which of the following statements will read the Oracle table **supervisors** as if it were a SAS data set?

a. input oralib.supervisors;

b. input 'oralib.supervisors$'n;

c. set oralib.supervisors;

d. set 'oralib.supervisors$'n;

7.03 Multiple Choice Poll – Correct Answer

Which of the following statements will read the Oracle table **supervisors** as if it were a SAS data set?

a. input oralib.supervisors;

b. input 'oralib.supervisors$'n;

c. set oralib.supervisors;

d. set 'oralib.supervisors$'n;

Printing an Oracle Table

```
libname oralib oracle
        user=edu001 pw=edu001
        path=dbmssrv schema=educ;

proc print data=oralib.supervisors;
   where state in ('NY' 'NJ');
run;

libname oralib clear;
```

Remember to release the database and associated resources by submitting a LIBNAME statement that contains the CLEAR option.

Viewing the Output

Partial PROC PRINT Output

Obs	EMPID	STATE	JOBCATEGORY
1	1834	NY	BC
2	1433	NJ	FA
3	1983	NY	FA
4	1420	NJ	ME
5	1882	NY	ME

Creating a SAS Data Set

```
libname oralib oracle user=edu001
        pw=edu001 path=dbmssrv schema=educ;

data nynjsup;
   set oralib.supervisors;
   where state in ('NY' 'NJ');
run;

proc print data=nynjsup;
run;

libname oralib clear;
```

Viewing the Output

PROC PRINT Output

Obs	EMPID	STATE	JOBCATEGORY
1	1834	NY	BC
2	1433	NJ	FA
3	1983	NY	FA
4	1420	NJ	ME
5	1882	NY	ME

Questions ???

Chapter
Review

1. In this SAS/ACCESS LIBNAME statement, what does **oracle** represent?

```
libname sports oracle
        user=edu001 pw=edu001
        path=dbmssrv schema=educ;
```

a. libref
b. option
c. table name
d. engine name

1. In this SAS/ACCESS LIBNAME statement, what does **oracle** represent?

```
libname sports oracle
        user=edu001 pw=edu001
        path=dbmssrv schema=educ;
```

a. libref
b. option
c. table name
d. engine name

2. Which PROC step successfully prints a list of all data sets in the **orionx** library without printing descriptor portions for the individual data sets?

a. proc contents data=orionx.nods _all_;
   run;
b. proc contents data=orionx._all_ nods;
   run;
c. proc print data=orionx._all_ noobs;
   run;
d. proc print data=orionx._all_ nods;
   run;

2. Which PROC step successfully prints a list of all data sets in the **orionx** library without printing descriptor portions for the individual data sets?

a. proc contents data=orionx.nods _all_;
run;

b. proc contents data=orionx._all_ nods;
run;

c. proc print data=orionx._all_ noobs;
run;

d. proc print data=orionx._all_ nods;
run;

3. When you use the SAS/ACCESS LIBNAME statement to assign a libref to a Microsoft Excel workbook, you can view the workbook using SAS Explorer or Microsoft Excel.

- ❍ True
- ❍ False

3. When you use the SAS/ACCESS LIBNAME statement to assign a libref to a Microsoft Excel workbook, you can view the workbook using SAS Explorer or Microsoft Excel.

- ❍ True
- ❍ False

4. When you use the SAS/ACCESS LIBNAME statement to assign a libref to a Microsoft Excel workbook, SAS treats each worksheet within the workbook as a library.

- ❍ True
- ❍ False

4. When you use the SAS/ACCESS LIBNAME statement to assign a libref to a Microsoft Excel workbook, SAS treats each worksheet within the workbook as a library.

- ❍ True
- ❍ False

5. What does the program shown here create?

```
libname sales excel 'c:\mydata\annual.xls';

data sales.qtr1_2011;
   set sasdata.qtr1_2011;
run;

data sales.qtr2_2011;
   set sasdata.qtr2_2011;
run;
```

a. a data set named **sales.qtr1_2011** and a data set named **sales.qtr2_2011**
b. an Excel workbook named **sales.qtr1_2011** and an Excel workbook named **sales.qtr2_2011**
c. an Excel workbook named **annual** that contains two worksheets, **qtr1_2011** and **qtr2_2011**
d. an Excel workbook named **sales** that contains two worksheets, **qtr1_2011** and **qtr2_2011**

5. What does the program shown here create?

```
libname sales excel 'c:\mydata\annual.xls';

data sales.qtr1_2011;
   set sasdata.qtr1_2011;
run;

data sales.qtr2_2011;
   set sasdata.qtr2_2011;
run;
```

a. a data set named **sales.qtr1_2011** and a data set named **sales.qtr2_2011**

b. an Excel workbook named **sales.qtr1_2011** and an Excel workbook named **sales.qtr2_2011**

c. an Excel workbook named **annual** that contains two worksheets, **qtr1_2011** and **qtr2_2011**

d. an Excel workbook named **sales** that contains two worksheets, **qtr1_2011** and **qtr2_2011**

6. What statement is used to read an Oracle table in a DATA step?

a. DATA statement
b. WHERE statement
c. SET statement
d. assignment statement

6. What statement is used to read an Oracle table in a DATA step?

a. DATA statement
b. WHERE statement
c. SET statement
d. assignment statement

7. A SAS name literal is a string that contains one or more special characters, enclosed in quotation marks, followed by the letter n.

- ❍ True
- ❍ False

7. A SAS name literal is a string that contains one or more special characters, enclosed in quotation marks, followed by the letter n.

- ❍ True
- ❍ False

8. When a date value is read from a worksheet, it is converted automatically to a SAS date. Which SAS format is used to display the value?

   a. MMDDYY8.
   b. MMDDYY10.
   c. DATE7.
   d. DATE9.

8. When a date value is read from a spreadsheet, it is converted automatically to a SAS date. Which SAS format is used to display the value?

a. MMDDYY8.
b. MMDDYY10.
c. DATE7.
d. DATE9.

9. When a SAS data set is created from a spreadsheet, the spreadsheet column headings are always stored as which of the following?

   a. variable names
   b. labels
   c. formats
   d. descriptor

9. When a SAS data set is created from a spreadsheet, the spreadsheet column headings are always stored as which of the following?

a. variable names
b. labels
c. formats
d. descriptor

10. A WHERE statement or a subsetting IF can be used to subset a Microsoft Excel worksheet.

- ❍ True
- ❍ False

10. A WHERE statement or a subsetting IF can be used to subset a Microsoft Excel worksheet.

- ❍ True
- ❍ False

Chapter 8: Reading Raw Data Files

8.1 Introduction to Reading Raw Data Files

8.2 Reading Standard Delimited Data

8.3 Reading Nonstandard Delimited Data

8.4 Handling Missing Data

Chapter 8: Reading Raw Data Files

8.1 Introduction to Reading Raw Data Files

8.2 Reading Standard Delimited Data

8.3 Reading Nonstandard Delimited Data

8.4 Handling Missing Data

Objectives

- Identify types of raw data files and input styles.
- Define the terms standard and nonstandard data.

Business Scenario

Information about Orion Star sales employees from Australia and the United States is stored in a raw data file.

Programmers need to be able to identify the layout and type of information in the raw data file.

Raw Data Files

A raw data file is also known as a *flat file*.

- They are text files that contain one record per line.
- A record typically contains multiple fields.
- Flat files do not have internal metadata.
- External documentation, known as a *record layout*, should exist.
- A record layout describes the fields and locations within each record.

Raw Data Files

Fields in a raw data file can be delimited or arranged in fixed columns.

Delimited File

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
```

Fixed Column File

```
          1    1    2    2    3    3    4    4    5    5    6
1---5----0----5----0----5----0----5----0----5----0----5----0--
120102Tom        Zhou            Sales Manager       108255AU
120103Wilson     Dawes           Sales Manager        87975AU
120121Irenie     Elvish          Sales Rep. II        26600AU
120122Christina  Ngan            Sales Rep. II        27475AU
```

Fields in Raw Data Files

In order for SAS to read a raw data file, you must specify the following information about each field:

- the location of the data value in the record
- the name of the SAS variable in which to store the data
- the type of the SAS variable

Reading Raw Data Files

There are different techniques, or *input styles*, for reading raw data files in SAS.

Input Style	Used for Reading
Column Input	Standard data in fixed columns
Formatted Input	Standard and nonstandard data in fixed columns
List Input	Standard and nonstandard data separated by blanks or some other delimiter

Standard and Nonstandard Data

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

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

Standard Numeric Data

58

67.23 | -23

5.67E5 | 00.99

1.2E-2

Nonstandard Numeric Data

(23) | $67.23

5,823 | 01/12/2010

12May2009

8.01 Multiple Answer Poll

What type of raw data files do you read?

a. delimited
b. fixed column
c. both delimited and fixed column
d. I do not read raw data files.

Chapter 8: Reading Raw Data Files

8.1 Introduction to Reading Raw Data Files

8.2 Reading Standard Delimited Data

8.3 Reading Nonstandard Delimited Data

8.4 Handling Missing Data

Objectives

- Use list input to create a SAS data set from a delimited raw data file.
- Examine the compilation and execution phases of the DATA step when reading a raw data file.
- Explicitly define the length of a variable.
- Examine behavior when a data error is encountered.

Business Scenario

Information about Orion Star sales employees is stored in a comma-delimited raw data file. The file contains both standard and nonstandard data fields.

List Input

Use list input to read delimited raw data files.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

- SAS considers a space (blank) to be the default delimiter.
- Both standard and nonstandard data can be read.
- Fields must be read sequentially, left to right.

8.02 Quiz

Which fields in this file can be read as standard numeric values?

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

8.02 Quiz – Correct Answer

Which fields in this file can be read as standard numeric values?

The employee ID and salary. The date fields are nonstandard and require special processing.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

Reading a Delimited Raw Data File

Use *INFILE* and *INPUT statements* in a DATA step to read a raw data file.

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $
         Last_Name $ Gender $ Salary
         Job_Title $ Country $;
run;
```

DATA *output-data-set*;
    INFILE "*raw-data-file***"** <**DLM='***delimiter***'**>;
    INPUT *variable* <**$**> *variable* <**$**> … ;
RUN;

p108d01

INFILE Statement

The INFILE statement identifies the raw data file to be read.

```
INFILE "&path\sales.csv" DLM=',';
```

INFILE "*raw-data-file***" <DLM='***delimiter***'>;**

- A full path is recommended.
- Using the **&path** macro variable reference makes the program more flexible.
- The DLM= option specifies alternate delimiters.

Be sure to use double quotation marks when referencing a macro variable within a quoted string.

INPUT Statement

The INPUT statement reads the data fields sequentially, left to right. Standard data fields require only a variable name and type.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
```

```
input Employee_ID First_Name $ Last_Name $
      Gender $ Salary Job_Title $ Country $;
```

INPUT *variable* <$> *variable* <$> …;

- The optional dollar sign indicates a character variable.
- Default length for ***all*** variables is eight bytes, regardless of type.

Viewing the Log

Partial SAS Log

```
249  data work.subset;
250     infile "&path\sales.csv" dlm=',';
251     input Employee_ID First_Name $ Last_Name $
252           Gender $ Salary Job_Title $ Country $;
253  run;

NOTE: The infile "s:\workshop\sales.csv" is:
      Filename=s:\workshop\sales.csv,
      RECFM=V,LRECL=256,File Size (bytes)=11340

NOTE: 165 records were read from the infile "s:\workshop\sales.csv".
      The minimum record length was 61.
      The maximum record length was 80.
NOTE: The data set WORK.SUBSET has 165 observations and 7 variables.
```

p108d01

Viewing the Output

```
proc print data=work.subset noobs;
run;
```

Partial PROC PRINT Output

Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
120121	Irenie	Elvish	F	26600	Sales Re	AU
120122	Christin	Ngan	F	27475	Sales Re	AU
120123	Kimiko	Hotstone	F	26190	Sales Re	AU

Some character values are truncated.

p108d01

Business Scenario

It is important to understand the processing that occurs when a DATA step reads a raw data file.

Compilation Phase

During compilation, SAS does the following:

- scans the step for syntax errors
- translates each statement into machine language
- creates an *input buffer* to hold one record at a time from the raw data file

- creates the program data vector (PDV) to hold one observation
- creates the descriptor portion of the output data set

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer

									1										2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

...

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

...

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer

									1										2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

PDV

Employee _ID N 8	First_ Name $ 8

With list input, the default length for character variables is eight bytes.

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer

									1										2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

PDV

Employee _ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_ Title $ 8	Country $ 8

...

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

Employee _ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_ Title $ 8	Country $ 8

Descriptor Portion of **work.subset**

Employee _ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_ Title $ 8	Country $ 8

8.03 Multiple Choice Poll

Which statement is true?

a. An input buffer is created only if you are reading data from a raw data file.
b. The PDV at compile time holds the variable name, type, byte size, and initial value.
c. The descriptor portion is the first item that is created at compile time.

8.03 Multiple Choice Poll – Correct Answer

Which statement is true?

a. An input buffer is created only if you are reading data from a raw data file.

b. The PDV at compile time holds the variable name, type, byte size, and initial value.

c. The descriptor portion is the first item that is created at compile time.

DATA Step Processing

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

Initialize PDV

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

									1										2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

									1										2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS reads a record into the input buffer.

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS scans until it reaches a delimiter.

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $

run
```

The value is converted from text to a floating-point numeric value and copied to the PDV.

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS skips the delimiter and scans to the next delimiter.

Input Buffer

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

The text value is copied to the PDV without conversion.

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102	Tom			.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

These actions continue for all variables in the INPUT statement.

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

...

Execution

Here is the output data set after the first iteration of the DATA step.

work.subset

Employee _ID	First _Name	Last _Name	Gender	Salary	Job _Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

Reinitialize PDV

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,	

All variables in the PDV are reinitialized.

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

	1									2														
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

	1									2														
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
	1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
120103	Wilson	Dawes	M	87975	Sales Ma	AU

...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

work.subset

Employee _ID	First _Name	Last _Name	Gender	Salary	Job _Title	County
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
						...

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
   infile "fpath\sales.csv"
   input Employee_ID First_Name $
         Last_Name $ Gender $
         Salary Job_Title $
         Country $;
run;
```

Continue until EOF

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee _ID N 8	First _ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _ Title $ 8	Country $ 8
.				.		

8.04 Multiple Choice Poll

Which statement is true of a DATA step when reading from a raw data file?

a. Data is read from the raw data file into the PDV.
b. The size of the input buffer adjusts automatically based on the length of the input record.
c. At the bottom of the DATA step, the contents of the PDV are output to the output SAS data set.

8.04 Multiple Choice Poll – Correct Answer

Which statement is true of a DATA step when reading from a raw data file?

a. Data is read from the raw data file into the PDV.
b. The size of the input buffer adjusts automatically based on the length of the input record.
c. At the bottom of the DATA step, the contents of the PDV are output to the output SAS data set.

Viewing the Output

```
proc print data=work.subset noobs;
run;
```

Partial PROC PRINT Output

Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
120121	Irenie	Elvish	F	26600	Sales Re	AU
120122	Christin	Ngan	F	27475	Sales Re	AU
120123	Kimiko	Hotstone	F	26190	Sales Re	AU
120124	Lucian	Daymond	M	26480	Sales Re	AU
120125	Fong	Hofmeist	M	32040	Sales Re	AU

Some character values are truncated.

p108d01

LENGTH Statement

The *LENGTH statement* defines the type and length of a variable.

LENGTH *variable(s)* **$** *length*;

```
data work.subset;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Put the LENGTH statement before the INPUT statement.

p108d02

Compilation

```
data work.subset;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Attributes are based on the LENGTH statement.

PDV

First_ Name $ 12	Last _Name $ 18	Gender $ 1	Job_Title $ 25	Country $ 2

...

Compilation

```
data work.subset;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

First_ Name $ 12	Last _Name $ 18	Gender $ 1	Job_Title $ 25	Country $ 2	Employee_ ID N 8	Salary N 8

Viewing the Output

```
proc print data=work.subset noobs;
run;
```

Partial PROC PRINT Output

First_ Name	Last_Name	Gender	Job_Title	Country	Employee_ ID	Salary
Tom	Zhou	M	Sales Manager	AU	120102	108255
Wilson	Dawes	M	Sales Manager	AU	120103	87975
Irenie	Elvish	F	Sales Rep. II	AU	120121	26600
Christina	Ngan	F	Sales Rep. II	AU	120122	27475
Kimiko	Hotstone	F	Sales Rep. I	AU	120123	26190

The character values are no longer truncated, but the order of the variables has changed.

p108d02

8.05 Quiz

Suppose you want the order of the variables to match the order of the fields. You can include the numeric variables in the LENGTH statement. Which of the following produces the correct results?

a.
```
length Employee_ID First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary Job_Title $ 25
       Country $ 2;
```

b.
```
length Employee_ID 8 First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary 8 Job_Title $ 25
       Country $ 2;
```

8.05 Quiz – Correct Answer

Suppose you want the order of the variables to match the order of the fields. You can include the numeric variables in the LENGTH statement. Which of the following produces the correct results?

a.
```
length Employee_ID First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary Job_Title $ 25
       Country $ 2;
```

b.
```
length Employee_ID 8 First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary 8 Job_Title $ 25
       Country $ 2;
```

Using a LENGTH Statement

The LENGTH statement identifies the character variables, so dollar signs can be omitted from the INPUT statement.

```
data work.subset;
   length Employee_ID 8 First_Name $ 12
          Last_Name $ 18 Gender $ 1
          Salary 8 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name Last_Name
         Gender Salary Job_Title Country;
run;
```

p108d03

Viewing the Output

Display the variables in creation order.

```
proc contents data=work.subset varnum;
run;
```

Partial PROC CONTENTS Output

```
Variables in Creation Order

 #    Variable        Type    Len

 1    Employee_ID     Num       8
 2    First_Name      Char     12
 3    Last_Name       Char     18
 4    Gender          Char      1
 5    Salary          Num       8
 6    Job_Title       Char     25
 7    Country         Char      2
```

Viewing the Output

Partial PROC PRINT Output

Employee_ ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country
120102	Tom	Zhou	M	108255	Sales Manager	AU
120103	Wilson	Dawes	M	87975	Sales Manager	AU
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU
120122	Christina	Ngan	F	27475	Sales Rep. II	AU
120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU
120124	Lucian	Daymond	M	26480	Sales Rep. I	AU
120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU

Questions ???

