

SAS[®] Programming 1: Essentials

Course Notes

SAS® Programming 1: Essentials Course Notes was developed by Charlot Bennett and Kathy Passarella. Additional contributions were made by Davetta Dunlap, Michele Ensor, Susan Farmer, Ted Meleky, Linda Mitterling, Bill Powers, Jim Simon, and Roger Staum. Editing and production support was provided by the Curriculum Development and Support Department.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

SAS® Programming 1: Essentials Course Notes

Copyright © 2013 SAS Institute Inc. Cary, NC, USA. All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

Book code E2132, course code LWPRG1/PRG1, prepared date 14Feb2013.

LWPRG1_003

ISBN 978-1-61290-489-4

Table of Contents

Course Description	ix
Prerequisites	x
Chapter 1 Introduction	1-1
1.1 Overview of SAS Foundation.....	1-3
1.2 Course Logistics	1-6
Demonstration: Investigating the Help Facility	1-12
1.3 Course Data Files.....	1-14
Demonstration: Creating Course Data Files.....	1-15
Exercises.....	1-16
Chapter 2 SAS® Programs.....	2-1
2.1 Introduction to SAS Programs	2-3
2.2 Submitting a SAS Program.....	2-7
Demonstration: Submitting a SAS Program: SAS Enterprise Guide.....	2-8
Demonstration: Submitting a SAS Program: SAS Windowing Environment	2-12
Exercises.....	2-18
2.3 SAS Program Syntax	2-20
Demonstration: Automatic Formatting with SAS Enterprise Guide.....	2-23
Demonstration: Diagnosing and Correcting Syntax Errors.....	2-27
Demonstration: Correcting Unbalanced Quotation Marks: SAS Windowing Environment	2-30
Demonstration: Correcting Unbalanced Quotation Marks: SAS Enterprise Guide	2-33
Exercises.....	2-35
2.4 Solutions	2-36
Solutions to Exercises	2-36

Solutions to Student Activities (Polls/Quizzes).....	2-38
Chapter 3 Accessing Data.....	3-1
3.1 Examining SAS Data Sets	3-3
Exercises.....	3-12
3.2 Accessing SAS Libraries	3-13
Demonstration: Browsing SAS Libraries: SAS Enterprise Guide	3-25
Demonstration: Browsing SAS Libraries: SAS Windowing Environment.....	3-29
Exercises.....	3-33
3.3 Solutions	3-34
Solutions to Exercises	3-34
Solutions to Student Activities (Polls/Quizzes).....	3-36
Chapter 4 Producing Detail Reports.....	4-1
4.1 Subsetting Report Data	4-3
Exercises.....	4-24
4.2 Sorting and Grouping Report Data	4-27
Exercises.....	4-38
4.3 Enhancing Reports	4-40
Exercises.....	4-48
4.4 Solutions	4-50
Solutions to Exercises	4-50
Solutions to Student Activities (Polls/Quizzes).....	4-54
Chapter 5 Formatting Data Values.....	5-1
5.1 Using SAS Formats.....	5-3
Exercises.....	5-10
5.2 User-Defined Formats.....	5-12
Demonstration: Defining and Using a Numeric Format	5-18

Exercises.....	5-23
5.3 Solutions	5-25
Solutions to Exercises	5-25
Solutions to Student Activities (Polls/Quizzes).....	5-28
Chapter 6 Reading SAS® Data Sets	6-1
6.1 Reading a SAS Data Set	6-3
Exercises.....	6-15
6.2 Customizing a SAS Data Set	6-17
Exercises.....	6-39
6.3 Solutions	6-42
Solutions to Exercises	6-42
Solutions to Student Activities (Polls/Quizzes).....	6-45
Chapter 7 Reading Spreadsheet and Database Data.....	7-1
7.1 Reading Spreadsheet Data	7-3
Demonstration: Reading Excel Data in SAS Enterprise Guide	7-10
Demonstration: Reading Excel Data in the SAS Windowing Environment	7-13
Demonstration: Creating a SAS Data Set	7-16
Exercises.....	7-17
7.2 Reading Database Data	7-20
7.3 Solutions	7-24
Solutions to Exercises	7-24
Solutions to Student Activities (Polls/Quizzes).....	7-26
Chapter 8 Reading Raw Data Files	8-1
8.1 Introduction to Reading Raw Data Files.....	8-3
8.2 Reading Standard Delimited Data.....	8-7
Demonstration: Examining Data Errors	8-33

Exercises.....	8-34
8.3 Reading Nonstandard Delimited Data	8-37
Demonstration: Using List Input: Importance of Colon Format Modifier.....	8-48
Exercises.....	8-51
8.4 Handling Missing Data	8-54
Exercises.....	8-60
8.5 Solutions	8-63
Solutions to Exercises	8-63
Solutions to Student Activities (Polls/Quizzes).....	8-65
Chapter 9 Manipulating Data.....	9-1
9.1 Using SAS Functions.....	9-3
Exercises.....	9-10
9.2 Conditional Processing	9-12
Exercises.....	9-37
9.3 Solutions	9-40
Solutions to Exercises	9-40
Solutions to Student Activities (Polls/Quizzes).....	9-44
Chapter 10 Combining Data Sets.....	10-1
10.1 Concatenating Data Sets	10-3
Exercises.....	10-22
10.2 Merging Data Sets One-to-One.....	10-25
10.3 Merging Data Sets One-to-Many.....	10-30
Exercises.....	10-47
10.4 Merging Data Sets with Non-Matches.....	10-49
Exercises.....	10-71
10.5 Solutions	10-73

Solutions to Exercises	10-73
Solutions to Student Activities (Polls/Quizzes).....	10-77
Chapter 11 Creating Summary Reports	11-1
11.1 The FREQ Procedure.....	11-3
Exercises.....	11-24
11.2 The MEANS and UNIVARIATE Procedures	11-28
Exercises.....	11-40
11.3 Using the Output Delivery System	11-44
Demonstration: Using the Output Delivery System.....	11-55
Exercises.....	11-62
11.4 Solutions	11-65
Solutions to Exercises	11-65
Solutions to Student Activities (Polls/Quizzes).....	11-69
Chapter 12 Learning More.....	12-1
12.1 Introduction.....	12-3
Appendix A Self-Study.....	A-1
A.1 Submitting Programs in UNIX and z/OS.....	A-3
Demonstration: Submitting a Program in the SAS Windowing Environment: UNIX	A-3
Demonstration: Submitting a Program in the SAS Windowing Environment: z/OS	A-7
A.2 Accessing Data.....	A-10
Exercises.....	A-22
A.3 Manipulating Data	A-23
Exercises.....	A-29
A.4 Combining Data Sets	A-30

Exercises.....	A-52
A.5 Advanced Reporting Techniques	A-54
Exercises.....	A-83
A.6 Introduction to SAS/GRAPH.....	A-91
Demonstration: Creating Bar and Pie Charts	A-101
Demonstration: Creating Plots	A-111
Demonstration: Enhancing Output.....	A-115
A.7 Solutions	A-118
Solutions to Exercises	A-118

Course Description

This course is for users who want to learn how to write SAS programs. It is the entry point to learning SAS programming and is a prerequisite to many other SAS courses. If you do not plan to write SAS programs and you prefer a point-and-click interface, you should attend the SAS® Enterprise Guide® 1: Querying and Reporting course.

To learn more...



For information about other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the Web at support.sas.com/training/ as well as in the Training Course Catalog.



For a list of other SAS books that relate to the topics covered in this Course Notes, USA customers can contact our SAS Publishing Department at 1-800-727-3228 or send e-mail to sasbook@sas.com. Customers outside the USA, please contact your local SAS office.

Also, see the Publications Catalog on the Web at support.sas.com/pubs for a complete list of books and a convenient order form.

Prerequisites

Before attending this course, you should have experience using computer software. Specifically, you should be able to

- understand file structures and system commands on your operating systems
- access data files on your operating systems.

No prior SAS experience is needed. If you do not feel comfortable with the prerequisites or are new to programming and think that the pace of this course might be too demanding, you can take the SAS® Programming Introduction: Basic Concepts course before attending this course. SAS® Programming Introduction: Basic Concepts is designed to introduce you to computer programming and presents a portion of the SAS® Programming 1: Essentials material at a slower pace.

Chapter 1 Introduction

1.1 Overview of SAS Foundation	1-3
1.2 Course Logistics	1-6
Demonstration: Investigating the Help Facility.....	1-12
1.3 Course Data Files	1-14
Demonstration: Creating Course Data Files	1-15
Exercises	1-16

1.1 Overview of SAS Foundation

Objectives

- Characterize SAS software.
- Describe the functionality of Base SAS and SAS Foundation tools.

3

What Is SAS?

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



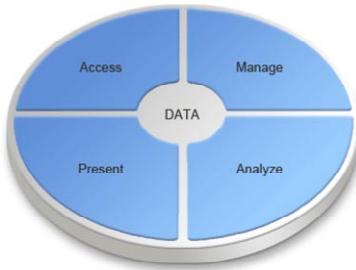
 sas

4

What Can You Do with SAS?

SAS software enables you to do the following:

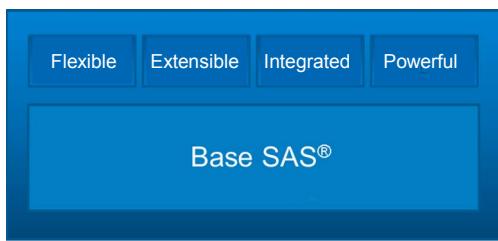
- access data across multiple sources
- manage data
- perform sophisticated analyses
- deliver information across your organization



5

What Is Base SAS?

Base SAS is the centerpiece of all SAS software.



Base SAS is a product within the SAS Foundation set of products that provides

- a highly flexible, highly extensible fourth-generation programming language
- a rich library of encapsulated programming procedures
- a graphic user interface for administering SAS tasks.

6

About This Class

This class focuses on writing SAS programs to do the following:



- access data in various forms
- create SAS data sets
- use prewritten procedures to analyze data and write basic reports
- combine data sets
- generate detail and summary reports in various formats, including HTML, RTF, PDF

7

1.01 Multiple Choice Poll

Have you worked with Base SAS?

- a. yes, just maintaining programs
- b. yes, writing some programs
- c. no, not at all

8

1.2 Course Logistics

Objectives

- Describe the data used in the course.
- Designate the editors and processing mode available for workshops.
- Specify the naming convention used for course files.
- Define the three levels of exercises.
- Navigate the Help facility.

11

Orion Star Sports & Outdoors

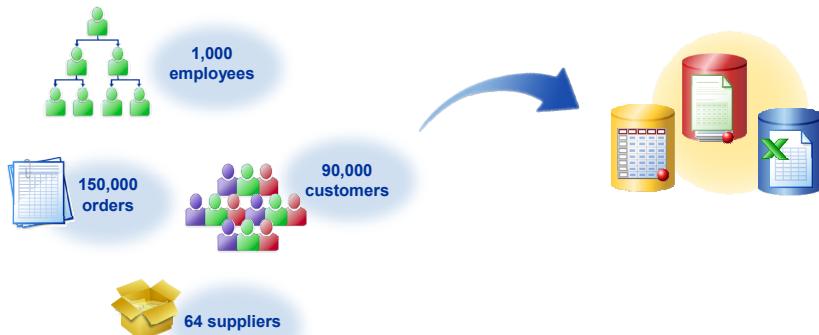
This course focuses on a fictitious global sports and outdoors retailer that has traditional stores, an online store, and a catalog business.



12

Orion Star Data

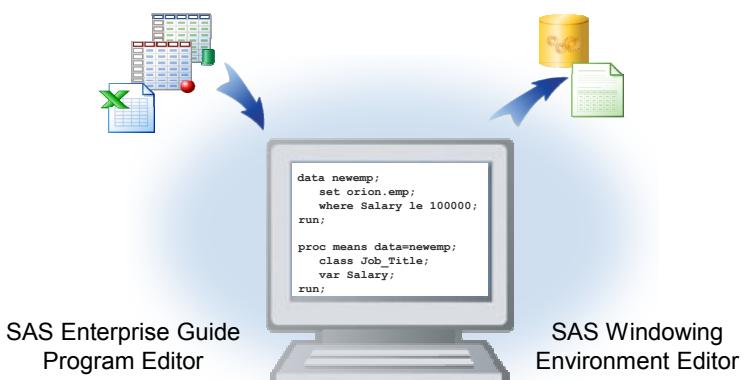
Large amounts of data are stored in transactional systems in various formats.



13

Orion Star Business Scenarios

In this course, you **write** SAS programs that access Orion Star data and create reports using an editor.



14

What Is SAS Enterprise Guide?



SAS Enterprise Guide is a powerful Windows client application that provides a GUI for transparently accessing the power of SAS.

It provides the following:

- a point-and-click interface with menus and wizards that enable the user to define tasks
- SAS code generation and execution based on user selections
- a full programming interface that can be used to write, edit, and submit SAS code

This class uses the programming interface.

15

What Is the SAS Windowing Environment?



The *SAS windowing environment* consists of a series of windows that you can use to edit and submit programs, and view the results.

The SAS windowing environment editor contains the following windows:

- the Enhanced Editor and Program Editor windows for preparing and submitting a program
- the Log window for viewing notes, warning messages, and error messages
- the Output window, which contains the output generated by most SAS procedures

16

	SAS Enterprise Guide	SAS Windowing Environment
Editor	Program Editor	Enhanced Editor or Program Editor
Formatting	automatic	manual
Syntax Help	context-sensitive	menu- or function key-based
Output	SAS Report	HTML

	SAS Enterprise Guide	SAS Windowing Environment
Projects	yes	no
Autocomplete	yes	no
Program Flow Analysis	yes	no

1.02 Multiple Choice Poll

Which editor will you use outside of this class to write SAS programs?

- a. SAS Enterprise Guide Program Editor
- b. the Program Editor in the SAS windowing environment
- c. a different editor
- d. I do not know.

17

Running SAS Programs

In this course, you invoke SAS in interactive mode (SAS Enterprise Guide or windowing environment) to **process** programs.



18

Running SAS Programs

There are other modes for processing SAS programs.

Batch Mode for z/OS (OS/390)	Noninteractive Mode
<p>Use any editor to create a file with SAS statements plus job control statements (JCL), and then submit the file to the operating system.</p> <p>Example file:</p> <pre>//jobname JOB ... // EXEC SAS //SYSIN DD * proc freq data=x.pay; tables ID; run;</pre>	<p>Use any editor to create a file with SAS statements, and then issue the SAS command referencing the file.</p> <p>Directory-based example:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">SAS <i>filename</i></div> <p>z/OS (OS/390) example:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">SAS INPUT(<i>filename</i>)</div>

19

1.03 Multiple Choice Poll

How will you invoke SAS outside of this class?

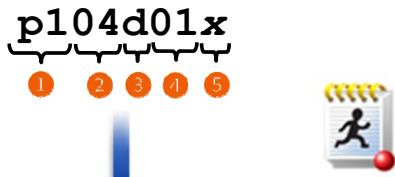
- a. interactive mode using SAS Enterprise Guide
- b. interactive mode using the windowing environment
- c. batch mode
- d. noninteractive mode
- e. I do not know.

20

Program Naming Conventions

In this course, you retrieve and save SAS programs using the structure below.

- ① course ID
- ② chapter #
- ③ type
a=activity
d=demo
e=exercise
s=solution
- ④ item #
- ⑤ placeholder



Programming 1, Chapter 4, Demo 1

21

Filename and Library Name References

In this course, macro variable references are used to give a more flexible approach for locating files.

Examples:

```
%let path=s:\workshop;
```

```
filename sales "&path\sales.dat";
```

```
infile "&path\payroll.dat";
```

22

Three Levels of Exercises

The course is designed to have you complete only **one** set of exercises. Select the level most appropriate for your skill set.

Level 1 Provides step-by-step instructions.

Level 2 Provides less information and guidance.

Challenge Provides minimal information and guidance.
You might need to use the Help facility.

23

Getting Help

In class, you can get product help in several ways, depending on the editor being used.

- Getting Started tutorials
- Help facilities included in the software
- web-based help, if web access is available



24



Investigating the Help Facility

The demonstration illustrates how to access and explore the Help facility in Enterprise Guide and in the windowing environment.

SAS Enterprise Guide

1. Invoke a SAS Enterprise Guide session.
2. Select **Help** from the menu.
3. Investigate the Help facility.

SAS Windowing Environment

1. Invoke a SAS session.
2. Select **Help** from the menu.
3. Investigate the Help facility.

1.04 Multiple Choice Poll

Were you able to open the Help facility in your session?

- a. I opened Help in Enterprise Guide.
- b. I opened Help in SAS.
- c. I was not able to open Help.

Extending Your Learning

After class, you will have access to an extended learning page that was created for this course. The page includes

- course data and program files
- a PDF file of the course notes
- other course-specific resources.



- ✍ This page might also be available during class.

28

1.3 Course Data Files

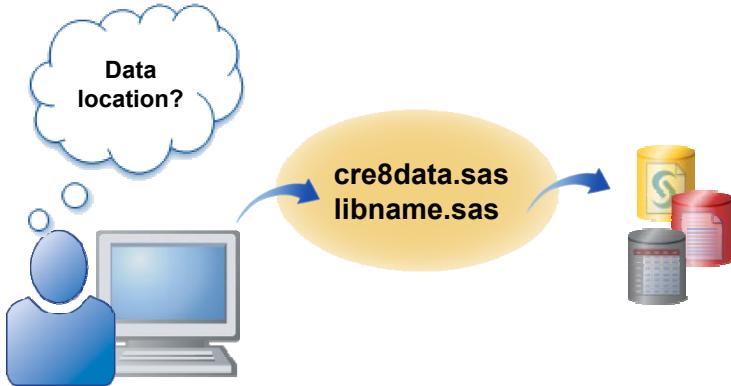
Objectives

- Execute a SAS program to create the course data files.
- Execute a SAS program to define the data location.

31

Business Scenario

Identify a location for the course data files and execute programs to create the files and define the location.



32



Creating Course Data Files

cre8data

The **cre8data** program creates data files for this course. The program must be executed once, at the start of the *course*.

1. Define target locations for your course data files. The default location for all course data is **s:\workshop**. If your data files are to be created in a location other than **s:\workshop**, you must identify a location for the SAS data files.
- Create the SAS data files here: _____
2. Select **File** ⇒ **Open** ⇒ **Program**.
 3. Navigate to the data folder, select **cre8data**, and click **Open**. The program is displayed in an editor.
 4. Note the default values for the %LET statement. If your files are to be created at a location other than **s:\workshop**, change the value assigned to PATH= to reflect the location of the SAS data files.



If your files are to be created in **s:\workshop**, then no change is needed.

5. Press F3 to submit the program.
6. Click the **Results** tab and verify that the output contains a list of data files.

Defining the Data Location

libname

The **libname** program tells SAS where to find the course data files. This program must be executed each time you start a new *session*.

1. Open the **libname** program. The LIBNAME statement starts with an asterisk. This means it is commented out and SAS will not execute this statement. At this point, you only want SAS to execute the %LET statement to tell SAS where the data files are located. You learn about the LIBNAME statement and the use of comments later in this course.

```
%let path=s:\workshop;
*libname orion "s:\workshop";
```



The data location might be different in your **libname** program. It was defined based on the data location specified in **cre8data**.

2. Submit the program. Click the **Log** tab and verify that there are no errors or warnings.



Exercises



You **must** complete the exercises to create the course data files. If you do not create the data files, all programs in this course will fail.

Required Exercise

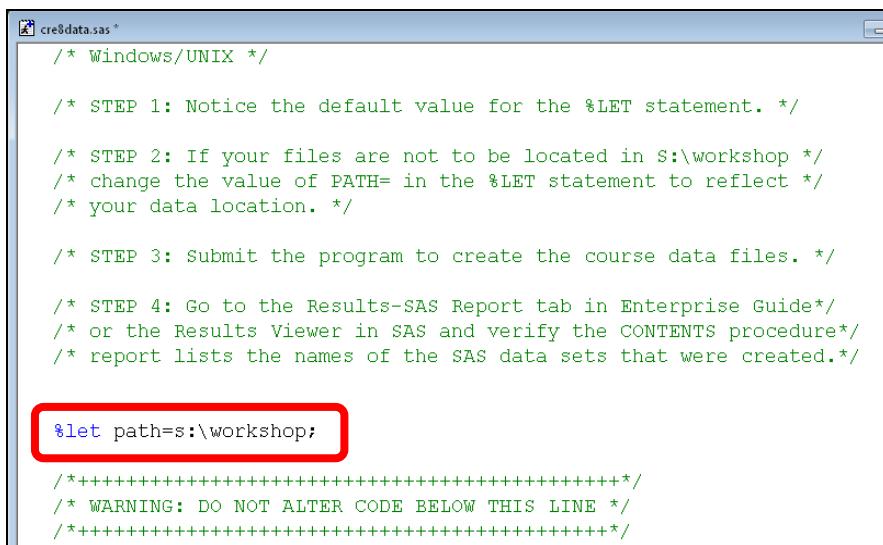
1. Creating Course Data

- a. The default location for all course data is **s:\workshop**. If your data files are to be created in a location other than **s:\workshop**, you must identify a location for your SAS data files.

Create the SAS data files here: _____

- b. Select **File** \Rightarrow **Open** \Rightarrow **Program**.

- c. Navigate to the data folder, select **cre8data**, and click **Open**. The program is displayed in an editor. Observe the default value in the %LET statement.



```

/* Windows/UNIX */

/* STEP 1: Notice the default value for the %LET statement. */

/* STEP 2: If your files are not to be located in S:\workshop */
/* change the value of PATH= in the %LET statement to reflect */
/* your data location. */

/* STEP 3: Submit the program to create the course data files. */

/* STEP 4: Go to the Results-SAS Report tab in Enterprise Guide*/
/* or the Results Viewer in SAS and verify the CONTENTS procedure*/
/* report lists the names of the SAS data sets that were created.*/

%let path=s:\workshop;

/*+++++*/
/* WARNING: DO NOT ALTER CODE BELOW THIS LINE */
/*+++++*/

```

- d. If your files are to be created at a location other than **s:\workshop**, change the value assigned to PATH= to reflect the location of your SAS data files.
 -  If your files are to be created in **s:\workshop**, then no change is needed.
- e. Press F3 to submit the program.
- f. Click the **Results** tab and verify that the output contains a list of data files, similar to the list below:

The CONTENTS Procedure				
Directory				
Libref	ORION			
Engine	V9			
Physical Name	s:\workshop			
Filename	s:\workshop			
# Name Member Type File 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

2. Defining the Data Location

- a. Open the **libname** program. Do not change anything in this program.
- b. Submit the program.
- c. Click the **Log** tab and verify that there are no errors or warnings.

Chapter 2 SAS® Programs

2.1 Introduction to SAS Programs	2-3
2.2 Submitting a SAS Program.....	2-7
Demonstration: Submitting a SAS Program: SAS Enterprise Guide	2-8
Demonstration: Submitting a SAS Program: SAS Windowing Environment	2-12
Exercises	2-18
2.3 SAS Program Syntax.....	2-20
Demonstration: Automatic Formatting with SAS Enterprise Guide	2-23
Demonstration: Diagnosing and Correcting Syntax Errors	2-27
Demonstration: Correcting Unbalanced Quotation Marks: SAS Windowing Environment	2-30
Demonstration: Correcting Unbalanced Quotation Marks: SAS Enterprise Guide.....	2-33
Exercises	2-35
2.4 Solutions	2-36
Solutions to Exercises	2-36
Solutions to Student Activities (Polls/Quizzes)	2-38

2.1 Introduction to SAS Programs

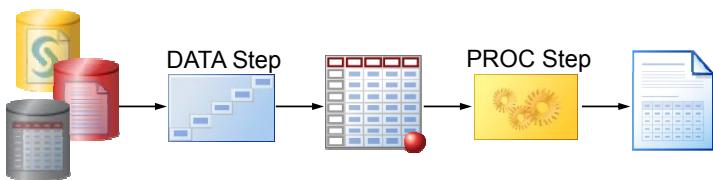
Objectives

- 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

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

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

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

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

Business Scenario

Orion Star programmers will create and execute SAS programs and view results in an interactive environment. They must become familiar with both environments.



17



Submitting a SAS Program: SAS Enterprise Guide

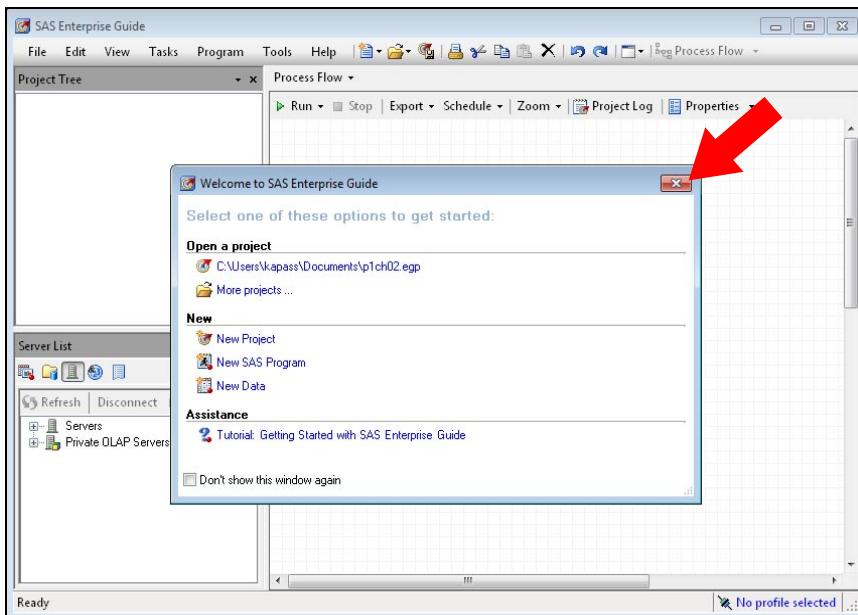
p102d01

- 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 .



Opening and Submitting a SAS Program

1. Open **p102d01**. To open a SAS program in Enterprise Guide, select **File** \Rightarrow **Open** \Rightarrow **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.

```

p102d01 ▾
Program* ▾
Save ▶ Run ▶ Stop | Selected Server: Local (Connected) ▾ Analyze 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;

```

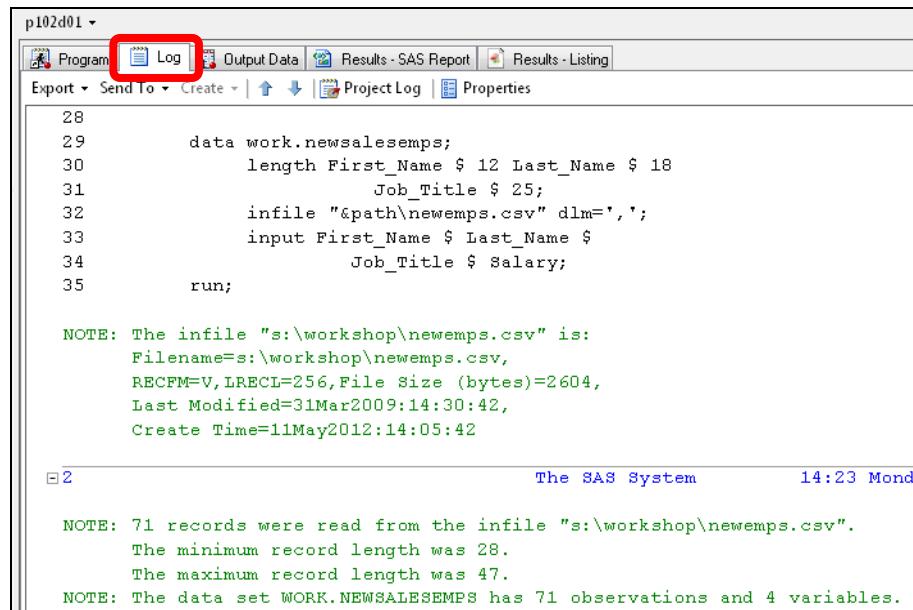
You can use the Program tab to access and edit existing SAS programs, write new programs, submit programs, and save programs to a file. The program is color-coded in the Program Editor.