Business Scenario

A raw data file contains information about Orion Star sales employees. It includes some invalid data values.

8.06 Quiz

What problems do you see with the data values for the last two data fields, **Salary** and **Country**?

Partial **sales3inv.csv**

```
120102,Tom,Zhou,Manager,108255,AU
120103,Wilson,Dawes,Manager,87975,AU
120121,Irenie,Elvish,Rep. II,26600,AU
120122,Christina,Ngan,Rep. II,n/a,AU
120123,Kimiko,Hotstone,Rep. I,26190,AU
120124,Lucian,Daymond,Rep. I,26480,12
120125,Fong,Hofmeister,Rep. IV,32040,AU
```

8.06 Quiz – Correct Answer

What problems do you see with the data values for the last two data fields, **Salary** and **Country**?

Partial **sales3inv.csv**

```
120102,Tom,Zhou,Manager,108255,AU
120103,Wilson,Dawes,Manager,87975,AU
120121,Irenie,Elvish,Rep. II,26600,AU
120122,Christina,Ngan,Rep. II,n/a,AU
120123,Kimiko,Hotstone,Rep. I,26190,AU
120124,Lucian,Daymond,Rep. I,26480,12
120125,Fong,Hofmeister,Rep. IV,32040,AU
```

Reading a Raw Data File with Data Errors

```
data work.sales;
   infile "&path\sales3inv.csv" dlm=',';
   input Employee_ID First $ Last $
         Job_Title $ Salary Country $;
run;

proc print data=work.sales;
run;
```

Salary is defined as numeric and **Country** as character.

p108d04

Viewing the Output

Partial PROC PRINT Output

Obs	Employee_ ID	First	Last	Job_ Title	Salary	Country
1	120102	Tom	Zhou	Manager	108255	AU
2	120103	Wilson	Dawes	Manager	87975	AU
3	120121	Irenie	Elvish	Rep. II	26600	AU
4	120122	Christina	Ngan	Rep. II	.	AU
5	120123	Kimiko	Hotstone	Rep. I	26190	AU
6	120124	Lucian	Daymond	Rep. I	26480	12
7	120125	Fong	Hofmeister	Rep. IV	32040	AU

- A missing value was stored in **Salary** for the input value *n/a*.
- The value *12* was successfully stored in **Country**.
- A data error occurred on observation 4 but not on observation 6.

Viewing the Log

Partial SAS Log

```
480  data work.sales;
481     infile "&path\sales3inv.csv" dlm=',';
482     input Employee_ID First $ Last $
483           Job_Title $ Salary  Country $;
484  run;

NOTE: The infile "s:\workshop\sales3inv.csv" is:
      Filename=s:\workshop\sales3inv.csv,
      RECFM=V,LRECL=256,File Size (bytes)=1972,

NOTE: Invalid data for Salary in line 4 31-33.
RULE:     ----+----1----+----2----+----3----+----4----+----5-
4         120122,Christina,Ngan,Rep. II,n/a,AU 36
Employee_ID=120122 First=Christin Last=Ngan Job_Title=Rep. II Salary=.
Country=AU _ERROR_=1 _N_=4
NOTE: 50 records were read from the infile "s:\workshop\sales3inv.csv".
NOTE: The data set WORK.SALES has 50 observations and 6 variables.
```

A data error occurs when a data value does not match the field specification.

Data Errors

When this kind of data error occurs, the following information is written to the SAS log:

- a note describing the error
- a column ruler
- the input record
- the contents of the PDV

```
NOTE: Invalid data for Salary in line 4 31-33.
RULE:     ----+----1----+----2----+----3----+----4----+----5-
4         120122,Christina,Ngan,Rep. II,n/a,AU 36
Employee_ID=120122 First=Christin Last=Ngan Job_Title=Rep. II Salary=.
Country=AU _ERROR_=1 _N_=4
```

A missing value is assigned to the corresponding variable, and execution continues.

Data Errors

Two temporary variables are created during the processing of every DATA step:

- **_N_** is the DATA step iteration counter.
- **_ERROR_** indicates data error status.
  - *0* indicates that no data error occurred on that record.
  - *1* indicates that one or more data errors occurred on that record.

```
NOTE: Invalid data for Salary in line 4 31-33.
RULE:     ----+----1----+----2----+----3----+----4----+----5-
4         120122,Christina,Ngan,Rep. II,n/a,AU 36
Employee_ID=120122 First=Christin Last=Ngan Job_Title=Rep. II Salary=.
Country=AU _ERROR_=1 _N_=4
```

Examining Data Errors

This demonstration illustrates SAS behavior when a data error occurs when reading a raw data file.

p108d04

8.07 Multiple Choice Poll

Submit program **p108a01** and examine the log.

Which statement best describes the reason for the error?

a. The data in the raw data file is invalid.
b. The programmer incorrectly read the data.

8.07 Multiple Choice Poll – Correct Answer

Which statement best describes the reason for the error?

a. The data in the raw data file is invalid.

b. The programmer incorrectly read the data.

Partial SAS Log

```
404     input Employee_ID First $ Last;
405  run;

NOTE: Invalid data for Last in line 1 16-17.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
1         120101,Patrick,Lu,M,163040,Director,AU,18AUG1976,01JUL2003 58
Employee_ID=120101 First=Patrick Last=. _ERROR_=1 _N_=1
NOTE: Invalid data for Last in line 2 15-24.
2         120104,Kareen,Billington,F,46230,Administration Manager,au,1
      61  1MAY1954,01JAN1981 78
Employee_ID=120104 First=Kareen Last=. _ERROR_=1 _N_=2
```

Last was read as numeric but needs to be read as character.

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 8: Reading Raw Data Files

8.1 Introduction to Reading Raw Data Files

8.2 Reading Standard Delimited Data

8.3 Reading Nonstandard Delimited Data

8.4 Handling Missing Data

Objectives

- Use informats to read character data.
- Use informats to read nonstandard data.
- Subset observations and add permanent attributes.

Business Scenario

Create a temporary SAS data set by reading both standard and nonstandard values from a comma-delimited raw data file.

The new data set will contain a subset of the input data and will include permanent attributes.

Considerations

Use modified list input to read all the fields from **sales.csv**. Store the date fields as SAS dates.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

Modified List Input

This DATA step uses *modified list input.* Instead of a LENGTH statement, an informat specifies the length for each character variable.

input *variable* <**:***informat.*> …;

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name :$12.
         Last_Name :$18. Gender :$1. Salary
         Job_Title :$25. Country :$2.;
run;
```

- The **$12.** informat defines a length of 12 for **First_Name** and allows up to 12 characters to be read.
- The **:** format modifier tells SAS to read until it encounters a delimiter.

p108d05

Modified List Input

Omitting the colon modifier causes unexpected results.

Partial **sales.csv**

reads 12 characters

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
```

```
input Employee_ID First_Name $12.
      Last_Name :$18. Gender :$1. Salary
      Job_Title :$25. Country :$2.;
```

PDV

Employee_ID N 8	First_Name $ 12	Last_Name $ 18	Gender $ 1
120102	Tom,Zhou,1	08255	S

Salary N 8	Job_Title $ 25	Country $ 2
.	11AUG1973	06

Reading Nonstandard Data

An informat is ***required*** to read nonstandard numeric data.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

In this example, informats are needed to specify the style of the date fields so that they can be read and converted to SAS dates.

8.08 Quiz

A ***format*** is an instruction that tells SAS how to display data values. What formats would you specify to display a SAS date in the styles shown below?

a) 01JAN2000

b) 01/16/2000

8.08 Quiz – Correct Answer

A ***format*** is an instruction that tells SAS how to display data values. What formats would you specify to display a SAS date in the styles shown below?

a) 01JAN2000 ⇨ **DATE9.**

b) 01/16/2000 ⇨ **MMDDYY10.**

Informats for Nonstandard Data

An *informat* is an instruction that SAS uses to ***read*** data values into a variable.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

DATE. **MMDDYY.**

The informat describes the data value and tells SAS how to convert it.

SAS Informats

SAS informats have the following form:

<$><*informat*><*w*>.

$	Indicates a character informat.
informat	Names the SAS informat or user-defined informat.
w	Specifies the width or number of columns to read or specifies the length of a character variable.
.	Is required syntax.

The width is typically not used with list input because SAS will read each field until it encounters a delimiter.

SAS Informats

Selected SAS Informats for Nonstandard Numeric Values

Informat	Definition
COMMA. DOLLAR.	Reads nonstandard numeric data and removes embedded commas, blanks, dollar signs, percent signs, and dashes.
COMMAX. DOLLARX.	Reads nonstandard numeric data and removes embedded non-numeric characters; reverses the roles of the decimal point and the comma.
EUROX.	Reads nonstandard numeric data and removes embedded non-numeric characters in European currency.
$CHAR.	Reads character values and preserves leading blanks.
$UPCASE.	Reads character values and converts them to uppercase.

SAS Informats

Informats are used to read and convert raw data.

Informat	Raw Data Value	SAS Data Value
COMMA. DOLLAR.	*$12,345*	*12345*
COMMAX. DOLLARX.	*$12.345*	*12345*
EUROX.	*€12.345*	*12345*
$CHAR.	*##Australia*	*##Australia*
$UPCASE.	*au*	*AU*

The character # represents a blank space.

SAS Informats

Use date informats to read and convert dates to SAS date values.

Informat	Raw Data Value	SAS Data Value
MMDDYY.	*010160* *01/01/60* *01/01/1960* *1/1/1960*	*0*
DDMMYY.	*311260* *31/12/60* *31/12/1960*	*365*
DATE.	*31DEC59* *31DEC1959*	*-1*

8.09 Quiz

Use the SAS Help facility or documentation to investigate the **DATE*w*.** informat and answer the following questions:

a) What does the ***w*** represent?

b) What is the default width of this informat?

8.09 Quiz – Correct Answer

Use the SAS Help facility or documentation to investigate the **DATE*w*.** informat and answer the following questions:

a) What does the ***w*** represent?
the width of the input field

b) What is the default width of this informat?
The default width is 7.

Using Informats to Read Nonstandard Data

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,6/1/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,7JAN1953,1/10/1978
```

DATE.
Default: 7

MMDDYY.
Default: 6

- An informat is needed to read a nonstandard value.
- There is no need to specify a width with list input.
- Most informats have default widths.

Only the highlighted portion of the field is read if the default width is specified.

Modified List Input

The colon format modifier (:) tells SAS to read until it encounters a delimiter.

```
input Employee_ID First_Name :$12.
      Last_Name :$18. Gender :$1.
      Salary Job_Title :$25. Country :$2.
      Birth_Date :date. Hire_Date :mmddyy.;
```

INPUT *variable* <**$**> *variable* <**:***informat*> …**;**

colon format modifier

p108d06

Viewing the Log

```
37   data work.sales;
38      infile "&path\sales.csv" dlm=',';
39      input Employee_ID First_Name :$12. Last_Name :$18.
40            Gender :$1. Salary Job_Title :$25. Country :$2.
41            Birth_Date :date. Hire_Date :mmddyy.;
42   run;

NOTE: The infile "s:\workshop\sales.csv" is:
      Filename=s:\workshop\sales.csv,
      RECFM=V,LRECL=256,File Size (bytes)=11340,

NOTE: 165 records were read from the infile "s:\workshop\sales.csv".
NOTE: The data set WORK.SALES has 165 observations and 9 variables.
```

p108d06

Viewing the Output

```
proc print data=work.sales;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Gender	Job_Title	Country	Employee_ ID	Salary	Birth_ Date	Hire_ Date
1	Tom	Zhou	M	Sales Manager	AU	120102	108255	4971	12205
2	Wilson	Dawes	M	Sales Manager	AU	120103	87975	-2535	6575
3	Irenie	Elvish	F	Sales Rep. II	AU	120121	26600	-4169	6575
4	Christina	Ngan	F	Sales Rep. II	AU	120122	27475	-523	8217
5	Kimiko	Hotstone	F	Sales Rep. I	AU	120123	26190	3193	10866

p108d06

8.10 Multiple Choice Poll

A new data set should contain observations only for Australian employees. Which of the following can be used to subset the data?

a. where Country='AU';

b. if Country='AU';

c. either a or b

d. You cannot subset when reading from a raw data file.

8.10 Multiple Choice Poll – Correct Answer

A new data set should contain observations only for Australian employees. Which of the following can be used to subset the data?

a. where Country='AU';

b. if Country='AU';

c. either a or b

d. You cannot subset when reading from a raw data file.

Additional SAS Statements

Additional SAS statements can be added to perform further processing in the DATA step.

```
data work.sales;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name :$12. Last_Name :$18.
         Gender :$1. Salary Job_Title :$25. Country :$2.
         Birth_Date :date. Hire_Date :mmddyy.;
   if Country='AU';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary dollar12. Hire_Date monyy7.;
run;
```

p108d07

Viewing the Output

```
proc print data=work.sales label;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Sales Title	Salary	Date Hired
1	Tom	Zhou	Sales Manager	$108,255	JUN1993
2	Wilson	Dawes	Sales Manager	$87,975	JAN1978
3	Irenie	Elvish	Sales Rep. II	$26,600	JAN1978
4	Christina	Ngan	Sales Rep. II	$27,475	JUL1982
5	Kimiko	Hotstone	Sales Rep. I	$26,190	OCT1989

p108d07

WHERE versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
SET statement	Yes	Yes
assignment statement	No	Yes
INPUT statement	No	Yes

Idea Exchange

What factors need to be considered when reading **salary.dat**, shown below?

Partial **salary.dat**

```
          1    1    2    2    3    3
1---5----0----5----0----5----0----5
Donny 5MAY2008 25 FL $43,132.50
Margaret 20FEB2008 43 NC $65,150
Dan 1JUN2008 27 FL $40,000.00
Subash 2FEB2008 45 NC $75,750
Antonio 25MAY2008 35 FL $43,500.50
```

Using List Input: Importance of Colon Format Modifier

This demonstration illustrates the use of informats when reading nonstandard data values.

p108a02

Business Scenario

You are working on a new project, but the raw data file has not been created yet. You can include in-stream data in a DATA step.

DATALINES Statement

The DATALINES statement supplies data within a program.

```
data work.newemps;
   input First_Name $ Last_Name $
         Job_Title $ Salary :dollar8.;
datalines;
Steven Worton Auditor $40,450
Merle Hieds Trainee $24,025
Marta Bamberger Manager $32,000
;
```

```
DATALINES;
…
;
```

- DATALINES is the last statement in the DATA step and immediately precedes the first data line.
- A null statement (a single semicolon) indicates the end of the input data.

p108d08

Viewing the Output

```
proc print data=work.newemps;
run;
```

PROC PRINT Output

Obs	First_ Name	Last_ Name	Job_ Title	Salary
1	Steven	Worton	Auditor	40450
2	Merle	Hieds	Trainee	24025
3	Marta	Bamberge	Manager	32000

p108d08

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 8: Reading Raw Data Files

8.1 Introduction to Reading Raw Data Files

8.2 Reading Standard Delimited Data

8.3 Reading Nonstandard Delimited Data

8.4 Handling Missing Data

Objectives

- Use the DSD option to read consecutive delimiters as missing values.
- Use the MISSOVER option to recognize missing values at the end of a record.

Business Scenario

Orion Star programmers have discovered that some files have records with missing data in one or more fields.

Missing Values in the Middle of the Record

The records in **phone2.csv** have a contact name, phone number, and a mobile number. The phone number is missing from some of the records.

Missing data is indicated by consecutive delimiters.

phone2.csv

```
         1     1     2     2     3     3     4     4
1---5----0----5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,, (919) 271-4592
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,, (714) 644-9090
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

8.11 Quiz

- Open and submit **p108a03**.
- Examine the SAS log. How many input records were read and how many observations were created?
- Examine the report. Does it look correct?

```
data work.contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path\phone2.csv" dlm=',';
   input Name $ Phone $ Mobile $;
run;

proc print data=work.contacts noobs;
run;
```

p108a03

8.11 Quiz – Correct Answer

- How many input records were read and how many observations were created? **five read, three created**
- Examine the report. Does it look correct? **no**

```
NOTE: 5 records were read from the infile "S:\workshop\phone2.csv".
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: SAS went to a new line when INPUT statement reached past the
end of a line.
NOTE: The data set WORK.CONTACTS has 3 observations and 3 variables.
```

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	Cornelia Krahl
Karen Ballinger	(714) 344-4321	Elke Wallstab

Consecutive Delimiters in List Input

List input treats two or more consecutive delimiters as a single delimiter and not as a missing value.

phone2.csv

```
          1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,, (919) 271-4592
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,, (714) 644-9090
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

When there is missing data in a record, SAS does the following:

- loads the next record to finish the observation
- writes a note to the log

DSD Option

Use the DSD option to correctly read **phone2.csv**.