2. To submit the program for execution, select **Program** \Rightarrow **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.

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**.



The screenshot shows the SAS Enterprise Guide interface with the 'Log' tab selected (indicated by a red box). The log window displays the following content:

```

28      data work.newsalesemps;
29          length First_Name $ 12 Last_Name $ 18
30              Job_Title $ 25;
31          infile "&path\newemps.csv" dlm=',';
32          input First_Name $ Last_Name $ 
33              Job_Title $ Salary;
34
35      run;

NOTE: The infile "s:\workshop\newemps.csv" is:
      Filename=s:\workshop\newemps.csv,
      RECFM=V,LRECL=256,File Size (bytes)=2604,
      Last Modified=31Mar2009:14:30:42,
      Create Time=11May2012:14:05:42

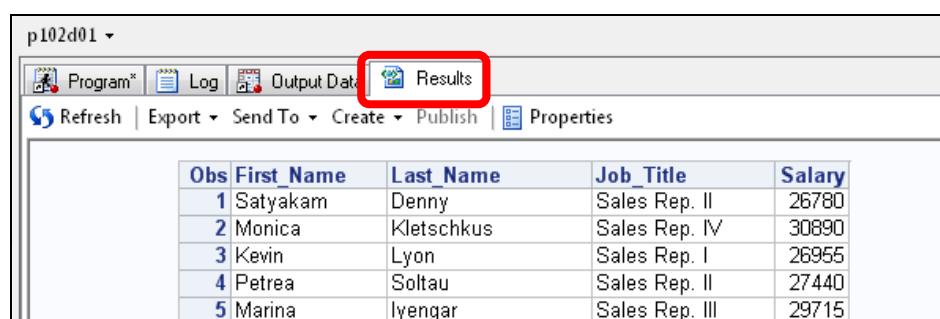
2                                     The SAS System      14:23 Mond
NOTE: 71 records were read from the infile "s:\workshop\newemps.csv".
      The minimum record length was 28.
      The maximum record length was 47.
NOTE: The data set WORK.NEWSALESEMP has 71 observations and 4 variables.

```

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.



The screenshot shows the SAS Enterprise Guide interface with the 'Results' tab selected (indicated by a red box). The results window displays a data grid with the following data:

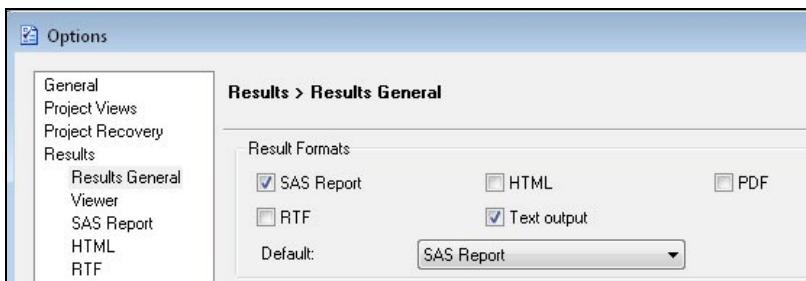
Obs	First_Name	Last_Name	Job_Title	Salary
1	Satyakam	Denny	Sales Rep. II	26780
2	Monica	Kletschkus	Sales Rep. IV	30890
3	Kevin	Lyon	Sales Rep. I	26955
4	Petrea	Soltau	Sales Rep. II	27440
5	Marina	Iyengar	Sales Rep. III	29715

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

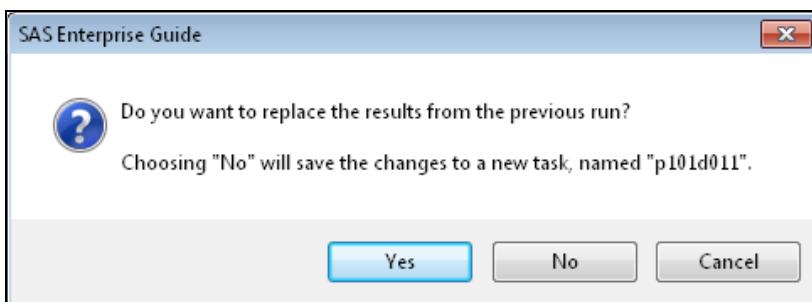
Selecting Output Formats

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.

The screenshot shows the SAS Enterprise Guide interface with a task window titled 'p102d01'. The 'Results - Listing' tab is selected, highlighted with a red box. The table displays the following data:

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

6. Reset the output formats to generate SAS Report or text output (or both) as desired.
- Text output is shown throughout this book.

Idea Exchange

Have you worked with SAS Enterprise Guide? If so, what do you like about it?



19



Submitting a SAS Program: SAS Windowing Environment

p102d01

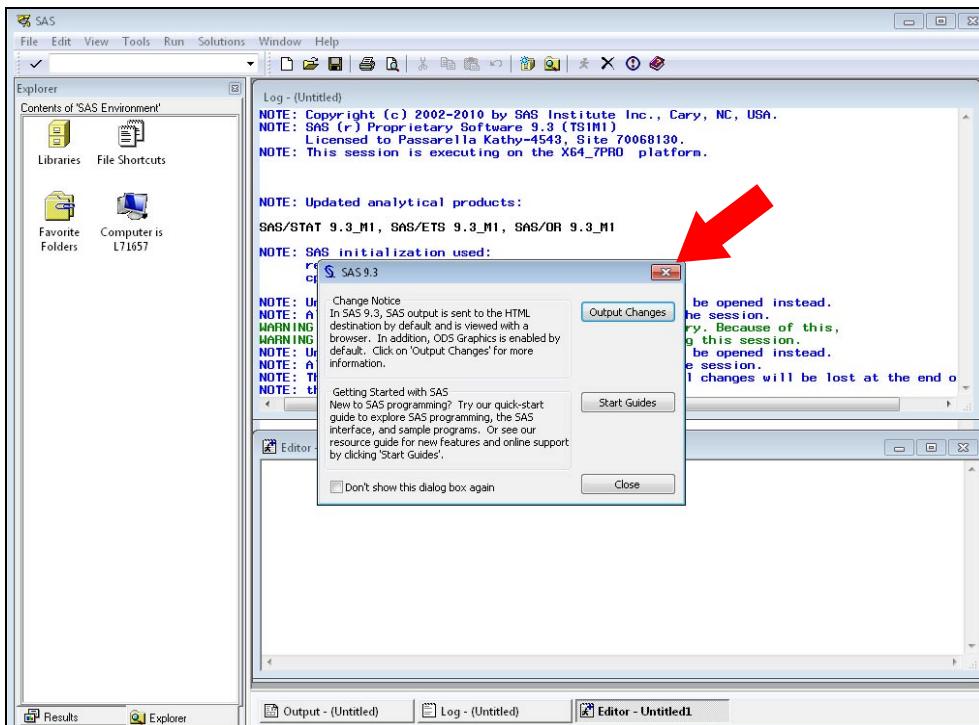
- Start a SAS session.
- Open and submit a SAS program.
- Examine the results.
- Select output formats.



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**.



Opening and Submitting a SAS Program

1. Open **p102d01.sas**. To open a SAS program, select **File** \Rightarrow **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.

```

p102d01.sas
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;

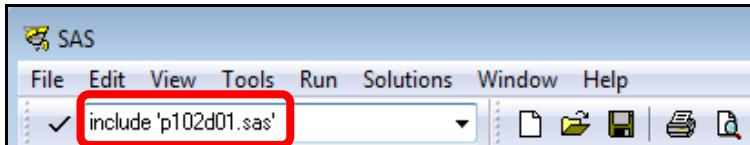
```

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.

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  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** \Rightarrow **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** \Rightarrow **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.

```

Log - (Untitled)
44
45  data work.newsalesemps;
46    length First_Name $ 12 Last_Name $ 18
47    Job_Title $ 25;
48    infile "&path\newemps.csv" dlm=',';
49    input First_Name $ Last_Name $ 
50      Job_Title $ Salary;
51  run;

NOTE: The infile "s:\workshop\newemps.csv" is:
      Filename=s:\workshop\newemps.csv,
      RECFM=V,LRECL=256,File Size (bytes)=2604,
      Last Modified=31Mar2009:14:30:42,
      Create Time=11May2012:14:05:42

NOTE: 71 records were read from the infile "s:\workshop\newemps.csv".
      The minimum record length was 28.
      The maximum record length was 47.
NOTE: The data set WORK.NEWSALESEMP has 71 observations and 4 variables.

```

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** \Rightarrow **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.

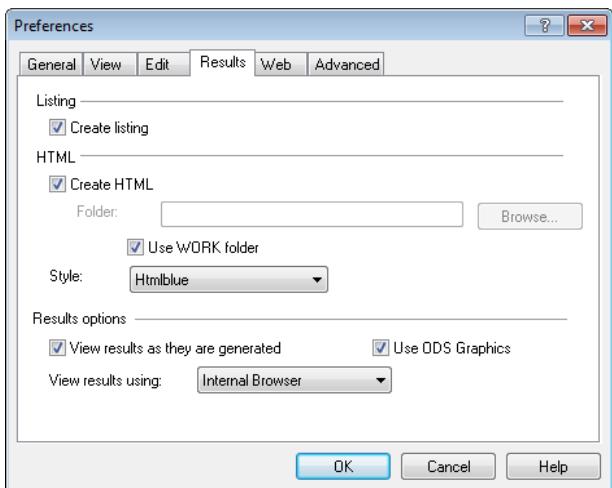
Selecting Output Formats

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** \Rightarrow **Options** \Rightarrow **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.



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

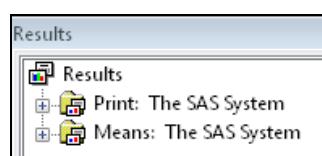
1. LISTING output, produced by request only, is displayed in the Output window. Click the **Output** tab to view the Output window.

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. III	27440
5	Marina	Iyengar	Sales Rep. III	29715

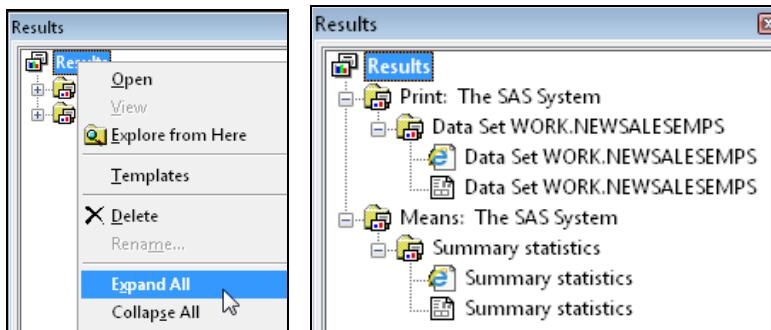
- In the SAS windowing environment, the content displayed in the Output window is the last page of output generated by the most recent program. Scroll up to see the top of the report.
 - To scroll vertically in the Output window, use the PAGE UP or PAGE DOWN keys on the keyboard, use the vertical scroll 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.

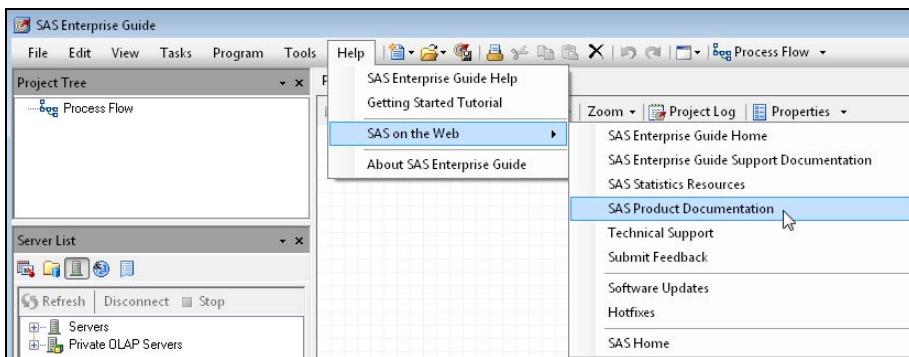


Click a plus sign to expand or collapse individual bookmarks, or right-click the word **Results** and select **Expand All/Collapse All**. Each bookmark contains an icon indicating the file type: HTML or LISTING. Double-click a bookmark to view the corresponding report in the appropriate viewer.



Accessing SAS Product Documentation on the Web

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



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



2.03 Multiple Choice Poll

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

- SAS windowing environment
- SAS Enterprise Guide
- 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

- 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)' .

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

rows: _____ columns: _____

- 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: _____

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

Level 2

2. Exploring Your Environment: SAS Enterprise Guide

- a. Customize the appearance and functionality of the Editor by selecting **Tools** \Rightarrow **Options** \Rightarrow **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** \Rightarrow **Options** \Rightarrow **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** \Rightarrow **Options** \Rightarrow **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** \Rightarrow **Options** \Rightarrow **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** \Rightarrow **Options** \Rightarrow **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.

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

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

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.

29

p102d02

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

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** \Rightarrow **Format Code**. The program is formatted automatically.
 - You can undo the automatic formatting by selecting **Edit** \Rightarrow **Undo** or pressing the CTRL and Z keys simultaneously.
4. Select **File** \Rightarrow **Save p102d02** to save the modified program file.
 - To customize the formatting, select **Program** \Rightarrow **Editor Option** and click the **Indenter** tab.

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 |
| 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*/;②
run;
/*
proc print data=work.newsalesemps; ③
run;
*/
proc means data=work.newsalesemps;
  *var Salary; ④
run;
```

34

p102d03



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.

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

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?

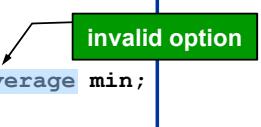
```

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 average min;
  var Salary;
run;

```



40

p102d04

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.

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;

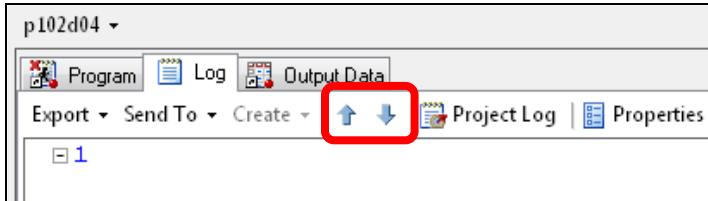
proc print data=work.newsalesemps
run;

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.NEWSALESEMP 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.

```

```

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, NOOBS, NOPRINT, NOTREADS, 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, SUMWT, 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.

- Edit the program.
 - Correct the spelling of DATA.
 - Put a semicolon at the end of the PROC PRINT statement.
 - 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;

```

```
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 or select **File** \Rightarrow **Save program** or **File** \Rightarrow **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** \Rightarrow **Save** or **File** \Rightarrow **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;
```



Correcting Unbalanced Quotation Marks: SAS Windowing Environment

p102d05

- Submit a SAS program that contains unbalanced quotation marks.

- 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;
```

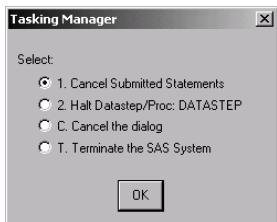
There are no messages of any kind following each step in the log because the steps did not execute. The absence of messages from SAS typically indicates unbalanced quotation marks. Another indication is the “DATA STEP running” message in the banner of the Editor window. This is because the RUN statement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



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.

```
*' ;*" ;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.



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.

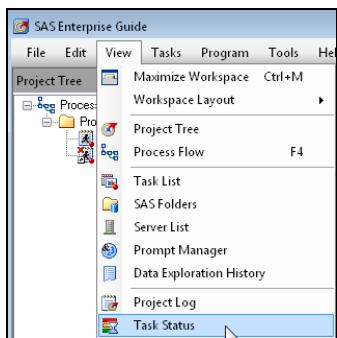
Stopping a Program

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.



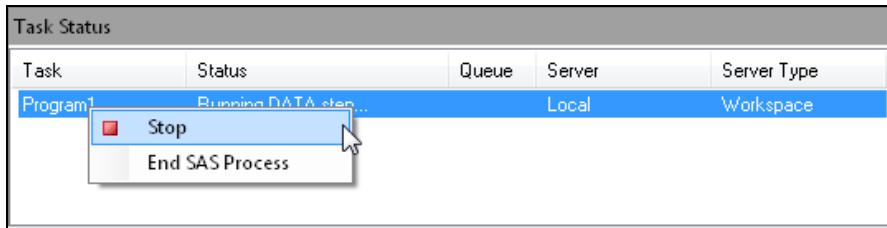
Method 2: Click **View** \Rightarrow **Task Status** to access the Task Status window.



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

Task Status				
Task	Status	Queue	Server	Server Type
Program1	Running DATA step...		Local	Workspace

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

- With the appropriate Editor window active, open the SAS program **p102e06**.
- Submit the program and use the notes in the SAS log to identify the error.
- Correct the error and resubmit the program.
- Save the corrected program.

Level 2

7. Diagnosing and Correcting Syntax Errors

- 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

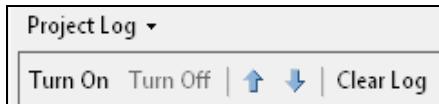
- a. Select **Tools** ⇒ **Options** ⇒ **Enhanced Editor**. Click the **Appearance** tab. Select the desired font size using the Size box in the Font section. Click **OK**.
- b. Select **Tools** ⇒ **Options** ⇒ **Preferences**. Click the **Results** tab. Select the **Create listing** box in the Listing 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.

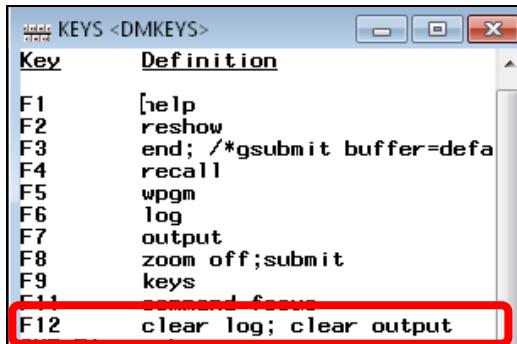
4. Enabling and Disabling the Project Log (SAS Enterprise Guide Only)

- 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.



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

- Submit the Keys command or select Tools \Rightarrow Options \Rightarrow Keys.
- Add the commands to the Definition column for the F12 key:



- Which key is programmed to submit a KEYS command? **the F9 key**
- Close the Keys window.
- 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

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;
```

The diagram illustrates the structure of the SAS program. It is divided into three main sections, each enclosed in a bracket and labeled as a 'PROC Step'. The first section, containing the DATA step code, is labeled 'DATA Step' with a blue background icon. The second and third sections, each containing a PROC step (PRINT and MEANS respectively) followed by a RUN statement, are both labeled 'PROC Step' with yellow background icons.

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;
run;

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

The diagram shows arrows pointing from the end of the DATA step code to the RUN statement, and from the end of the PROC PRINT step code to the PROC MEANS statement, indicating where each step ends.

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

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

2.06 Quiz – Correct Answer

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

```

data work.emps;
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;

```

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

Chapter 3 Accessing Data

3.1 Examining SAS Data Sets.....	3-3
Exercises	3-12
3.2 Accessing SAS Libraries	3-13
Demonstration: Browsing SAS Libraries: SAS Enterprise Guide	3-25
Demonstration: Browsing SAS Libraries: SAS Windowing Environment	3-29
Exercises	3-33
3.3 Solutions	3-34
Solutions to Exercises	3-34
Solutions to Student Activities (Polls/Quizzes)	3-36

3.1 Examining SAS Data Sets

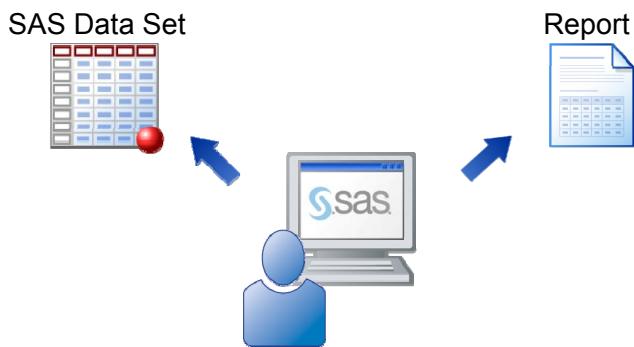
Objectives

- 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.

3

Business Scenario

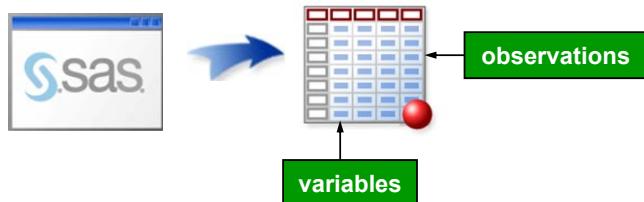
Many SAS data sets related to the Orion Star project already exist. The programmers need to know how to display the structure and contents of the data sets.



4

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.



5

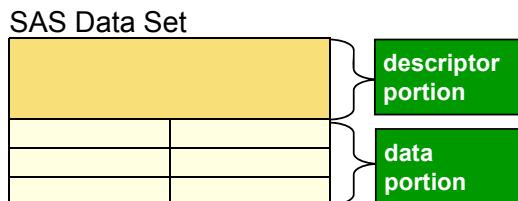
SAS Data Set Terminology

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

6

SAS Data Set Terminology

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



7

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.NEWSALESEMP	
Engine	V9	
Created	Mon, Feb 27, 2012 01:28 PM	
Observations	71	
Variables	4	
...		
First_Name	\$ 12	Last_Name \$ 18
		Job_Title \$ 25
		Salary N 8

general properties

variable properties

8

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;**

9

p103d01

Viewing the Output

Partial PROC CONTENTS Output

The CONTENTS Procedure			
Data Set Name	WORK.NEWSALESEMP	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

10

The default PROC CONTENTS report has three parts:

- general data set properties
- engine/host information
- variable properties

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.

11

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

The diagram illustrates the structure of the data portion. It shows a table with four columns: First_Name, Last_Name, Job_Title, and Salary. Brackets below the table group the first three columns as 'character values' and the last column as 'numeric values'. Brackets to the right of the table group all four columns as 'variable names' and 'data values'. Callouts point from the column headers to 'variable names' and from the data cells to 'data values'.

13

Variable names are part of the descriptor portion, not the data portion.

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;

14

p103d02

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

15

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.



16

The same naming rules apply to SAS data set names.

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

17

A variable name can contain special characters if you place the name in quotation marks and immediately follow it with the letter N (for example, 'Flight#''n). This is called a *SAS name literal*. In order to use SAS name literals as variable names, the VALIDVARNAME= option must be set to ANY.

```
options validvarname=any;
```

This setting is the default in SAS Enterprise Guide.

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.

19

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.

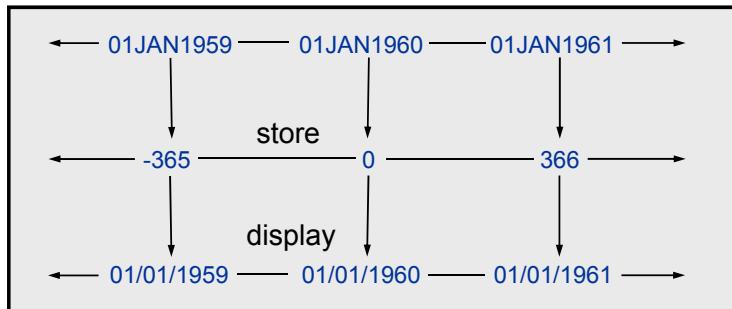
20

By default, a period is used for a missing numeric value. This default can be altered with the MISSING= SAS system option.

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

21

SAS can perform calculations on dates starting from 1582 A.D.

SAS can read either two- or four-digit year values. If SAS encounters a two-digit year, the YEARCUTOFF= system option is used to specify to which 100-year span the two-digit year should be attributed. For example, by setting the YEARCUTOFF= option to 1950, the 100-year span from 1950 to 2049 is used for two-digit year values.

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).

22



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. Examining the Data Portion of a SAS Data Set

- Retrieve the starter program **p103e01**.
- After the PROC CONTENTS step, add a PROC PRINT step to display all observations, all variables, and the Obs column for the data set named **work.donations**.
- Submit the program to create the PROC PRINT report below. The results contain 124 observations.

Partial PROC PRINT Output

Obs	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Total
1	120265	.	.	.	25	25
2	120267	15	15	15	15	60
3	120269	20	20	20	20	80
...						
123	121145	35	35	35	35	140
124	121147	10	10	10	10	40

Level 2

2. Examining the Descriptor and Data Portions of a SAS Data Set

- Retrieve the starter program **p103e02**.
- After the DATA step, add a PROC CONTENTS step to display the descriptor portion of **work.newpacks**.
- Submit the program and answer the following questions:

How many observations are in the data set? _____

How many variables are in the data set? _____

What is the length (byte size) of the variable **Product_Name**? _____

- After the PROC CONTENTS step, add a PROC PRINT to display the data portion of **work.newpacks**.

Submit the program to create the following PROC PRINT report:

Obs	Supplier_Name	Country	Product_Name
1	Top Sports	DK	Black/Black

2	Top Sports	DK	X-Large Bottlegreen/Black
3	Top Sports	DK	Comanche Women's 6000 Q Backpack. Bark
...			
14	Luna sastreria S.A.	ES	Hammock Sports Bag
15	Miller Trading Inc	US	Sioux Men's Backpack 26 Litre.

Challenge

3. Working with Times and Datetimes

- a. Retrieve and submit the starter program **p103e03**.
- b. Notice the values of **CurrentTime** and **CurrentDateTime** in the PROC PRINT output.
- c. Use the SAS Help facility or product documentation to investigate how times and datetimes are stored in SAS.
- d. Complete the following sentences:

A SAS time value represents the number of _____.

A SAS datetime value represents the number of _____.

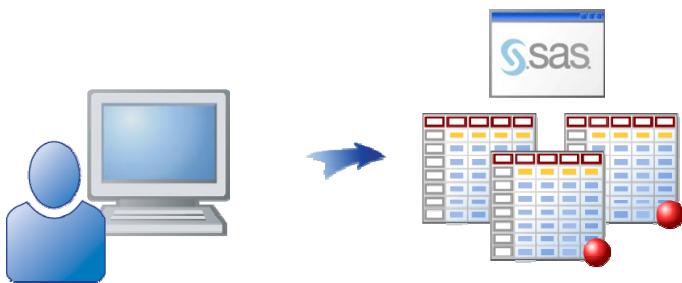
3.2 Accessing SAS Libraries

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.

Business Scenario

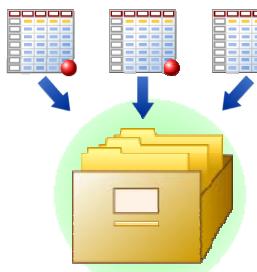
Orion Star programmers need to access existing SAS data sets, so they need to understand how the data sets are stored in SAS.



28

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.

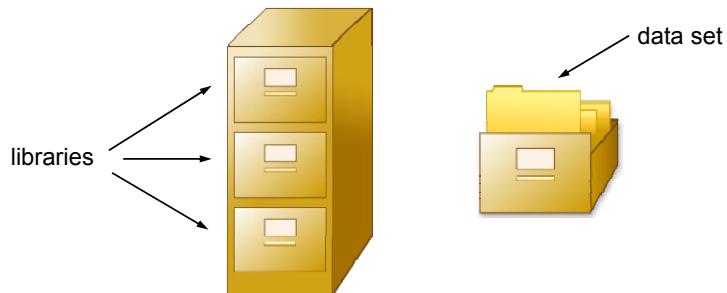


Each file is a member of the library.

29

SAS Libraries

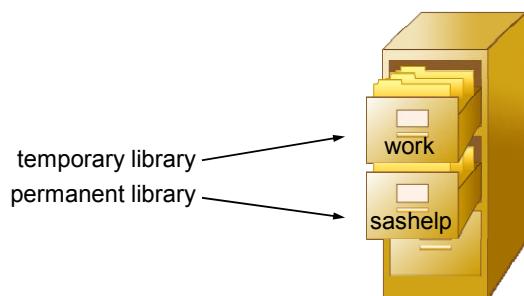
You can think of a SAS library as a drawer in a filing cabinet and a SAS data set as one of the files in the drawer.



30

How SAS Libraries Are Defined

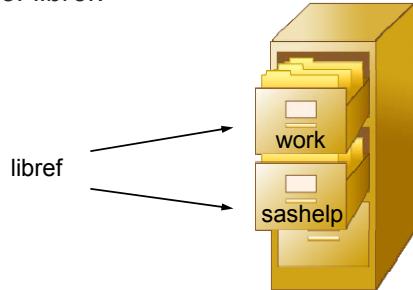
When a SAS session starts, SAS automatically creates one temporary and at least one permanent SAS library that you can access. These libraries are open and ready to be used.



31

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*.

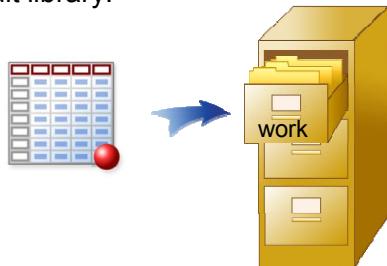


A libref is a shortcut to the library.

32

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.

33

Permanent Libraries

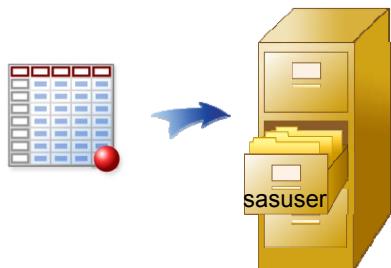
Sashelp is a permanent library that contains sample SAS data sets you can access during your SAS session.



34

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.

35

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.

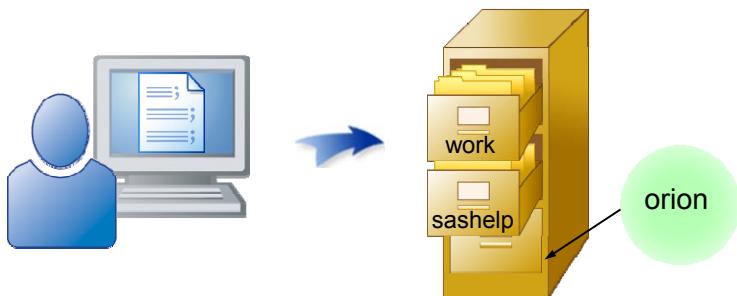


When a data set is in the temporary **work** library, you can use a one-level name (for example, **newsalesemps**).

36

Business Scenario

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



37

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.

38

User-Defined Libraries

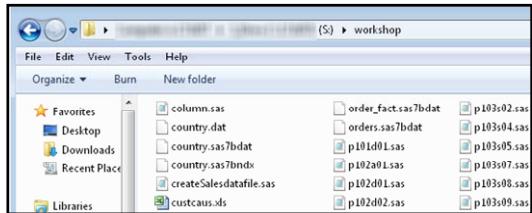
Operating Environment	A SAS library is...	Example
Microsoft Windows	A folder	s:\workshop
UNIX	A directory	~/workshop
z/OS (OS/390)	A sequential file	userid.workshop.sasdata

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

39

Accessing a Permanent Library

Step 1 Identify the location of the library.



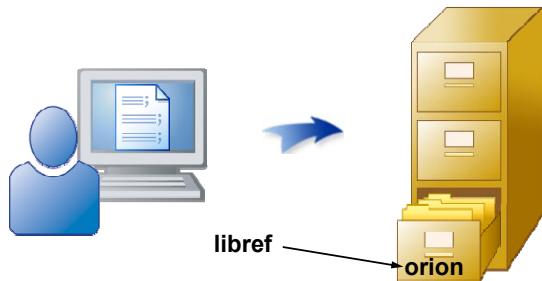
In this example, **s:\workshop**, a Microsoft Windows folder, is used as the SAS library.

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

40

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.

41

Rules for naming a libref:

- The name must be 8 characters or less.
- The name must begin with a letter or underscore followed by letters, underscores, and numerals.

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.

42

- In the Microsoft Windows environment, an existing folder is used as a SAS library. The LIBNAME statement cannot create a new folder.
- In the UNIX environment, an existing directory is used as a SAS library. The LIBNAME statement cannot create a new directory.
- In the z/OS environment, a sequential file is used as a SAS library. z/OS (OS/390) users can use a SAS LIBNAME statement, a DD statement, or a TSO ALLOCATE command. These statements and commands can create a new library.

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
```