```
data work.contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path\phone2.csv" dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=work.contacts noobs;
run;
```

INFILE "*raw-data-file***"** <**DLM=**> **DSD**;

DSD Option

The DSD option in the INFILE statement does the following:

- sets the default delimiter to a comma
- treats consecutive delimiters as missing values
- enables SAS to read values with embedded delimiters if the value is surrounded by quotation marks

INFILE "*raw-data-file***"** <**DLM=**> **DSD**;

The DLM= option can be used with the DSD option but is not needed for comma-delimited files.

Viewing the Output

Adding the DSD option gives the correct results.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano		(919) 271-4592
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger		(714) 644-9090
Elke Wallstab	(910) 763-5561	(910) 545-3421

Partial SAS Log

```
NOTE: 5 records were read from the infile "S:\workshop\phone2.csv".
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and 3
variables.
```

Questions ???

Business Scenario

Orion Star programmers have discovered that some files have observations with missing data at the end of the record, so there are fewer fields in the record than specified in the INPUT statement.

Missing Values at the End of a Record

The raw data file **phone.csv** contains missing values at the end of some records.

phone.csv

```
          1     1     2                  4     4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,(919) 871-7830
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,(714) 344-4321
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

missing values

The DSD option is not appropriate because the missing data is not marked by consecutive delimiters.

MISSOVER Option

The *MISSOVER option* prevents SAS from loading a new record when the end of the current record is reached.

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path\phone.csv" dlm=',' missover;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

INFILE "*raw-data-file***"** <**DLM=**> **MISSOVER;**

If SAS reaches the end of a record without finding values for all fields, variables without values are set to missing.

p108d10

Viewing the Output

Partial SAS Log

```
NOTE: 5 records were read from the infile "S:\workshop\phone.csv".
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and 3
variables.
```

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger	(714) 344-4321	
Elke Wallstab	(910) 763-5561	(910) 545-3421

INFILE Options

INFILE "*raw-data-file*" <DLM=> <DSD> <MISSOVER>;

Option	Description
DLM=	Specifies an alternate delimiter.
DSD	Sets the default delimiter to a comma, treats consecutive delimiters as missing values, and allows embedded delimiters when the data value is enclosed in quotation marks.
MISSOVER	Sets variables to missing if the end of the record is reached before finding values for all fields.

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter Review

1. In the first iteration of this program, SAS does the following:
   - reads a record from the raw data file into the input buffer
   - scans the input buffer and copies the values to the PDV
   - writes the values to the output data set
   - reinitializes the input buffer
   - reads the next record from the raw data file

```
data work.profit;
   infile 'c:\mydata\income.csv' dlm=',';
   input Amount SalesRep $ Customer $;
run;
```

❍ True

❍ False

1. In the first iteration of this program, SAS does the following:
   - reads a record from the raw data file into the input buffer
   - scans the input buffer and copies the values to the PDV
   - writes the values to the output data set
   - reinitializes the input buffer
   - reads the next record from the raw data file.

```
data work.profit;
   infile 'c:\mydata\income.csv' dlm=',';
   input Amount SalesRep $ Customer $;
run;
```

❍ True

❍ False

2. Which INFILE statement correctly specifies the raw data file shown here?

Partial **salestotals.dat**

```
         1    1    2    2    3    3    4    4    5
1---5----0----5----0----5----0----5----0----5----0
```

```
14528*instore*06/15/2008*215.65*1650072*red
14529*online*06/15/2008*183.98*1650039*white
14530*online*06/16/2008*107.50*1650450*green
14531*instore*06/17/2008*350.78*1652903*graphite
```

a. infile 'c:\mydata\salestotals.dat';
b. infile 'c:\mydata\salestotals.dat' dlm=*;
c. infile 'c:\mydata\salestotals.dat' dlm=',';
d. infile 'c:\mydata\salestotals.dat' dlm='*';

2. Which INFILE statement correctly specifies the raw data file shown here?

Partial **salestotals.dat**

```
          1    1    2    2    3    3    4    4    5
1---5----0----5----0----5----0----5----0----5----0
14528*instore*06/15/2008*215.65*1650072*red
14529*online*06/15/2008*183.98*1650039*white
14530*online*06/16/2008*107.50*1650450*green
14531*instore*06/17/2008*350.78*1652903*graphite
```

a. infile 'c:\mydata\salestotals.dat';
b. infile 'c:\mydata\salestotals.dat' dlm=*;
c. infile 'c:\mydata\salestotals.dat' dlm=',';
d. infile 'c:\mydata\salestotals.dat' dlm='*';

3. Which of the following INPUT statements creates the data set shown here, assuming that the DATA step does not contain a LENGTH statement?

Partial SAS Data Set **customers**

Customer_ID	Last_Name	First_Name	Total_Sales
123049	Kim	Jason	545
123050	Weston	Ingrid	832

a. input Customer_ID $ Last_Name $ First_Name $ Total_Sales;

b. input customer_id $ last_name $ first_name $ total_sales;

c. input Last_Name $ First_Name $ Total_Sales Customer_ID $;

3. Which of the following INPUT statements creates the data set shown here, assuming that the DATA step does not contain a LENGTH statement?

Partial SAS Data Set **customers**

Customer_ID	Last_Name	First_Name	Total_Sales
123049	Kim	Jason	545
123050	Weston	Ingrid	832

a. input Customer_ID $ Last_Name $ First_Name $ Total_Sales;

b. input customer_id $ last_name $ first_name $ total_sales;

c. input Last_Name $ First_Name $ Total_Sales Customer_ID $;

4. The INPUT statement below correctly reads this space-delimited raw data file.

```
          1    1    2    2    3    3
1---5----0----5----0----5----0----5
Donny 5MAY2008 25 FL $43,123.50
Margaret 20FEB2008 43 NC 65,150
```

```
input name $ hired date9. age state $
      salary comma10.;
```

- ❍ True
- ❍ False

4. The INPUT statement below correctly reads this space-delimited raw data file.

```
          1    1    2    2    3    3
1---5----0----5----0----5----0----5
Donny 5MAY2008 25 FL $43,123.50
Margaret 20FEB2008 43 NC 65,150
```

```
input name $ hired date9. age state $
      salary comma10.;
```

- ❍ True
- ❍ False

5. By default, SAS creates character variables with a length of _____ bytes for list input.

a. 6
b. 8
c. 10
d. 12

5. By default, SAS creates character variables with a length of _____ bytes for list input.

a. 6
b. 8
c. 10
d. 12

6. Which of the following values can SAS store in a character variable that has a length of 8 bytes?

   a. Sales Manager
   b. Regional Manager
   c. 12036578
   d. $123,293.50
   e. 06/15/2008

6. Which of the following values can SAS store in a character variable that has a length of 8 bytes?

   a. Sales Manager
   b. Regional Manager
   c. 12036578
   d. $123,293.50
   e. 06/15/2008

7. To explicitly define the length of a variable read from a raw data file, you use a LENGTH statement after the INPUT statement in a DATA step.

- ❍ True
- ❍ False

7. To explicitly define the length of a variable read from a raw data file, you use a LENGTH statement after the INPUT statement in a DATA step.

- ❍ True
- ❍ False

8. Which INFILE statement correctly specifies the raw data file shown here?

Partial **sales.dat**

```
          1    1    2    2    3    3    4    4    5
1---5----0----5----0----5----0----5----0----5----0
14528*instore*06/15/2008*215.65**red
14529*online*06/15/2008*183.98*1650039*white
14530**06/16/2008*107.50*1650450*green
14531*instore*06/17/2008*350.78*1652903*graphite
```

a. infile 'c:\mydata\sales.dat';
b. infile 'c:\mydata\sales.dat' dsd dlm='*';
c. infile 'c:\mydata\sales.dat' dlm=*;
d. infile 'c:\mydata\sales.dat' dlm='*';

8. Which INFILE statement correctly specifies the raw data file shown here?

Partial **sales.dat**

```
          1     1     2     2     3     3     4     4     5
1---5----0----5----0----5----0----5----0----5----0----5----0
14528*instore*06/15/2008*215.65**red
14529*online*06/15/2008*183.98*1650039*white
14530**06/16/2008*107.50*1650450*green
14531*instore*06/17/2008*350.78*1652903*graphite
```

a. infile 'c:\mydata\sales.dat';
b. infile 'c:\mydata\sales.dat' dsd dlm='*';
c. infile 'c:\mydata\sales.dat' dlm=*;
d. infile 'c:\mydata\sales.dat' dlm='*';

9. Which of the following statements specifies in-stream data, or the lines of data that you enter directly in a DATA step?

   a. DATALINES
   b. INFILE
   c. INPUT
   d. INSTREAM

9. Which of the following statements specifies in-stream data, or the lines of data that you enter directly in a DATA step?

   a. DATALINES
   b. INFILE
   c. INPUT
   d. INSTREAM

10. Which of the following statements cannot be used in a DATA step that reads a raw data file as input?

a. KEEP
b. IF
c. FORMAT
d. WHERE

10. Which of the following statements cannot be used in a DATA step that reads a raw data file as input?

a. KEEP
b. IF
c. FORMAT
d. WHERE

Chapter 9: Manipulating Data

9.1 Using SAS Functions

9.2 Conditional Processing

Chapter 9: Manipulating Data

9.1 Using SAS Functions

9.2 Conditional Processing

Objectives

- Create data values using SAS functions.

Business Scenario

Orion Star management plans to give a $500 bonus to each employee in his or her hire month.

Considerations

Create a new data set with three new variables:

- **Bonus**, which is a constant 500
- **Compensation**, which is the sum of **Salary** and **Bonus**
- **BonusMonth**, which is the month in which the employee was hired

Considerations

Partial **orion.sales**

Employee _ID	First _Name	Last _ Name	Gender	Salary	Job_ Title	Country	Birth_ Date	Hire_ Date
120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114

Partial **work.comp**

Employee _ID	First _Name	Last _ Name	Bonus	Compensation	Bonus Month
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1
120121	Irenie	Elvish	500	27100	1

Drop **Gender**, **Salary**, **Job_Title**, **Country**, **Birth_Date**, and **Hire_Date** from **work.comp**.

9.01 Multiple Choice Poll

Which of the following statements creates a numeric variable **Bonus** with a value of 500?

a. Bonus=$500;

b. Bonus=500;

c. label Bonus='500';

d. format Bonus 500.;

Partial **work.comp**

Bonus	Compensation	Bonus Month
500	108755	6
500	88475	1
500	27100	1

9.01 Multiple Choice Poll – Correct Answer

Which of the following statements creates a numeric variable **Bonus** with a value of 500?

a. Bonus=$500;

b. Bonus=500;

c. label Bonus='500';

d. format Bonus 500.;

Partial **work.comp**

Bonus	Compensation	Bonus Month
500	108755	6
500	88475	1
500	27100	1

You use an assignment statement to set the value of the variable Bonus equal to 500. Numeric constants do not include commas or currency symbols.

SAS Functions

SAS functions can be used in an assignment statement. A *function* is a routine that accepts arguments and returns a value.

variable=function-name(*argument1*, *argument2*, …);

Some functions manipulate character values, compute descriptive statistics, or manipulate SAS date values.

- Arguments are enclosed in parentheses and separated by commas.
- A function can return a numeric or character result.

SUM Function

Use the *SUM function* to create **Compensation**. The SUM function is a descriptive statistics function that returns the sum of its arguments.

```
Compensation=sum(Salary,Bonus);
```

SUM(*argument1*,*argument2*, ...**)**

- The arguments must be numeric.
- Missing values are ignored by SUM and other descriptive statistics functions.

MONTH Function

Use the *MONTH function* to extract the month of hire from **Hire_Date**.

```
BonusMonth=month(Hire_Date);
```

MONTH(*SAS-date***)**

Other date functions can do the following:

- extract information from SAS date values
- create SAS date values

Date Functions: Extracting Values

Syntax	Description
YEAR(*SAS-date*)	Extracts the year from a SAS date and returns a four-digit year.
QTR(*SAS-date*)	Extracts the calendar quarter from a SAS date and returns a number from 1 to 4.
MONTH(*SAS-date*)	Extracts the month from a SAS date and returns a number from 1 to 12.
DAY(*SAS-date*)	Extracts the day of the month from a SAS date and returns a number from 1 to 31.
WEEKDAY(*SAS-date*)	Extracts the day of the week from a SAS date and returns a number from 1 to 7, where 1 represents Sunday.

Date Functions: Creating SAS Dates

Syntax	Description
TODAY() DATE()	Returns the current date as a SAS date value.
MDY(*month*,*day*,*year*)	Returns a SAS date value from numeric month, day, and year values.

Examples
CurrentDate=today();
y2k=mdy(01,1,2000);
NewYear=mdy(Mon,Day,2013);

Using SAS Functions

A function call can be used alone in an assignment statement.

```
BonusMonth=month(Hire_Date);
AnnivBonus=mdy(BonusMonth,15,2008);
```

A function call can be part of any SAS expression.

```
if month(Hire_Date)=12;
```

A function call can be an argument to another function.

```
AnnivBonus=mdy(month(Hire_Date),15,2012);
```

Using SAS Functions

Create **Bonus**, **Compensation**, and **BonusMonth**.

```
data work.comp;
   set orion.sales;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
run;
```

```
175  data work.comp;
176     set orion.sales;
177     Bonus=500;
178     Compensation=sum(Salary,Bonus);
179     BonusMonth=month(Hire_Date);
180  run;

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.COMP has 165 observations and 12 variables.
```

orion.sales has nine variables.

p109d01

Viewing the Output

```
proc print data=work.comp noobs;
   var Employee_ID First_Name Last_Name
       Bonus Compensation BonusMonth;
run;
```

Partial PROC PRINT Output

Employee_ID	First_ Name	Last_Name	Bonus	Compensation	Bonus Month
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1
120121	Irenie	Elvish	500	27100	1
120122	Christina	Ngan	500	27975	7
120123	Kimiko	Hotstone	500	26690	10

p109d01

9.02 Quiz

A DROP statement has been added to this DATA step. Will the program calculate **Compensation** and **BonusMonth** correctly?

```
data work.comp;
   set orion.sales;
   drop Gender Salary Job_Title Country
        Birth_Date Hire_Date;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
run;
```

p109a01

9.02 Quiz – Correct Answer

A DROP statement has been added to this DATA step. Will the program calculate **Compensation** and **BonusMonth** correctly?

```
23   data work.comp;
24      set orion.sales;
25      drop Gender Salary Job_Title Country
26               Birth_Date Hire_Date;
27      Bonus=500;
28      Compensation=sum(Salary,Bonus);
29      BonusMonth=month(Hire_Date);
30   run;
NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.COMP has 165 observations and 6 variables.
```

Yes. A drop flag is set for the dropped variables, but the variables are in the PDV and therefore available for processing. DROP is a compile-time only statement.

Viewing the Output

```
proc print data=work.comp noobs;
run;
```

Partial PROC PRINT Output

Employee_ID	First_ Name	Last_Name	Bonus	Compensation	Bonus Month
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1
120121	Irenie	Elvish	500	27100	1
120122	Christina	Ngan	500	27975	7
120123	Kimiko	Hotstone	500	26690	10

p109a01

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 9: Manipulating Data

9.1 Using SAS Functions

9.2 Conditional Processing

Objectives

- Process data conditionally using IF-THEN/ELSE statements.
- Execute multiple statements conditionally using DO and END statements.
- Control the length of character variables using the LENGTH statement.

Business Scenario

Orion Star management plans to give each sales employee a bonus based on his or her job title.

Considerations

Create a new data set, **work.comp**, using **orion.sales** as input. Include a new variable, **Bonus**, with a value based on **Job_Title**.

Job_Title	Bonus
Sales Rep. IV	1000
Sales Manager	1500
Senior Sales Manager	2000
Chief Sales Officer	2500

IF-THEN Statements

The IF-THEN statement executes a SAS statement for observations that meet a specific condition.