43

3.04 Multiple Choice Poll

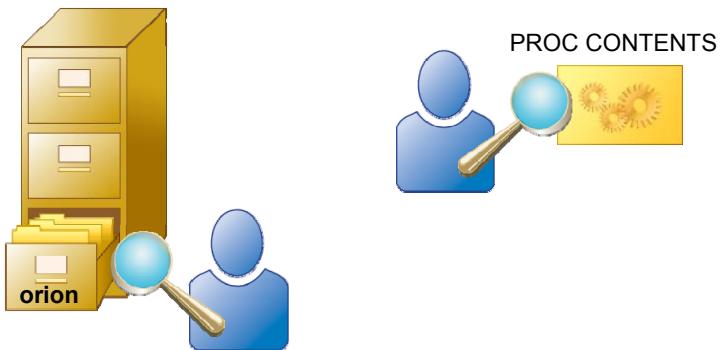
Which of the following correctly assigns the libref **myfile**s to a SAS library in the **c:\mysasfiles** folder?

- a. libname orion myfile "c:\mysasfiles";
- b. libname myfile "c:\mysasfiles";
- c. libref orion myfile "c:\mysasfiles";
- d. libref myfile "c:\mysasfiles";

44

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.



46

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_.

47

p103d03

Viewing the Output

Partial PROC CONTENTS Output

The CONTENTS Procedure				
Directory				
#	Name	Member Type	File 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

48

A member type of DATA indicates a standard SAS data set. INDEX indicates a file that enables SAS to access observations in the SAS data set quickly and efficiently.

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_FormalName
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	

49

p103d04

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.

50

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;
```

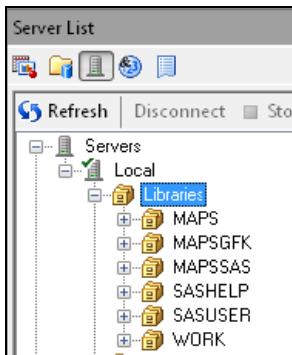
51



Browsing SAS Libraries: SAS Enterprise Guide

libname.sas, p103d03

1. In the Server List, select **Servers** \Rightarrow **Local** \Rightarrow **Libraries** to display the active libraries.



2. Open the program **libname.sas**.

```
%let path=s:\workshop;
*libname orion "s:\workshop";
```

The first statement creates a macro variable named **path** and assigns it a full path to the folder containing the course data. The second statement, a LIBNAME statement, associates the libref, **orion**, with the same data location. The LIBNAME statement is commented out.

3. Modify the program by removing the asterisk from the start of the LIBNAME statement.
4. Save the modified program.
5. Submit the program.

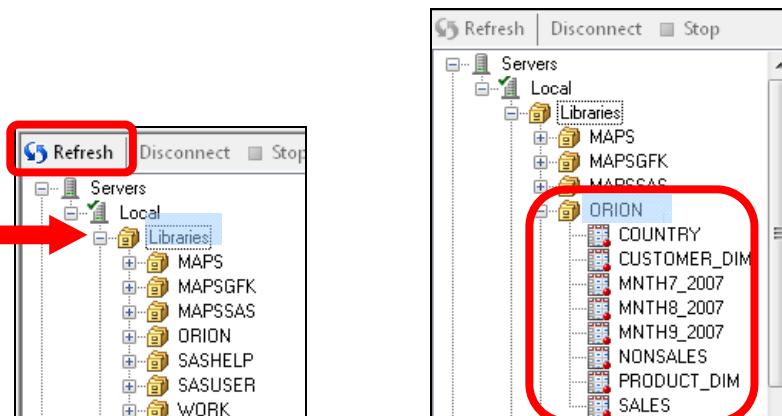


When working with the course data and programs, be sure to run **libname.sas** at the start of every Enterprise Guide session to set the **path** variable and assign the **orion** libref.

- Check the log to confirm that the **orion** libref was assigned. The physical name reflects the location of your SAS data files and might differ from the name shown below.

```
15      %let path=s:\workshop;
16      libname orion "s:\workshop";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:          V9
      Physical Name:  s:\workshop
```

- In the Server List, click **Libraries** and then click **Refresh**. The **orion** library is in the active library list.
- Expand **ORION** to see the list of data sets.



- Double-click the **COUNTRY** data set to open it in a data grid.

COUNTRY						
	Country	Country_Name	Population	Country_ID	Continent_ID	Country_Former
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	

- Click to close the data grid.

- In the Server List, right-click **COUNTRY** and select **Properties**. The Properties window is displayed with the **General** tab selected. Click the **Columns** tab to see information about the variables.



Click to close the Properties window.

12. Explore the **orion** library programmatically using the CONTENTS procedure. Open and submit program p103d03 to generate a list of library members. Partial output is shown below.

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

```
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
          INDEX   17408  13Oct10:19:04:39
4  CUSTOMER   DATA    33792  04Nov11:09:52:27
```

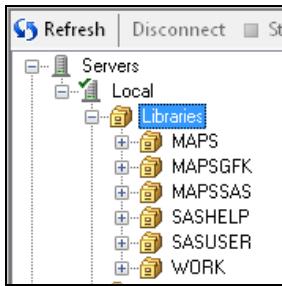
13. Open a new Program window. Type and submit the following statement to clear the **orion** libref.

```
libname orion clear;
```

14. Check the log and the Server List to verify that the **orion** libref was deassigned.

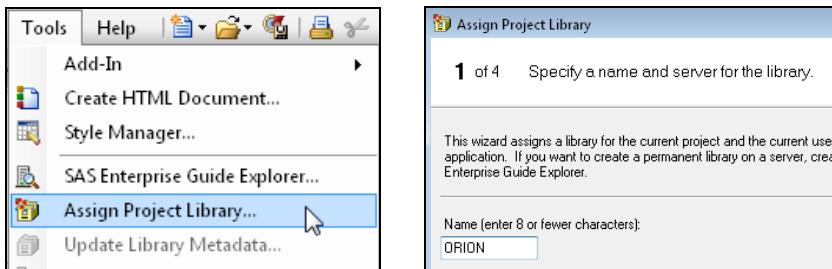
```
15      libname orion clear;
NOTE: Libref ORION has been deassigned.
```

In the Server List, click **Libraries** and click **Refresh**. Verify that **orion** is no longer defined.



Assigning a Library Interactively in SAS Enterprise Guide

1. Select Tools \Rightarrow Assign Project Library to invoke the Assign Project Library Wizard.



2. Enter the following values into the wizard windows:

Step 1 of 4: Type the name **orion**. (The name is displayed in uppercase.) Click **Next**.

Step 2 of 4: Enter the location of the data files or browse to the location. Click **Next**.

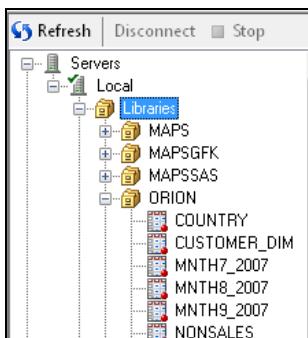
Step 3 of 4: Do not enter any options. Click **Next**.

Step 4 of 4: Click **Finish**.

SAS Enterprise Guide submits a LIBNAME statement based on your wizard entries. Check the log for success. A sample log is shown below.

```
15      LIBNAME ORION BASE "S:\workshop";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:      BASE
      Physical Name: S:\workshop
```

3. In the Server List, click **Libraries** and select **Refresh**. Expand **ORION** to see its members.



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

53



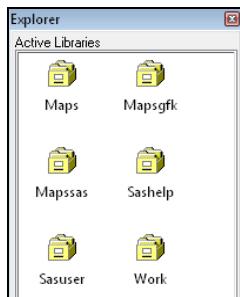
Browsing SAS Libraries: SAS Windowing Environment

libname, p103d03

1. Click the Explorer tab to activate the Explorer window or select **View** \Rightarrow **Contents Only**.



2. Double-click **Libraries** to show all available libraries. The active libraries are displayed.



3. Open **libname.sas**.

```
%let path=s:\workshop;
*libname orion "s:\workshop";
```

The first statement creates a macro variable named **path** and assigns it a full path to the folder containing the course data. The second statement, a LIBNAME statement, associates the libref, **orion**, with the same data location. The LIBNAME statement is commented out.

4. Modify the program by removing the asterisk from the start of the LIBNAME statement. Save the modified program.
5. Submit the program.

 When working with the course data and programs, be sure to run **libname.sas** at the start of every SAS session to set the **path** variable and assign the **orion** libref.

6. Check the log to confirm that the **orion** libref was assigned. The physical name reflects the location of your SAS data files and might differ from the name shown below.

```
1. 412 %let path=s:\workshop;
413 libname orion "s:\workshop";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:          V9
      Physical Name:  s:\workshop
```

7. In the Explorer window, verify that **orion** is now displayed as an active library.

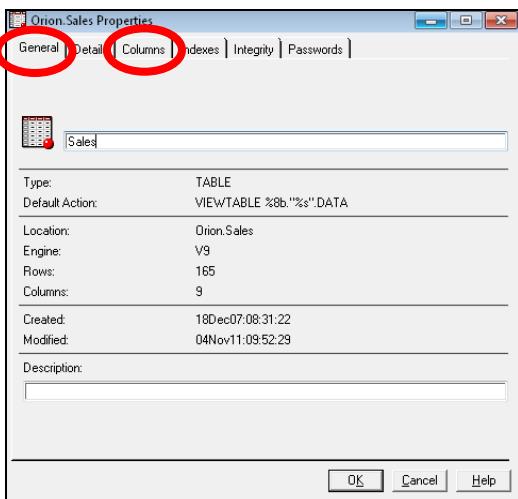


8. Double-click the **Orion** library to show all members of that library.



9. Right-click the **Sales** data set. Select **Properties**.

The default view provides general information about the data set. You can view information about the variables in the data set by clicking the **Columns** tab at the top of the Properties window.



10. Click **X**, **OK**, or **Cancel** to close the Properties window.
11. Double-click the **Sales** data set or right-click the file and select **Open**. The data set **orion.sales** opens in a **VIEWTABLE** window.

VIEWTABLE: Orion.Sales								
	Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_Date
1	120102	Tom	Zhou	M	108255	Sales Manager	AU	4971
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU	-2535
3	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-4169
4	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-523
5	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	3193

Variable labels are displayed by default. You can display variable names instead of variable labels by selecting **View** \Rightarrow **Column Names**. In addition to browsing SAS data sets, you can use the **VIEWTABLE** window to edit and create data sets, and to customize your view of a SAS data set.

12. Click **X** to close the **VIEWTABLE** window.
13. Explore the **orion** library programmatically. Open and submit **p103d03** to generate a list of library members.

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

14. View the output. Partial output is shown below.

The CONTENTS Procedure				
Directory				
Libref	ORION			
Engine	V9			
Physical Name	S:\workshop			
Filename	S:\workshop			
#	Name	Member Type	File 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

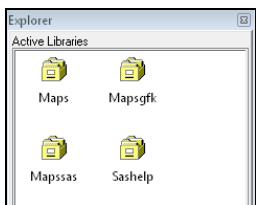
15. With the Explorer window active, click  to return to the Active Libraries display.

16. Open a new Editor window. Type and submit the following statement to clear the **orion** libref.

```
libname orion clear;
```

17. Check the log and the Explorer window to verify that the **orion** libref was deassigned.

```
15      libname orion clear;
NOTE: Libref ORION has been deassigned.
```



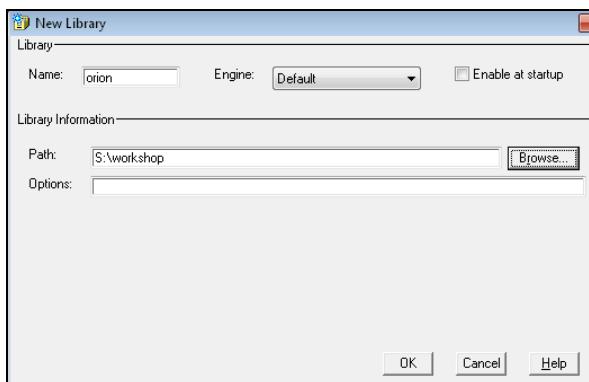
Assigning a Library Interactively in the SAS Windowing Environment

1. From the menu bar, select **Tools** \Rightarrow **New Library** or click  (the New Library tool). The New Library window is displayed.



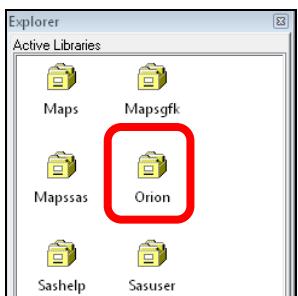
2. Do the following in the New Library window:

- Type the name **orion**.
- Enter a full path to the course data or browse to it.
- Do not enter any options.
- Click **OK**.





- Select **Enable at startup** to have this library defined automatically when a SAS session starts.
3. No notes are written to the log when this method is used to assign a library. Use the Explorer window to verify that **orion** was assigned.



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

56



Exercises

You must submit a LIBNAME statement before beginning this exercise session.

- a. Open **libname.sas**. Modify the program by removing the asterisk from the LIBNAME statement.
- b. Submit the modified program.
- c. Verify that the libref **orion** was successfully assigned.
- d. Save the program.

Level 1

4. Accessing a Permanent Data Set

- a. Use an interactive facility to explore the **orion** library and answer the following questions:

How many observations are in the **orion.country** data set? _____

How many variables are in the **orion.country** data set? _____

What is the name of the last country in the data set? _____

- b. Submit a PROC CONTENTS step to generate a list of all members in the **orion** library. What is the name of the last member listed? _____

Level 2

5. Viewing General Data Set Properties

- a. Examine the general data set properties of **orion.staff**.

b. What sort information is stored for this data set? _____

Challenge

6. SAS Autoexec File

Use the Help facility or product documentation to investigate the SAS autoexec file and answer the following questions.

What is the name of the file? _____

What is its purpose? _____

How is it created? _____

How could this be useful in a SAS session? _____

3.3 Solutions

Solutions to Exercises

1. Examining the Data Portion of a SAS Data Set

```
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;

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

2. Examining the Descriptor and Data Portions of a SAS Data Set

```
data work.newpacks;
```

```

input Supplier_Name $ 1-20 Supplier_Country $ 23-24
      Product_Name $ 28-70;
datalines;
Top Sports           DK   Black/Black
Top Sports           DK   X-Large Bottlegreen/Black
...
Miller Trading Inc  US   Sioux Men's Backpack 26 Litre.
;

proc contents data=work.newpacks;
run;

proc print data=work.newpacks noobs;
  var Product_Name Supplier_Name;
run;

```

- How many observations are in the data set? **13**
- How many variables are in the data set? **3**
- What is the length (byte size) of the variable **Product_Name**? **43**

3. Working with Times and Datetimes

- A SAS time value represents the number of seconds since **midnight of the current day**.
- A SAS datetime value represents the number of seconds between **midnight January 1, 1960**, and **an hour/minute/second within a specified date**.

4. Accessing a Permanent Data Set

- a. How many observations are in the **orion.country** data set? **7**

How many variables? **6**

What is the name of the last country in the data set? **South Africa**

- b. Submit a PROC CONTENTS step to generate a list of all members in the **orion** library.

```

proc contents data=orion._all_ nods;
run;

```

What is the name of the last member listed? **US_SUPPLIERS**

5. Viewing General Data Set Properties

```

proc contents data=orion.staff;
run;

```

- a. Examine the general data set properties of **orion.staff**.
- b. What sort information is stored for this data set? **The General Information section indicates that the data set is sorted. The Variable section indicates that it is sorted by Employee_ID using the ANSI character set, and has been validated.**

6. SAS Autoexec File

What is the name of the file? **autoexec.sas**

What is its purpose? **It contains SAS statements that are executed immediately after SAS initializes. These SAS statements can be used to invoke SAS programs automatically, set up certain variables for use during your SAS session, or set system options.**

How is it created? **With any text editor, but the recommended method is to use a SAS text editor (such as the Enhanced Editor window) and save it using the Save As dialog box.**

How could this be useful in a SAS session? **It can be used to set the path macro variable and to automatically submit a LIBNAME statement.**

Solutions to Student Activities (Polls/Quizzes)

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;
```

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

18

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.

23

3.04 Multiple Choice Poll – Correct Answer

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

- a. libname orion myfile "c:\mysasfiles";
- b. libname myfile "c:\mysasfiles";**
- c. libref orion myfile "c:\mysasfiles";
- d. libref myfile "c:\mysasfiles";

45

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.

54

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

Chapter 4 Producing Detail Reports

4.1 Subsetting Report Data.....	4-3
Exercises	4-24
4.2 Sorting and Grouping Report Data	4-27
Exercises	4-38
4.3 Enhancing Reports.....	4-40
Exercises	4-48
4.4 Solutions	4-50
Solutions to Exercises	4-50
Solutions to Student Activities (Polls/Quizzes)	4-54

4.1 Subsetting Report Data

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.

3

Business Scenario

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



Obs	Last_Name	First_Name	Salary
1	xxxxxx	xxxxxxxxxx	99999
2	xxxxxx	xxxxxxxxxx	99999
3	xxxxxx	xxxxxxxxxx	99999
-----			99999

4

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.

5

p104d01

The columns are listed, left to right, in the order the variables are stored in the data set.

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

6

p104d01

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

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

p104d01

7

Viewing the Log

Partial SAS Log

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

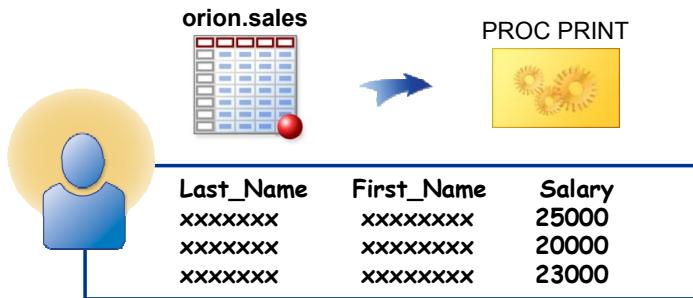
NOTE: There were 165 observations read from the data set ORION.SALES.

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

8

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.



9

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;

10

p104d02

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;
```

11

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

12

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

13

p104d02

WHERE Statement

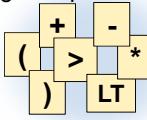
The WHERE expression defines the condition (or conditions) for selecting observations.

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

14

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.



15

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.

16

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

17

The caret (^), tilde (~), and the not sign (~) all indicate a logical not. Use the character available on your keyboard, or use the mnemonic equivalent.

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.

18

Setup for the Poll

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

19

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.

20

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;

22

p104d03

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);
```

23

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');
```

24

- NOT modifies a condition by finding the complement of the specified criteria. Use the character available on your keyboard, or use the mnemonic equivalent.
- AND finds observations that satisfy both conditions.
- OR finds observations that satisfy one or both conditions.

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.

25

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='.';`

26

Business Scenario

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



Last_Name	First_Name	Country	Job_Title
xxxxxxxxxx	xxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxx	xx	xxxxxxxxxxxx
xxxxxxxxxx	xxxxxx	xx	xxxxxxxxxxxx

28

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

29

p104d04

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.

30

p104d04

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.

31

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

32

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	

33



Special WHERE operators cannot be used in IF-THEN or subsetting IF statements.

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';
```

34

BETWEEN-AND Operator

Equivalent Statements

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

35

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.

36

p104d05

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);

```

37

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

38

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?

39

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=. ;
```

41

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=.;
```

42

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%';
```

43

When using the LIKE operator:

- Consecutive underscores can be specified.
- A percent sign and an underscore can be specified in the same pattern.
- The operator is case sensitive.

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

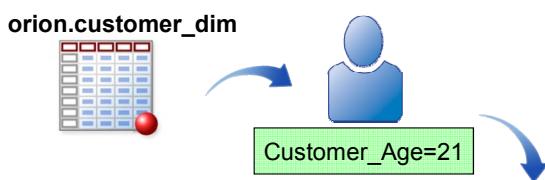
↑

last name, first name

44

Business Scenario

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



Obs	Customer_ID	Customer_Name	Customer_Gender	Customer_Country	Customer_Group	Customer_Age_Group	Customer_Type
1	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXXX	XXXXXXXXXX
2	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXXX	XXXXXXXXXX
3	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXXX	XXXXXXXXXX

47

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.

48

p104d06

Viewing the Output

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

PROC PRINT Output

Obs	Customer_ID	Customer_Name	Customer_Gender	Customer_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
Obs	Customer_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.

49

Report width and wrapping depends on the LINESIZE system option, the ODS destination, or both.

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.

50

p104d07

Note that the ID variable was removed from the VAR statement. If it were not removed, it would be displayed twice: once as the leftmost column and again in the position specified by the VAR statement.

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

51



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 orion.order_fact with the PRINT Procedure

- Retrieve the starter program **p104e01**. Run the program and view the output. Observe that there are 617 observations. Observations might be displayed over two lines, depending on output settings.
- Add a SUM statement to display the sum of **Total_Retail_Price**. The last several lines of the report are shown below.

Obs	Order_Type	Product_ID	Quantity	Total_Retail_Price	CostPrice_Per_Unit	Discount
610	1	240700100007	2	\$45.70	\$9.30	.
611	1	240700100017	2	\$19.98	\$11.40	40%
612	1	240700400003	2	\$24.80	\$5.60	.
613	1	240800100042	3	\$760.80	\$105.30	.
614	1	240500200016	3	\$95.10	\$14.50	.
615	1	240500200122	2	\$48.20	\$11.50	.
616	1	240700200018	4	\$75.20	\$10.30	.
617	1	220101400130	2	\$33.80	\$5.70	.
=====						
\$100,077.46						

- Add a WHERE statement to select only the observations with **Total_Retail_Price** more than 500. Submit the program. Verify that 35 observations were displayed.

What do you notice about the Obs column? _____

Did the sum of **Total_Retail_Price** change to reflect only the subset? _____

- Add an option to suppress the Obs column. Verify that there are 35 observations in the results.

How can you verify the number of observations in the results? _____

- Add an ID statement to use **Customer_ID** as the identifying variable. Submit the program. The results contain 35 observations.

How did the output change? _____

- Add a VAR statement to display **Customer_ID**, **Order_ID**, **Order_Type**, **Quantity**, and **Total_Retail_Price**.

What do you notice about **Customer_ID**? _____

- Modify the VAR statement to address the issue with **Customer_ID**.

Level 2

2. Displaying orion.customer_dim with the PRINT Procedure

- Write a PRINT step to display **orion.customer_dim**.

- b. Modify the program to display a subset of **orion.customer_dim** by selecting only the observations for customers between the ages of 30 and 40. Also, suppress the Obs column. The resulting report should contain 17 observations.
- c. Add a statement to use **Customer_ID** instead of Obs as the identifying column. Submit the program and verify the results.
- d. Add a statement to limit the variables to those shown in the report below.

Customer_ID	Customer_Name	Age	Customer_Type
4	James Kvarniq	33	Orion Club members low activity
9	Cornelia Krahel	33	Orion Club Gold members medium activity
11	Elke Wallstab	33	Orion Club members high activity
...
54655	Lauren Marx	38	Internet/Catalog Customers
70201	Angel Borwick	38	Orion Club Gold members low activity

Challenge

3. Producing a Default Listing Report of **orion.order_fact**

 This exercise assumes you are creating LISTING output in the SAS Windowing Environment.

- a. Produce a default listing report of **orion.order_fact**. The output might wrap onto a second line.
- b. Investigate the use of the LINESIZE= SAS system option to adjust the width of the lines. What are the minimum and maximum values for the LINESIZE= option? _____

Submit an OPTIONS statement with LINESIZE= set to the highest allowed value. Resubmit the step, and observe the horizontal scroll bar if displayed.

Reset the line size to 96 when finished.

- c. Another way to create compact output is to request vertical headings. Investigate the HEADING= option in the PROC PRINT statement, and then experiment with it to generate vertical headings, and then horizontal headings.

How do you specify vertical headings? _____

How do you specify horizontal headings? _____

4. Producing a Default Listing Report of **orion.product_dim**

 This exercise assumes you are creating LISTING output in the SAS Windowing Environment.

- a. Produce a default listing report to display **orion.product_dim**. Notice that the column width varies from one page to the next, depending on the width of the values displayed on each page.
- b. Investigate the WIDTH= option of the PROC PRINT statement, and modify the program to use this option with a value of UNIFORM.

How are the results different when using the WIDTH=UNIFORM option? _____

c. Why might the procedure run more slowly with this option? _____

d. How can you save computer resources and still display columns consistently across pages? _____

4.2 Sorting and Grouping Report Data

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.

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

56

Creating a Sorted Report

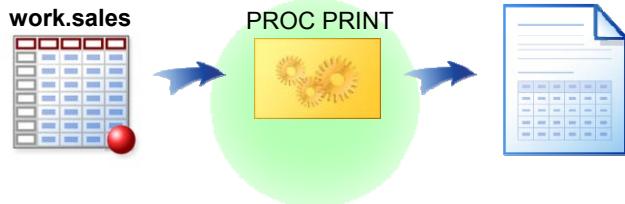
Step 1 Use the SORT procedure to create a new data set, **work.sales**, ordering the observations by the value of **Salary**.



57

Creating a Sorted Report

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



58

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.

59

p104d08

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.
```

60

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

61

p104d08

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.

62

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;
```

63

Business Scenario

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

-----Country=AU-----							
Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Birth_Date	Hire_Date
9999	xxxx	xxxxx	x	99999	xxxxxx	9999	9999
9999	xxxx	xxxxx	x	99999	xxxxxx	9999	9999
-----Country=US-----							
Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Birth_Date	Hire_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

66

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**.

67

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);

68

p104d09

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;**

69

Specifying Multiple BY Variables

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

PROC SORT



by **Country**, ascending

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

PROC SORT



by **Salary**, descending

- This sorting continues for every specified BY variable.

70

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).

71

p104d09

Viewing the Output

Partial PROC PRINT Output

----- Country=AU -----					
Employee_ID	First_Name	Last_Name	Gender	Salary	Hire_Date
120102	Tom	Zhou	M	108255	12205
120103	Wilson	Dawes	M	87975	... 6575
120168	Selina	Barcoe	M	36605	18567

----- Country=US -----					
Employee_ID	First_Name	Last_Name	Gender	Salary	Hire_Date
120261	Harry	Highpoint	M	243190	11535
121143	Louis	Favaron	M	95090	... 15157
121064	Asishana	Polky	M	84260	13027

72

4.06 Quiz

Open and submit p104a02. View the log.

Why did the program fail?

73

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
-----			999999
Country			999999
----- Country=US -----			
First_Name	Last_Name	Gender	Salary
XXXXXXX	XXXXXXX	X	99999
XXXXXXX	XXXXXXX	X	99999
-----			999999
Country			999999
-----			9999999
			9999999

subtotals

grand total

75

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;
```

76

p104d10

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
			1900015
Country			

----- Country=US -----			
First_Name	Last_Name	Gender	Salary
Harry	Highpoint	M	243190
Louis	Favaron	M	95090
Dennis	Lansberry	M	84260
...			
Tulsidas	Ould	M	22710
			3241405
Country			

subtotal for AU

subtotal for US

grand total

77

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;
```

78

p104a03

4.07 Multiple Choice Poll

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

- The WHERE statement in the PROC SORT step.
- The WHERE statement in the PROC PRINT step.
- Both WHERE statements are needed.
- The WHERE statements are equally efficient.

79



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

5. Sorting **orion.employee_payroll** and Displaying the New Data Set

- Open **p104e05**. Add a PROC SORT step before the PROC PRINT step to sort **orion.employee_payroll** by **Salary**, placing the sorted observations into a temporary data set named **sort_salary**.
- Modify the PROC PRINT step to display the new data set. Verify that your output matches the report below.

Obs	Employee_ID	Employee_Gender	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependent
1	121084	M	22710	3150	12784	.	M	3
2	120191	F	24015	1112	17167	17347	S	0
...								
422	120261	M	243190	4800	11535	.	0	1
423	120262	M	268455	5042	11932	.	M	2
424	120259	M	433800	2946	12297	.	M	1

6. Sorting **orion.employee_payroll** and Displaying Grouped Observations

- Open **p104e06**. Add a PROC SORT step before the PROC PRINT step to sort **orion.employee_payroll** by **Employee_Gender**, and within gender by **Salary** in descending order. Place the sorted observations into a temporary data set named **sort_salary2**.
- Modify the PROC PRINT step to display the new data set with the observations grouped by **Employee_Gender**.

----- Employee_Gender=F -----							
Obs	Employee_ID	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependents
1	120260	207885	3258	10532	.	M	2
2	120719	87420	4770	14641	.	M	1
3	120661	85495	-400	10227	17347	M	3
...							
190	120196	24025	10257	17167	17347	S	0
191	120191	24015	1112	17167	17347	S	0
----- Employee_Gender=M -----							
Obs	Employee_ID	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependents
192	120259	433800	2946	12297	.	M	1
193	120262	268455	5042	11932	.	M	2
...							
423	120190	24100	10566	17837	18017	M	2
424	121084	22710	3150	12784	.	M	3

Level 2

7. Sorting orion.employee_payroll and Displaying a Subset of the New Data Set

- Sort **orion.employee_payroll** by **Employee_Gender**, and by descending **Salary** within gender. Place the sorted observations into a temporary data set named **sort_sal**.
- Print a subset of the **sort_sal** data set. Select only the observations for active employees (those without a value for **Employee_Term_Date**) who earn more than \$65,000. Group the report by **Employee_Gender**, and include a total and subtotals for **Salary**. Suppress the Obs column. Display only **Employee_ID**, **Salary**, and **Marital_Status**. The results contain 18 observations.

----- Employee_Gender=F -----		
Employee_ID	Salary	Marital_Status
120260	207885	M
120719	87420	M
...		
120677	65555	M
-----	-----	-----
Employee_Gender	605190	
----- Employee_Gender=M -----		

		Employee_ID	Salary	Marital_Status
		120259	433800	M
		120262	268455	M
		...		
		120268	76105	S
		<hr/>		
		Employee_Gender	2072410	
			=====	
			2677600	

Challenge

8. Retaining the First Observation of Each BY Group

- Sort `orion.orders` by `Customer_ID`. Place the sorted observations in a temporary data set.
- Display the sorted data set. The resulting report should contain 490 observations. `Customer_ID` is listed multiple times for customers that placed more than one order.
- Investigate an option that causes PROC SORT to retain only the first observation in each BY group.
- Add the appropriate option to the PROC SORT step to retain only the first observation in each BY group. The results contain 75 observations with no duplicate values for `Customer_ID`.
- Explore the DUPOUT= option to write duplicate observations to a separate output data set.

4.3 Enhancing Reports

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.

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	9999	xxxxxxxxxx	99999

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

85

Displaying Titles and Footnotes

Use TITLE and FOOTNOTE statements to enhance the report.

```

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

footnote1 'Confidential';

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

title;
footnote;

```

TITLEn 'text';

FOOTNOTEn 'text';

86

p104d11

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

87

TITLE Statement

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

`TITLEn '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.

88

FOOTNOTE Statement

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

```
FOOTNOTEn '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.

89

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.

90

Cancelling All Titles and Footnotes

- The null TITLE statement cancels all titles.

```
title;
```

- The null FOOTNOTE statement cancels all footnotes.

```
footnote;
```

91

Changing and Cancelling Titles and Footnotes

PROC PRINT Code

Resultant Title(s)

<pre>title1 'The First Line'; title2 'The Second Line'; proc print data=orion.sales; run;</pre>	The First Line The Second Line
<pre>title2 'The Next Line'; proc print data=orion.sales; run;</pre>	The First Line The Next Line
<pre>title 'The Top Line'; proc print data=orion.sales; run;</pre>	The Top Line
<pre>title3 'The Third Line'; proc print data=orion.sales; run;</pre>	The Top Line The Third Line
<pre>title; proc print data=orion.sales; run;</pre>	

102

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;
```

103

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;

105



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';
footnotel '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

107

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.

108

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

109

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.

110

p104d13

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

111



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

9. Displaying Titles and Footnotes in a Detail Report

- Open and submit **p104e09** to display all observations for Australian Sales Rep IVs.
- Add a VAR statement to display only the variables shown in the report below.
- Add TITLE and FOOTNOTE statements to include the titles and footnotes shown in the report below.
- Submit the program and verify the output. The results contain five observations as shown below.
- Submit a null TITLE and null FOOTNOTE statement to clear all titles and footnotes.

Australian Sales Employees Senior Sales Representatives					
Obs	Employee_ID	First_Name	Last_Name	Gender	Salary
7	120125	Fong	Hofmeister	M	32040
10	120128	Monica	Kletschkus	F	30890
17	120135	Alexei	Platts	M	32490
41	120159	Lynelle	Phoumirath	F	30765

48	120166	Fadi	Nowd	M	30660
Job_Title: Sales Rep. IV					

10. Displaying Column Headings in a Detail Report

- a. Open and submit p104e10. Modify the program to define and use the following labels:

Variable	Label
Employee_ID	Employee ID
First_Name	First Name
Last_Name	Last Name
Salary	Annual Salary

Submit the program and verify the output.

Entry-level Sales Representatives				
Employee ID	First Name	Last Name	Gender	Annual Salary
121023	Shawn	Fuller	M	26010
121028	William	Smades	M	26585
121029	Kuo-Chung	Mcelwee	M	27225
...				
121138	Hershell	Tolley	M	27265
121140	Saunders	Briggi	M	26335

Job_Title: Sales Rep. I

- b. Modify the program to use a blank space as the SPLIT= character to generate two-line column headings. Submit the modified program and verify that two-line column labels are displayed.

Entry-level Sales Representatives				
Employee ID	First Name	Last Name	Gender	Annual Salary
121023	Shawn	Fuller	M	26010
121028	William	Smades	M	26585
121029	Kuo-Chung	Mcelwee	M	27225
...				
121138	Hershell	Tolley	M	27265
121140	Saunders	Briggi	M	26335

Job_Title: Sales Rep. I

Level 2

11. Writing an Enhanced Detail Report

- a. Write a program to display a subset of **orion.employee_addresses** as shown below. The program should sort the observations by **State**, **City**, and **Employee_Name** and then display the sorted observations grouped by **State**. The resulting report should contain 311 observations.

US Employees by State				
----- State=CA -----				
Employee				Zip
ID	Name		City	Code
120656	Amos, Salley		San Diego	92116
120759	Apr, Nishan		San Diego	92071
121017	Arizmendi, Gilbert		San Diego	91950
121062	Armant, Debra		San Diego	92025
121049	Bataineh, Perrior		San Diego	92126
...				

4.4 Solutions

Solutions to Exercises

1. Displaying orion.order_fact with the PRINT Procedure

```
proc print data=orion.order_fact noobs;
  where Total_Retail_Price>500;
  id Customer_ID;
  var Order_ID Order_Type Quantity Total_Retail_Price;
  sum Total_Retail_Price;
run;
```

- a. Run the program and view the output.
- b. Add a SUM statement and verify the resulting sum.
- c. What do you notice about the Obs column? **The numbers are not sequential. The original observation numbers are displayed.**

Did the sum of **Total_Retail Price** change to reflect only the subset? **Yes**

- d. If the Obs column is suppressed, how can you verify the number of observations in the results? **Check the log.**
- e. When the ID statement was added, how did the output change? **Customer_ID is the leftmost column and is displayed on each line for an observation.**
- f. When the VAR statement is added, what do you notice about **Customer_ID?** **There are two Customer_ID columns. The first column is the ID field, and a second one is included because Customer_ID is listed in the VAR statement.**
- g. Remove the duplicate column by removing **Customer_ID** from the VAR statement.

2. Displaying orion.customer_dim with the PRINT Procedure

```
proc print data=orion.customer_dim noobs;
  where Customer_Age between 30 and 40;
```

```
id Customer_ID;
var Customer_Name Customer_Age Customer_Type;
run;
```

3. Producing a Default Listing Report of orion.order_fact (SAS Windowing Environment)

```
options ls=max;

proc print data=orion.order_fact;
run;

options ls=96;

proc print data=orion.order_fact headings=v;
run;
```

- a. Submit a simple PROC PRINT step.
- b. The minimum value for LINESIZE= is 64 and the maximum size is MAX.

Use this statement to reset the line size to 96: **options ls=96;**

- c. HEADINGS=V forces all column headings to display vertically.

HEADINGS=H forces all column headings to display horizontally.

4. Producing a Default Listing Report of orion.product_dim (SAS Windowing Environment)

```
proc print data=orion.product_dim width=uniform;
run;
```

- a. Submit a simple PROC PRINT step.
- b. Add the WIDTH=uniform option. How are the results different? **Each column has the same column width on each page.**
- c. Why might the procedure run more slowly with this option? **With this option, PROC PRINT must read through the entire data set twice.**
- d. How can you save computer resources and still display columns consistently across pages? **Use a format on every column to explicitly specify a field width so that PROC PRINT reads the data only once.**

5. Sorting orion.employee_payroll and Displaying the New Data Set

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

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

6. Sorting orion.employee_payroll and Displaying Grouped Observations

```
proc sort data=orion.employee_payroll out=work.sort_salary2;
  by Employee_Gender descending Salary;
run;
```

```
proc print data=work.sort_salary2;
  by Employee_Gender;
run;
```

7. Sorting orion.employee_payroll and Displaying a Subset of the New Data Set

```
proc sort data=orion.employee_payroll out=work.sort_sal;
  by Employee_Gender descending Salary;
run;

proc print data=work.sort_sal noobs;
  by Employee_Gender;
  sum Salary;
  where Employee_Term_Date is missing and Salary>65000;
  var Employee_ID Salary Marital_Status;
run;
```

8. Retaining the First Observation of Each BY Group

```
proc sort data=orion.orders out=work.custorders nodupkey
  dupout=work.duplicates;
  by Customer_ID;
run;

title 'Unique Customers';
proc print data=work.custorders;
run;

title 'Duplicate Customer Observations';
proc print data=work.duplicates;
run;
title;
```

9. Displaying Titles and Footnotes in a Detail Report

```
title1 'Australian Sales Employees';
title2 'Senior Sales Representatives';
footnotel 'Job_Title: Sales Rep. IV';

proc print data=orion.sales noobs;
  where Country='AU' and Job_Title contains 'Rep. IV';
  var Employee_ID First_Name Last_Name Gender Salary;
run;
title;
footnote;
```

10. Display Column Headings in a Detail Report

a.

```
title 'Entry-level Sales Representatives';
footnote 'Job_Title: Sales Rep. I';

proc print data=orion.sales noobs label;
  where Country='US' and Job_Title='Sales Rep. I';
  var Employee_ID First_Name Last_Name Gender Salary;
```

```

label Employee_ID="Employee ID"
      First_Name="First Name"
      Last_Name="Last Name"
      Salary="Annual Salary";
run;

title;
footnote;

```

b.

```

title 'Entry-level Sales Representatives';
footnote 'Job_Title: Sales Rep. I';

proc print data=orion.sales noobs split=' ';
  where Country='US' and Job_Title='Sales Rep. I';
  var Employee_ID First_Name Last_Name Gender Salary;
  label Employee_ID="Employee ID"
        First_Name="First Name"
        Last_Name="Last Name"
        Salary="Annual Salary";
run;

title;
footnote;

```

11. Writing an Enhanced Detail Report

```

proc sort data=orion.employee_addresses out=work.address;
  where Country='US';
  by State City Employee_Name;
run;

title "US Employees by State";
proc print data=work.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;

```

Solutions to Student Activities (Polls/Quizzes)

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;
```

21

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='.';**

27

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);

```

40

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

45

4.05 Quiz – Correct Answer

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

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

64

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.
```

74

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.

80

p104a03s

4.08 Quiz – Correct Answer

Which footnote or footnotes appear in the second procedure output?

- | | |
|--|--|
| a. Non Sales Employees | c. Non Sales Employees
Confidential |
| b. Orion Star
Non Sales Employees | 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;
```

104

Chapter 5 Formatting Data Values

5.1 Using SAS Formats	5-3
Exercises	5-10
5.2 User-Defined Formats	5-12
Demonstration: Defining and Using a Numeric Format	5-18
Exercises	5-23
5.3 Solutions	5-25
Solutions to Exercises	5-25
Solutions to Student Activities (Polls/Quizzes)	5-28

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

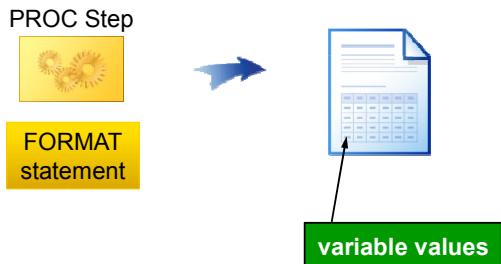


Last_Name	First_Name	Country	Job_Title	Salary	Hire_Date
Zhou	Tom	AU	Sales Manager	\$108,255	06/01/1993
Dawes	Wilson	AU	Sales Manager	\$87,975	01/01/1978
Elvish	Irenie	AU	Sales Rep. II	\$26,600	01/01/1978

4

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

Viewing the Output

Partial PROC PRINT Output

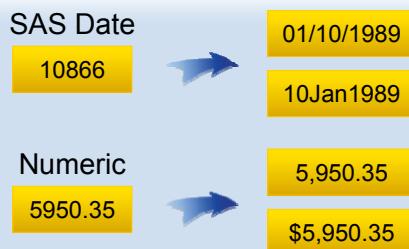
Last_Name	First_Name	Country	Job_Title	Salary	Hire_Date
Zhou	Tom	AU	Sales Manager	\$108,255	06/01/1993
Dawes	Wilson	AU	Sales Manager	\$87,975	01/01/1978
Elvish	Irenie	AU	Sales Rep. II	\$26,600	01/01/1978
Ngan	Christina	AU	Sales Rep. II	\$27,475	07/01/1982
Hotstone	Kimiko	AU	Sales Rep. I	\$26,190	10/01/1989

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

SAS Formats

SAS formats have the following form:

`<$>format<w>.<d>`

\$	Indicates a character format.
<i>format</i>	Names the SAS format.
<i>w</i>	Specifies the total format width, including decimal places and special characters.
.	Is required syntax. Formats always contain a period (.) as part of the name.
<i>d</i>	Specifies the number of decimal places to display in numeric formats.

9

SAS Formats

Selected SAS formats:

Format	Definition
\$ <i>w</i> .	Writes standard character data.
<i>w.d</i>	Writes standard numeric data.
COMMA <i>w.d</i>	Writes numeric values with a comma that separates every three digits and a period that separates the decimal fraction.
DOLLAR <i>w.d</i>	Writes numeric values with a leading dollar sign, a comma that separates every three digits, and a period that separates the decimal fraction.
COMMAX <i>w.d</i>	Writes numeric values with a period that separates every three digits and a comma that separates the decimal fraction.
EUROX <i>w.d</i>	Writes numeric values with a leading euro symbol (€), a period that separates every three digits, and a comma that separates the decimal fraction.

10

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
COMMADOT12.2	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.

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

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 ddmmmyyyy.
Term_Date mmmmyyyy.;`
- 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.

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

- Open **p105e01** and submit. Review the output.
- Modify the PROC PRINT step to display only **Employee_ID**, **Salary**, **Birth_Date**, and **Employee_Hire_Date**.
- 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.

Obs	Employee_ID	Salary	Birth_Date	Employee_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

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					
Employee_ID	First Name	Last Name	Title	Salary	Date 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.

Employee_ID	First_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"

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

23

p105d02

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

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

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.

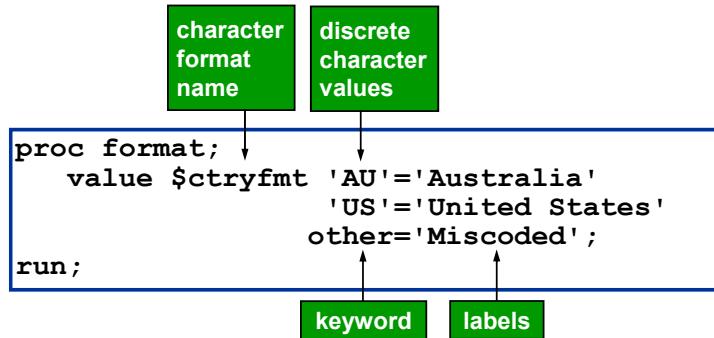
5.03 Multiple Answer Poll

Which names are invalid for user-defined formats?

- a. \$stfmt
- b. \$3levels
- c. _4years
- d. salranges
- e. dollar

29

Defining a Character Format



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

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

Specifying Ranges of Values

Use PROC FORMAT to specify the salary range for each tier.



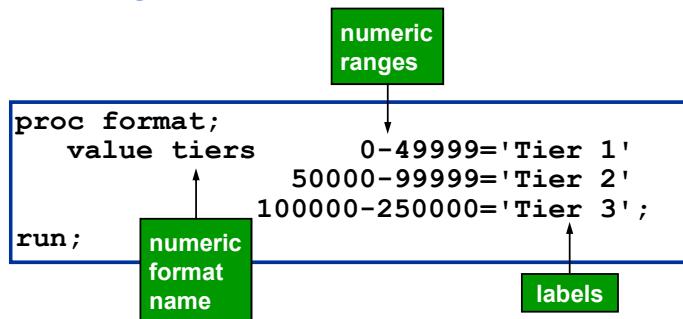
PROC FORMAT



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



37

p105d04



Defining and Using a Numeric Format

p105d04

This demonstration illustrates the use of a user-defined numeric format.

```

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?

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;
```

40

p105d05

LOW and HIGH Keywords

```

proc format;
  value tiers      low-<50000 = 'Tier 1'
                  50000-<100000='Tier 2'
                  100000-high   ='Tier 3';
run;

```

The diagram illustrates the range of values for the LOW and HIGH keywords in a format statement. A blue box encloses the code. Inside, a green box labeled "the lowest possible value" points to the first condition "low-<50000". Another green box labeled "the highest possible value" points to the last condition "100000-high".

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

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

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



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

4. Creating User-Defined Formats

- Retrieve the starter program **p105e04**.
- Create a character format named \$GENDER that displays gender codes as follows:

F	Female
M	Male

- Create a numeric format named MNAME that displays month numbers as follows:

1	January
2	February
3	March

- Add a PROC PRINT step to display the data set, applying these two user-defined formats to the **Employee_Gender** and **BirthMonth** variables, respectively.
- Submit the program to produce the following report. The results contain 113 observations.

Employees with Birthdays in Q1			
Obs	Employee_ID	Employee_Gender	Birth Month
1	120103	Male	January
2	120107	Female	January
3	120108	Female	February
...			
112	121142	Male	February
113	121148	Male	January

Level 2

5. Defining Ranges in User-Defined Formats

- Retrieve the starter program **p105e05**.
- Create a character format named \$GENDER that displays gender codes as follows:

F	Female
M	Male
Any other value	Invalid code

- c. Create a numeric format named SALRANGE that displays salary ranges as follows:

At least 20,000 but less than 100,000	Below \$100,000
At least 100,000 and up to 500,000	\$100,000 or more
missing	Missing salary
Any other value	Invalid salary

- d. In the PROC PRINT step, apply these two user-defined formats to the **Gender** and **Salary** variables, respectively. Submit the program to produce the following report:

Partial PROC PRINT Output

Salary and Gender Values for Non-Sales Employees				
Obs	Employee_ID	Job_Title	Salary	Gender
1	120101	Director	\$100,000 or more	Male
2	120104	Administration Manager	Below \$100,000	Female
3	120105	Secretary I	Below \$100,000	Female
4	120106	Office Assistant II	Missing salary	Male
5	120107	Office Assistant III	Below \$100,000	Female
6	120108	Warehouse Assistant II	Below \$100,000	Female
7	120108	Warehouse Assistant I	Below \$100,000	Female
8	120110	Warehouse Assistant III	Below \$100,000	Male
9	120111	Security Guard II	Below \$100,000	Male
10	120112		Below \$100,000	Female
11	120113	Security Guard II	Below \$100,000	Female
12	120114	Security Manager	Below \$100,000	Invalid code
13	120115	Service Assistant I	Invalid salary	Male

Challenge

6. Exploring Format Storage Options

User-defined formats are stored in the **formats** catalog in the **work** library, **work.formats**. Use the SAS Help Facility or product documentation to explore permanent format catalogs in PROC FORMAT.

What option enables you to store the formats in a permanent library? _____

What option causes SAS to look for formats in permanent libraries? _____

5.3 Solutions

Solutions to Exercises

1. Displaying Formatted Values in a Detail Report

```
proc print data=orion.employee_payroll;
  var Employee_ID Salary Birth_Date Employee_Hire_Date;
  format Salary dollar11.2 Birth_Date mmddyy10.
        Employee_Hire_Date date9.;
```

```
run;
```

- a. Submit **p105e01**
- b. Add a VAR statement.
- c. Add a FORMAT statement.

2. Displaying Formatted Values in a Detail Report

```
title1 'US Sales Employees';
title2 'Earning Under $26,000';

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 dollar10. Hire_Date monyy7.;
run;
title;
footnote;
```

3. Exploring Functions by Category

```
proc print data=orion.sales noobs;
  var Employee_ID First_Name Last_Name Job_Title;
  format First_Name Last_Name $upcase. Job_Title $quote. ;
run;
```

4. Creating User-Defined Formats

```
data Q1Birthdays;
  set orion.employee_payroll;
  BirthMonth=month(Birth_Date);
  if BirthMonth le 3;
run;

proc format;
  value $gender
    'F'='Female'
    'M'='Male';
  value mname
    1='January'
    2='February'
    3='March';
run;

title 'Employees with Birthdays in Q1';
proc print data=Q1Birthdays;
  var Employee_ID Employee_Gender BirthMonth;
  format Employee_Gender $gender.
        BirthMonth mname.;
```

```
run;
title;
```

5. Defining Ranges in User-Defined Formats

```
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;

title1 'Salary and Gender Values';
title2 'for Non-Sales Employees';

proc print data=orion.nonsales;
  var Employee_ID Job_Title Salary Gender;
  format Salary salrange. Gender $gender.;
run;
title;
```

6. Exploring Format Storage Options

What option enables you to store the formats in a permanent library? **LIBRARY=**

What option causes SAS to look for formats in permanent libraries? **FMTSEARCH=**

Solutions to Student Activities (Polls/Quizzes)

5.01 Quiz – Correct Answer

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.”

The `Zw.d` format writes standard numeric data with leading zeros. It is similar to the `w.d` format except that `Zw.d` pads right-aligned output with zeros instead of blanks.

14

5.02 Quiz – Correct Answer

Which FORMAT statement creates the output shown below?

- a. `format Birth_Date Hire_Date mmddyy10.
Term_Date monyy7.;`
- b. `format Birth_Date Hire_Date ddmmmyyyy.
Term_Date mmmmyyyy.;`
- c. `format Birth_Date Hire_Date ddmmyy10.
Term_Date monyy7.;`

output →

	Birth_Date	Hire_Date	Term_Date
	21/05/1969	15/10/1992	MAR2007

18

5.03 Multiple Answer Poll – Correct Answer

Which names are invalid for user-defined formats?

- a. \$stfmt
- b.** \$3levels
- c. _4years
- d. salranges
- e.** dollar

Character formats must have a dollar sign as the first character and a letter or underscore as the second character.

User-defined formats cannot be given the name of a format provided by SAS.

30

5.04 Quiz – Correct Answer

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;
```

41

p105d05

Chapter 6 Reading SAS® Data Sets

6.1	Reading a SAS Data Set.....	6-3
	Exercises	6-15
6.2	Customizing a SAS Data Set	6-17
	Exercises	6-39
6.3	Solutions	6-42
	Solutions to Exercises	6-42
	Solutions to Student Activities (Polls/Quizzes)	6-45

6.1 Reading a SAS Data Set

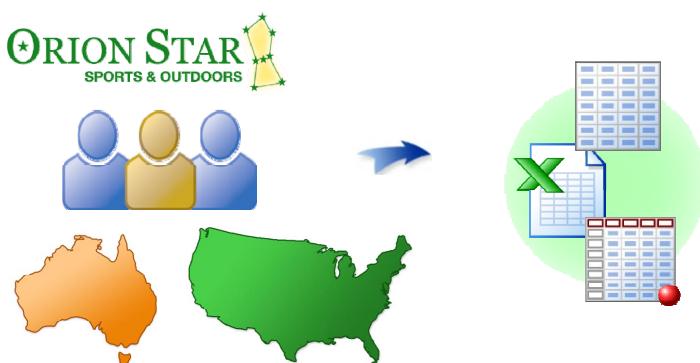
Objectives

- 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.