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   ...
run;
```

IF *expression* **THEN** *statement*;

- *expression* defines a condition.
- *statement* can be any executable SAS statement.
- If *expression* is true, then *statement* executes.

Conditional Processing

The value assigned to **Bonus** is determined by testing for various values of **Job_Title**.

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	.

p109d02
…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

false

PDV

Employee_ID	Last_Name	…	Job_Title	Bonus
120102	Zhou	…	Sales Manager	.

…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales R          V' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

true

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	.

…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales M[illegible]r' then
      Bonus=1500;
   if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

false

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Senior Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

false

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

…

Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   if Job_Title='Sales Manager' then
      Bonus=1500;
   if Job_Title='Sales Manager'
      then Bonus=2000;
   if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

Employee_ID	Last_Name	…	Job_Title	Bonus
120102	Zhou	…	Sales Manager	1500

Conditional Processing

```
data work.comp;
    se                    s;
    if Job_Title='Sales Rep. IV' then
       Bonus=1000;
    if Job_Title='Sales Manager' then
       Bonus=1500;
    if Job_Title='Senior Sales Manager'
       then Bonus=2000;
    if Job_Title='Chief Sales Officer'
       then Bonus=2500;
run;
```

Continue until EOF

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

Viewing the Output

```
proc print data=work.comp;
   var Last_Name Job_Title Bonus;
run;
```

Partial PROC PRINT Output

Obs	Last_Name	Job_Title	Bonus
1	Zhou	Sales Manager	1500
2	Dawes	Sales Manager	1500
3	Elvish	Sales Rep. II	.
4	Ngan	Sales Rep. II	.
5	Hotstone	Sales Rep. I	.
6	Daymond	Sales Rep. I	.
7	Hofmeister	Sales Rep. IV	1000
8	Denny	Sales Rep. II	.
9	Clarkson	Sales Rep. II	.
10	Kletschkus	Sales Rep. IV	1000
11	Roebuck	Sales Rep. III	.
12	Lyon	Sales Rep. I	.

p109d02

9.03 Multiple Choice Poll

In the previous program, is it possible for more than one condition to be true for a single observation?

a. Yes, more than one condition can be true.

b. No, the conditions are mutually exclusive, so only one condition can be true.

9.03 Multiple Choice Poll – Correct Answer

In the previous program, is it possible for more than one condition to be true for a single observation?

a. Yes, more than one condition can be true.

b. No, the conditions are mutually exclusive, so only one condition can be true.

For each observation there is only one value for Job_Title. If that value matches one of the conditions, then it cannot match any other condition.

Using the ELSE Statement

Use the *ELSE statement* when testing mutually exclusive conditions.

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

IF *expression* **THEN** *statement*;
<ELSE IF *expression* **THEN** *statement*;**>**
<ELSE IF *expression* **THEN** *statement*;**>**

p109d03

Conditional Processing

When an expression is true, the associated statement is executed and subsequent ELSE statements are skipped.

IF-THEN Statements

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	.

p109d03

…

IF-THEN Statements

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

false

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	.

…

IF-THEN Statements

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. I[illegible]en
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

true

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	.

…

IF-THEN Statements

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

IF-THEN Statements

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. IV' then
      Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

IF-THEN Statements

```
data work.comp;
    set sales;
    if Job_Title='Sales Rep. IV' then
       Bonus=1000;
    else if Job_Title='Sales Manager' then
       Bonus=1500;
    else if Job_Title='Senior Sales Manager'
       then Bonus=2000;
    else if Job_Title='Chief Sales Officer'
       then Bonus=2500;
run;
```

Continue until EOF

PDV

Employee_ID	Last_Name
120102	Zhou

…

Job_Title	Bonus
Sales Manager	1500

Viewing the Output

```
proc print data=work.comp;
   var Last_Name Job_Title Bonus;
run;
```

Partial PROC PRINT Output

Obs	Last_Name	Job_Title	Bonus
1	Zhou	Sales Manager	1500
2	Dawes	Sales Manager	1500
3	Elvish	Sales Rep. II	.
4	Ngan	Sales Rep. II	.
5	Hotstone	Sales Rep. I	.
6	Daymond	Sales Rep. I	.
7	Hofmeister	Sales Rep. IV	1000
8	Denny	Sales Rep. II	.
9	Clarkson	Sales Rep. II	.
10	Kletschkus	Sales Rep. IV	1000

p109d03

Business Scenario: Part 2

Orion Star management wants to modify the bonus plan as defined below.

Job_Title	Bonus
Sales Rep. III	1000
Sales Rep. IV	1000
Sales Manager	1500
Senior Sales Manager	2000
Chief Sales Officer	2500
All other titles	500

Using Conditional Processing

```
data work.comp;
   set orion.sales;
   if Job_Title='Sales Rep. III' or
      Job_Title='Sales Rep. IV' then
         Bonus=1000;
   else if Job_Title='Sales Manager' then
      Bonus=1500;
   else if Job_Title='Senior Sales Manager'
      then Bonus=2000;
   else if Job_Title='Chief Sales Officer'
      then Bonus=2500;
   else Bonus=500;
run;
```

compound condition

IF *expression* **THEN** *statement*;
<**ELSE IF** *expression* **THEN** *statement*;>
<…>
<**ELSE** *statement*;>

p109d04

Conditional Processing

An optional final ELSE statement gives an alternative action if none of the conditions are true.

Viewing the Output

```
proc print data=work.comp;
   var Last_Name Job_Title Bonus;
run;
```

Partial PROC PRINT Output

Obs	Last_Name	Job_Title	Bonus
1	Zhou	Sales Manager	1500
2	Dawes	Sales Manager	1500
3	Elvish	Sales Rep. II	500
4	Ngan	Sales Rep. II	500
5	Hotstone	Sales Rep. I	500
6	Daymond	Sales Rep. I	500
7	Hofmeister	Sales Rep. IV	1000
8	Denny	Sales Rep. II	500
9	Clarkson	Sales Rep. II	500
10	Kletschkus	Sales Rep. IV	1000

p109d04

Business Scenario

Orion Star managers are considering a country-based bonus. Create a new SAS data set named **work.bonus** using **orion.sales** as input. The value of the new variable, **Bonus**, is based on **Country**.

IF-THEN/ELSE Statements

If **orion.sales** has been validated and ***only*** includes the **Country** values *US* and *AU*, the conditional clause can be omitted from the ELSE statement.

```
data work.bonus;
   set orion.sales;
   if Country='US' then Bonus=500;
   else Bonus=300;
run;
```

IF *expression* **THEN** *statement*;
ELSE *statement*;

All observations not equal to *US* get a bonus of 300.

Viewing the Output

```
proc print data=work.bonus;
   var First_Name Last_Name Country Bonus;
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus
60	Billy	Plested	AU	300
61	Matsuoka	Wills	AU	300
62	Vino	George	AU	300
63	Meera	Body	AU	300
64	Harry	Highpoint	US	500
65	Julienne	Magolan	US	500
66	Scott	Desanctis	US	500
67	Cherda	Ridley	US	500
68	Priscilla	Farren	US	500
69	Robert	Stevens	US	500

p109d05

9.04 Quiz

Program **p109a02** reads **orion.nonsales**, a non-validated data set. Open and submit the program and review the results. Why is **Bonus** set to 300 in observations 125, 197, and 200?

```
data work.bonus;
   set orion.nonsales;
   if Country='US' then Bonus=500;
   else Bonus=300;
run;
```

9.04 Quiz – Correct Answer

Program **p109a02** reads **orion.nonsales**, a non-validated data set. Open and submit the program and review the results. Why is **Bonus** set to 300 in observations 125, 197, and 200?

```
data work.bonus;
   set orion.nonsales;
   if Country='US' then Bonus=500;
   else Bonus=300;
run;
```

The Country variable has some mixed case values in orion.nonsales. Observations with a country value of *US* are assigned 500. All others are assigned 300, including *us*.

Testing for Invalid Data

You can test for multiple values of **Country**.

```
data work.bonus;
   set orion.nonsales;
   if Country in ('US','us')
      then Bonus=500;
   else Bonus=300;
run;
```

You can use the UPCASE function in the expression.

```
data work.bonus;
   set orion.nonsales;
   if upcase(Country)='US'
      then Bonus=500;
   else Bonus=300;
run;
```

p109a02s

Cleaning Invalid Data

You can clean the data before checking the value.

```
data work.bonus;
   set orion.nonsales;
   Country=upcase(Country);
   if Country='US'
      then Bonus=500;
   else Bonus=300;
run;
```

It is a best practice to clean the data at the source, but in some cases, that is not possible. With this method, you are creating a clean data set.

p109d06

Questions ???

Business Scenario

Orion Star employees will receive a bonus once or twice a year. In addition to **Bonus**, add a new variable, **Freq**, that is equal to the following:

- *Once a Year* for United States employees
- *Twice a Year* for Australian employees

Once a Year

Twice a Year

IF-THEN/ELSE Statements

Only ***one*** executable statement is allowed in IF-THEN and ELSE statements.

IF *expression* **THEN** *statement*;
ELSE IF *expression* **THEN** *statement*;
ELSE *statement*;

For this business scenario, ***two*** statements must be executed for each true expression.

```
if Country='US' then
```

```
Bonus=500;
```

```
Freq='Once a Year';
```

DO Group

Multiple statements are permitted in a *DO group*.

```
data work.bonus;
   set orion.sales;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

DO group

Each DO group ends with an END statement.

p109d07

IF-THEN DO/ELSE DO Statements

Multiple statements are also permitted in an ELSE DO group.

Viewing the Output

```
proc print data=work.bonus;
   var First_Name Last_Name Country Bonus
       Freq;
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus	Freq
60	Billy	Plested	AU	300	Twice a Yea
61	Matsuoka	Wills	AU	300	Twice a Yea
62	Vino	George	AU	300	Twice a Yea
63	Meera	Body	AU	300	Twice a Yea
64	Harry	Highpoint	US	500	Once a Year
65	Julienne	Magolan	US	500	Once a Year
66	Scott	Desanctis	US	500	Once a Year
67	Cherda	Ridley	US	500	Once a Year
68	Priscilla	Farren	US	500	Once a Year
69	Robert	Stevens	US	500	Once a Year

truncation

p109d07

Compilation

```
data work.bonus;
   set orion.sales;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8

Compilation

```
data work.bonus;
   set orion.sales;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Bonus N 8

...

Compilation

```
data work.bonus;
   set orion.sales;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if ( then do;
      Bonus=500;
      Freq='Twice a Year';
   end;
run;
```

11 characters

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Bonus N 8	Freq $ 11

...

Compilation

```
data work.bonus;
   set orion.sales;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

12 characters

length does not change

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Bonus N 8	Freq $ 11

9.05 Quiz

How would you prevent **Freq** from being truncated?

9.05 Quiz – Correct Answer

How would you prevent **Freq** from being truncated?

Possible solutions:

- **Pad the first occurrence of the Freq value with blanks to be the length of the longest possible value.**
- **Switch conditional statements to place the longest value of Freq in the first conditional statement.**
- **Add a LENGTH statement to declare the byte size of the variable up front.**

Defining Character Variables

Set the length of the variable **Freq** to avoid truncation.

```
data work.bonus;
   set orion.sales;
   length Freq $ 12;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

LENGTH *variable(s)* <$> *length*;

- It is a good practice to use a LENGTH statement any time you create a new character variable.

p109d08

Compilation

```
data work.bonus;
    set orion.sales;
    length Freq $ 12;
    if Country='US' then do;
        Bonus=500;
        Freq='Once a Year';
    end;
    else if Country='AU' then do;
        Bonus=300;
        Freq='Twice a Year';
    end;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8

Compilation

```
data work.bonus;
   set orion.sales;
   length Freq $ 12;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Freq $ 12

...

Compilation

```
data work.bonus;
   set orion.sales;
   length Freq $ 12;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Freq $ 12	Bonus N 8

...

Compilation

```
data work.bonus;
   set orion.sales;
   length Freq $ 12;
   if Country='US' then do;
      Bonus=500;
      Freq='Once a Year';
   end;
   else if Country='AU' then do;
      Bonus=300;
      Freq='Twice a Year';
   end;
run;
```

length does not change

PDV

Employee_ID N 8	First_Name $ 12	...	Hire_Date N 8	Freq $ 12	Bonus N 8

Viewing the Output

```
proc print data=work.bonus;
   var First_Name Last_Name Country
       Bonus Freq;
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus	Freq
60	Billy	Plested	AU	300	Twice a Year
61	Matsuoka	Wills	AU	300	Twice a Year
62	Vino	George	AU	300	Twice a Year
63	Meera	Body	AU	300	Twice a Year
64	Harry	Highpoint	US	500	Once a Year
65	Julienne	Magolan	US	500	Once a Year
66	Scott	Desanctis	US	500	Once a Year
67	Cherda	Ridley	US	500	Once a Year
68	Priscilla	Farren	US	500	Once a Year
69	Robert	Stevens	US	500	Once a Year

no truncation

p109d08

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter
Review

1. Based on this program and the observation shown in the PDV, what variable's value will be assigned to **Amount**?

```
data payroll;
   set salaries;
   if PayClass='Monthly' then Amount=Salary;
   else if PayClass='Hourly' then do;
      Amount=HrlyWage*Hrs;
      if Hrs>40 then Msg='CHECK TIMECARD';
   end;
   else Amount=JobRate;
run;
```

PDV

Emp ID	Pay Class	Hrs	Amo unt	Job Rate	Msg
1201	Contract	30	.	.	

a. not specified

b. HrlyWage*Hrs

c. JobRate

d. Salary

1. Based on this program and the observation shown in the PDV, what variable's value will be assigned to **Amount**?

```
data payroll;
   set salaries;
   if PayClass='Monthly' then Amount=Salary;
   else if PayClass='Hourly' then do;
      Amount=HrlyWage*Hrs;
      if Hrs>40 then Msg='CHECK TIMECARD';
   end;
   else Amount=JobRate;
run;
```

PDV

Emp ID	Pay Class	Hrs	Amo unt	Job Rate	Msg
1201	Contract	30	.	.	

a. not specified

b. HrlyWage*Hrs

c. JobRate

d. Salary

2. Which of the following SAS functions returns a number from 1 to 12?

   a. YEAR(*SAS-date*)
   b. MONTH(*SAS-date*)
   c. WEEKDAY(*SAS-date*)
   d. TODAY(*SAS-date*)

2. Which of the following SAS functions returns a number from 1 to 12?

   a. YEAR(*SAS-date*)
   b. MONTH(*SAS-date*)
   c. WEEKDAY(*SAS-date*)
   d. TODAY(*SAS-date*)

3. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   drop Gender Salary Country;
   Compensation=sum(Salary,Bonus);
run;
```

a. 6
b. 7
c. 10
d. None. This program contains a logic error.

3. The data set **orion.sales** contains nine variables. Given this DATA step, how many variables does the descriptor portion of **work.comp** contain?

```
data work.comp;
   set orion.sales;
   drop Gender Salary Country;
   Compensation=sum(Salary,Bonus);
run;
```

a. 6
b. 7
c. 10
d. None. This program contains a logic error.

4. Which DATA step ensures that all observations are assigned a nonmissing value for **Bonus**?

a.
```
data work.bonus;
   set orion.sales;
   if Country='US' then Bonus=500;
   else if Country='AU' then Bonus=300;
run;
```

b.
```
data work.bonus;
   set orion.sales;
   if Country='US' then Bonus=500;
   else Bonus=300;
run;
```

4. Which DATA step ensures that all observations are assigned a nonmissing value for **Bonus**?

a.
```
data work.bonus;
   set orion.sales;
   if Country='US' then Bonus=500;
   else if Country='AU' then Bonus=300;
run;
```

b.
```
data work.bonus;
   set orion.sales;
   if Country='US' then Bonus=500;
   else Bonus=300;
run;
```

5. In the DATA step below, what is the length of the new variable, **Type**?

```
data orion.newloan;
   set orion.records;
   TotalPaid=sum(TotLoan+Interest);
   if Code='1' then Type='Fixed';
   else Type='Variable';
   length Type $ 10;
run;
```

a. 5
b. 8
c. 10
d. It depends on the first value of **Type** in **orion.records**.

5. In the DATA step below, what is the length of the new variable, **Type**?

```
data orion.newloan;
   set orion.records;
   TotalPaid=sum(TotLoan+Interest);
   if Code='1' then Type='Fixed';
   else Type='Variable';
   length Type $ 10;
run;
```

a. 5
b. 8
c. 10
d. It depends on the first value of **Type** in **orion.records**.

6. In the program below, what is the value of **Benefit** for the observation shown?

```
data work.total;
   set payroll.june;
   Benefit=sum(Ins,Health_Award);
run;
```

PDV

Emp_ID	Salary	Ins	Bonus	Health_Award	Benefit
KBA	54000	800	250	.	.

a. a missing value
b. 55050
c. 800
d. 0

6. In the program below, what is the value of **Benefit** for the observation shown?

```
data work.total;
   set payroll.june;
   Benefit=sum(Ins,Health_Award);
run;
```

PDV

Emp_ID	Salary	Ins	Bonus	Health_Award	Benefit
KBA	54000	800	250	.	.

a. a missing value

b. 55050

c. 800

d. 0

7. Which of these statements does ***not*** correctly specify a SAS function?

   a. Deadline=sum(TimeSpent, Last_Name);
   b. GreatDay=today();
   c. FingersToes=sum(10,10);
   d. BirthdayYear=year('12dec1987'd);

7. Which of these statements does ***not*** correctly specify a SAS function?

   a. Deadline=sum(TimeSpent, Last_Name);
   b. GreatDay=today();
   c. FingersToes=sum(10,10);
   d. BirthdayYear=year('12dec1987'd);

8. Given what you know about how SAS processes the DROP and KEEP statements, would these two DATA steps create the same data set?

- ❍ Yes
- ❍ No

```
data work.subset1;
   set orion.sales;
   drop Salary;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
run;
```

```
data work.subset1;
   set orion.sales;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
   drop Salary;
run;
```

8. Given what you know about how SAS processes the DROP and KEEP statements, would these two DATA steps create the same data set?

- ❍ Yes
- ❍ No

```
data work.subset1;
   set orion.sales;
   drop Salary;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
run;
```

```
data work.subset1;
   set orion.sales;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
   drop Salary;
run;
```

9. Which of the following determines the length of a new variable at compile time?

   a. INPUT statement

   b. assignment statement

   c. LENGTH statement

   d. all of the above

9. Which of the following determines the length of a new variable at compile time?

a. INPUT statement

b. assignment statement

c. LENGTH statement

d. all of the above

10. Use a DO group in a DATA step when you want to execute multiple statements for a true IF-THEN expression.

- ❍ True
- ❍ False

10. Use a DO group in a DATA step when you want to execute multiple statements for a true IF-THEN expression.

- ❍ True
- ❍ False

Chapter 10: Combining Data Sets

10.1 Concatenating Data Sets

10.2 Merging Data Sets One-to-One

10.3 Merging Data Sets One-to-Many

10.4 Merging Data Sets With Non-Matches

Chapter 10: Combining Data Sets

10.1 Concatenating Data Sets

10.2 Merging Data Sets One-to-One

10.3 Merging Data Sets One-to-Many

10.4 Merging Data Sets With Non-Matches

Objectives

- Concatenate two or more SAS data sets using the SET statement in a DATA step.
- Change the names of variables using the RENAME= data set option.

Business Scenario

You have been asked to combine the data sets containing information about Orion Star employees from Denmark and France into a new data set.

Considerations

Concatenate like-structured data sets **empsdk** and **empsfr** to create a new data set named **empsall1**.

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France
Sophie	F	France

Both data sets contain the same variables.

Using a DATA Step

Use a DATA step to concatenate the data sets. List the data sets in the SET statement.