3

Business Scenario

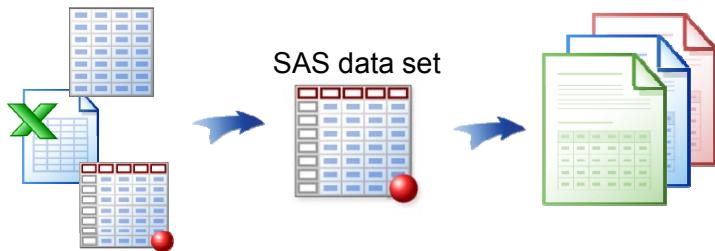
Information about Orion Star sales employees resides in several input sources.



4

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.



5

6.01 Multiple Answer Poll

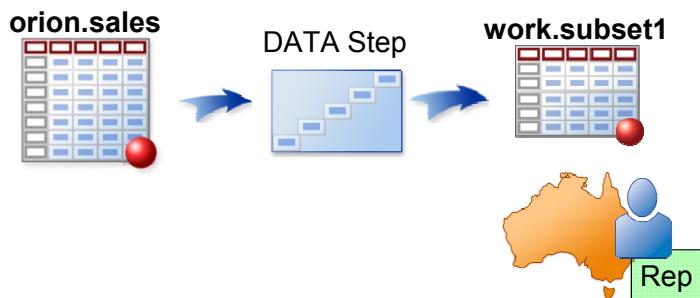
Which types of files will you read into SAS?

- a. SAS data sets
- b. Excel worksheets
- c. database tables
- d. raw data files
- e. other
- f. not sure

6

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.



7

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;

8

p106d01

DATA Statement

The *DATA statement* begins a DATA step and provides the name of the SAS data set to create.

```
data work.subset1;           DATA output-SAS-data-set;
  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.

9

p106d01

SET Statement

The *SET statement* reads observations from an existing SAS data set for further processing in the DATA step.

```
data work.subset1;           SET input-SAS-data-set;
  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.

10

p106d01

WHERE Statement

The *WHERE statement* selects observations from a SAS data set that meet a particular condition.

```
data work.subset1;
  set orion.sales;
  where Country='AU' and
        Job_Title contains 'Rep';
run;
```

WHERE WHERE-expression;

The variables named in the WHERE expression must exist in the input SAS data set.

11

p106d01

Using a WHERE statement might improve the efficiency of your SAS programs because SAS only processes the observations that meet the condition or conditions in the WHERE expression.

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.

12

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

13

p106d01

Setup for the Poll

Consider the DATA step below.

```
data us;
  set orion.sales;
  where Country='US';
run;
```

14

p106a01

6.02 Multiple Choice Poll

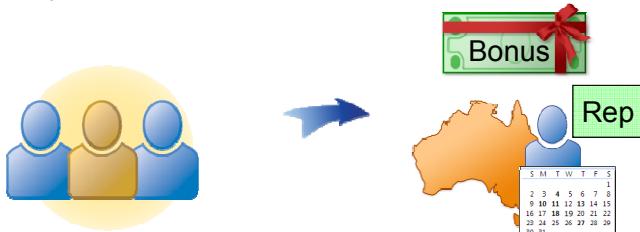
Considering this DATA step, which statement is true?

- 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.

15

Business Scenario: Part 2

Orion Star management wants to give a 10% bonus to each Australian Sales representative hired before January 1, 2000.



18

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?



19

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;
```

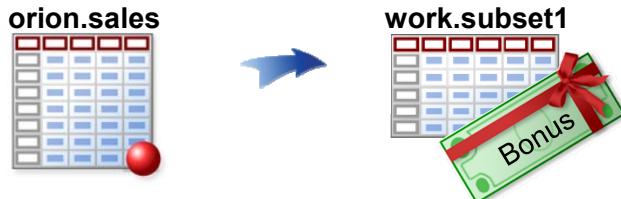
- ✍ A SAS date constant is a date written in the form 'ddmmm<yy>yy'd.

20

p106d02

Considerations

Create a data set that includes the new variable, **Bonus**, which represents a 10% bonus.



21

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;
```

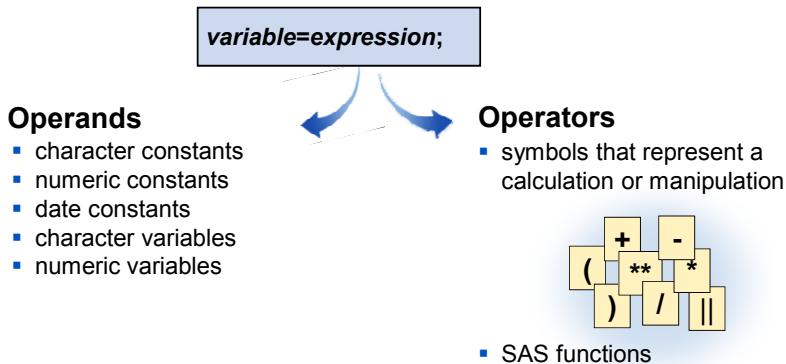
variable=expression;

22

p106d02a

Assignment Statement

The *expression* consists of operands and operators.



23

The operators can be character or arithmetic operators or SAS functions. A function is a routine that accepts arguments, performs a calculation or manipulation using the arguments, and returns a single value.

Sample Assignment Statements

Example	Type
<code>Salary=26960 ;</code>	Numeric constant
<code>Gender='F' ;</code>	Character constant
<code>Hire_Date='21JAN1995'd ;</code>	Date constant
<code>BonusMonth=month(Hire_Date) ;</code>	SAS function
<code>Bonus=Salary*.10 ;</code>	Arithmetic expression

24

The MONTH function accepts a SAS date and returns the month portion of the date as an integer between 1 and 12. You investigate this and other SAS functions in a later chapter.

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.

25

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.

26

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

27

p106d02a

No format was specified for **Bonus**, so PROC PRINT uses a BESTw.d format. One decimal position is sufficient to display the values on this page.

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;**

28



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. Creating a SAS Data Set

- Retrieve and submit the starter program **p106e01**.

What is the name of the variable that contains gender values? _____

What are the two observed gender values? _____

- Add a DATA step before the PROC PRINT step to create a new data set named **work.youngadult** using the data set **orion.customer_dim** as input. Include a WHERE statement to select only female customers.

Submit the program and confirm that **work.youngadult** was created with 30 observations and 11 variables.

- Modify the program to select female customers whose age is between 18 and 36. Submit the program and confirm that **work.youngadult** was created with 15 observations and 11 variables.
- Modify the program to select 18- to 36-year-old female customers who have the word *Gold* in their **Customer_Group** value. Submit the program and confirm that **work.youngadult** was created with 5 observations and 11 variables.
- Add an assignment statement to the DATA step to create a new variable, **Discount**, and assign it a value of .25.
- Modify the PROC PRINT step to print the new data set as shown below. Use an ID statement to display **Customer_ID** instead of the Obs column. Results should contain five observations.

Customer_ID	Customer_Name	Customer_Age	Customer_Gender	Customer_Group	Discount
5	Sandrina Stephano	28	F	Orion Club Gold members	0.25
9	Cornelia Krahil	33	F	Orion Club Gold members	0.25
45	Dianne Patchin	28	F	Orion Club Gold members	0.25
49	Annmarie Leveille	23	F	Orion Club Gold members	0.25
2550	Sanelisiwe Collier	19	F	Orion Club Gold members	0.25

Level 2

2. Creating a SAS Data Set

- Write a DATA step to create a new data set named **work.assistant** using the data set **orion.staff** as input.
- The **work.assistant** data set should contain only the observations where **Job_Title** contains **Assistant** and **Salary** is less than \$26,000.

- c. Create two new variables, **Increase** and **New_Salary**.
 - **Increase** is **Salary** multiplied by 0.10.
 - **New_Salary** is **Salary** added to **Increase**.
- d. Generate a detail listing report as shown below. Display **Employee_ID** as the identifier in place of the Obs column. The results should contain five observations.

Employee_ID	Job_Title	Salary	Increase	New_Salary
120685	Warehouse Assistant I	\$25,130.00	\$2,513.00	\$27,643.00
120688	Warehouse Assistant I	\$25,905.00	\$2,590.50	\$28,495.50
120690	Warehouse Assistant I	\$25,185.00	\$2,518.50	\$27,703.50
121010	Service Assistant I	\$25,195.00	\$2,519.50	\$27,714.50
121011	Service Assistant I	\$25,735.00	\$2,573.50	\$28,308.50

Challenge

3. Using the SOUNDS-LIKE Operator to Select Observations

- a. Write a DATA step to create a new data set named **work.tony** using **orion.customer_dim** as input.
- b. Include a WHERE statement in the DATA step to select observations in which the **Customer_FirstName** value sounds like *Tony*.

 Documentation on the SOUNDS-LIKE operator can be found in the SAS Help facility or product documentation by searching for “sounds-like operator.”

- c. Write a PROC PRINT step to create the following report:

Obs	Customer_FirstName	Customer_LastName
1	Tonie	Asmussen
2	Tommy	Mcdonald

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.

33

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.



34

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.
```

35

p106d03

Viewing the Output

```
proc print data=work.subset1;
run;
```

Partial PROC PRINT Output

	First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
Obs						
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

36

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.

37

p106d04

When you use a KEEP statement, be sure to name every variable to be written to the new SAS data set, including any variables created within the step, such as **Bonus**.

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.
```

38

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

39

p106d04

Business Scenario: Behind the Scenes

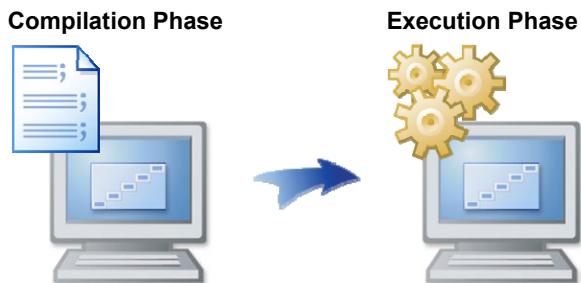
Orion Star programmers need to understand the internal processing that occurs when a DATA step is submitted.



41

DATA Step Processing

SAS processes the DATA step in two phases:



42

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.

43

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;
```

44

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

45

...

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		



46

...

Compilation

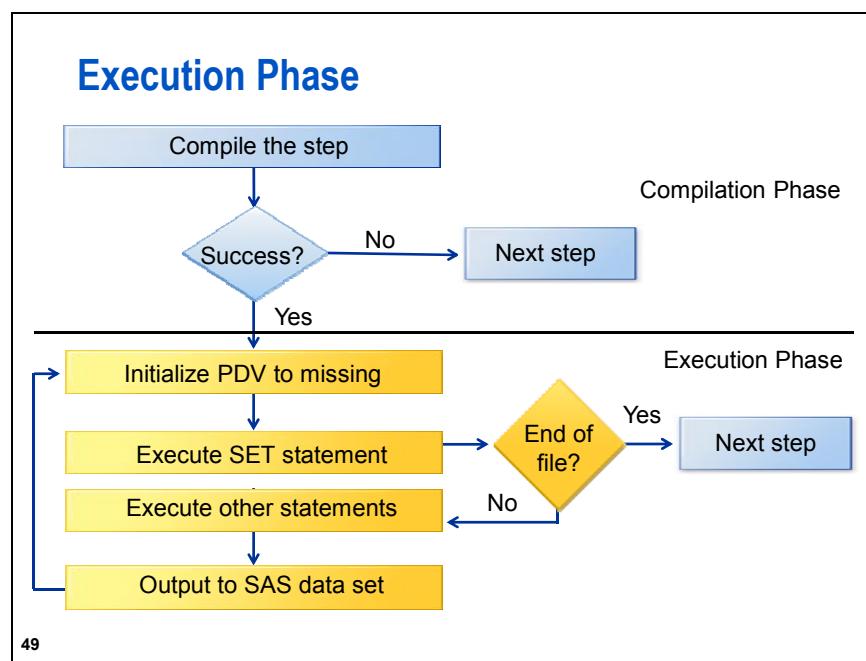
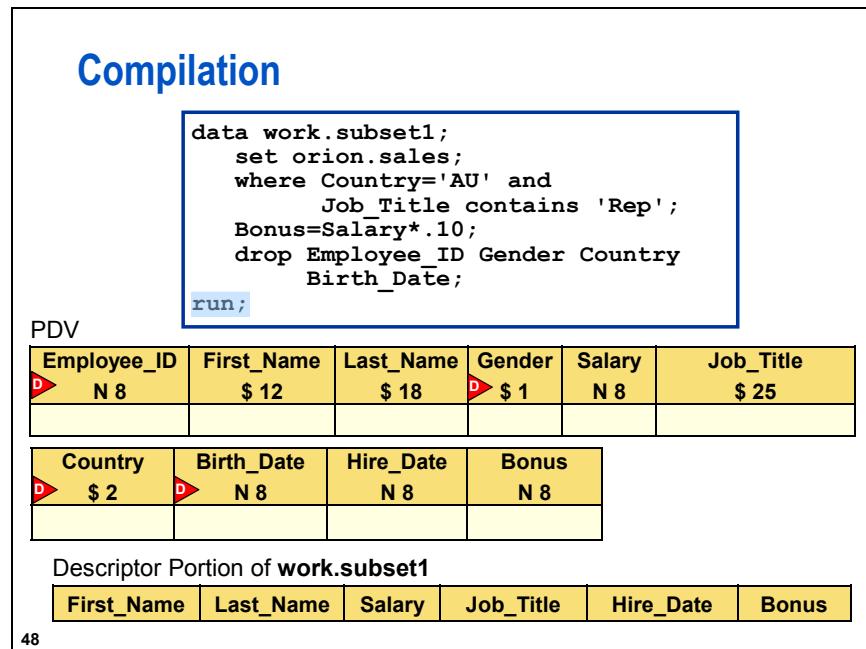
```
data work.subset1;
  set orion.sales;
  where Country='AU' and
    Job_Title contains 'Rep';
  Bonus=Salary*.10;
  drop Employee_ID Gender Country
    Birth_Date;
run;
```

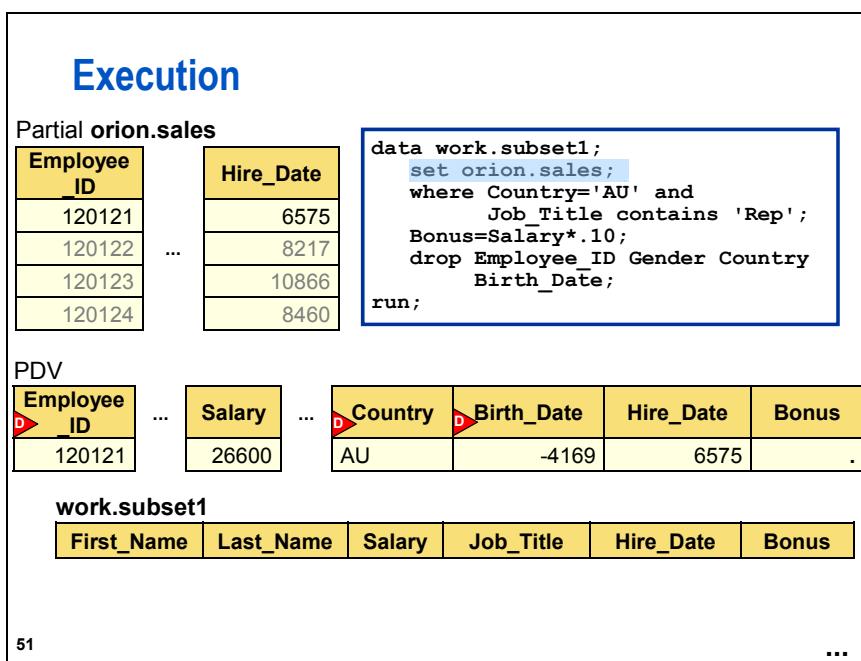
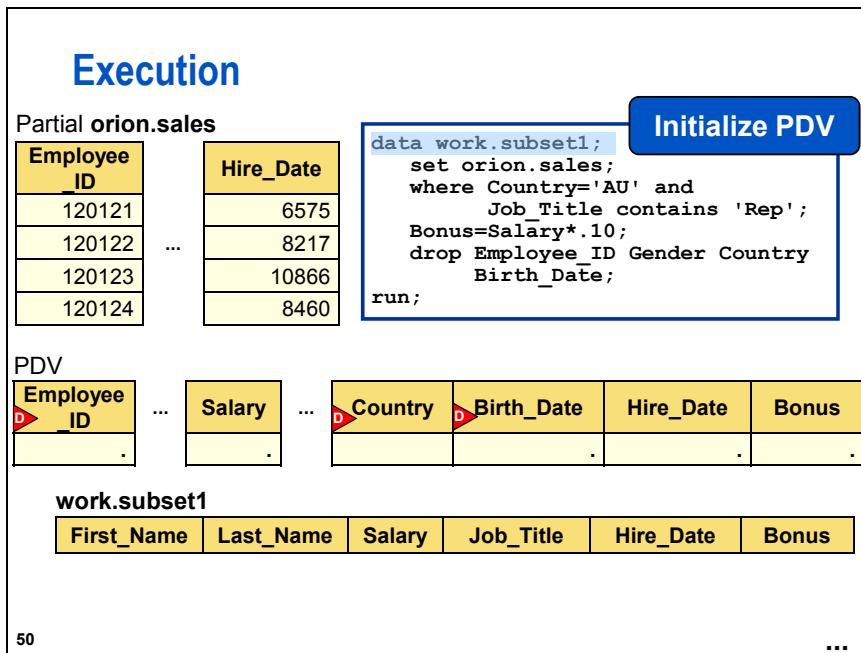
PDV

Employee_ID D N 8	First_Name \$ 12	Last_Name \$ 18	Gender D \$ 1	Salary N 8	Job_Title \$ 25
Country D \$ 2	Birth_Date D N 8	Hire_Date N 8	Bonus N 8		

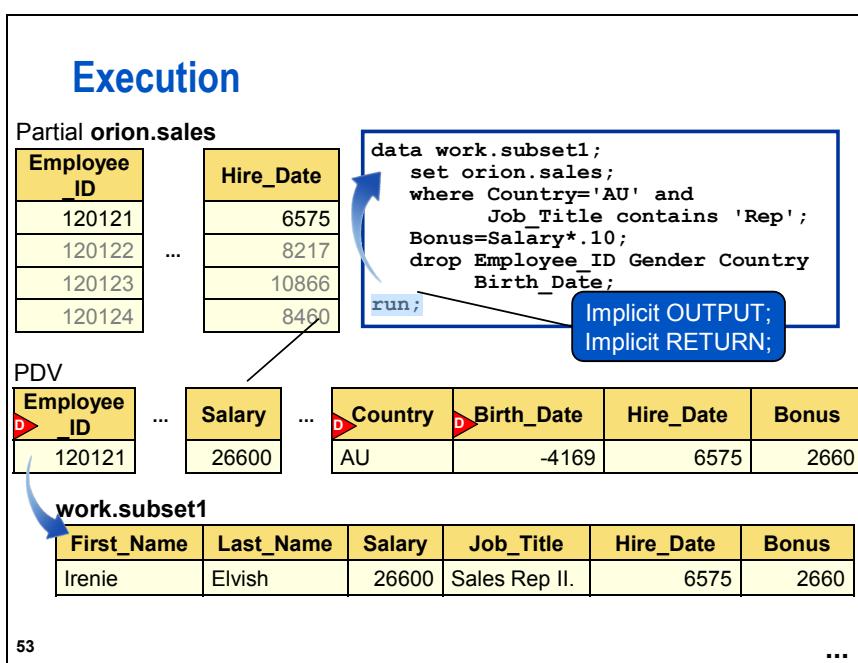
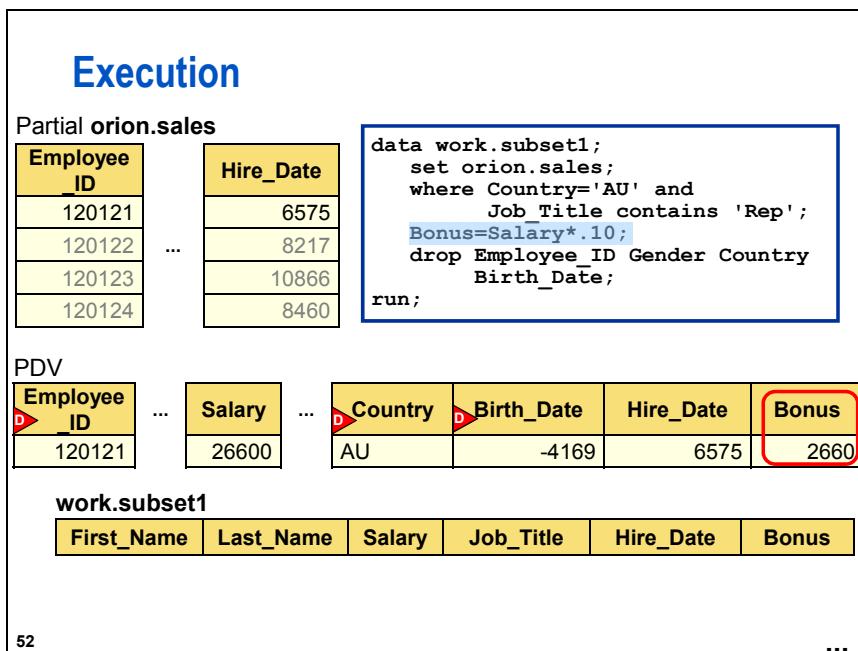
47

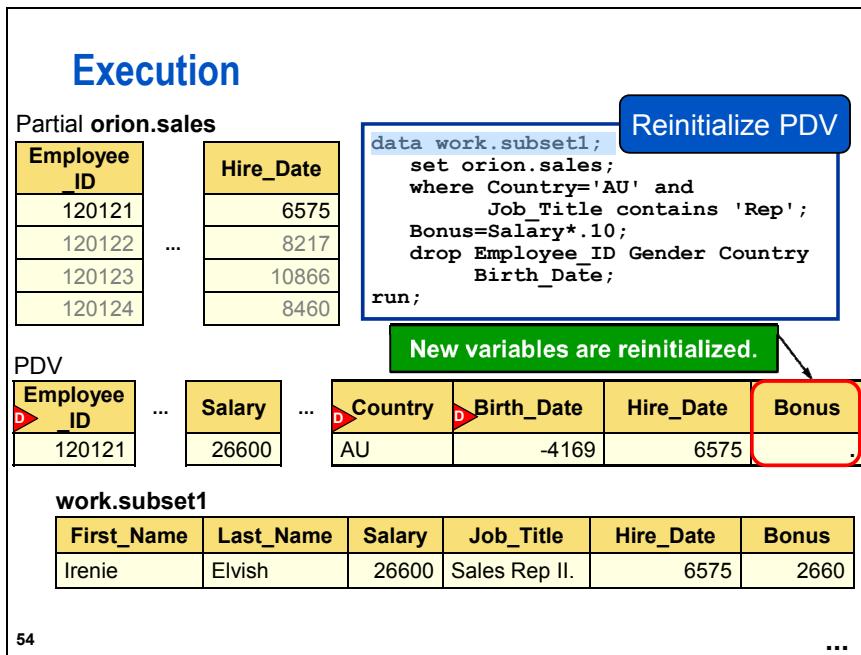
...



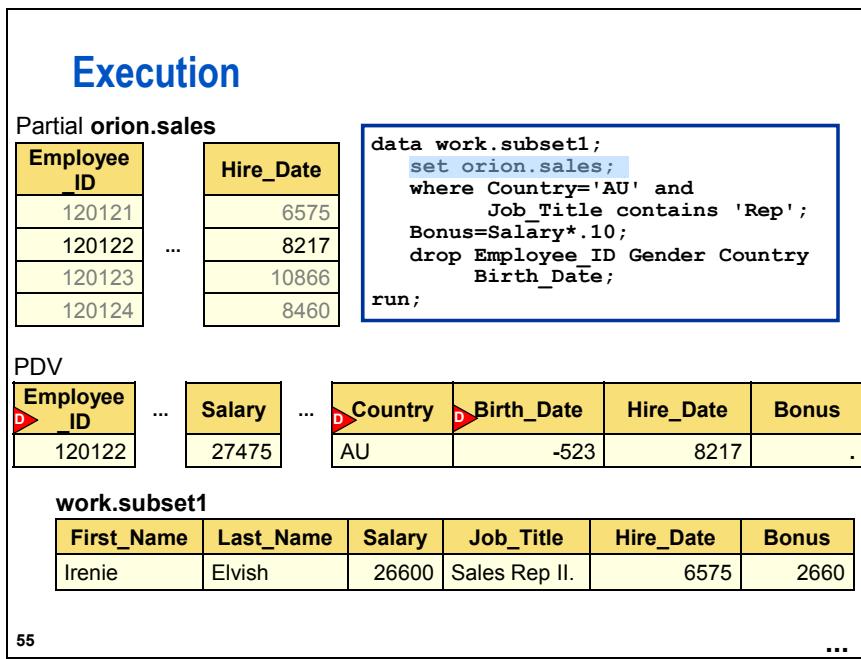


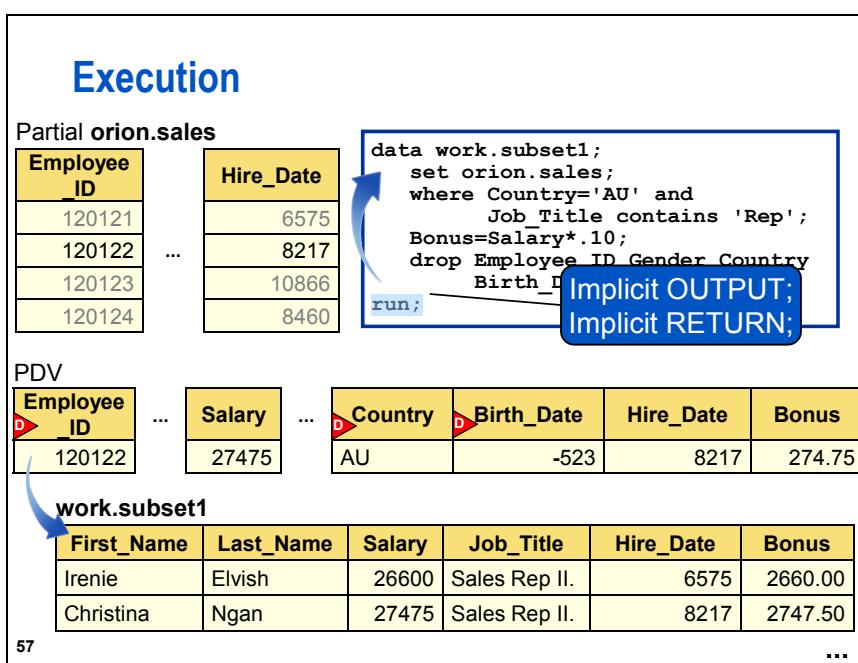
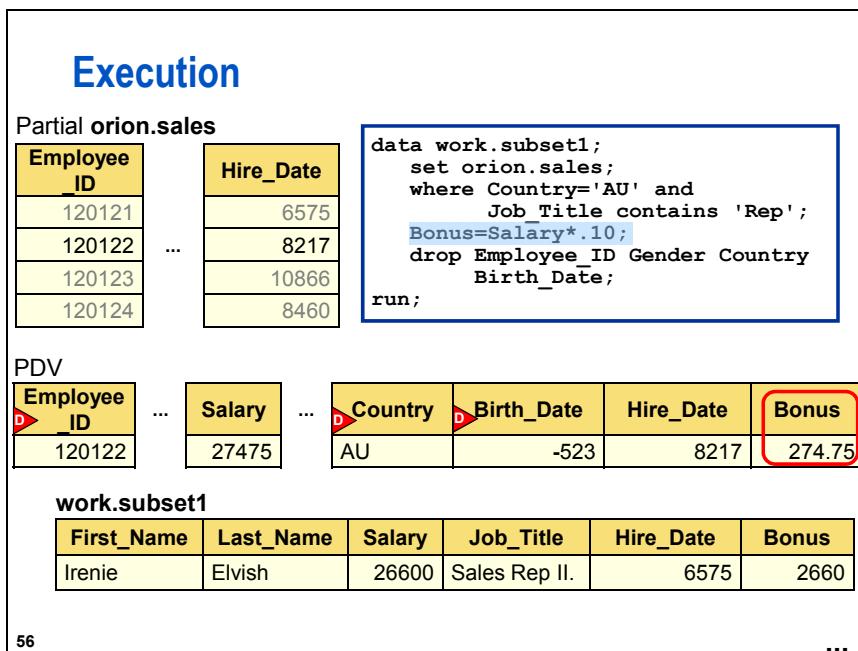
The WHERE statement is executed before the other executable statements.





Only the new variables are reinitialized. The variables that come from the input data set are *not* reinitialized because they are overwritten when the next observation is read into the PDV. Values in the PDV are overwritten even if values in the next observation are missing.





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_Title=
Bonus=Salary*.10;
drop Employee_ID Gender Country
Birth_Date;
run;
```

Continue until EOF

PDV

Employee_ID	Salary	Country	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	2660.00
Christina	Ngan	27475	Sales Rep II.	8217	2747.50

58

Viewing the Output

```
proc print data=work.subset1;
run;
```

Partial PROC PRINT Output

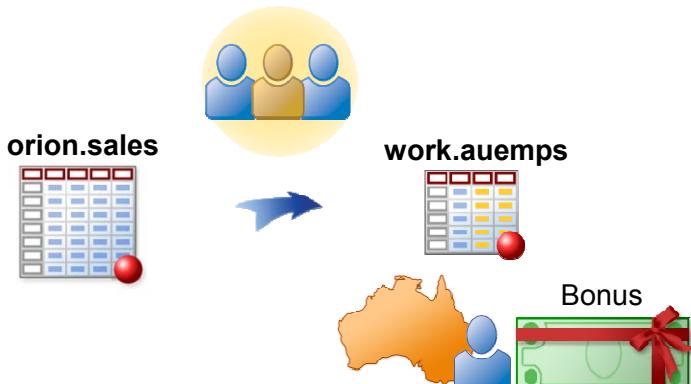
Obs	First_Name	Last_Name	Salary	Job_Title	Hire_Date	Bonus
1	Irenie	Elvish	26600	Sales Rep. II	6575	2660.0
2	Christina	Ngan	27475	Sales Rep. II	8217	2747.5
3	Kimiko	Hotstone	26190	Sales Rep. I	10866	2619.0
4	Lucian	Daymond	26480	Sales Rep. I	8460	2648.0
5	Fong	Hofmeister	32040	Sales Rep. IV	8460	3204.0

p106d03

59

Business Scenario: Part 4

Create a data set that contains all Australian employees whose **Bonus** is at least \$3000.