```
data empsall1;
   set empsdk empsfr;
run;
```

SET *SAS-data-set1 SAS-data-set2* . . .**;**

- The SET statement reads observations from each data set in the order in which they are listed.
- Any number of data sets can be included in the SET statement.

Compilation

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country

empsall1

First	Gender	Country

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

Initialize PDV

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country

empsall1

First	Gender	Country

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Lars	M	Denmark

empsall1

First	Gender	Country

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	Country
Lars	M	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Lars	M	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark

Data set variables are not reinitialized.

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Kari	F	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	Country
Kari	F	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Kari	F	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark

PDV is not reinitialized.

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Jonas	M	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	Country
Jonas	M	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Jonas	M	Denmark

PDV is not reinitialized.

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

EOF

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Jonas	M	Denmark

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

Reinitialize PDV

PDV

First	Gender	Country

PDV is reinitialized before processing the next data set.

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Pierre	M	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set emps
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	Country
Pierre	M	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Pierre	M	France

PDV is not reinitialized.

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Sophie	F	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
    set empsdk empsfr;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	Country
Sophie	F	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France
Sophie	F	France

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
Sophie	F	France

```
data empsall1;
   set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Sophie	F	France

PDV is not reinitialized.

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France
Sophie	F	France

...

Execution

empsdk

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark

empsfr

First	Gender	Country
Pierre	M	France
phie	F	France

EOF

```
data empsall1;
    set empsdk empsfr;
run;
```

PDV

First	Gender	Country
Sophie	F	France

empsall1

First	Gender	Country
Lars	M	Denmark
Kari	F	Denmark
Jonas	M	Denmark
Pierre	M	France
Sophie	F	France

Viewing the Log

Partial SAS Log

```
145  data empsall1;
146     set empsdk empsfr;
147  run;

NOTE: There were 3 observations read from the data set WORK.EMPSDK.
NOTE: There were 2 observations read from the data set WORK.EMPSFR.
NOTE: The data set WORK.EMPSALL1 has 5 observations and 3
variables.
```

Unlike-Structured Data Sets

Concatenate **empscn** and **empsjp** to create a new data set named **empsall2**.

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

The data sets do not contain the same variables.

```
data empsall2;
   set empscn empsjp;
run;
```

p110d02

10.01 Quiz

How many variables will be in **empsall2** after concatenating **empscn** and **empsjp**?

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp;
run;
```

10.01 Quiz – Correct Answer

How many variables will be in **empsall2** after concatenating **empscn** and **empsjp**?

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

Four variables: First, Gender, Country, and Region

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp;
run;
```

PDV

First	Gender	Country

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp;
run;
```

PDV

First	Gender	Country	Region

Final Results

empsall2

First	Gender	Country	Region
Chang	M	China	
Li	M	China	
Ming	F	China	
Cho	F		Japan
Tomi	M		Japan

- **Region** has missing values due to PDV initialization.
- **Country** has missing values due to PDV reinitialization before processing the second data set.

Business Scenario

Rename variables in one or more data sets to align columns.

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

Rename **Region** to **Country**.

RENAME= Data Set Option

The *RENAME= data set option* changes the name of a variable.

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

SAS-data-set **(RENAME=(***old-name-1***=***new-name-1*
*old-name-2***=***new-name-2*
...
*old-name-n***=***new-name-n***))**

- The RENAME= option must be specified in parentheses immediately after the appropriate SAS data set name.
- The name change affects the PDV and the output data set. It has no effect on the input data set.

RENAME= Data Set Option

Multiples variables can be renamed in one or more data sets.

```
set empscn(rename=(Country=Region))
    empsjp;
```

```
set empscn(rename=(First=Fname
                   Country=Region))
    empsjp(rename=(First=Fname));
```

```
set empscn
    empsjp(rename=(Region=Country));
```

10.02 Quiz

Which statement has correct syntax?

a.
```
set empscn(rename(Country=Location))
    empsjp(rename(Region=Location));
```

b.
```
set empscn(rename=(Country=Location))
    empsjp(rename=(Region=Location));
```

c.
```
set empscn rename=(Country=Location)
    empsjp rename=(Region=Location);
```

10.02 Quiz – Correct Answer

Which statement has correct syntax?

a.
```
set empscn(rename(Country=Location))
    empsjp(rename(Region=Location));
```

b.
```
set empscn(rename=(Country=Location))
    empsjp(rename=(Region=Location));
```

c.
```
set empscn rename=(Country=Location)
    empsjp rename=(Region=Location);
```

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

PDV

First	Gender	Country

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

PDV

First	Gender	Country

...

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

PDV

First	Gender	Country

...

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

PDV

First	Gender	Country

...

Compilation

empscn

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China

empsjp

First	Gender	Region
Cho	F	Japan
Tomi	M	Japan

```
data empsall2;
   set empscn empsjp(rename=(Region=Country));
run;
```

PDV

First	Gender	Country

Final Results

The **Region** values are stored in **Country**.

empsall2

First	Gender	Country
Chang	M	China
Li	M	China
Ming	F	China
Cho	F	Japan
Tomi	M	Japan

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 10: Combining Data Sets

10.1 Concatenating Data Sets

10.2 Merging Data Sets One-to-One

10.3 Merging Data Sets One-to-Many

10.4 Merging Data Sets With Non-Matches

Objectives

- Prepare data sets for merging using the SORT procedure.
- Merge SAS data sets one-to-one based on a common variable.

Match-Merging

One-to-One

A single observation in one data set is related to exactly one observation in another data set based on the values of one or more selected variables.

One-to-Many

A single observation in one data set is related to more than one observation in another data set based on the values of one or more selected variables.

Non-matches

At least one observation in one data set is unrelated to any observation in another data set based on the values of one or more selected variables.

Match-Merging: Sorting the Data Sets

The data sets in a match-merge must be sorted by the common variable or variables being matched.

```
PROC SORT  DATA=input-SAS-data-set
                <OUT=output-SAS-data-set>;
     BY <DESCENDING> by-variable(s);
RUN;
```

10.03 Multiple Choice Poll

Which of the following BY statements correctly sorts by descending salary within gender?

a. by descending salary within gender;
b. by descending salary gender;
c. by gender descending salary;
d. by gender salary descending;

10.03 Multiple Choice Poll – Correct Answer

Which of the following BY statements correctly sorts by descending salary within gender?

a. by descending salary within gender;
b. by descending salary gender;
c. by gender descending salary;
d. by gender salary descending;

The keyword descending is placed before the variable to which it applies.

Business Scenario

Merge the Australian employee data set with a phone data set to obtain each employee's home phone number, storing the results in a new data set.

empsau

First	Gender	EmpID

+

phoneh

EmpID	Phone

empsauh

First	Gender	EmpID	Phone

Match-Merging

The *MERGE statement* in a DATA step joins observations from two or more SAS data sets into single observations.

```
data empsauh;
    merge empsau phoneh;
    by EmpID;
run;
```

MERGE *SAS-data-set1 SAS-data-set2 . . .*;
BY <DESCENDING> *BY-variable(s)*;

A *BY statement* indicates a match-merge and lists the variable or variables to match.

p110d04

MERGE and BY Statements

Requirements for match-merging:

- Two or more data sets are listed in the MERGE statement.
- The variables in the BY statement must be common to all data sets.
- The data sets must be sorted by the variables listed in the BY statement.

One-to-One Merge

One observation in **empsau** matches exactly one observation in **phoneh**.

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phoneh

EmpID	Phone
121150	+61(2)5555-1793
121151	+61(2)5555-1849
121152	+61(2)5555-1665

The data sets are sorted by **EmpID**.

Final Results

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phoneh

EmpID	Phone
121150	+61(2)5555-1793
121151	+61(2)5555-1849
121152	+61(2)5555-1665

empsauh

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1793
Kylie	F	121151	+61(2)5555-1849
Birin	M	121152	+61(2)5555-1665

10.04 Quiz

- Retrieve program **p110a01**.
- Complete the program to match-merge the sorted SAS data sets referenced in the PROC SORT steps.
- Submit the program. Correct and resubmit, if necessary.

What are the modified, completed statements?

10.04 Quiz – Correct Answer

What are the modified, completed statements?

```
proc sort data=orion.employee_payroll
          out=work.payroll;
   by Employee_ID;
run;

proc sort data=orion.employee_addresses
          out=work.addresses;
   by Employee_ID;
run;

data work.payadd;
   merge work.payroll work.addresses;
   by Employee_ID;
run;
```

p110a01s

Questions ???

Chapter 10: Combining Data Sets

10.1 Concatenating Data Sets

10.2 Merging Data Sets One-to-One

10.3 Merging Data Sets One-to-Many

10.4 Merging Data Sets With Non-Matches

Objectives

- Merge SAS data sets one-to-many based on a common variable.

Business Scenario

Merge the Australian employee information data set with the phone data set to obtain the phone numbers for each employee.

Considerations

In this one-to-many merge, one observation in **empsau** matches one or more observations in **phones**.

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

The data sets are sorted by **EmpID**.

Match-Merging

Merge the two data sets by **EmpID** and create a new data set named **empphones**.

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

p110d05

Match-Merging

BY variable

BY group

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

- The common variable is called the *BY variable*.
- The value of the BY variable is the *BY value*.
- A group of observations with the same BY value is a *BY group*.

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

Initialize PDV

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

PDV

First	Gender	EmpID	Type	Phone
		.		

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Do the **EmpID** values match?

Yes

PDV

First	Gender	EmpID	Type	Phone
		.		

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads both observations into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Home	+61(2)5555-1793

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1793

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Data set variables are not reinitialized.

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1793

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Do the **EmpID** values match?

No

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Home	+61(2)5555-1793

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Did **EmpID** change?

No

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Home	+61(2)5555-1793

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads the matching observation into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1794

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phonehW;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1794

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Do the **EmpID** values match?

Yes

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1794

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Did **EmpID** change?

Yes

PDV

First	Gender	EmpID	Type	Phone
Togar	M	121150	Work	+61(2)5555-1794

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reinitialize PDV

PDV

First	Gender	EmpID	Type	Phone
		.		

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads both observations into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Kylie	F	121151	Home	+61(2)5555-1849

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Kylie	F	121151	Home	+61(2)5555-1849

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Do the **EmpID** values match?

Yes

PDV

First	Gender	EmpID	Type	Phone
Kylie	F	121151	Home	+61(2)5555-1849

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Did **EmpID** change?

Yes

PDV

First	Gender	EmpID	Type	Phone
Kylie	F	121151	Home	+61(2)5555-1849

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

Reinitialize PDV

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Type	Phone
		.		

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads both observations into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Work	+61(2)5555-1850

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Work	+61(2)5555-1850

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Did **EmpID** change?

No

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Work	+61(2)5555-1850

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads the matching observation into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Home	+61(2)5555-1665

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Home	+61(2)5555-1665

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Did **EmpID** change?

No

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Home	+61(2)5555-1665

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

Reads the matching observation into the PDV.

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Cell	+61(2)5555-1666

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

```
data empphones;
   merge empsau phones;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Cell	+61(2)5555-1666

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

EOF

```
data empphones;
    merge empsau phones;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Type	Phone
Birin	M	121152	Cell	+61(2)5555-1666

...

Final Results

empphones

First	Gender	EmpID	Type	Phone
Togar	M	121150	Home	+61(2)5555-1793
Togar	M	121150	Work	+61(2)5555-1794
Kylie	F	121151	Home	+61(2)5555-1849
Birin	M	121152	Work	+61(2)5555-1850
Birin	M	121152	Home	+61(2)5555-1665
Birin	M	121152	Cell	+61(2)5555-1666

Questions ???

10.05 Quiz

In a one-to-many merge, does it matter which data set is listed first in the MERGE statement?

- Open **p110a02** and submit
- Reverse the order of the data sets and submit again.
- Observe the results. How are they different?

10.05 Quiz – Correct Answer

In a one-to-many merge, does it matter which data set is listed first in the MERGE statement?

- Open **p110a02** and submit
- Reverse the order of the data sets and submit again.
- Observe the results. How are they different?

The results are the same, but the order of the variables is different.

p110a02s

Many-to-One Merge

One or more rows in one data set match exactly one row in the other data set.

phones

EmpID	Type	Phone
121150	Home	+61(2)5555-1793
121150	Work	+61(2)5555-1794
121151	Home	+61(2)5555-1849
121152	Work	+61(2)5555-1850
121152	Home	+61(2)5555-1665
121152	Cell	+61(2)5555-1666

empsau

EmpID	First	Gender
121150	Togar	M
121151	Kylie	F
121512	Birin	M

```
data phones;
   merge phones empsau;
   by EmpID;
run;
```

p110d06

Viewing the Output

PROC PRINT Output

Obs	EmpID	Type	Phone	First	Gender
1	121150	Home	+61(2)5555-1793	Togar	M
2	121150	Work	+61(2)5555-1794	Togar	M
3	121151	Home	+61(2)5555-1849	Kylie	F
4	121152	Work	+61(2)5555-1850	Birin	M
5	121152	Home	+61(2)5555-1665	Birin	M
6	121152	Cell	+61(2)5555-1666	Birin	M

- The results are the same as the one-to-many merge.
- The order of variables is different.

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 10: Combining Data Sets

10.1 Concatenating Data Sets

10.2 Merging Data Sets One-to-One

10.3 Merging Data Sets One-to-Many

10.4 Merging Data Sets With Non-Matches

Objectives

- Control the observations in the output data set by using the IN= option.

Business Scenario

An Orion Star manager in Australia requested an inventory of company phone numbers.

Merge with Non-Matches

There are observations in **empsau** that do not have a match in **phonec**, and some in **phonec** that do not match any in **empsau**.

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

The data sets are sorted by **EmpID**.

Match-Merging

Merge **empsau** and **phonec** by **EmpID** to create a new data set named **empsauc**.

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

Initialize PDV

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

PDV

First	Gender	EmpID	Phone
		.	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Do the **EmpID** values match?

Yes

PDV

First	Gender	EmpID	Phone
		.	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reads both observations into the PDV.

PDV

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Do the **EmpID** values match?

No

PDV

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Did **EmpID** change?

Yes

PDV

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

Reinitialize PDV

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Phone
		.	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Which **EmpID** value sequentially comes first?

121151

PDV

First	Gender	EmpID	Phone
		.	

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reads that observation into the PDV.

PDV

First	Gender	EmpID	Phone
Kylie	F	121151	

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Phone
Kylie	F	121151	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Do the **EmpID** values match?

Yes

PDV

First	Gender	EmpID	Phone
Kylie	F	121151	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Did **EmpID** change?

Yes

PDV

First	Gender	EmpID	Phone
Kylie	F	121151	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

Reinitialize PDV

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Phone
		.	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reads both observations into the PDV.

PDV

First	Gender	EmpID	Phone
Birin	M	121152	+61(2)5555-1667

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Phone
Birin	M	121152	+61(2)5555-1667

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Did **EmpID** change?

Yes

PDV

First	Gender	EmpID	Phone
Birin	M	121152	+61(2)5555-1667

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

Reinitialize PDV

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Phone
		.	

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reads the observation into the PDV.

PDV

First	Gender	EmpID	Phone
		121153	+61(2)5555-1348

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
   merge empsau phonec;
   by EmpID;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

First	Gender	EmpID	Phone
		121153	+61(2)5555-1348

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

EOF

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

First	Gender	EmpID	Phone
		121153	+61(2)5555-1348

Final Results

empsauc

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795
Kylie	F	121151	
Birin	M	121152	+61(2)5555-1667
		121153	+61(2)5555-1348

The final results include both matches and non-matches.

10.06 Quiz

Which data set contributed information to the last observation in the output data set?

a. empsau
b. phonec
c. both empsau and phonec
d. There is insufficient information.

empsauc

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795
Kylie	F	121151	
Birin	M	121152	+61(2)5555-1667
		121153	+61(2)5555-1348

10.06 Quiz – Correct Answer

Which data set contributed information to the last observation in the output data set?

a. empsau
b. phonec
c. both empsau and phonec
d. There is insufficient information.

empsauc

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795
Kylie	F	121151	
Birin	M	121152	+61(2)5555-1667
		121153	+61(2)5555-1348

Questions ???

Business Scenario

An Orion Star manager requested three phone inventory reports.

- employees with company phones
- employees without company phones
- phones with an invalid employee ID

IN= Data Set Option

The *IN= data set option* creates a variable that indicates whether the data set contributed to building the current observation.

MERGE *SAS-data-set* **(IN=***variable***) …**

variable is a temporary numeric variable that has two possible values:

0	Indicates that the data set did ***not*** contribute to the current observation.
1	Indicates that the data set ***did*** contribute to the current observation.

IN= Data Set Option

MERGE statement examples:

```
merge empsau(in=Emps)
      phonec(in=Cell);
```

```
merge empsau(in=E)
      phonec(in=P);
```

```
merge empsau(in=AU)
      phonec;
```

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
run;
```

match

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
Togar	M	121150	1	+61(2)5555-1795	1

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
run;
```

non-match

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
Kylie	F	121151	1		0

...

Execution

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
run;
```

match

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
Birin	M	121152	1	+61(2)5555-1667	1

...

10.07 Quiz

What are the values of **Emps** and **Cell**?

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
run;
```

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
		121153		+61(2)5555-1348	

10.07 Quiz – Correct Answer

What are the values of **Emps** and **Cell**?

empsau

First	Gender	EmpID
Togar	M	121150
Kylie	F	121151
Birin	M	121152

phonec

EmpID	Phone
121150	+61(2)5555-1795
121152	+61(2)5555-1667
121153	+61(2)5555-1348

EOF

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
run;
```

non-match

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
		121153	0	+61(2)5555-1348	1

PDV Results

PDV

First	Gender	EmpID	D Emps	Phone	D Cell
Togar	M	121150	1	+61(2)5555-1795	1
Kylie	F	121151	1		0
Birin	M	121152	1	+61(2)5555-1667	1
		121153	0	+61(2)5555-1348	1

The variables created with the IN= data set option are only available during DATA step execution.