61

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.

62

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;
```

63

p106a03

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.

65

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.

66

Viewing the Output

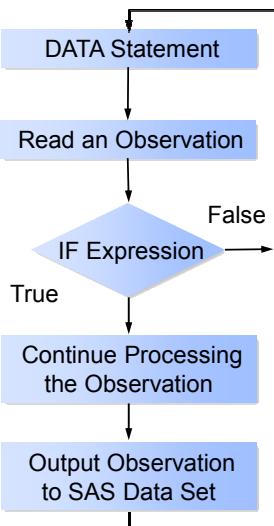
```
proc print data=work.auemps;
  var First_Name Last_Name Salary Bonus;
run;
```

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

67

Processing the Subsetting IF Statement



68

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;

```

69



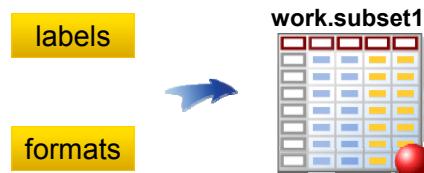
WHERE versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
SET statement	Yes	Yes
assignment statement	No	Yes

70

Business Scenario: Part 5

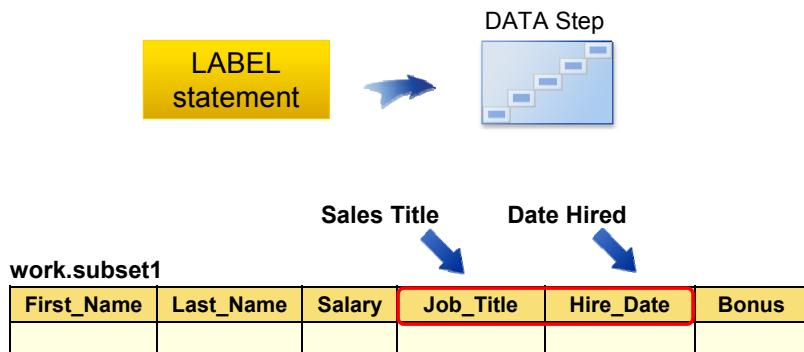
Define permanent labels and formats for some of the variables in the new data set.



72

LABEL Statement

The LABEL statement assigns descriptive labels to variables.



73

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' ...>;

74

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	

75

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

76

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

77

p106d06

PROC PRINT is the only SAS procedure that requires either the LABEL option or the SPLIT= option to use custom labels.

6.05 Quiz

What column heading will be displayed for **Job_Title** in the program below?

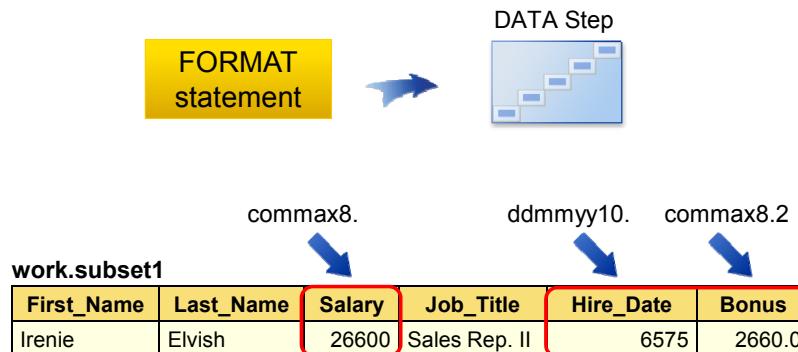
```
data work.us;
  set orion.sales;
  where Country='US';
  Bonus=Salary*.10;
  label Job_Title='Sales Title';
  drop Employee_ID Gender Country
    Birth_Date;
run;
proc print data=work.subset1 label;
  label Job_Title='Title';
run;
```

78

p106a05

FORMAT Statement

The FORMAT statement associates formats with variables.



80

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 ...;

81

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.	

82

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

83

p106d07



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

4. Subsetting Observations Based on Two Conditions

- a. Retrieve the starter program **p106e04**.
- b. Modify the DATA step to select only the observations with **Emp_Hire_Date** values on or after July 1, 2010. Subset the observations as they are being read into the program data vector.
- c. In the DATA step, write another statement to select only the observations that have an increase greater than 3000.
- d. The new data set should contain only the following variables: **Employee_ID**, **Emp_Hire_Date**, **Salary**, **Increase**, and **NewSalary**.
- e. Add permanent labels for **Employee_ID**, **Emp_Hire_Date**, and **NewSalary** as shown in the report below.
- f. Add permanent formats to display **Salary** and **NewSalary** with dollar signs, commas, and two decimal places, and **Increase** with commas and no decimal places.
- g. Submit a PROC CONTENTS step to verify that the labels and formats are stored in the descriptor portion of the new data set, **work.increase**.

Partial PROC CONTENTS Output

Alphabetic List of Variables and Attributes						
#	Variable	Type	Len	Format	Informat	Label
3	Emp_Hire_Date	Num	8	DATE9..	DATE9..	Hire Date
1	Employee_ID	Num	8	12..		Employee ID
4	Increase	Num	8	COMMA5..		
5	NewSalary	Num	8	DOLLAR10.2		New Annual Salary
2	Salary	Num	8	DOLLAR10.2		Employee Annual Salary

- h. Some variables have labels and formats that were not defined in this program. How were these created? _____
- i. Submit the program to create the PROC PRINT report below, with labels split over multiple lines. Results should contain 10 observations.

Obs	Employee ID	Employee		New	
		Annual Salary	Hire Date	Increase	Annual Salary
1	120128	\$30,890.00	01NOV2010	3,089	\$33,979.00
2	120144	\$30,265.00	01OCT2010	3,027	\$33,291.50
3	120161	\$30,785.00	01OCT2010	3,079	\$33,863.50
...					
9	121085	\$32,235.00	01JAN2011	3,224	\$35,458.50
10	121107	\$31,380.00	01JUL2010	3,138	\$34,518.00

Level 2

5. Subsetting Observations Based on Three Conditions

- a. Write a DATA step to create **work.delays** using **orion.orders** as input.
- b. Create a new variable, **Order_Month**, and set it to the month of **Order_Date**.
Hint: Use the MONTH function.
- c. Use a WHERE statement and a subsetting IF statement to select only the observations that meet all of the following conditions:
 - **Delivery_Date** values that are more than four days beyond **Order_Date**
 - **Employee_ID** values that are equal to 99999999
 - **Order_Month** values occurring in August
- d. The new data set should include only **Employee_ID**, **Customer_ID**, **Order_Date**, **Delivery_Date**, and **Order_Month**.
- e. Add permanent labels for **Order_Date**, **Delivery_Date**, and **Order_Month** as shown below.
- f. Add permanent formats to display **Order_Date** and **Delivery_Date** as MM/DD/YYYY.
- g. Add a PROC CONTENTS step to verify that the labels and formats were stored permanently.

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
2	Customer_ID	Num	8	12.	Customer ID
4	Delivery_Date	Num	8	MMDDYY10.	Date Delivered
1	Employee_ID	Num	8	12.	Employee ID
3	Order_Date	Num	8	MMDDYY10.	Date Ordered
5	Order_Month	Num	8		Month Ordered

- h. Write a PROC PRINT step to create the report below. Results should contain nine observations.

Obs	Employee_ID	Customer_ID	Order_Date	Delivery_Date	Order_Month
1	99999999	70187	08/13/2007	08/18/2007	8
2	99999999	52	08/20/2007	08/26/2007	8
3	99999999	16	08/27/2007	09/04/2007	8
4	99999999	61	08/29/2007	09/03/2007	8
5	99999999	2550	08/10/2008	08/15/2008	8
6	99999999	70201	08/15/2008	08/20/2008	8
7	99999999	9	08/10/2009	08/15/2009	8
8	99999999	71	08/30/2010	09/05/2010	8
9	99999999	70201	08/24/2011	08/29/2011	8

Challenge

6. Using an IF-THEN/DELETE Statement to Subset Observations

- a. Write a DATA step to create **work.bigdonations** using **orion.employee_donations** as input.
- b. Use the SUM function to create a new variable, **Total**, which holds the sum of the four quarterly donations.

- c. Use the N function to create a new variable, **NumQtrs**, which holds the count of nonmissing values in **Qtr1**, **Qtr2**, **Qtr3**, and **Qtr4**. Explore the N function in the SAS Help facility or online documentation.
- d. The new data set should not include the charities or method of payment.
- e. The final data set should contain only observations meeting the following two conditions:
 - **Total** values greater than or equal to 50
 - **NumQtrs** value equal to 4
 Use an IF-THEN/DELETE statement to eliminate the observations where the conditions are not met. Explore the use of IF-THEN/DELETE in the SAS Help facility or online documentation.
- f. Store permanent labels in the new data set as shown in the report below.
- g. Create the following report to verify that the labels were stored:

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
1	Employee_ID	Num	8	12.	Employee ID
7	NumQtrs	Num	8		
2	Qtr1	Num	8		First Quarter
3	Qtr2	Num	8		Second Quarter
4	Qtr3	Num	8		Third Quarter
5	Qtr4	Num	8		Fourth Quarter
6	Total	Num	8		

- h. Create the report below. The results should contain 50 observations.

Employee ID	First Quarter	Second Quarter	Third Quarter	Fourth Quarter	Total	Num Qtrs
120267	15	15	15	15	60	4
120269	20	20	20	20	80	4
120271	20	20	20	20	80	4
120275	15	15	15	15	60	4
120660	25	25	25	25	100	4

6.3 Solutions

Solutions to Exercises

1. Creating a SAS Data Set

- a. What is the name of the variable that contains gender values? **Customer_Gender**

What are the possible values of this variable? **M or F**

```
data work.youngadult;
  set orion.customer_dim;
  where Customer_Gender='F' and
        Customer_Age between 18 and 36 and
        Customer_Group contains 'Gold';
  Discount=.25;
```

```

run;

proc print data=work.youngadult;
  var Customer_Name Customer_Age
      Customer_Gender Customer_Group Discount;
  id Customer_ID;
run;

```

2. Creating a SAS Data Set

```

data work.assistant;
  set orion.staff;
  where Job_Title contains 'Assistant' and
        Salary<26000;
  Increase=Salary*.10;
  New_Salary=Salary+Increase;
run;

proc print data=work.assistant;
  id Employee_ID;
  var Job_Title Salary Increase New_Salary;
  format Salary Increase New_Salary dollar10.2;
run;

```

3. Using the SOUNDS-LIKE Operator to Select Observations

```

data work.tony;
  set orion.customer_dim;
  where Customer_FirstName=* 'Tony';
run;

proc print data=work.tony;
  var Customer_FirstName Customer_LastName;
run;

```

4. Subsetting Observations Based on Two Conditions

```

data work.increase;
  set orion.staff;
  where Emp_Hire_Date='01JUL2010'd;
  Increase=Salary*0.10;
  if Increase>3000;
  NewSalary=Salary+Increase;
  label Employee_ID='Employee ID'
        Salary='Annual Salary'
        Emp_Hire_Date='Hire Date'
        NewSalary='New Annual Salary';
  format Salary NewSalary dollar10.2 Increase comma5. ;
  keep Employee_ID Emp_Hire_Date Salary Increase NewSalary;
run;

proc print data=work.increase split=' ';
run;

```

The existing labels and formats were inherited from the input data set.

5. Subsetting Observations Based on Three Conditions

```

data work.delays;
  set orion.orders;
  where Order_Date+4<Delivery_Date
    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=work.delays;
run;

proc print data=work.delays;
run;

```

6. Using an IF-THEN/DELETE Statement to Subset Observations

```

data work.bigdonations;
  set orion.employee_donations;
  Total=sum(Qtr1,Qtr2,Qtr3,Qtr4);
  NumQtrs=n(Qtr1,Qtr2,Qtr3,Qtr4);
  if Total<50 or NumQtrs<4 then delete;
  label Qtr1='First Quarter'
    Qtr2='Second Quarter'
    Qtr3='Third Quarter'
    Qtr4='Fourth Quarter';
  drop Recipients Paid_By;
run;

proc contents data=work.bigdonations;
run;

proc print data=work.bigdonations label noobs;
run;

```

Solutions to Student Activities (Polls/Quizzes)

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;
```

16

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;** $\rightarrow 4+10/2 \rightarrow 4+5 \rightarrow 9$

num=(y+z)/2; $\rightarrow 14/2 \rightarrow 7$

b. **num=x+z/2;** $\rightarrow .+10/2 \rightarrow .+5 \rightarrow .$

29

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.

64

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.

79

Chapter 7 Reading Spreadsheet and Database Data

7.1	Reading Spreadsheet Data	7-3
	Demonstration: Reading Excel Data in SAS Enterprise Guide.....	7-10
	Demonstration: Reading Excel Data in the SAS Windowing Environment.....	7-13
	Demonstration: Creating a SAS Data Set.....	7-16
	Exercises	7-17
7.2	Reading Database Data.....	7-20
7.3	Solutions	7-24
	Solutions to Exercises	7-24
	Solutions to Student Activities (Polls/Quizzes)	7-26

7.1 Reading Spreadsheet 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.

3

Business Scenario

The Sales Manager has requested a report about Orion Star sales employees from Australia and the United States.

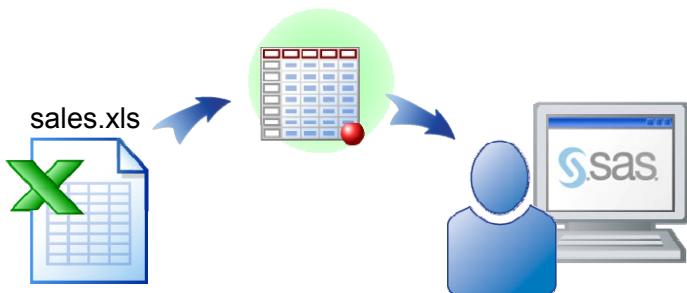
The input data is in an Excel workbook.



4

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.



5

Examine the Workbook

Partial sales.xls

two worksheets

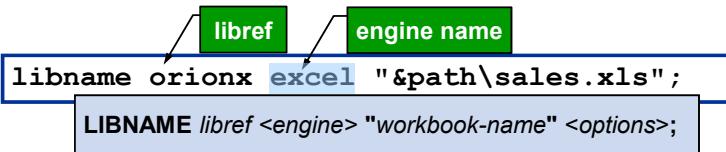
cells formatted as dates

	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	Irene	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	Lucien	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	Setyakam	Denny	M	26780	Sales Rep. II	AU	20-Sep-92	8/1/2010

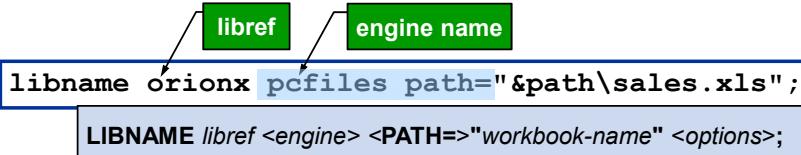
6

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.



When the bit counts differ, use the PC Files Server.



7

SAS/ACCESS provides data connectivity and integration between SAS and third-party data sources, including Microsoft Excel workbooks and various databases. SAS/ACCESS uses data access engines to read, write, and update data regardless of the data source or platform.

Both SAS and Microsoft Office offer 32-bit and 64-bit versions. Different SAS/ACCESS engines are needed based on the products’ “bitness.” If the “bitness” of both products is the same, use the default SAS/ACCESS Excel engine. If the “bitness” differs, use the PC Files Server engine and specify PATH= in front of *workbook-name*.

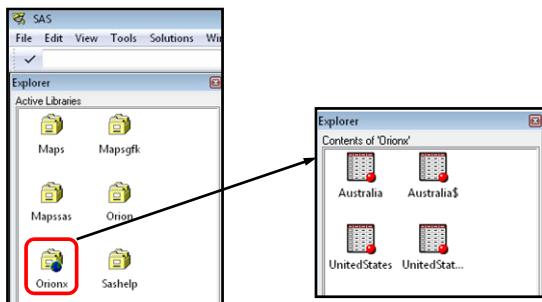
The table below summarizes the possible bit combinations and the appropriate engine to use for each.

SAS	Microsoft Office	SAS/ACCESS Engine
9.4	32-bit	PC Files Server
9.4	64-bit	Default
9.3 32-bit	32-bit	Default
9.3 32-bit	64-bit	PC Files Server
9.3 64-bit	32-bit	PC Files Server
9.3 64-bit	64-bit	Default
SAS 9.2 or earlier	32-bit	Default
SAS 9.2 or earlier	64-bit	PC Files Server

For more information, see the usage note “Installing SAS® 9.3 PC Files Server and using it to convert 32-bit Microsoft Office files to SAS® 64-bit files”: <http://support.sas.com/kb/43/802.html>.

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.

8

The named ranges might or might not exist, depending on how the Excel worksheets were created. They are included here to show the difference between a named range and a worksheet name.

CONTENTS Procedure

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

The CONTENTS Procedure

Directory

Libref	ORIONX
Engine	PCFILES
Physical Name	s:\workshop\sales.xls
Schema/Owner	.

#	Name	Member Type	DBMS	Member Type
1	Australia	DATA	TABLE	
2	Australia\$	DATA	SYSTEM	TABLE
3	UnitedStates	DATA	TABLE	
4	UnitedStates\$	DATA	SYSTEM	TABLE

9

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

10

The column headings are used to create variable names. In the SAS windowing environment, embedded spaces in column names are replaced with underscores. In SAS Enterprise Guide, the column headings are used without modification because special characters are allowed in variable names. Set the VALIDVARNAME=V7 option in SAS Enterprise Guide to cause Enterprise Guide to behave the same as the windowing environment.

Some fields are missing in the PROC CONTENTS output because Excel metadata is incomplete.

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

11

SAS Name Literals

A SAS *name literal* is a string within quotation marks, followed by the letter n.

```
orionx. 'Australia$'n
```

The diagram shows the string 'orionx. 'Australia\$'n' enclosed in a blue rectangular box. A bracket is placed under the entire string, with the label 'SAS name literal' in a green box below it.

SAS name literals permit special characters in data set names.

12

Printing an Excel Worksheet

```
libname orionx pcfiles path=&path\sales.xls;
proc print data=orionx. 'Australia$'n;
run;
```

Partial PROC PRINT Output

Employee_Obs	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

13

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;
```

14

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

15

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.

16

p107d01

7.01 Quiz

Which PROC PRINT step displays the worksheet containing employees from the United States?

- `proc print data=orionx.'UnitedStates';
run;`
- `proc print data=orionx.'UnitedStates$';
run;`
- `proc print data=orionx.'UnitedStates'n;
run;`
- `proc print data=orionx.'UnitedStates$n;
run;`

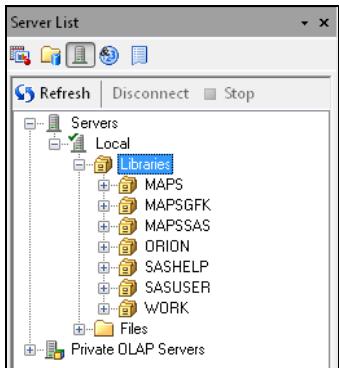
17



Reading Excel Data in SAS Enterprise Guide

p107d01

1. In the Server List window, select **Servers** \Rightarrow **Local** \Rightarrow **Libraries**. If **ORION** is expanded, click the minus sign to collapse it.



2. Open **p107d01** and submit **only** the LIBNAME statement.
3. Check the log for success.

```
16      libname orionx pcfiles path="&path\sales.xls";
NOTE: Libref ORIONX was successfully assigned as follows:
      Engine:      PCFILES
      Physical Name: s:\workshop\sales.xls
```

4. In the Server List window, select **Libraries** and click **Refresh** to refresh the list. Verify that **ORIONX** is displayed as an active library. Expand **ORIONX** to see its contents. The named ranges might not exist.

5. In the Server List window, right-click **Australia\$** and select **Properties**. Then click the **Column** tab to see the variable information. Notice that the variable names contain embedded blanks. SAS Enterprise Guide allows special characters in variable names without the need for a name literal.



Close the Properties window.

- Return to the Program window and submit the PROC CONTENTS step to see the same information in a report. Again, notice that there are embedded blanks in the variable names.

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 VALIDVARNAME= option instructs SAS Enterprise Guide to follow the same rules as the SAS windowing environment in regard to variable names. To set this option, return to the Program window, and uncomment and submit the OPTIONS statement.

```
options validvarname=v7;
```

- Resubmit the PROC CONTENTS step and observe that the embedded blanks have been replaced with underscores.

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

- In the Server List window, double-click **Australia\$** to see the data displayed in a data grid.

Australia\$										
Filter and Sort Query Builder Data ▾ Describe ▾ Graph ▾ Analyze ▾ Export ▾ Send To ▾ Help										
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	

Close the data grid.

10. Return to the Program window and submit the PROC PRINT step to see the same information in a report.

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

11. Return to the Program window and submit the final LIBNAME statement to clear the libref, releasing the spreadsheet. Check the log or the Server List window for success. You might need to refresh the Server List window.

-  To restore the default variable name behavior, submit another OPTIONS statement to set VALIDVARNAME=ANY.



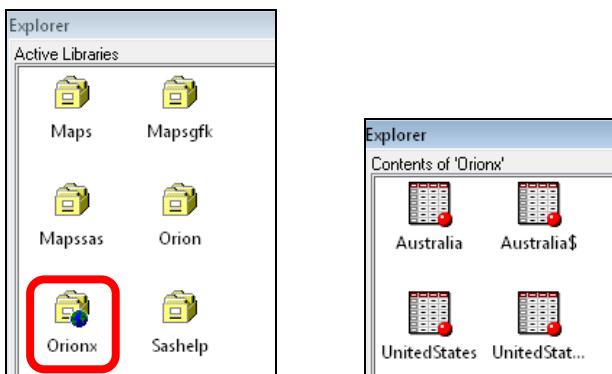
Reading Excel Data in the SAS Windowing Environment

p107d01

1. Activate the Explorer window and double-click **Libraries** to display the active libraries.
2. Open **p107d01** and submit **only** the LIBNAME statement.
3. Check the log for success.

```
64 libname orionx pcfiles path="&path\sales.xls";
NOTE: Libref ORIONX was successfully assigned as follows:
      Engine:      PCFILES
      Physical Name: s:\workshop\sales.xls
```

4. Use the Explorer window to verify that **orionx** is active. Double-click **orionx** to drill into it to see its contents. The named ranges might not exist.



5. In the Explorer window, right-click **Australia\$** and select **View Columns** to see the variable information. Notice that the variable names contain underscores in place of embedded blanks, and the Label column contains the original spreadsheet column headings.

General Details Columns Indexes Integrity					
Find column name: <input type="text"/>					
Column Name	Type	Length	Format	Informat	Label
Employee_ID	Num...	8			Employee ID
First_Name	Text	10	\$10.	\$10.	First Name
Last_Name	Text	12	\$12.	\$12.	Last Name
Gender	Text	1	\$1.	\$1.	Gender
Salary	Num...	8			Salary
Job_Title	Text	14	\$14.	\$14.	Job Title
Country	Text	2	\$2.	\$2.	Country
Birth_Date	Num...	8	DATE9.	DATE9.	Birth Date
Hire_Date	Num...	8	DATE9.	DATE9.	Hire Date

Close the Properties window.

6. Return to the Editor and submit the PROC CONTENTS step to see the same information in a report.

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	12	\$12.	\$12.	First Name
4	Gender	Char	1	\$1.	\$1.	Gender
9	Hire_Date	Num	8	DATE9.	DATE9.	Hire Date
6	Job_Title	Char	20	\$20.	\$20.	Job Title
3	Last_Name	Char	18	\$18.	\$18.	Last Name
5	Salary	Num	8			Salary

In the Explorer window, double-click **Australia\$** to see the data portion. The labels are displayed as column headings. Close the table.

	Employee ID	First Name	Last Name	Gender	Salary	Job Title	Country	Birth Date
1	120102 Tom	Zhou	M		108255	Sales Manager	AU	11AUG1973
2	120103 Wilson	Dawes	M		87975	Sales Manager	AU	22JAN1953
3	120121 Irene	Elvish	F		26600	Sales Rep. II	AU	02AUG1948
4	120122 Christina	Ngan	F		27475	Sales Rep. II	AU	27JUL1958
5	120123 Kimiko	Hotstone	F		26190	Sales Rep. I	AU	28SEP1968

7. Return to the Editor and submit the PROC PRINT step to see the same information in a report.

ID	Last_Name	Job_Title	Employee_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

8. Submit the final LIBNAME statement to clear the libref, releasing the spreadsheet. Check the log or Explorer window for success.