- They are not written to the SAS data set.
- Their value can be tested using conditional logic.

Matches Only

Add a subsetting IF statement to select the employees that have company phones.

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
   if Emps=1 and Cell=1;
run;
```

empsauc

First	Gender	EmpID	Phone
Togar	M	121150	+61(2)5555-1795
Birin	M	121152	+61(2)5555-1667

Non-Matches from empsau

Select the employees that do not have company phones.

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
   if Emps=1 and Cell=0;
run;
```

empsauc

First	Gender	EmpID	Phone
Kylie	F	121151	

Non-Matches from phonec

Select the phones associated with an invalid employee ID.

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
   if Emps=0 and Cell=1;
run;
```

empsauc

First	Gender	EmpID	Phone
		121153	+61(2)5555-1348

All Non-Matches

```
data empsauc;
   merge empsau(in=Emps)
         phonec(in=Cell);
   by EmpID;
   if Emps=0 or Cell=0;
run;
```

empsauc

First	Gender	EmpID	Phone
Kylie	F	121151	
		121153	+61(2)5555-1348

Use the OR operator, not the AND operator.

Alternate Syntax

When checking a variable for a value of *1* or *0* as in the previous scenario, you can use the following syntax:

Instead of	You can use
if Emps=1 and Cell=1;	if Emps and Cell;
if Emps=1 and Cell=0;	if Emps and not Cell;
if Emps=0 and Cell=1;	if not Emps and Cell;
if Emps=0 or Cell=0;	If not Emps or not Cell;

Alternate Syntax

Both programs create a report of employees without cell phones.

```
data empsphone;
   merge empsact(in=inEmps)
         phoneact(in=inCell);
   by EmpID;
   if inEmps=1 and inCell=0;
run;
```

```
data empsphone;
   merge empsact(in=inEmps)
         phoneact(in=inCell);
   by EmpID;
   if inEmps and not inCell;
run;
```

p110d09

Questions ???

Business Scenario

Merge Orion Star customer information with customer type data to obtain a customer description. The new data set should include only US customers.

Considerations

The **orion.customer** data set is not sorted by **Customer_Type_ID**, the common variable. The subsetting variable, **Country**, is defined in only one data set.

orion.customer

Customer_ ID	Country	Customer_ Name	. . .	Birth_ Date	Customer_Type_ ID

orion.customer_type

Customer_ Group	Customer_Group_ ID	Customer_ Type	Customer_Type_ ID

10.08 Quiz

Open and submit **p110a03**. Correct the program and resubmit. What change is needed to correct the error?

```
proc sort data=orion.customer
          out=cust_by_type;
   by Customer_Type_ID;
run;

data customers;
   merge cust_by_type orion.customer_type;
   by Customer_Type_ID;
   where Country='US';
run;
```

10.08 Quiz – Correct Answer

What change is needed to correct the error?

```
397  proc sort data=orion.customer
398             out=cust_by_type;
399     by Customer_Type_ID;
400  run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUST_BY_TYPE has 77 observations and 12 variables.

401
402  data customers;
403     merge cust_by_type orion.customer_type;
404     by Customer_Type_ID;
405     where Country='US';
ERROR: Variable Country is not on file ORION.CUSTOMER_TYPE.
406  run;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.CUSTOMERS may be incomplete.  When this step was
stopped there were 0 observations and 15 variables.
```

Country is not defined in both data sets. Replace the WHERE statement with a subsetting IF.

Subsetting IF

Use a subsetting IF when the subsetting variable is not in all data sets named in the MERGE statement.

```
proc sort data=orion.customer
          out=cust_by_type;
   by Customer_Type_ID;
run;

data customers;
   merge cust_by_type  orion.customer_type;
   by Customer_Type_ID;
   if Country='US';
run;
```

p110a03s

Viewing the Output

Partial SAS Log

```
407  proc sort data=orion.customer
408            out=cust_by_type;
409     by Customer_Type_ID;
410  run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUST_BY_TYPE has 77 observations and 12 variables.

411
412  data customers;
413     merge cust_by_type orion.customer_type;
414     by Customer_Type_ID;
415     if Country='US';
416  run;

NOTE: There were 77 observations read from the data set WORK.CUST_BY_TYPE.
NOTE: There were 8 observations read from the data set ORION.CUSTOMER_TYPE.
NOTE: The data set WORK.CUSTOMERS has 28 observations and 15 variables.
```

WHERE versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
SET statement	Yes	Yes
assignment statement	No	Yes
INPUT statement	No	Yes
SET/MERGE statement (multiple data sets)		
Variable in ALL data sets	Yes	Yes
Variable not in ALL data sets	No	Yes

Questions ???

Exercise

This exercise reinforces the concepts discussed previously.

Chapter Review

1. Which of the following statements is true about merging SAS data sets by using the DATA step?

   a. Merging combines observations from two or more data sets into a single observation in new data set.
   b. SAS can merge data sets based on the position of observations in the original data set or by the values of one or more common variables.
   c. Match-merging is merging by values of one or more common variables.
   d. To match-merge data sets, all input data sets must be sorted or indexed on the BY variable or variables.
   e. all of the above

1. Which of the following statements is true about merging SAS data sets by using the DATA step?

   a. Merging combines observations from two or more data sets into a single observation in new data set.
   b. SAS can merge data sets based on the position of observations in the original data set or by the values of one or more common variables.
   c. Match-merging is merging by values of one or more common variables.
   d. To match-merge data sets, all input data sets must be sorted or indexed on the BY variable or variables.
   e. all of the above

2. Which of the following programs concatenates the data sets **sales** and **products**, in that order?

a.
```
data newsales;
   set products sales;
run;
```

b.
```
data newsales;
   set sales products;
run;
```

c.
```
data newsales;
   set sales;
   set products;
run;
```

2. Which of the following programs concatenates the data sets **sales** and **products**, in that order?

a.
```
data newsales;
   set products sales;
run;
```

b.
```
data newsales;
   set sales products;
run;
```

c.
```
data newsales;
   set sales;
   set products;
run;
```

3. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
   merge managers staff;
   by EmpID;
run;
```

a. all of the observations from **managers**, and only those observations from **staff** with matching values for **EmpID**
b. all of the observations from **staff**, and only those observations from **managers** with matching values for **EmpID**
c. all observations from **staff** and all observations from **managers**, whether or not they have matching values
d. only those observations from **staff** and **managers** with matching values for **EmpID**

3. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
   merge managers staff;
   by EmpID;
run;
```

a. all of the observations from **managers**, and only those observations from **staff** with matching values for **EmpID**

b. all of the observations from **staff**, and only those observations from **managers** with matching values for **EmpID**

c. all observations from **staff** and all observations from **managers**, whether or not they have matching values

d. only those observations from **staff** and **managers** with matching values for **EmpID**

4. If you concatenate the data sets below in the order shown, what is the value of **Sale** in observation 2 of the new data set?

Reps

ID	Name
1	Nay Rong
2	Kelly Windsor
3	Julio Meraz
4	Richard Krabill

Close

ID	Sale
1	$28,000
2	$30,000
2	$40,000
3	$15,000
3	$20,000
3	$25,000
4	$35,000

a. missing
b. $30,000
c. $40,000
d. You cannot concatenate these data sets.

4. If you concatenate the data sets below in the order shown, what is the value of **Sale** in observation 2 of the new data set?

Reps

ID	Name
1	Nay Rong
2	Kelly Windsor
3	Julio Meraz
4	Richard Krabill

Close

ID	Sale
1	$28,000
2	$30,000
2	$40,000
3	$15,000
3	$20,000
3	$25,000
4	$35,000

a. missing
b. $30,000
c. $40,000
d. You cannot concatenate these data sets.

5. What happens if you submit the following program to merge **donors1** and **donors2**, shown below?

```
data merged;
   merge donors1 donors2;
   by ID;
run;
```

donors1

ID	Type	Units
2304	O	16
1129	A	48
1129	A	50
1129	A	57
2486	B	63

donors2

ID	Code	Units
6488	65	27
1129	63	32
5438	62	39
2304	61	45
1387	64	67

a. The **merged** data set contains some missing values because not all observations have matching observations in the other data set.

b. The **merged** data set contains eight observations.

c. The DATA step produces errors.

5. What happens if you submit the following program to merge **donors1** and **donors2**, shown below?

```
data merged;
   merge donors1 donors2;
   by ID;
run;
```

donors1

ID	Type	Units
2304	O	16
1129	A	48
1129	A	50
1129	A	57
2486	B	63

donors2

ID	Code	Units
6488	65	27
1129	63	32
5438	62	39
2304	61	45
1387	64	67

a. The **merged** data set contains some missing values because not all observations have matching observations in the other data set.

b. The **merged** data set contains eight observations.

c. The DATA step produces errors.

6. Suppose you want to concatenate these data sets. Which DATA step creates an output data set that combines the values of **Color** and **Hue** in the single variable **Color**?

a.
```
data widgets_all;
   set widget1(rename=(Color=Hue))
       widget2;
run;
```

b.
```
data widgets_all;
   set widget1
       widget2(rename=(Hue=Color));
run;
```

c.
```
data widgets_all;
   set widget1
       widget2(Hue=Color);
run;
```

Widget1

Tag	Color	Model
77904	blue	AB42
56012	red	BA25
35499	orange	FC36

Widget2

Tag	Hue	Model
89325	red	SP17
65888	yellow	BA12
00167	green	PG20

6. Suppose you want to concatenate these data sets. Which DATA step creates an output data set that combines the values of **Color** and **Hue** in the single variable **Color**?

a.
```
data widgets_all;
   set widget1(rename=(Color=Hue))
       widget2;
run;
```

b.
```
data widgets_all;
   set widget1
       widget2(rename=(Hue=Color));
run;
```

c.
```
data widgets_all;
   set widget1
       widget2(Hue=Color);
run;
```

Widget1

Tag	Color	Model
77904	blue	AB42
56012	red	BA25
35499	orange	FC36

Widget2

Tag	Hue	Model
89325	red	SP17
65888	yellow	BA12
00167	green	PG20

7. What is the syntax error in this DATA step?

```
data returns_qtr1;
   set returns_jan(rename=(ID=CustID)
                   (Return=Item))
       returns_feb(rename=(Dt=Date))
       returns_mar;
run;
```

a. You cannot specify more than two data sets in the SET statement.
b. There are too many sets of parentheses in the RENAME= option.
c. You cannot specify multiple variables in the RENAME= option.
d. The BY statement is missing.

7. What is the syntax error in this DATA step?

```
data returns_qtr1;
   set returns_jan(rename=(ID=CustID)
                   (Return=Item))
       returns_feb(rename=(Dt=Date))
       returns_mar;
run;
```

a. You cannot specify more than two data sets in the SET statement.

b. There are too many sets of parentheses in the RENAME= option.

c. You cannot specify multiple variables in the RENAME= option.

d. The BY statement is missing.

8. In the second iteration of this DATA step, after the data has been merged, what are the values of **C** and **A**?

```
data client_amount;
   merge clients(in=C)
         amounts(in=A);
   by Name;
run;
```

Clients

Name	EmpID
Ankerton	11123
Davis	22298
Masters	33351
Wolmer	44483

Amounts

Name	Date	Amt
Ankerton	08OCT96	92
Ankerton	15OCT96	43
Davis	04OCT96	16
Masters	.	27
Thomas	21OCT96	15

a. C=1, A=0
b. C=0, A=1
c. C=1, A=1
d. missing
e. unknown

8. In the second iteration of this DATA step, after the data has been merged, what are the values of **C** and **A**?

```
data client_amount;
   merge clients(in=C)
         amounts(in=A);
   by Name;
run;
```

Clients

Name	EmpID
Ankerton	11123
Davis	22298
Masters	33351
Wolmer	44483

Amounts

Name	Date	Amt
Ankerton	08OCT96	92
Ankerton	15OCT96	43
Davis	04OCT96	16
Masters	.	27
Thomas	21OCT96	15

a. C=1, A=0
b. C=0, A=1
c. C=1, A=1
d. missing
e. unknown

9. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
   merge managers (in=M)
         staff (in=S);
   by EmpID;
   if M=0 and S=1;
run;
```

a. only the observations from **staff** that have no match in **managers**
b. only the observations from **managers** that have no match in **staff**
c. all observations from both **managers** and **staff**, whether or not they match
d. no observations

9. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
   merge managers (in=M)
         staff (in=S);
   by EmpID;
   if M=0 and S=1;
run;
```

a. only the observations from **staff** that have no match in **managers**

b. only the observations from **managers** that have no match in **staff**

c. all observations from both **managers** and **staff**, whether or not they match

d. no observations

10. What is the relationship of the data set **first** to the data set **second** when merged by the variable **ID**?

a. one-to-one
b. one-to-many
c. many-to-one
d. many-to-many
e. non-matching

first

Name	ID	Age
Togar	121150	39
Kylie	121152	34
Birin	121153	32
Gloria	121154	12
James	121155	36
Gene	121156	28
Tom	121157	27

second

ID	Date
121150	02/15/05
121152	05/22/07
121153	03/04/06
121154	11/22/05
121155	07/08/06
121156	12/15/06
121157	04/30/07

10. What is the relationship of the data set **first** to the data set **second** when merged by the variable **ID**?

a. one-to-one
b. one-to-many
c. many-to-one
d. many-to-many
e. non-matching

first

Name	ID	Age
Togar	121150	39
Kylie	121152	34
Birin	121153	32
Gloria	121154	12
James	121155	36
Gene	121156	28
Tom	121157	27

second

ID	Date
121150	02/15/05
121152	05/22/07
121153	03/04/06
121154	11/22/05
121155	07/08/06
121156	12/15/06
121157	04/30/07

Started on	Monday, October 30, 2023, 4:23 PM
State	Finished
Completed on	Monday, October 30, 2023, 5:23 PM
Time taken	59 mins 54 secs
Grade	**39.00** out of 61.00 (**63.93**%)

Question **1**
Correct
1.00 points out of 1.00

What are the two types of steps?

Select one:

- a. DATA and PROC ✔
- b. RUN and QUIT
- c. CHARACTER and NUMERIC
- d. DATA and DESCRIPTOR

The two types of SAS steps are DATA and PROC. Character and numeric are the two SAS data types, RUN and QUIT are the two explicity step boundaries and the two parts of a data set include the DATA and DESCRIPTOR portions.

The correct answer is: DATA and PROC

Question **2**
Correct
1.00 points out of 1.00

How many steps are in the program shown below?

```
proc print data=sashelp.cars;
    var make model msrp mpg_city
    mpg_highway horsepower weight;
    format weight comma8.;
run;
```

Select one:

- a. 5
- b. 3
- c. 4
- d. 1 ✔

There is one PROC PRINT step in this program. There are four statements, and five lines of code.

The correct answer is: 1

Question **3**
Correct
1.00 points out of 1.00

Given the code shown below, which two statements are true? (Choose two.)

```
data work.homeruns23;
    set sashelp.baseball;
    /* where nhome=23; */
run;
```

Select one or more:

- a. **WORK.HOMERUNS23** contains only the rows where **NHOME** is equal to *23*.
- b. **WORK.HOMERUNS23** contains the same number of rows as **SASHELP.BASEBALL**. ✔
- c. The WHERE statement is executed.
- d. The WHERE statement is ignored. ✔

The WHERE statement is ignored because it is commented. Commented statements are not executed. Because the WHERE statement is not executed, all the rows are read from **SASHELP.BASEBALL**.

The correct answers are: The WHERE statement is ignored., **WORK.HOMERUNS23** contains the same number of rows as **SASHELP.BASEBALL**.

0.00 points out of 1.00

```
35        var make model msrp mpg_city
36        mpg_highway horsepower weight
37        format weight comma8.;
                        _______
                        22
                        201
ERROR: Variable FORMAT not found.
ERROR 22-322: Syntax error, expecting one of the following: a name, ;,
      -, /, :, _ALL_, _CHARACTER_, _CHAR_, _NUMERIC_.
ERROR 201-322: The option is not recognized and will be ignored.
38    run;
```

Which statement contains the error?

Select one:

- a. FORMAT ✖
- b. VAR
- c. PROC PRINT
- d. RUN

The FORMAT statement is underlined, but the error is a missing semicolon in the VAR statement.

The correct answer is: VAR

Question **5**

Incorrect

0.00 points out of 1.00

Which statement aids in debugging logic errors?

Select one:

- a. TRACELOG
- b. DEBUG ✖
- c. PUTLOG
- d. _ERROR_

The PUTLOG statement writes messages that you specify to the SAS log and can be helpful when debugging logic errors. The automatic variable **_ERROR_** is *0* by default but is set to *1* when a data error occurs.

The correct answer is: PUTLOG

Question **6**

Correct

1.00 points out of 1.00

Examine the SAS log shown below:

```
1  proc print data=sashelp.snacks;
2      var Product Price QtySold
3  run;