Business Scenario

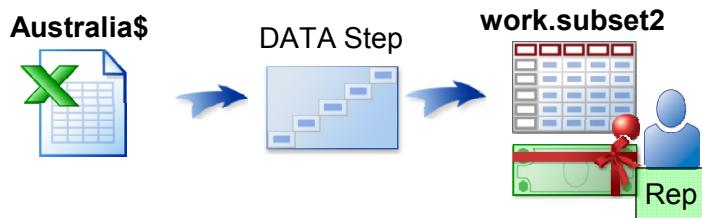
Create a SAS data set using a Microsoft Excel workbook as input.



20

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

21

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

22



Creating a SAS Data Set

p107d02

 If you are using SAS Enterprise Guide, remember to uncomment and submit the OPTIONS statement to set VALIDVARNAME=V7.

```
libname orionxls pcfiles path="&path\sales.xls";
*options validvarname=v7; /* Needed for SAS Enterprise Guide */

data work.subset2;
  set orionxls.'Australia'$'n;
  where Job_Title contains 'Rep';
  Bonus=Salary*.10;
  keep First_Name Last_Name Salary Bonus
        Job_Title Hire_Date;
  label Job_Title='Sales Title'
        Hire_Date='Date Hired';
format Salary comma10. Hire_Date mmddyy10.
           Bonus comma8.2;
run;

proc contents data=work.subset2;
run;

proc print data=work.subset2 label;
run;

libname orionxls clear;
```

1. Open **p107d02** and submit the LIBNAME statement.
2. Use the Server List window or SAS Explorer to explore the active libraries and drill into **orionxls**. Observe that the two spreadsheets, **Australia\$** and **UnitedStates\$**, are displayed as if they were SAS data sets.
3. Submit the DATA step.
4. Check the log. Verify that 61 observations were written to **work_subset2**.

If you are using SAS Enterprise Guide, the following error occurs if the VALIDVARNAME=V7 option was not set:

```
ERROR: Variable Job_Title is not on file ORIONXLS.'Australia$'n.
```

Job_Title is not found in **Australia\$**. **Job Title** is found instead, because SAS Enterprise Guide permits blanks and special characters in variable names. Setting the VALIDVARNAME=v7 option causes SAS Enterprise Guide to use the same variable naming rules as the SAS windowing environment. Be sure to set the option as explained in the note above.

5. Run the PROC CONTENTS step. Verify that the formats and labels were stored in the descriptor portion of **work_subset2**. Notice that all columns have labels, not just the columns that were assigned labels in the DATA step.

Alphabetic List of Variables and Attributes						
#	Variable	Type	Len	Format	Informat	Label
6	Bonus	Num	8	COMMA8.2		
1	First_Name	Char	10	\$10.	\$10.	First Name
5	Hire_Date	Num	8	MMDDYY10.	DATE9.	Date Hired
4	Job_Title	Char	14	\$14.	\$14.	Sales Title
2	Last_Name	Char	12	\$12.	\$12.	Last Name
3	Salary	Num	8	COMMA10.		Salary

6. Submit the PROC PRINT step and verify that the results contain 61 observations and that the labels were displayed and formats applied.

First Name	Last Name	Salary	Sales Title	Date Hired	Bonus
Irenie	Elvish	26,600	Sales Rep. II	01/01/1978	2,660.00
Christina	Ngan	27,475	Sales Rep. II	07/01/1982	2,747.50
Kimiko	Hotstone	26,190	Sales Rep. I	10/01/1989	2,619.00
...					
Alban	Kingston	28,830	Sales Rep. III	10/01/1996	2,883.00
Alena	Moody	26,205	Sales Rep. II	09/01/2010	2,620.50

7. Submit the LIBNAME statement to clear the resources.
8. Check the log for success.



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. Accessing an Excel Worksheet

- Retrieve the starter program **p107e01**.
- Add a LIBNAME statement before the PROC CONTENTS step to create a libref named **CUSTFM** that references the Excel workbook, **custfm.xls**.

 If you are using SAS Enterprise Guide, you need to set the VALIDVARNAME=V7 option if not already set in the session.
- Submit the LIBNAME statement and the PROC CONTENTS step to create the following partial PROC CONTENTS report:

Part 1 of 3

The CONTENTS Procedure			
Directory			
Libref	CUSTFM		
Engine	EXCEL		
Physical Name	custfm.xls		
User	Admin		
 DBMS			
Member Member			
#	Name	Type	Type
1	Females\$	DATA	TABLE
2	Males\$	DATA	TABLE

- Add a SET statement in the DATA step to read the worksheet containing the male data.
- Add a KEEP statement in the DATA step to include only the **First_Name**, **Last_Name**, and **Birth_Date** variables in the new data set.
- Add a FORMAT statement in the DATA step to display **Birth_Date** as a four-digit year.
- Add a LABEL statement to change the column heading of **Birth_Date** to **Birth Year**.
- Submit the program including the final LIBNAME statement and create the report below. Results should contain 47 observations.
- Verify that the worksheet was released.

Partial PROC PRINT Output

Obs	First Name	Last Name	Birth Year
1	James	Kvarniq	1974
2	David	Black	1969
3	Markus	Sepke	1988
4	Ulrich	Heyde	1939
5	Jimmie	Evans	1954

Level 2

2. Accessing an Excel Worksheet

- Write a LIBNAME statement to create a libref named **PROD** that references the Excel workbook **products.xls**.
- Write a PROC CONTENTS step to view all of the contents of **PROD**.
- Submit the program to determine the names of the four worksheets in **products.xls**.
- Write a DATA step that reads the worksheet containing sports data and creates a new data set named **work.golf**.

The data set **work.golf** should

- include only the observations where **Category** is equal to *Golf*
 - not include the **Category** variable
 - include a label of **Golf Products** for the **Name** variable.
- Write a LIBNAME statement to clear the **PROD** libref.
 - Create the report below. The results should contain 56 observations.

Partial PROC PRINT Output

Obs	Golf Products
1	Ball Bag
2	Red/White/Black Staff 9 Bag
3	Tee Holder
4	Bb Softspikes - Xp 22-pack
5	Bretagne Performance Tg Men's Golf Shoes L.

Challenge

3. Creating an Excel Spreadsheet

- Open **p107e03**. Insert a LIBNAME statement to associate the libref **out** with the Excel workbook **employees.xls** in the default data folder.
- Modify the program so that it creates a spreadsheet named **salesemps** in the **employees.xls** workbook.
- Submit the SAS program and verify that it created the data set **out.salesemps** with 71 observations and 4 variables as shown in the partial SAS log below.

NOTE: 71 records were read from the infile "s:\workshop\newemps.csv".

The minimum record length was 28.

The maximum record length was 47.

NOTE: The data set OUT.salesemps has 71 observations and 4 variables.



The program will fail if the workbook already exists. Use Windows Explorer to navigate to the data folder and delete **employees.xls**.

4. Using the IMPORT and EXPORT Procedures

Use the SAS Help facility or online documentation to investigate the IMPORT and EXPORT procedures. Describe the chief differences from using the SAS/ACCESS LIBNAME statement.

7.2 Reading Database Data

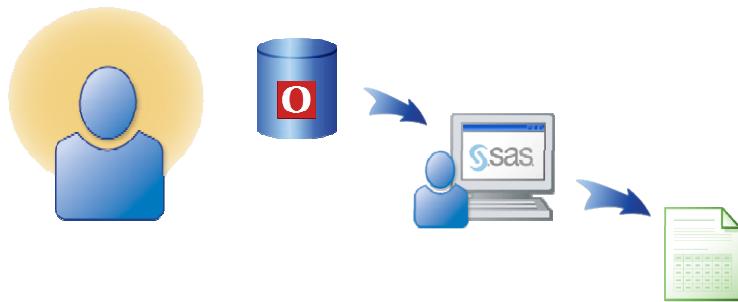
Objectives

- 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.

28

Business Scenario

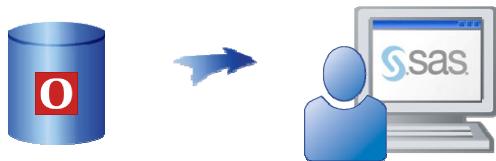
The Northeast Sales Manager requested a report listing supervisors from New York and New Jersey. The input data is in an Oracle database.



29

Business Scenario

Use SAS/ACCESS to read the tables within the database as if they were SAS data sets.



30

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;
  
```

LINBNAME libref engine <SAS/ACCESS options>;

This example uses the LIBNAME statement supported by the SAS/ACCESS Interface to Oracle.

31

The engine name, such as Oracle or DB2, is the SAS/ACCESS component that reads and writes to your DBMS. The engine name is required.

USER= specifies an optional Oracle user name. USER= must be used with PASSWORD=.

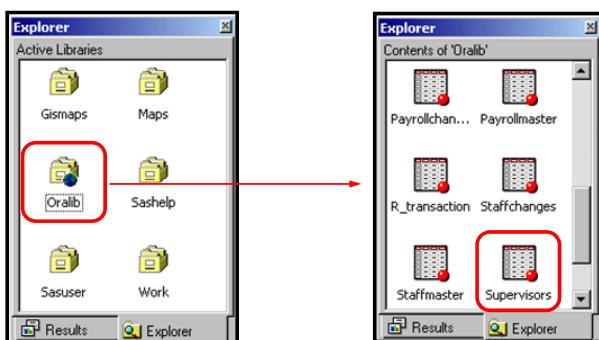
PASSWORD= (or PW=) specifies an optional Oracle password associated with the Oracle user name.

PATH= specifies the Oracle driver, node, and database. SAS/ACCESS uses the same Oracle path designation that you use to connect to Oracle directly.

SCHEMA= enables you to read database objects, such as tables and views, in the specified schema. If this option is omitted, you connect to the default schema for your DBMS.

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.

32

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';

33

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.

35

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

36

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;

```

37

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

38

7.3 Solutions

Solutions to Exercises

1. Accessing an Excel Worksheet

```

libname custfm pcfiles path="&PATH\custfm.xls";
*options validvarname=v7; /* needed for SAS Enterprise Guide */

```

```

proc contents data=custfm._all_;
run;

data work.males;
  set custfm.'Males$\n';
  keep First_Name Last_Name Birth_Date;
  format Birth_Date year4.;
  label Birth_Date='Birth Year';
run;

proc print data=work.males label;
run;

libname custfm clear;

```

2. Accessing an Excel Worksheet

```

libname prod pcfiles path="&PATH\products.xls";
*options validvarname=v7; /* SAS Enterprise Guide only */

proc contents data=prod._all_;
run;

data work.golf;
  set prod.'Sports$\n';
  where Category='Golf';
  drop Category;
  label Name='Golf Products';
run;

libname prod clear;

proc print data=work.golf label;
run;

```

3. Creating an Excel Spreadsheet

```

/* delete employees.xls from the data directory if it exists */

libname out pcfiles path="&path\employees.xls";
*options validvarname=v7; /* SAS Enterprise Guide only */

data out.salesemps;
  length First_Name $ 12 Last_Name $ 18
        Job_Title $ 25;
  infile "&path\newemps.csv" dlm=',';
  input First_Name $ Last_Name $
        Job_Title $ Salary;
run;
libname out clear;

```

4. Using the IMPORT and EXPORT Procedures

Explore the IMPORT and EXPORT procedures. Describe the chief differences from using the SAS/ACCESS LIBNAME statement.

When you use the SAS/ACCESS LIBNAME statement, you are

- accessing the most recent data in the workbook or database.
- not necessarily making a SAS copy of the data. You can use PROC PRINT (or other procedures) directly on a sheet or table through a dynamic "pipeline."

When you import, you are

- duplicating storage, in a sense
- creating a static SAS copy of the data, which might get out of date unless you remember to re-import
- allowing for easier programming (for example, no name literals).

Solutions to Student Activities (Polls/Quizzes)

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;`

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

23

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';

34

Chapter 8 Reading Raw Data Files

8.1	Introduction to Reading Raw Data Files.....	8-3
8.2	Reading Standard Delimited Data.....	8-7
	Demonstration: Examining Data Errors	8-33
	Exercises	8-34
8.3	Reading Nonstandard Delimited Data	8-37
	Demonstration: Using List Input: Importance of Colon Format Modifier.....	8-48
	Exercises	8-51
8.4	Handling Missing Data	8-54
	Exercises	8-60
8.5	Solutions	8-63
	Solutions to Exercises	8-63
	Solutions to Student Activities (Polls/Quizzes).....	8-65

8.1 Introduction to Reading Raw Data Files

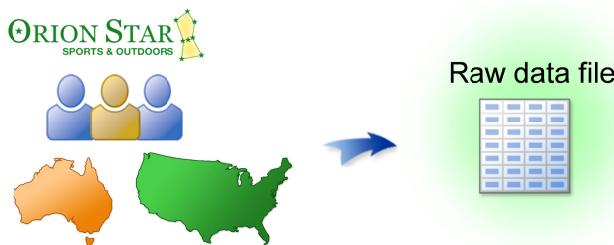
Objectives

- Identify types of raw data files and input styles.
- Define the terms standard and nonstandard data.

3

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.

4

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.

5

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	---
120102	Tom	Zhou		Sales Manager			108255AU						
120103	Wilson	Dawes		Sales Manager			87975AU						
120121	Irenie	Elvish		Sales Rep. II			26600AU						
120122	Christina	Ngan		Sales Rep. II			27475AU						

6

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

7

Reading Raw Data Files

There are different techniques, or *input styles*, for reading raw data files in SAS.

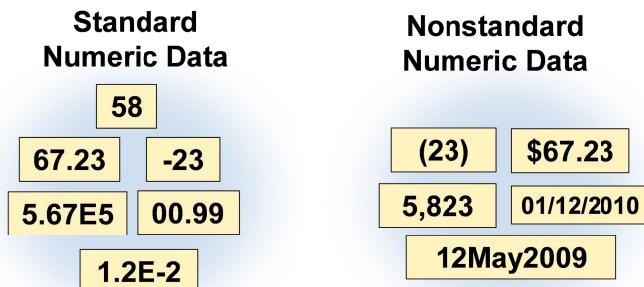
Input Style	Used for Reading
Column Input	Standard data in fixed columns
Formatted Input	Standard and nonstandard data in fixed columns
List Input	Standard and nonstandard data separated by blanks or some other delimiter

8

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.



9

8.01 Multiple Answer Poll

What type of raw data files do you read?

- delimited
- fixed column
- both delimited and fixed column
- I do not read raw data files.

10

8.2 Reading Standard Delimited 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.

12

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.



13

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.

14

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
```

15

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;
```

17

p108d01

DLM= is an alias for DELIMITER=*delimiter(s)*. It is used to specify an alternate delimiter (other than a blank) to be used for list input. *Delimiter(s)* is one or more characters enclosed in quotation marks. If more than one delimiter is specified, each is treated as a delimiter. The delimiter is case sensitive.

- To specify a tab delimiter on Windows or UNIX, type **dlm='09'x**.
- To specify a tab delimiter on z/OS (OS/390), type **dlm='05'x**.

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.

18

A %LET statement was submitted in the **libname.sas** program to create the user-defined macro variable, **path**. The INFILE statement references the value of the macro variable by placing an ampersand before its name. The macro variable will exist for the duration of the SAS session unless it is changed or deleted.

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.

19

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.
```

20

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

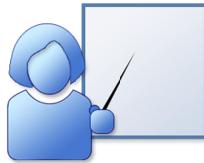
Some character values are truncated.

21

p108d01

Business Scenario

It is important to understand the processing that occurs when a DATA step reads a raw data file.

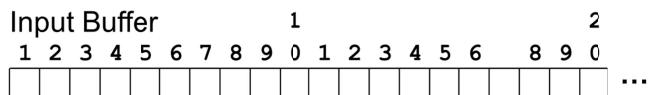


22

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

23

Compilation

```
data work.subset;
  infile "&path\sales.csv" dlm=',';
  input Employee_ID First_Name $ Last_Name $ 
    Gender $ Salary Job_Title $ Country $;
run;
```

24

p108d01

...

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

25

...

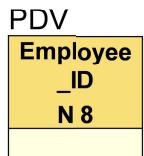
The default length of the input buffer depends on the operating system. It can be modified using the LRECL= option in the INFILE statement.

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



Attributes are based on the INPUT statement.

26

...

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	First_Name
N 8	\$ 8

With list input, the default length for character variables is eight bytes.

27

...

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	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8

28

...

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

29

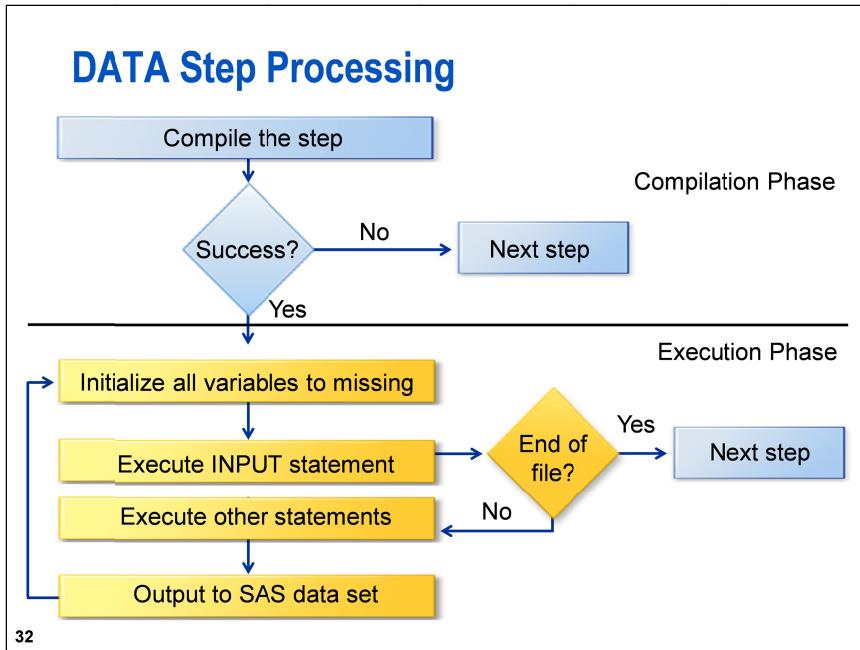
...

8.03 Multiple Choice Poll

Which statement is true?

- An input buffer is created only if you are reading data from a raw data file.
- The PDV at compile time holds the variable name, type, byte size, and initial value.
- The descriptor portion is the first item that is created at compile time.

30



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;
```

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	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
.				.		
...						

34

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, ...
```

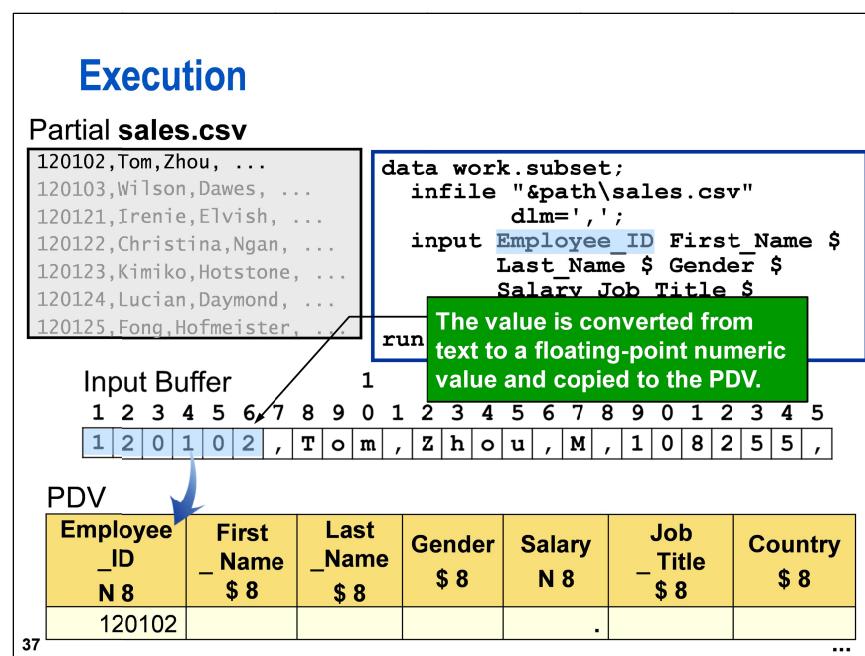
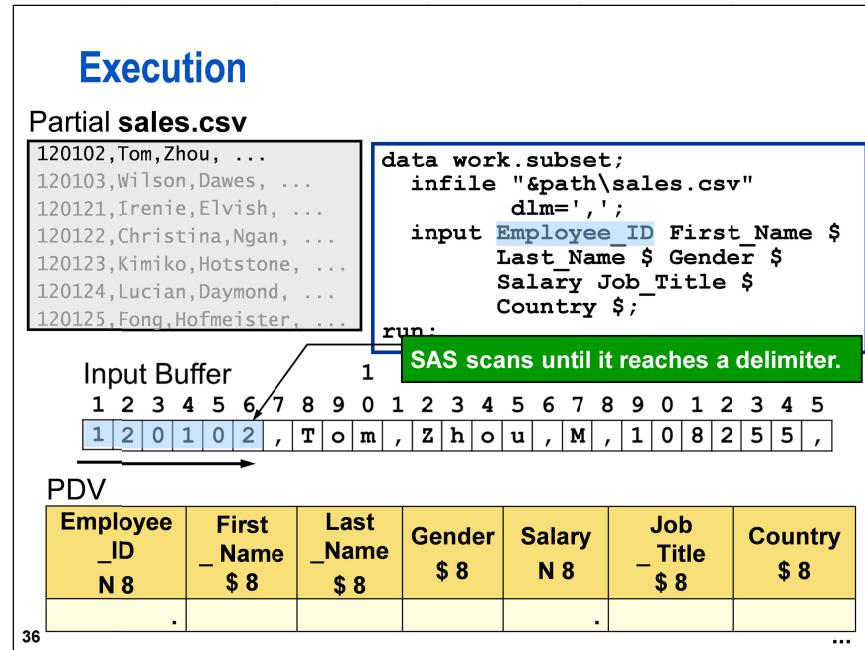
```
data work_subset;
  infile "&path\sales.csv"
    dlm=',';
  input Employee_ID First_Name $
    Last_Name $ Gender $
    Salary Job_Title $
    Country $;
run;
```

Input Buffer 1 SAS reads a record into the 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	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
.				.		
...						

35



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										
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
120102				.		
...						

38

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										
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
120102	Tom			.		
...						

39

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	2	,	T	o	m	,	z	h	o	u	,	M	,	1	0	8	2	5	5	,

These actions continue for all variables in the INPUT statement.

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

40

There is usually no delimiter after the last field in a record, so SAS stops reading when it encounters an end-of-record marker.

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	First_Name	Last_Name	Gender	Salary	Job_Title	Country
N 8	\$ 8	\$ 8	\$ 8	N 8	\$ 8	\$ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

41

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

42

...

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;

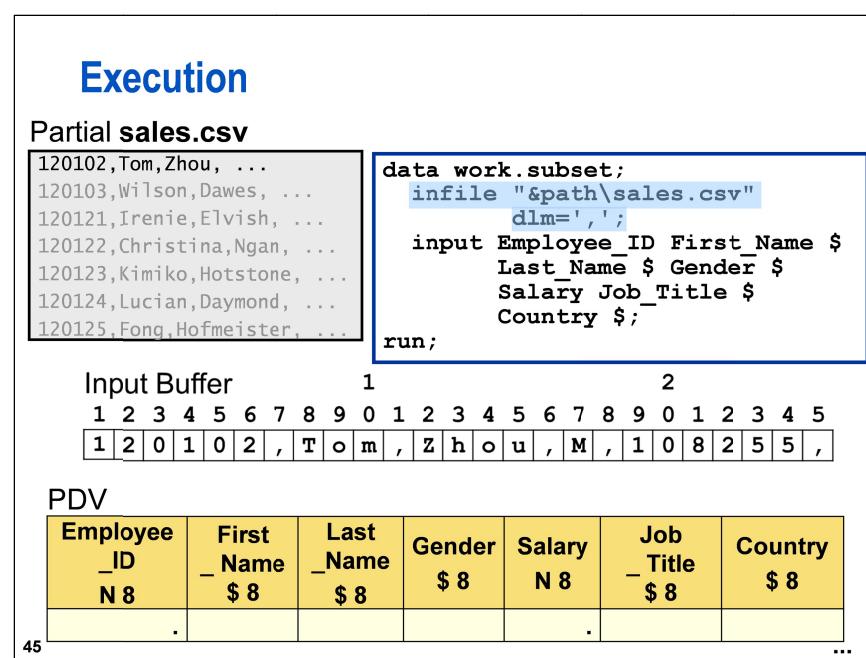
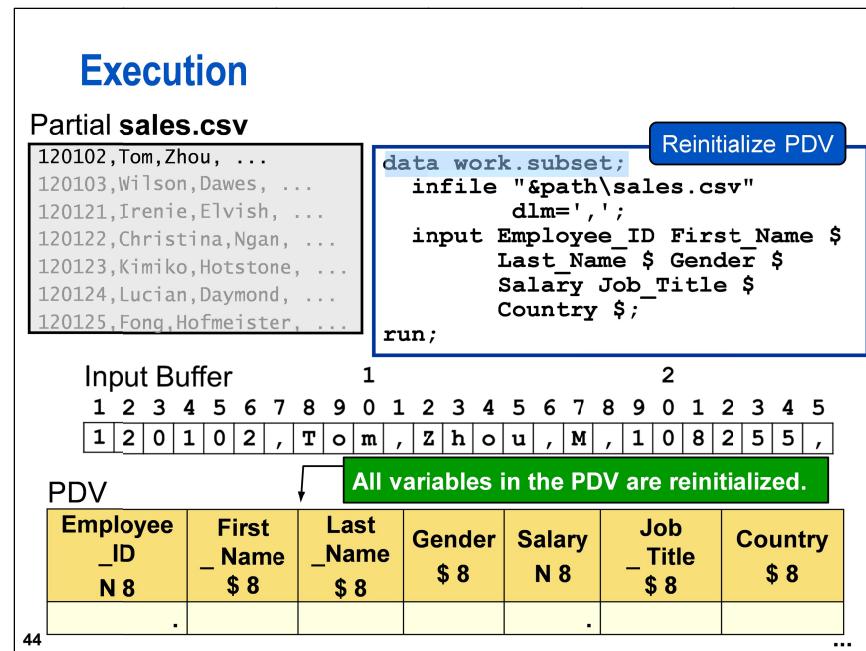
Input Buffer 1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 1 | 2 | 0 | 1 | 0 | 2 | , T o m , z h o u , M , 1 0 8 | 2 | 5 | 5 |,

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

43

...



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
.				.		
...						

46

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
120103	Wilson	Dawes	M	87975	Sales Ma	AU
...						

47

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

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	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	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
...						

48

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"
    Continue until EOF
  input Employee_ID First_Name $
    Last_Name $ Gender $
    Salary Job_Title $
    Country $;
run;
```

Input Buffer 1

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	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
.				.		

49

8.04 Multiple Choice Poll

Which statement is true of a DATA step when reading from a raw data file?

- Data is read from the raw data file into the PDV.
- The size of the input buffer adjusts automatically based on the length of the input record.
- At the bottom of the DATA step, the contents of the PDV are output to the output SAS data set.

50

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	Davmond	M	26480	Sales Re	AU
120125	Fong	Hofmeist	M	32040	Sales Re	AU

Some character values are truncated.

52

p108d01

Some character values contain unnecessary trailing blanks, although this is not obvious from a PROC PRINT report.

LENGTH Statement

The *LENGTH statement* defines the type and length of a variable.

```
data work.subset;
  LENGTH variable(s) $ length;
  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.

53

p108d02

The LENGTH statement is used primarily for character variables.

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

Attributes are based on the LENGTH statement.

First_Name \$ 12	Last_Name \$ 18	Gender \$ 1	Job_Title \$ 25	Country \$ 2

54

...

The name, type, and length of a variable are determined at the variable's first use. These specifications can be in a LENGTH statement or the INPUT statement, whichever appears first in the DATA step. The name is used exactly as specified at first use, including the case.

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



55

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.

56

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;
```

57

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;
```

59

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

60

Viewing the Output

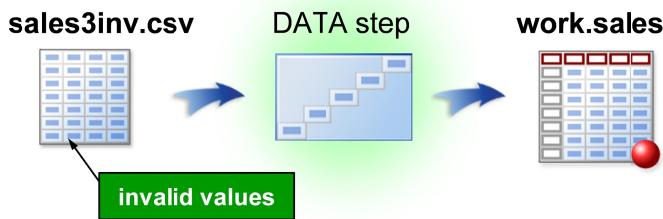
Partial PROC PRINT Output

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

61

Business Scenario

A raw data file contains information about Orion Star sales employees. It includes some invalid data values.



63

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
```

64

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.

66

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.

67

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,
      Line Length=256, Encoding=1252, EOL=CR, EOB=EOF.

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.

68

Even though these are referred to as *data errors*, they generate notes, not error messages. Syntax errors stop the DATA step, whereas data errors allow processing to continue.

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.

69

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
```

70

`_ERROR_` is reset to 0, and `_N_` is incremented by 1 at the beginning of each iteration.



Examining Data Errors

p108d04

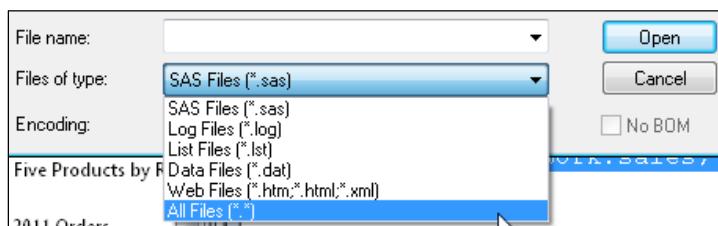
```
data work.sales;
  infile "&path\sales3inv.csv" dlm=',';
  input Employee_ID First $ Last $ 
        Job_Title $ Salary   Country $;
run;

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

1. Open the raw data file `sales3inv.csv`.

In SAS Enterprise Guide, select **File** \Rightarrow **Open** \Rightarrow **Data**. Navigate to your data folder.

In the SAS windowing environment, select **File** \Rightarrow **Open Program**. Select **All Files (*.*)** from the File of type menu.



Select `sales3inv.csv` from the list of files.

2. View the **Salary** values. There are three invalid values and one missing value.

3. Open and submit **p108d04**.
4. Examine the log. The DATA step creates an output data set, **work.sales**, with 50 observations. The three invalid **Salary** values were identified in the log, and the corresponding observations have a missing value for **Salary**. The missing **Salary** value was not reported as a data error because missing is a valid value in SAS.
5. Examine the output. Notice that there are four observations with missing **Salary** values.

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.

72

Use the SAS system option **ERRORS=n** to specify the maximum number of observations for which error messages about data input errors are printed.

```
options errors=5;
```



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. Reading a Comma-Delimited Raw Data File

- a. Open **p108e01**. Add the appropriate LENGTH, INFILE, and INPUT statements to read the comma-delimited raw data file named the following:

Windows	"&path\newemps.csv"
UNIX	"&path/newemps.csv"
z/OS (OS/390)	"&path..rawdata(newemps)"

Partial Raw Data File

```
Satyakam,Denny,Sales Rep. II,26780
Monica,Kletschkus,Sales Rep. IV,30890
Kevin,Lyon,Sales Rep. I,26955
Petrea,Soltau,Sales Rep. II,27440
Marina,Iyengar,Sales Rep. III,29715
```

- b. Read the following fields from the raw data file:

Name	Type	Length
First	Character	12
Last	Character	18
Title	Character	25
Salary	Numeric	8

- c. Submit the program to create the report below. The results should contain 71 observations.

Partial PROC PRINT Output

Obs	First	Last	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

Level 2**2. Reading a Space-Delimited Raw Data File**

- a. Write a DATA step to create a new data set named **work.qtrdonation**, reading the space-delimited raw data file named the following:

Windows	"&path\donation.dat"
UNIX	"&path/donation.dat"
z/OS (OS/390)	"&path..rawdata(donation)"

Partial Raw Data File

```
120265 . . . 25
120267 15 15 15 15
120269 20 20 20 20
120270 20 10 5 .
120271 20 20 20 20
```

- b. Read the following fields from the raw data file:

Name	Type	Length
IDNum	Character	6
Qtr1	Numeric	8
Qtr2	Numeric	8
Qtr3	Numeric	8
Qtr4	Numeric	8

- c. Write a PROC PRINT step to create the report below. The results contain 124 observations.

Partial PROC PRINT Output

Obs	IDNum	Qtr1	Qtr2	Qtr3	Qtr4
1	120265	.	.	.	25
2	120267	15	15	15	15
3	120269	20	20	20	20
4	120270	20	10	5	.
5	120271	20	20	20	20

Challenge

3. Reading a Tab-Delimited Raw Data File

- a. Create a temporary data set, **managers2**, using the tab-delimited raw data file named the following:

Windows	"&path\managers2.dat"
UNIX	"&path/ managers2.dat"
z/OS (OS/390)	"&path..rawdata(managers2)"

Raw Data File

120102	Tom	Zhou	M	108255	Sales Manager
120103	Wilson	Dawes	M	87975	Sales Manager
120261	Harry	Highpoint	M	243190	Chief Sales Officer
121143	Louis	Favaron	M	95090	Senior Sales Manager
121144	Renee	Capachietti	F	83505	Sales Manager
121145	Dennis	Lansberry	M	84260	Sales Manager

- b. Read the following fields from the raw data file:

Name	Type
ID	Numeric
First	Character
Last	Character

Name	Type
Gender	Character
Salary	Numeric
Title	Character

- c. The new data set should contain only **First**, **Last**, and **Title**.
- d. Generate the report below. The results should contain six observations.

Obs	First	Last	Title
1	Tom	Zhou	Sales Manager
2	Wilson	Dawes	Sales Manager
3	Harry	Highpoint	Chief Sales Officer
4	Louis	Favaron	Senior Sales Manager
5	Renee	Capachietti	Sales Manager
6	Dennis	Lansberry	Sales Manager

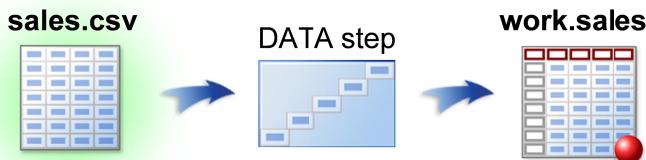
8.3 Reading Nonstandard Delimited Data

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.

78

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
```

79

Modified List Input

This DATA step uses *modified list input*. Instead of a LENGTH statement, an informat specifies the length for each character variable.

```
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.

80

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

81

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.

82

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

83

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.

85

SAS Informats

SAS informats have the following form:

`<$><informat><w>.`

\$	Indicates a character informat.
<i>informat</i>	Names the SAS informat or user-defined informat.
<i>w</i>	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.

86

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.
COMMAM. 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.

87

SAS Informats

Informats are used to read and convert raw data.

Informat	Raw Data Value	SAS Data Value
COMMA. DOLLAR.	\$12,345	12345
COMMAM. DOLLARX.	\$12.345	12345
EUROX.	€12.345	12345
\$CHAR.	##Australia	##Australia
\$UPCASE.	au	AU

 The character # represents a blank space.

88

SAS Informats

Use date informats to read and convert dates to SAS date values.

Informat	Raw Data Value	SAS Data Value
MMDDYY.	010160	0
	01/01/60	
	01/01/1960	
	1/1/1960	
DDMMYY.	311260	365
	31/12/60	
	31/12/1960	
DATE.	31DEC59	-1
	31DEC1959	

89

8.09 Quiz

Use the SAS Help facility or documentation to investigate the **DATEw.** informat and answer the following questions:

- a) What does the **w** represent?
- b) What is the default width of this informat?

90

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,AP,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.

92

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

93

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.

```

94

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

95

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.

96

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;
```

98

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

99

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

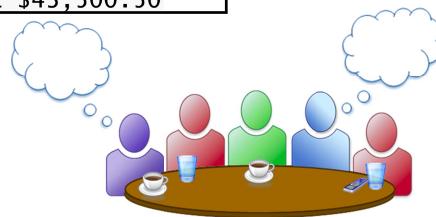
100

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
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	



101



Using List Input: Importance of Colon Format Modifier

p108a02

- Open **p108a02** and examine the INPUT statement.
 - The INFILE statement does not contain DLM= because the file is space delimited.
 - HireDate** and **Salary** are nonstandard numeric fields, so an informat is needed.
- In the SAS windowing environment, select **File** \Rightarrow **Open Program**, change the value for **Files of type** to **Data Files (*.dat)**, and select **salary.dat**.
 - Files with a .dat extension can be imported in SAS Enterprise Guide but not viewed in an editor.

Partial salary.dat

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

- Submit part 1 and view the log and output. The expected output is shown below.

```
/* Part 1 - using colon format modifiers*/
data work.salaries;
  infile "&path\salary.dat";
  input Name $ HireDate :date. Age State $ Salary :comma. ;
run;

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

Partial PROC PRINT Output

Obs	Name	Date	Age	State	Hire Salary
1	Donny	17657	25	FL	43132.5
2	Margaret	17582	43	NC	65150.0
3	Dan	17684	27	FL	40000.0
4	Subash	17564	45	NC	75750.0
5	Antonio	17677	35	FL	43500.5

4. Now let's see what happens when a colon format modifier is omitted from **Salary**. Submit part 2.

```
/* Part 2 - omit the colon format modifier for Salary */
data work.salaries;
  infile "&path\salary.dat";
  input Name $ HireDate :date. Age State $ Salary comma. ;
run;

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

5. Examine the log. There are no errors or warnings.

```
923  /* Part 2 - omit the colon format modifier for Salary */
924  data work.salaries;
925    infile "&path\salary.dat";
926    input Name $ HireDate :date. Age State $ Salary comma. ;
927  run;

NOTE: The infile "s:\workshop\salary.dat" is:
      Filename=s:\workshop\salary.dat,
NOTE: 8 records were read from the infile "s:\workshop\salary.dat".
NOTE: The data set WORK.SALARIES has 8 observations and 5 variables.
```

6. Examine the output.

Obs	Name	Date	Age	State	Salary
1	Donny	17657	25	FL	.
2	Margaret	17582	43	NC	6
3	Dan	17684	27	FL	.
4	Subash	17564	45	NC	7
5	Antonio	17677	35	FL	.

The **Salary** values are incorrect. Why are the values either missing or only one digit in length?

- The **comma. informat** has a default width of 1, so SAS reads 1 column from the input file. This reads the first character of the **Salary** value.
- When the first column contains a dollar sign, a missing value is assigned to the numeric variable. This does not cause a data error because the **comma informat** removes non-numeric characters, including the dollar sign. When the first column contains a digit, that digit becomes the value of the variable.



In some cases, omitting a colon results in missing or invalid values. In other cases, it results in data errors.

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.



103

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

104

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	Bamberg	Manager	32000

105

p108d08



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

4. Reading Nonstandard Data from a Comma-Delimited Raw Data File

- a. Open **p108e04**. Add the appropriate LENGTH, INFILE, and INPUT statements to read the comma-delimited raw data file named the following:

Windows	"&path\custca.csv"
UNIX	"&path/custca.csv"
z/OS (OS/390)	"&path..rawdata(custca)"

Partial Raw Data File

```
Bill,Cuddy,11171,M,16/10/1986,21,15-30 years
Susan,Krasowski,17023,F,09/07/1959,48,46-60 years
Andreas,Rennie,26148,M,18/07/1934,73,61-75 years
Lauren,Krasowski,46966,F,24/10/1986,21,15-30 years
Lauren,Marx,54655,F,18/08/1969,38,31-45 years
```

Read the following fields:

Name	Type	Length
First	Character	20
Last	Character	20
ID	Numeric	8
Gender	Character	1
BirthDate	Numeric	8
Age	Numeric	8
AgeGroup	Character	12

- b. Use FORMAT and DROP statements in the DATA step to create a data set that results in the report below when displayed with a PROC PRINT step. Include an appropriate title. The results should contain 15 observations.

Partial PROC PRINT Output

Canadian Customers					
Obs	First	Last	Gender	AgeGroup	Birth Date
1	Bill	Cuddy	M	15-30 years	OCT1986
2	Susan	Krasowski	F	46-60 years	JUL1959
3	Andreas	Rennie	M	61-75 years	JUL1934
4	Lauren	Krasowski	F	15-30 years	OCT1986
5	Lauren	Marx	F	31-45 years	AUG1969

Level 2

5. Reading a Delimited Raw Data File with Nonstandard Data Values

- a. Write a DATA step to create a temporary data set, **prices**, reading the delimited raw data file named the following:

Windows	"&path\pricing.dat"
UNIX	"&path/ pricing.dat"
z/OS (OS/390)	"&path..rawdata(pricing)"

All data fields are numeric.

Partial Raw Data File

210200100009*09JUN2011*31DEC9999*\$15.50*\$34.70
210200100017*24JAN2011*31DEC9999*\$17.80*22.80
210200200023*04JUL2011*31DEC9999*\$8.25*\$19.80
210200600067*27OCT2011*31DEC9999*\$28.90*47.00
210200600085*28AUG2011*31DEC9999*\$17.85*\$39.40

- b. Generate the report below. The results should contain 16 observations.

Partial PROC PRINT Output

2011 Pricing					
Obs	ProductID	StartDate	EndDate	Cost	Sales Price
1	210200100009	06/09/2011	12/31/9999	15.50	34.70
2	210200100017	01/24/2011	12/31/9999	17.80	22.80
3	210200200023	07/04/2011	12/31/9999	8.25	19.80
4	210200600067	10/27/2011	12/31/9999	28.90	47.00
5	210200600085	08/28/2011	12/31/9999	17.85	39.40

Challenge**6. Reading In-Stream Delimited Data**

- a. Open p108e06. Write a DATA step to read the delimited in-stream data shown below.

 An INFILE statement is required. Use SAS Help or online documentation to explore the use of DATALINES as a file specification in an INFILE statement.

```
120102/Tom/Zhou/M/108,255/Sales Manager/01Jun1993
120103/Wilson/Dawes/M/87,975/Sales Manager/01Jan1978
120261/Harry/Highpoint/M/243,190/Chief Sales Officer/01Aug1991
121143/Louis/Favaron/M/95,090/Senior Sales Manager/01Jul2001
121144/Renee/Capachietti/F/83,505/Sales Manager/01Nov1995
121145/Dennis/Lansberry/M/84,260/Sales Manager/01Apr1980
```

- b. Generate the report below:

Orion Star Management Team						
First	Last	Title	ID	Gender	Salary	HireDate
Tom	Zhou	Sales Manager	120102	M	108255	06/01/1993
Wilson	Dawes	Sales Manager	120103	M	87975	01/01/1978
Harry	Highpoint	Chief Sales Officer	120261	M	243190	08/01/1991
Louis	Favaron	Senior Sales Manager	121143	M	95090	07/01/2001
Renee	Capachietti	Sales Manager	121144	F	83505	11/01/1995
Dennis	Lansberry	Sales Manager	121145	M	84260	04/01/1980

8.4 Handling Missing Data

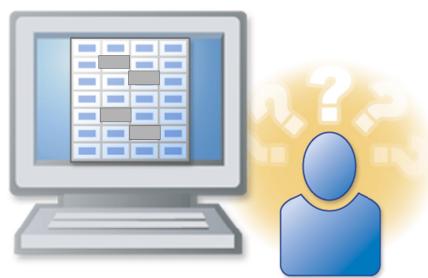
Objectives

- 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.

109

Business Scenario

Orion Star programmers have discovered that some files have records with missing data in one or more fields.



110

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.

phone2.csv										Missing data is indicated by consecutive delimiters.
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 Krah1,	(212)	891-3241,	(212)	233-5413						
Karen Ballinger	,,	(714)	644-9090							
Elke Wallstab,	(910)	763-5561,	(910)	545-3421						

111

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;
```

112

p108a03

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	---
James Kvarniq, (704) 293-8126, (701) 281-8923							
Sandrina Stephano,, (919) 271-4592							
Cornelia Krah, (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

114

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

115

p108d09

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.

116

Viewing the Output

Adding the DSD option gives the correct results.

PROC PRINT Output

Name	Phone	Mobile
James Kvarnig	(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.
```

117

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.



119

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	missing values	4	4
1	---	5	-----0-----0-----5	-----0-----0-----5	
James Kvarniq,(704) 293-8126,(701) 281-8923					
Sandrina Stephano,(919) 871-7830					
Cornelia Krah1,(212) 891-3241,(212) 233-5413					
Karen Ballinger,(714) 344-4321					
Elke Wallstab,(910) 763-5561,(910) 545-3421					

The DSD option is not appropriate because the missing data is not marked by consecutive delimiters.

120

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.

121

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

122

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.

123



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

7. Reading a Comma-Delimited File with Missing Values

- a. Open **p108e07** and insert INFILE and INPUT statements to read the comma-delimited raw data named the following:

Windows	"&path\donation.csv"
UNIX	"&path/donation.csv"
z/OS (OS/390)	"&path..rawdata(donation)"

Partial Raw Data File

```
120265, , , , 25
120267, 15, 15, 15, 15
120269, 20, 20, 20, 20
120270, 20, 10, 5
120271, 20, 20, 20, 20
```

- b. There might be missing data in the middle or at the end of a record. Read the following fields from the raw data file:

Name	Type
Employee ID	Numeric
Quarter 1	Numeric
Quarter 2	Numeric
Quarter 3	Numeric
Quarter 4	Numeric

- c. Write a PROC PRINT step to generate the report below. The results should include 124 observations.

Obs	EmpID	Q1	Q2	Q3	Q4
1	120265	.	.	.	25
2	120267	15	15	15	15
3	120269	20	20	20	20
4	120270	20	10	5	.
5	120271	20	20	20	20

Level 2

8. Reading a Delimited File with Missing Values

- a. Write a DATA step to create a temporary data set, **prices**, using the asterisk-delimited raw data file named the following:

Windows	"&path\prices.dat"
UNIX	"&path/prices.dat"
z/OS (OS/390)	"&path..rawdata(prices)"

Partial Raw Data File

210200100009*09JUN2007*31DEC9999*\$15.50*\$34.70
210200100017*24JAN2007*31DEC9999*\$17.80
210200200023*04JUL2007*31DEC9999*\$8.25*\$19.80
210200600067*27OCT2007*31DEC9999*\$28.90
210200600085*28AUG2007*31DEC9999*\$17.85*\$39.40

There might be missing data at the end of some records. Read the following fields from the raw data file:

Name	Type	Length
ProductID	Numeric	8
StartDate	Numeric	8
EndDate	Numeric	8
UnitCostPrice	Numeric	8

Name	Type	Length
UnitSalesPrice	Numeric	8

- b. Define labels and formats in the DATA step to create a data set that generates the following output when used in the PROC PRINT step. The results should contain 259 observations.

Partial PROC PRINT Output

2007 Prices					
Obs	Product ID	Start of Date Range	End of Date Range	Cost Price per Unit	Sales Price per Unit
1	210200100009	06/09/2007	12/31/9999	15.50	34.70
2	210200100017	01/24/2007	12/31/9999	17.80	.
3	210200200023	07/04/2007	12/31/9999	8.25	19.80
4	210200600067	10/27/2007	12/31/9999	28.90	.
5	210200600085	08/28/2007	12/31/9999	17.85	39.40

Challenge

9. Reading a Delimited File with Missing Values and Embedded Delimiters

- a. Write a DATA step to create a temporary data set, **salesmgmt**, using the raw data file named the following:

Windows	"&path\managers.dat"
UNIX	"&path/managers.dat"
z/OS (OS/390)	"&path..rawdata(managers)"

Partial Raw Data File

```
120102/Tom/Zhou/M//Sales Manager/AU/11AUG1969/'06/01/1989'
120103/Wilson/Dawes/M/87975/Sales Manager/AU/22JAN1949/'01/01/1974'
120261/Harry/Highpoint/M/243190//US/21FEB1969/'08/01/1987'
121143/Louis/Favaron/M/95090/Senior Sales Manager/US/26NOV1969/'07/01/1997'
121144/Renee/Capachietti/F/83505/Sales Manager/US/28JUN1964
121145/Dennis/Lansberry/M/84260/Sales Manager/US/22NOV1949/'04/01/1976'
```

- b. **ID** is a numeric value. The **salesmgmt** data set should contain only the variables shown in the report below.
- c. Write a PROC PRINT step to generate the report below. The results should contain six observations.

PROC PRINT Output

Orion Star Managers					
Obs	ID	Last	Title	HireDate	Salary
1	120102	Zhou	Sales Manager	01JUN1989	.
2	120103	Dawes	Sales Manager	01JAN1974	87975
3	120261	Highpoint		01AUG1987	243190
4	121143	Favaron	Senior Sales Manager	01JUL1997	95090

5	121144	Capachietti	Sales Manager	.	83505
6	121145	Lansberry	Sales Manager	01APR1976	84260

8.5 Solutions

Solutions to Exercises

1. Reading a Comma-Delimited Raw Data File

```
data work.newemployees;
length First $ 12 Last $ 18 Title $ 25;
infile "&path\newemps.csv" dlm=',';
input First $ Last $ Title $ Salary;
run;

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

2. Reading a Space-Delimited Raw Data File

```
data work.qtrdonation;
length IDNum $ 6;
infile "&path\donation.dat";
input IDNum $ Qtr1 Qtr2 Qtr3 Qtr4;
run;

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

3. Reading a Tab-Delimited Raw Data File

```
data work.managers2;
length First Last $ 12 Title $ 25;
infile "&path\managers2.dat" dlm='09'x;
input ID First $ Last $ Gender $ Salary Title $;
keep First Last Title;

proc print data=work.managers2;
run;
```

4. Reading Nonstandard Data from a Comma-Delimited Raw Data File

```
data work.canada_customers;
length First Last $ 20 Gender $ 1 AgeGroup $ 12;
infile "&path\custca.csv" dlm=',';
input First $ Last $ ID Gender $
      BirthDate :ddmmyy. Age AgeGroup $;
format BirthDate monyy7.;
drop ID Age;
run;

title 'Canadian Customers';
proc print data=work.canada_customers;
run;
title;
```

5. Reading a Delimited Raw Data File with Nonstandard Values

```
data work.prices;
  infile "&path\pricing.dat" dlm='*';
  input ProductID StartDate :date. EndDate :date.
    Cost :dollar. SalesPrice :dollar. ;
  format StartDate EndDate mmddyy10.
    Cost SalesPrice 8.2;
run;

title '2011 Pricing';
proc print data=work.prices;
run;
title;
```

6. Reading In-Stream Delimited Data

```
data work.managers;
  infile datalines dlm=''' ;
  input ID First :$12. Last :$12. Gender $ Salary :comma.
    Title :$25. HireDate :date. ;
  datalines;
120102/Tom/Zhou/M/108,255/Sales Manager/01Jun1993
120103/Wilson/Dawes/M/87,975/Sales Manager/01Jan1978
120261/Harry/Highpoint/M/243,190/Chief Sales Officer/01Aug1991
121143/Louis/Favaron/M/95,090/Senior Sales Manager/01Jul2001
121144/Renee/Capachietti/F/83,505/Sales Manager/01Nov1995
121145/Dennis/Lansberry/M/84,260/Sales Manager/01Apr1980
;

title 'Orion Star Management Team';
proc print data=work.managers noobs;
  format HireDate mmddyy10. ;
run;
title;
```

7. Reading a Comma-Delimited File with Missing Values

```
data work.donations;
  infile "&path\donation.csv" dsd missover;
  input EmpID Q1 Q2 Q3 Q4;
run;

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

8. Reading a Delimited File with Missing Values

```
data work.prices;
  infile "&path\prices.dat" dlm='*' missover;
  input ProductID StartDate :date. EndDate :date.
    UnitCostPrice :dollar. UnitSalesPrice :dollar. ;
  label ProductID='Product ID'
    StartDate='Start of Date Range'
    EndDate='End of Date Range'
```

```

UnitCostPrice='Cost Price per Unit'
UnitSalesPrice='Sales Price per Unit';
format StartDate EndDate mmddyy10.
      UnitCostPrice UnitSalesPrice 8.2;
run;

title '2007 Prices';
proc print data=work.prices label;
run;
title;

```

9. Reading a Delimited File with Missing Values and Embedded Delimiters

```

data work.salesmgmt;
length First Last $ 12 Gender $ 1 Title $ 25 Country $ 2;
format BirthDate HireDate date9.;
infile "&path\managers.dat" dsd dlm='/' missover;
input ID First Last Gender Salary Title Country
      BirthDate :date. HireDate :mmddyy. ;
run;

title 'Orion Star Managers';
proc print data=work.salesmgmt;
  var ID Last Title HireDate Salary ;
run;
title;

```

Solutions to Student Activities (Polls/Quizzes)

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

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.

31

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.

51

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;`

58

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

65

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.

73

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.**

84

8.09 Quiz – Correct Answer

Use the SAS Help facility or documentation to investigate the **DATEw.** 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.

91

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.

97

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 KrahI
Karen Ballinger	(714) 344-4321	Elke Wallstab

Chapter 9 Manipulating Data

9.1 Using SAS Functions	9-3
Exercises	9-10
9.2 Conditional Processing	9-12
Exercises	9-37
9.3 Solutions	9-40
Solutions to Exercises	9-40
Solutions to Student Activities (Polls/Quizzes)	9-44

9.1 Using SAS Functions

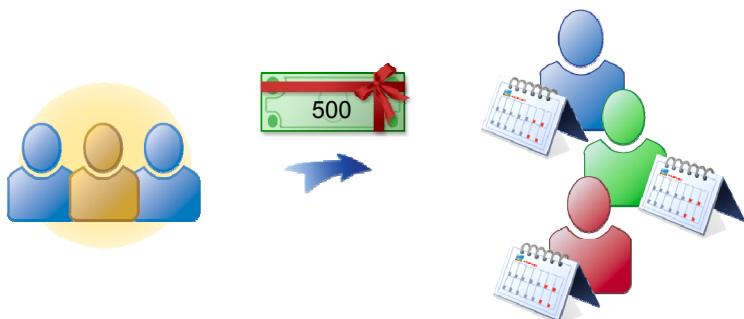
Objectives

- Create data values using SAS functions.

3

Business Scenario

Orion Star management plans to give a \$500 bonus to each employee in his or her hire month.

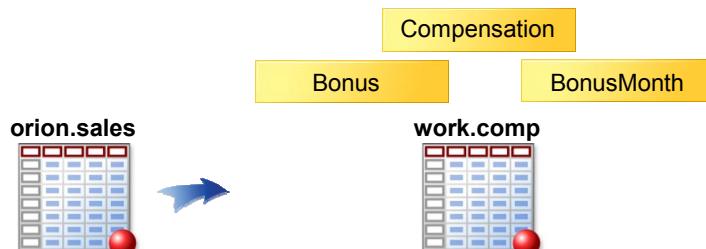


4

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



5

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**.

6

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

7

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.

9

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.

10

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

11

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.

12

Date Functions: Creating SAS Dates

Syntax	Description
TODAY() DATE()	Returns the current date as a SAS date value.
MDY(<i>month,day,year</i>)	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);
```

13

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);
```

14

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;
```

orion.sales has
nine variables.

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.COMP has 165 observations and 12 variables.

15

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

16

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;
```

17

p109a01

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

19

p109a01



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. Creating New Variables

- Retrieve the starter program **p109e01**.
- In the DATA step, create three new variables:
 - Increase**, which is **Salary** multiplied by 0.10
 - NewSalary**, which is **Salary** added to **Increase**
 - BdayQtr**, which is the quarter in which the employee was born
- The new data set should include only **Employee_ID**, **Salary**, **Birth_Date**, and the three new variables.
- Store permanent formats to display **Salary**, **Increase**, and **NewSalary** with commas.
- Modify the program to create the report below, including labels. The results should contain 424 observations.