ERROR: Variable RUN not found.
```

What type of error message is displayed?

Select one:

- a. logic error
- b. syntax error ✔
- c. PROC error
- d. run-time error

A syntax error message is written to the log when the programming statements do not conform to the rules of the SAS language. Leaving off the semicolon for the VAR statement is considered a syntax error.

The correct answer is: syntax error

1.00 points out of 1.00

Select one:

- a. _1EMPLOYEE NAME
- b. $1EMPLOYEE_NAME
- c. 1EMPLOYEE_NAME
- d. _1EMPLOYEE_NAME ✔

Valid variable or data set names must begin with a letter or an underscore(_) and can continue with letters, numbers and underscores up to 32 characters.

The correct answer is: _1EMPLOYEE_NAME

Question **8**

Correct

1.00 points out of 1.00

Which statement is true with regard to character variables?

Select one:

- a. The byte size limit is 32.
- b. One character represents one byte. ✔
- c. A character value must contain a letter.
- d. Values are stored using floating-point representation.

In SAS, one character equals one byte. Character values can have letters, numbers, and/or underscores is true for character names. For values, you can have anything: letters, numbers, and any special character such as the space or underscore. The byte size limit is 32,767. Only numeric values are stored using floating point representation.

The correct answer is: One character represents one byte.

Question **9**

Correct

1.00 points out of 1.00

Which assignment statement creates a character variable?

Select one:

a.
```
anniversary = "18apr2022";
```
✔

b.
```
anniversary = "18APR22"d;
```

c.
```
anniversary = "18apr2022"d;
```

d.
```
anniversary = 18Apr22;
```

Date constants are numerics, and "*18Apr22"* would cause an error. A value in quotation marks (and without a trailing *d*, *t*, or *dt*) is a character value.

The correct answer is:
```
anniversary = "18apr2022";
```

Question **10**

Incorrect

0.00 points out of 1.00

A SAS date is the number of days since January 1, 1960 and the specific date. The starting point of January 1, 1960 is represented as ___ on a number line.

Enter your numeric answer for the space above.

Answer:

1960

✖

The starting point of January 1, 1960 is represented as 0 on a number line. SAS dates represent the number of days since January 1, 1960.

The correct answer is: 0

0.50 points out of 1.00

Select one or more:

a.
```
where Name = '.';
```
✖

b.
```
where Name is missing;
```
✔

c.
```
where Name = missing;
```

d.
```
where Name = " ";
```

Missing character values are represented with a blank. The IS MISSING operator checks for the existence of either character or numeric values depending the data type of the variable used in the statement. To explicitly check for missing character, you can use " ", for example

```
where Name = " ";
```

The correct answers are:
```
where Name is missing;
```

,
```
where Name = " ";
```

Question **12**

Incorrect

0.00 points out of 1.00

Which statement is true regarding the LIBNAME statement for reading SAS data sets?

Select one:

- a. The LIBNAME statement belongs in the DATA step after the DATA statement.
- b. SAS9 is the default engine if an engine is NOT specified on the LIBNAME statement.
- c. A library reference on the LIBNAME statement specifies a shortcut name for the storage location of SAS data sets.
- d. A LIBNAME statement specifies a library reference, an engine, a storage location, and a data set name. ✖

The LIBNAME statement specifies a shortcut name for the storage location of SAS data sets, also known as a library reference. This statement is a global statement that can stand-alone in the program. V9 is the default engine if an engine is not specified.

The correct answer is: A library reference on the LIBNAME statement specifies a shortcut name for the storage location of SAS data sets.

Question **13**

Correct

1.00 points out of 1.00

Which two LIBNAME statements create a library reference for accessing SAS data sets? (Choose two.)

Select one or more:

a.
```
libname test = "C:\User\Maria\Data";
```

b.
```
libname test base "C:\User\Maria\Data";
```
✔

c.
```
libname "C:\User\Maria\Data" test;
```

d.
```
libname test "C:\User\Maria\Data";
```
✔

The base library is implied, and it may be explicitly specified.

The correct answers are:
```
libname test "C:\User\Maria\Data";
```

,

1.00 points out of 1.00

Select one:

- a. PROC CONTENTS shows a listing of the SAS programs that are available in the default location.
- b. PROC CONTENTS displays the global options that have been assigned.
- c. PROC CONTENTS creates a report of the SAS products that are installed on the given computer.
- d. PROC CONTENTS creates a report of the descriptor portion of a SAS data set. ✔

PROC CONTENTS creates a report of the descriptor portion of a SAS data set. PROC SETINIT creates a report of the SAS products installed on the given computer and PROC OPTIONS displays the global options that have been assisgned.

The correct answer is: PROC CONTENTS creates a report of the descriptor portion of a SAS data set.

Question **15**

Correct

1.00 points out of 1.00

The **Customer** data set shown below has five variables:

CustomerID	Name	Phone	Age	LoyaltyGroup
1	Sara	410-555-4040	18	Gold
2	Colton	919-555-3489	29	Silver
3	Heidi	201-555-1770	45	Bronze
4	Brian	212-555-6780	33	Gold

The following DATA step is written:

```
data A(keep=CustomerID Age);
    set Customer(drop=Name Phone);
run;
```

How many variables will be in the new data set?

Select one:

- a. 5
- b. 3
- c. 2 ✔
- d. 1

The KEEP= data set option on the DATA statement lists two variables. These are the two variables that will be in the new data set. The DROP= data set option on the SET statement excludes the two variables listed from being processed when the INPUT data set is read, and the remaining three variables are processed.

The correct answer is: 2

0.00 points out of 1.00

Select one:

a.
```
rename (Mass,MassMetric)
```

b.
```
rename Mass as MassMetric
```
✖

c.
```
rename Mass=MassMetric
```

d.
```
rename=(Mass=MassMetric)
```

The old name, **Mass**, goes before the equals sign. The new name, **MassMetric**, goes after the equals sign.

```
rename=(Mass=MassMetric)
```

The correct answer is:
```
rename=(Mass=MassMetric)
```

Question **17**

Incorrect

0.00 points out of 1.00

Which DATA statement correctly creates three SAS data sets?

Select one:

a.
```
data=one two three;
```

b.
```
data one, two, three;
```

c.
```
data=one data=two data=three;
```
✖

d.
```
data one two three;
```

List the names of the multiple data sets you want to create on the DATA statement separated by a space and end the statement with a semicolon.

The correct answer is:
```
data one two three;
```

Question **18**

Incorrect

0.00 points out of 1.00

Given the SAS program shown below, after running the program, which statement is true?

```
data Setosa Versicolor;
    set sashelp.iris;
    if Species = “Setosa” then output Setosa;
run;
```

Select one:

a. **sashelp.iris** will only contain rows where **Species** is *Setosa*.

b. **Versicolor** will contain all rows where **Species** is not *Setosa*.

c. **Versicolor** will contain no rows.

d. No data set is created. There is an error in the log. ✖

Rows are explicitly output to **Setosa**. No rows are explicitly output to **Versicolor**.

The correct answer is: **Versicolor** will contain no rows.

0.00 points out of 1.00

Select one:

a.
```
set new;
```

b.
```
set=new;
```

c.
```
set data=new;
```
✖

d.
```
set data new;
```

List the name of the temporary data set on the SET statement and end the statement with a semicolon.

The correct answer is:
```
set new;
```

Question **20**

Correct

1.00 points out of 1.00

Which statement reads the temporary data set **October**?

Select one:

a.
```
read October;
```

b.
```
set October;
```
✔

c.
```
data October;
```

d.
```
in October;
```

The SET statement reads in data sets.

The correct answer is:
```
set October;
```

Question **21**

Incorrect

0.00 points out of 1.00

Which sentence is true regarding a DATA step with the MERGE and BY statements?

Select one:

a. The data sets must be sorted by the variable on the BY statement.

b. The DATA step with the MERGE statement combines data vertically.

c. The MERGE statement can combine only two data sets.

d. The variable on the BY statement must exist in one data set. ✖

If the data sets are NOT sorted by the variable on the BY statement, the program will fail. The DATA step with the MERGE statement combines data horizontally. The MERGE statement can combine two or more data sets. The variable on the BY statement must exist in all data sets.

The correct answer is: The data sets must be sorted by the variable on the BY statement.

1.00 points out of 1.00

```
    merge patient(in=inP) appointment(in=inA);
    by PatientID;
    insert-statement-here;
run;
```

Select one:

a.
```
if inP=1 and inA=0;
```

b.
```
if inP=1;
```

c.
```
if inP=1 and inA=1;
```

d.
```
if inP=1 or inA=1;
```

If the returned observation is in both **patient** and **appointment**, then the value for the data set contributor options (**inP** and **inA**) will both equal *1*.

The correct answer is:
```
if inP=1 and inA=1;
```

Question **23**

Correct

1.00 points out of 1.00

Review the data sets **sales_region1** and **sales_region2** and the DATA step below.

sales_region1

Obs	ProductID	Quantity	Location
1	0635	34	Greensboro
2	0385	12	High Point

sales_region2

Obs	ProdID	quantity	location
1	0689	25	Austin
2	0143	48	Dallas

```
data sales;
    set sales_region1 sales_region2;
    if quantity >= 0;
run;
```

How many columns will be in the data set **sales**?
Enter your numeric answer below.

Answer:

4

The four columns in sales will be **ProductID**, **Quantity**, **Location**, and **ProdID**.

The correct answer is: 4

0.00 points out of 1.00

Select one:

- a. SET statement ✖
- b. DO statement
- c. IF statement
- d. LENGTH statement

The LENGTH statement specifies the number of bytes for storing character and numeric variables at compilation of the step and is not an executable statement.

The correct answer is: LENGTH statement

Question **25**

Correct

1.00 points out of 1.00

Which action occurs first in the execution phase of the DATA step?

Select one:

- a. Calculations in the program are executed.
- b. The observation in the PDV is written to the output data set.
- c. The PDV is initialized. ✔
- d. An observation from the input data set is read into the PDV.

The first action in the execution phase of the DATA step is initializing the PDV, then the observations are read into the PDV, calculations in the program are executed and finally the observations in the PDV is written to the output data set.

The correct answer is: The PDV is initialized.

Question **26**

Incorrect

0.00 points out of 1.00

Review the data sets **sales_region1** and **sales_region2** and the DATA step below.

sales_region1

Obs	ProductID	Quantity	Location
1	0635	34	Greensboro
2	0385	12	High Point

sales_region2

Obs	ProdID	quantity	location
1	0689	25	Austin
2	0143	48	Dallas

```
data sales;
    set sales_region1(drop=Location) sales_region2;
    keep Quantity;
run;
```

Including **_ERROR_** and **_N_**, how many variables are in the program data vector?

Select one:

- a. 6
- b. 5
- c. 3 ✖
- d. 8

The program data vector will contain **ProductID**, **Quantity**, **ProdID**, **location**, **_ERROR_**, and **_N_**.

The correct answer is: 6

1.00 points out of 1.00

```
data birds;
    set sashelp.bird;
    insert-statement-here;
run;
```

Select one:

a.
```
where Text like 'error';
```

b.
```
where Text = 'error';
```

c.
```
where find 'error' in Text;
```

d.
```
where Text contains 'error';
```

✔

The CONTAINS operator tests whether a string is part of the variable's values.

The correct answer is:
```
where Text contains 'error';
```

Question **28**

Correct

1.00 points out of 1.00

Given the following program, which statement subsets only the observations where the value of **Discount** is greater than *5,000*?

```
data discount;
    set sashelp.cars;
    Discount=MSRP*.10;
    insert-statement-here;
run;
```

Select one:

a.
```
where Discount gt 5,000;
```

b.
```
if Discount > 5000;
```

✔

c.
```
where Discount > 5000;
```

d.
```
if Discount gt 5,000;
```

The subsetting IF statement subsets only those observations that meet the condition. The condition can include variables from the existing input data set or newly created variables in the step. When subsetting for numeric variables, numeric variables can include the digits 0 through 9 plus positive, negative, and E-notation. The WHERE statement is a preproccessor; you can use only variables from the existing input data set, not newly created variables in the step.

The correct answer is:
```
if Discount > 5000;
```

1.00 points out of 1.00

Select one:

a.
```
data underweight;
    set patients;
    BMI = weight_kg / height_meters **2;
    if BMI <= 18.5;
run;
```
✔

b.
```
data underweight;
    set patients(if= (BMI <=18.5));
    BMI = weight_kg / height_meters **2;
run;
```

c.
```
data underweight;
    set patients(where= (BMI <= 18.5));
    BMI = weight_kg / height_meters **2;
run;
```

d.
```
data underweight;
    set patients;
    BMI = weight_kg / height_meters **2;
    where BMI <= 18.5;
run;
```

BMI is calculated in the program data vector. The WHERE statement is evaluated before data is loaded into the program data vector. IF= is not a data set option. The subsetting IF statement filters rows by the given condition.

The correct answer is:
```
data underweight;
    set patients;
    BMI = weight_kg / height_meters **2;
    if BMI <= 18.5;
run;
```

Question **30**

Partially correct

0.50 points out of 1.00

Which two sentences are true regarding the assignment statement? (Choose two.)

Select one or more:

a. A variable can be dropped from a data set.

b. Character and numeric variables can be created.

c. The value of a variable can be updated. ✔

d. Formats can be assigned to a variable. ✖

The assignment statement is used to create a numeric or character variable. If the variable does not exist in the input data set, the assignment statement creates a new variable. If variable already exist in the input data set, the assignment statement updates the variable's value.

The correct answers are: Character and numeric variables can be created., The value of a variable can be updated.

1.00 points out of 1.00

```
    first = 5;
    second = .;
    third = 10;
    fourth = first + second + third;
run;
```

What is the value of **fourth** in the data set **work.sum**?

Select one:

- a. *first + second + third*
- b. *15*
- c. . ✔
- d. *5 + . + 10*

SAS sets the results of a calculation to missing if a missing value is used in an arithmetic expression.

The correct answer is: **.**

Question **32**

Correct

1.00 points out of 1.00

The **BirthDate** variable is equal to a value of *9205*, which corresponds to the date of Friday, March 15, 1985. Given the following statement, what is the value of **BirthMnth**?

```
BirthMnth=month(BirthDate);
```

Select one:

- a. *March*
- b. *03*
- c. *3* ✔
- d. *MAR*

The MONTH function extracts the numeric month value from the specified date value. The MONTH function returns only the integer 1-12 without a leading zero.

The correct answer is: *3*

Question **33**

Correct

1.00 points out of 1.00

Given the following output, which statement creates the variable **FirstLast**?

First	Last	FirstLast
Jane	Smith	Jane Smith
Todd	Brown	Todd Brown

Select one:

- a. `FirstLast=catx(First, ,Last);`
- b. `FirstLast=catx( ,First,Last);`
- c. `FirstLast=catx(' ',First,Last);` ✔
- d. `FirstLast=catx(First,' ',Last);`

The CATX function requires the delimiter to be specified first, followed by the variables you wish to concatenate.

The correct answer is: `FirstLast=catx(' ',First,Last);`

Question **34**

Correct

1.00 points out of 1.00

When using IF/THEN conditional logic in a DATA step, how many executable statements are permitted after the THEN?

Enter your numeric answer in the space.

Answer:

1

✔

The IF/THEN conditional logic statement in a DATA step permits one executable statement after the THEN. If you have more than one executable

0.00 points out of 1.00

```
    set sashelp.baseball ;
    if League='American' then Country='US';
    else if League='National' then Country='Other';
    length Country $ 6;
run;
```

Select one:

- a. 6 ✖
- b. 5
- c. 2
- d. 3

SAS determines the length of character variables from its first occurrence in the DATA step. Using the LENGTH statement after the variable's first occurrence in the DATA step does not change the length. To avoid truncation, use the LENGTH statement before the first occurrence of the character variable. 2 is the byte size because the first place SAS sees **Country** is the first IF/THEN, which has **Country='US'.** Therefore, the byte size is 2 because of US.

The correct answer is: 2

Question **36**

Correct

1.00 points out of 1.00

Which DATA statement is needed to complete this program?

```
insert-statement-here
    set sashelp.cars;
    if type="Sedan" or type="Wagon" then output cars;
    else if type="Truck" or type="SUV" then output trucks;
run;
```

Select one:

- a.
```
data cars trucks;
```
✔
- b.
```
data sedan wagon truck suv;
```
- c.
```
data Sedan Wagon Truck SUV;
```
- d.
```
data;
```

Here, in a DATA step, you must first specify the name of the new SAS data set in the DATA statement. On the conditional statements, the use of **output cars** and **output trucks** is the reason the statement **data cars trucks;** is correct.

The correct answer is:
```
data cars trucks;
```

0.00 points out of 1.00

Month	Principle
1	250.00
2	250.63
3	251.25
4	251.88

Select one:

a.
```
data savings;
    Month = 1;
    Principal = 250;
    format Principal comma10.2;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
run;
```

b.
```
data savings;
    Month = 1;
    Principal = 250;
    format Principal comma10.2;
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
run;
```

c.
```
data savings;
    output;
    Month = 1;
    Principal = 250;
    format Principal comma10.2;
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
run;
```

d.
```
data savings;
    Month = 1;
    Principal = 250;
    format Principal comma10.2;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
run;
```

✖

Because there is no looping in these DATA steps, there must be one OUTPUT statement per observation to create the pictured data set.

The correct answer is:
```
data savings;
    Month = 1;
```

```
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
    Month = Month + 1;
    Principal = Principal * (1+.03/12);
    output;
run;
```

1.00 points out of 1.00

Month	Sales
1	5
1	5
1	5
2	5
2	5
2	5
3	10
3	10
3	10

The following code is submitted:

```
data Total;
    set sales;
    by month;
    if first.month=1 then TotalSales=0;
    TotalSales + Sales;
    insert-statement-here
run;
```

The data set **Total** is shown below:

Month	Sales	TotalSales
1	5	5
1	5	10
1	5	15
2	5	5
2	5	10
2	5	15
3	10	10
3	10	20
3	10	30

Which subsetting IF statement correctly selects the final row of each month to produce the data below?

Month	Sales	TotalSales
1	5	15
2	5	15
3	10	30

Select one:

a.
```
if last=1;
```

b.
```
if last.month=1;
```

✔

c.
```
if last.month=0;
```

processed and written to the new SAS data set

The correct answer is:
```
if last.month=1;
```

Question **39**

Incorrect

0.00 points out of 1.00

Given the input data set **sales** shown below:

Location	Quantity
Austin	15
Dallas	25
Houston	5
Amarillo	.

The following code is submitted:
```
data accumulate;
    set sales;
    insert-statement-here
run;
```

Which statement correctly creates an accumulating column for total **Quantity**?

Select one:

a.
```
if Quantity ^= . then Total = sum(Total, Quantity);
```

b.
```
Total = Total + Quantity;
```

c.
```
Total = sum(Total, Quantity);
```

d.
```
Total + Quantity;
```

Only the SUM statement retains values from row to row, without a separate RETAIN statement.

The correct answer is:
```
Total + Quantity;
```

Question **40**

Correct

1.00 points out of 1.00

Given the following DATA step, how many observations are in the output data set **work.savings**?
```
data work.savings;
    amount=5000;
    do year=1 to 5;
        do month=1 to 12;
            amount+(amount*0.03/12);
        end;
       output;
    end;
run;
```

Enter your numeric answer in the space below.

Answer:

5

The explicit OUTPUT statement is outside of the inner DO loop. The step outputs one observation per each iteration of the outer DO loop up to YEAR=5, when **YEAR** passes the value of *5*, execution of the DO loop stops.

The correct answer is: 5

0.00 points out of 1.00

```
    do i=0 to 10 by 2;
        equation=i/2;
    end;
run;
```

Select one:

- a. .
- b. 10 ✖
- c. 12
- d. 11

The value of the index variable **i** is *12* in the output data set **work.mydata**. The value of the index variable is incremented at the bottom of the DO loop, when **i=10**, **i** gets incremented by 2, **i=12** passes the value of the STOP value of *10*, so when the step returns to the top of the loop, **i** is no longer within range of the STOP value and the DO loop stop execution. After the DO loop stops, the implied OUTPUT is executed, so the value of *12* is written to the output data set.

The correct answer is: 12

Question **42**

Correct

1.00 points out of 1.00

Which step permanently adds labels and formats to a data set?

Select one:

- a. PROC PRINT step
- b. PROC LABELS step
- c. DATA step ✔
- d. PROC FORMAT step

The DATA step creates data and any attributes assigned by the LABEL and FORMAT statements in the DATA step are permanently stored with the data set.

The correct answer is: DATA step

Question **43**

Correct

1.00 points out of 1.00

The data set **HeadofState1900** has the variable **Country** with a length of 20 and the variable **Title** with a length of 30. The following SAS program is submitted:

```
data HeadOfState1900_cleaned;
    length Country $ 15;
    set HeadofState1900;
    Country = propcase(Country);
    Title = upcase(Title);
    Date = mdy(1,1,1900);
    format Date date9.;
run;
```

What are the lengths of the variables **Country** and **Title** in the output data set?

Select one:

- a. **Country** has a length of 15, and **Title** has a length of 8.
- b. **Country** has a length of 20, and **Title** has a length of 30.
- c. **Country** has a length of 8, and **Title** has a length of 8.
- d. **Country** has a length of 15, and **Title** has a length of 30. ✔

The first time **Country** is referenced in the DATA step, it has a length of 15. The first time **Title** is referenced (through the SET statement), it has a length of 30.

The correct answer is: **Country** has a length of 15, and **Title** has a length of 30.

1.00 points out of 1.00

Select one:

- a. NONUM
- b. NOPRINT
- c. NOOBS ✔
- d. NOROW

The NOOBS option suppresses the display of the observation number in the PROC PRINT report.

The correct answer is: NOOBS

Question **45**

Correct

1.00 points out of 1.00

Which is a default behavior of PROC PRINT?

Select one:

- a. display observation numbers ✔
- b. display only the top 10 observations of a data set
- c. display variable labels
- d. display variables in alphabetical order

By default, PROC PRINT displays all variables and observations from a dataset. It does not display labels, and it does display row numbers.

The correct answer is: display observation numbers

Question **46**

Correct

1.00 points out of 1.00

Which PROC MEANS option controls the number of decimal places displayed in the report?

Select one:

- a. DECIMALS=
- b. MEANDEC
- c. MAXDEC= ✔
- d. NODEC

The MAXDEC= option is one of the options that controls how your data are summarized. It is the only option that specifies the number of decimal places to be displayed.

The correct answer is: MAXDEC=

0.00 points out of 1.00

SepalLength	Sepal Length (mm)	104	62.6	6.2	54.0	79.0
SepalWidth	Sepal Width (mm)	104	29.6	4.2	22.0	44.0
PetalLength	Petal Length (mm)	104	46.5	12.7	12.0	69.0
PetalWidth	Petal Width (mm)	104	15.7	5.8	2.0	25.0

The Means Procedure

Select one:

a.
```
proc means data=sashelp.iris maxdec=1;
    class Species;
run;
```

✖

b.
```
proc means data=sashelp.iris maxdec=1;
    where SepalLength > 53;
run;
```

c.
```
proc means data=sashelp.iris maxdec=1;
    by Species;
run;
```

d.
```
proc means data=sashelp.iris maxdec=1;
    var petalWidth;
run;
```

The default of the MEANS procedure is to generate statistics for all numeric variables in the data set if no VAR statement is applied in the step.

The correct answer is:
```
proc means data=sashelp.iris maxdec=1;
    where SepalLength > 53;
run;
```

Question **48**

Correct

1.00 points out of 1.00

Which statement is true regarding the FREQ procedure?

Select one:

a. By default, the FREQ procedure creates a frequency table for each character variable in the input data set.

b. The FREQ procedure can produce only a one-way frequency table.

c. The ORDER=FREQ option sorts the input data set in the FREQ procedure.

d. The TABLES statement specifies the variables that are analyzed in the FREQ procedure. ✔

The TABLES statement specifies the variables that are analyzed in the FREQ procedure. By default, a PROC FREQ step without a TABLES statement creates a frequency table for each character and numeric variable in the input data set. You can produce one-way or n-way tables and the ORDER=FREQ option produces a report with descending frequency counts.

The correct answer is: The TABLES statement specifies the variables that are analyzed in the FREQ procedure.

1.00 points out of 1.00

Product	Frequency	Cumulative Frequency
Boot	52	52
Men's Casual	45	97
Men's Dress	50	147
Sandal	49	196
Slipper	52	248
Sport Shoe	51	299
Women's Casual	45	344
Women's Dress	51	395

The FREQ procedure

```
proc freq data=sashelp.shoes;
    tables product / insert-option(s)-here;
run;
```

Select one:

- a. NOCUM NOPERCENT
- b. NOPERCENT ✔
- c. NOCUM NOFREQ
- d. NOFREQ

NOPERCENT suppresses the percent and cumulative percent statistics.

The correct answer is: NOPERCENT

Question **50**

Incorrect

0.00 points out of 1.00

Which PROC FREQ step displays the data in the report by descending frequency?

Select one:

a.
```
proc freq data=sashelp.cars / order=freq;
    tables type;
run;
```

b.
```
proc freq data=sashelp.cars;
    tables type / order=freq;
run;
```

c.
```
proc freq data=sashelp.cars;
    tables type order=freq;
run;
```

✖

d.
```
proc freq data=sashelp.cars order=freq;
    tables type;
run;
```

The ORDER=FREQ option must be specified in the PROC FREQ statement itself in order to create a report by descending frequency.

The correct answer is:
```
proc freq data=sashelp.cars order=freq;
    tables type;
run;
```

1.00 points out of 1.00

Select one:

- a. **title=clear;** clears out all previously defined titles.
- b. The TITLE statement must be in a PROC step.
- c. A report can only use one TITLE statement.
- d. The TITLE statement is a global statement. ✔

The TITLE statement is a global statement that can be used outside or inside of a PROC step. A report can have from 1 to 10 titles and a null TITLE statement (TITLE;) clears all previously defined titles.

The correct answer is: The TITLE statement is a global statement.

Question **52**

Correct

1.00 points out of 1.00

The following PROC PRINT step is submitted.

```
proc print data=work.expenses;
run;
```

The following report is created:

Obs	Name	Date	Amount
1	Jill	21735	1134.22
2	Tony	21741	7500.00
3	Fred	21745	10765.95

The following report is desired:

Obs	Name	Date	Amount
1	Jill	05JUL2019	1,134
2	Tony	11JUL2019	7,500
3	Fred	15JUL2019	10,766

Which FORMAT statement can be added to the PROC PRINT step to produce the desired report?

Select one:

a.
```
format Date date9 Amount comma8;
```

b.
```
format Date=date9. Amount=comma8.;
```

c.
```
format Date date9. Amount comma8.;
```
✔

d.
```
format Date(date9) Amount(comma8);
```

The FORMAT statement **format Date date9. Amount comma8.;** associates the DATE9. numeric format with the variable **Date** and displays the date as 05JUL2019. It also associates the COMMA8. format with the variable **Amount**, displaying the values with commas.

The correct answer is:
```
format Date date9. Amount comma8.;
```

0.00 points out of 1.00

```
    class position;
    var nRuns nRBI;
run;
```

Select one:

a.
```
where position='1B' or '2B' or '3B';
```

✖

b.
```
where position in ('1B' '2B' '3B');
```

c.
```
if position in ('1B','2B','3B');
```

d.
```
if position='1B' or '2B' or '3B';
```

Subsetting IF statements can only appear in DATA steps, therefore the two options with IF statements are not correct. In SAS, WHERE statements can be used in both DATA and PROC steps. The syntax for the IN operator is appropriate for subsetting for a list of values. The = operator would be appropriate if position = was specified in front of each value such as **position = '1B'**, **position ='2B'**, **position ='3B'**.

The correct answer is:
```
where position in ('1B' '2B' '3B');
```

Question **54**

Incorrect

0.00 points out of 1.00

Which program sorts the data set **sashelp.cars** in descending order by **Make**?

Select one:

a.
```
proc sort data=sashelp.cars;
    by Make descending;
run;
```

✖

b.
```
proc sort data=sashelp.cars;
    by descending Make;
run;
```

c.
```
proc sort data=sashelp.cars desc;
    order Make;
run;
```

d.
```
proc sort data=sashelp.cars order=desc;
    sortby Make;
run;
```

By default, SAS sorts data in ascending order, from lowest to highest. To have your data sorted in the opposite order (which is what is called for in this question) you must add the keyword DESCENDING to the BY statement before each variable that should be sorted in the reverse order.

The correct answer is:
```
proc sort data=sashelp.cars;
    by descending Make;
run;
```

Question **55**

Correct

1.00 points out of 1.00

Which statement about PROC SORT is true?

Select one:

a. PROC SORT can sort the data by only one variable.

b. PROC SORT will display the sorted data in a report.

c. PROC SORT overwrites the original data set by default. ✔

d. PROC SORT sorts the data in descending order by default.

The PROC SORT step overwrites the original data set by default, sorts the data in ascending order by default, and can sort the data by one or more variables.

The correct answer is: PROC SORT overwrites the original data set by default.

1.00 points out of 1.00

Select one:

a.
```
proc sort data=sashelp.class out=sortclass;
    by Age Name;
run;
```

b.
```
proc sort data=sashelp.class data=sortclass;
    by Age Name;
run;
```

c.
```
proc sort data=sashelp.class;
    by Name Age;
    output sortclass;
run;
```

d.
```
proc sort data=sashelp.class;
    by Name Age;
    output out=sortclass;
run;
```

The OUT= option specifies the name of the sorted data set you want to create. The BY statement specifies the sort variables in the order listed.

The correct answer is:
```
proc sort data=sashelp.class out=sortclass;
    by Age Name;
run;
```

Question **57**

Incorrect

0.00 points out of 1.00

The following SAS program is submitted:

```
proc sort data=SASUSER.PROJECTS out=PSORT;
    by Product Code descending Date Cost;
run;
```

Which of the following is true concerning the submitted program?

Select one:

a. The DESCENDING option applies only to the **Code** variable. ✖

b. The DESCENDING option applies to the **Date** and **Cost** variables.

c. The DESCENDING option applies only to the **Date** variable.

d. The DESCENDING option applies to the **Product** and **Code** variables.

The DESCENDING option must proceed the variable you wish to be in the descending order.

The correct answer is: The DESCENDING option applies only to the **Date** variable.

0.00 points out of 1.00

Select one:

a.
```
proc format;
    value 1,3,5='odds'
          2,4,6='evens';
run;
```

b.
```
proc format;
    value codes 1-3='low'
                4-6='medium'
                7-9='high';
run;
```

c.
```
proc format;
    value mult 0,1,5='No'
                  2='Yes'
                6-8='Unsure';
run;
```

✖

d.
```
proc format;
    value grade 1='A'
                2='B'
                3='C';
run;
```

In the VALUE statement, after the keyword VALUE, you must specify the name of the format before specifying the values to format.

The correct answer is:
```
proc format;
    value 1,3,5='odds'
          2,4,6='evens';
run;
```

Question **59**

Correct

1.00 points out of 1.00

Which PROC EXPORT step correctly creates the **storms.csv** file from the **mydata.storms** data set?

Select one:

a.
```
proc export data=mydata.storms outfile="storms.csv";
            dbms=csv;
run;
```

b.
```
proc export data=mydata.storms outfile="storms.csv"
            dbms=csv;
run;
```

✔

c.
```
proc export data=storms.csv outfile=mydata.storms;
            dbms=csv;
run;
```

d.
```
proc export data=mydata.storms outfile=storms.csv
            dbms=csv;
run;
```

The general form for PROC EXPORT statement is:

```
PROC EXPORT DATA = data-set OUTFILE = 'filename' options;
```

The OUTFILE= option specifies the filename and extension in a set of quotes. You can specify the file type by adding the DBMS= option to the PROC EXPORT statement. The DBMS= option is **not** a standalone statement.

The correct answer is:
```
proc export data=mydata.storms outfile="storms.csv"
            dbms=csv;
run;
```

1.00 points out of 1.00

Select one:

a.
```
libname exlib "c:\Class.xlsx";
```

b.
```
libname exlib xlsx "c:\Class";
```

c.
```
libname exlib xlsx "c:\Class.xlsx";
```

✔

d.
```
libname xlsx exlib "c:\Class.xlsx";
```

In SAS, the syntax for the LIBNAME statement is:

```
LIBNAME libref  XLSX  fully-qualified path of the Excel workbook including the XLSX extension;
```

The correct answer is:
```
libname exlib xlsx "c:\Class.xlsx";
```

Question **61**

Correct

1.00 points out of 1.00

Which program correctly sends the PROC PRINT results to an RTF file named **shoes**?

Select one:

a.
```
ods rtf out='C:\shoes.rtf';
proc print data=sashelp.shoes;
run;
ods rtf close;
```

b.
```
ods file='C:\shoes.rtf';
proc print data=sashelp.shoes;
run;
ods close;
```

c.
```
ods rtf file='C:\shoes.rtf';
proc print data=sashelp.shoes;
run;
ods rtf close;
```

✔

d.
```
ods rtf 'C:\shoes.rtf';
proc print data=sashelp.shoes;
run;
ods rtf close;
```

In order to send the PROC PRINT results to an RTF file, you must use the ODS statement that specifies the destination and the FILE= option to specify the name of the file in quotation marks. You then need to use the ODS RTF CLOSE statement to specify that you want to stop writing to the RTF file.

The correct answer is:
```
ods rtf file='C:\shoes.rtf';
proc print data=sashelp.shoes;
run;
ods rtf close;
```

```
proc sort data=orion.employee_addresses out=SASProj.address;
        where Country='US';
        by State City Employee_Name;
run;
proc print data=SASProj.address noobs split=' ';
        var Employee_ID Employee_Name City Postal_Code;
        label Employee_ID='Employee ID'
                Employee_Name='Name'
                Postal_Code='Zip Code';
  by State;
run;


proc print data=orion.sales label noobs;
        where Country='US' and Salary<26000;
        var Employee_ID First_Name Last_Name Job_Title Salary Hire_Date;
        label
                First_Name='First Name'
                Last_Name='Last Name'
                Job_Title=' Title'
                Hire_Date='Date Hired';
        format Salary dollarlO. Hire_Date monyy7.;
run;
data SASProj.delays;
        set orion.orders;
        where (Delivery_Date - Order_Date) > 4 and Employee_ID = 99999999;
        Order_Month= month(Order_Date);
        if Order_Month=8;
        label Order_Date='Date Ordered'
                Delivery_Date='Date Delivered'
                Order_Month='Month Ordered';
                format Order_Date Delivery_Date mmddyy10.;
        keep Employee_ID Customer_ID Order_Date Delivery_Date Order_Month;
run;
proc contents data=SASProj.delays;
run;
proc print data=SASProj.delays;
run;
```

```
proc format;
        value $gender
                'F'='Female'
                'M'='Male'
                other='Invalid code';
        value salrange .='Missing salary'
                20000-<100000='Below $100,000'
                100000-500000='$100,000 or more'
                other='Invalid salary';
run;
proc print data=orion.nonsales;
        var Employee_ID Job_Title Salary Gender;
        format Salary salrange. Gender $gender.;
run;


options validvarname=v7;
*libname out excel path="&path\employees.xls";
data out.salesemps;
        length First_Name $ 12 Last_Name $ 18 Job_Title $ 25;
        infile "/home/u63514409/SASUFS/newemps.csv" dlm=',';
        input First_Name $ Last_Name $ Job_Title $ Salary;
run;
libname out clear;
data SASProj.ManagementTeam;
  infile datalines dlm='/' truncover;
  input ID First $ Last $ Gender $ Salary :comma8. Title $ HireDate :date11.;
  format HireDate mmddyy10. Salary comma8.;
  datalines;
120102/Tom/Zhou/M/108,255/Sales Manager/01Jun1993
120103/Wilson/Dawes/M/87,975/Sales Manager/01Jan1978;
run;
title1 'Section 4.2:a-b';
title2 'Orion Star Management Team';
footnote 'Motsamai LT (2016206381)';
proc print data=SASProj.ManagementTeam noobs;
  var First Last Title ID Gender Salary HireDate;
run;
```