

SAS® Programming 3: Advanced Techniques and Efficiencies

Course Notes

SAS® Programming 3: Advanced Techniques and Efficiencies Course Notes was developed by Susan Farmer and Johnny Johnson. Additional contributions were made by Kay Alden, Nicole Ball, Rebecca Callaway, Cindy Cragin, Bruce Dawless, Brian Gayle, Linda Jolley, Linda Mitterling, Matthew Perry, Kent Reeve, Christine Riddiough, Lorilyn Russell, Mark Stranieri, Jim Simon, Theresa Stemler, and Jane Stroupe. 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 3: Advanced Techniques and Efficiencies Course Notes

Copyright © 2014 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 E2562, course code LWPG3/PG3, prepared date 27Aug2014.

LWPG3_001

ISBN 978-1-62959-126-1

Table of Contents

Course Description	vii
Prerequisites	viii
Chapter 1 Introduction	1-1
1.1 An Overview of SAS Foundation.....	1-3
1.2 Course Logistics	1-6
1.3 Creating the Course Data	1-13
Demonstration: Creating Course Data Files.....	1-13
Exercises.....	1-14
Chapter 2 Efficient SAS® Programming	2-1
2.1 Identifying Computer Resources Related to Efficiency.....	2-3
Exercises.....	2-19
2.2 Solutions	2-21
Solutions to Exercises	2-21
Solutions to Student Activities (Polls/Quizzes).....	2-22
Chapter 3 Controlling I/O Processing and Memory	3-1
3.1 SAS DATA Step Processing.....	3-3
Demonstration: Viewing the Descriptor Portion of a SAS Data Set.....	3-8
3.2 Controlling I/O.....	3-9
Exercises.....	3-21
3.3 Reducing the Length of Numeric Variables.....	3-24
Exercises.....	3-37
3.4 Compressing SAS Data Sets.....	3-40
Exercises.....	3-49

3.5	Using SAS Views.....	3-51
	Demonstration: Creating a SAS DATA View.....	3-56
	Demonstration: Creating and Using a PROC SQL View.....	3-64
	Exercises.....	3-72
3.6	Solutions	3-76
	Solutions to Exercises	3-76
	Solutions to Student Activities (Polls/Quizzes).....	3-94
Chapter 4 Accessing Observations.....		4-1
4.1	Access Methods	4-3
4.2	Accessing Observations by Number	4-5
	Exercises.....	4-26
4.3	Creating an Index.....	4-27
	Exercises.....	4-54
4.4	Using an Index	4-56
	Exercises.....	4-70
4.5	Solutions	4-74
	Solutions to Exercises	4-74
	Solutions to Student Activities (Polls/Quizzes).....	4-89
Chapter 5 DATA Step Arrays		5-1
5.1	Introduction to Lookup Techniques	5-3
5.2	One-Dimensional Arrays	5-9
	Exercises.....	5-26
5.3	Multidimensional Arrays	5-30
	Exercises.....	5-44
5.4	Loading a Multidimensional Array from a SAS Data Set.....	5-48
	Exercises.....	5-69

5.5	Solutions	5-74
	Solutions to Exercises	5-74
	Solutions to Student Activities (Polls/Quizzes).....	5-87
Chapter 6	DATA Step Hash and Hiter Objects	6-1
6.1	Introduction.....	6-3
6.2	Hash Object Methods.....	6-8
	Exercises.....	6-28
6.3	Loading a Hash Object from a SAS Data Set	6-30
	Exercises.....	6-48
6.4	DATA Step Hiter Object.....	6-52
	Exercises.....	6-70
6.5	Solutions	6-73
	Solutions to Exercises	6-73
	Solutions to Student Activities (Polls/Quizzes).....	6-87
Chapter 7	Combining Data Horizontally.....	7-1
7.1	DATA Step Merges and SQL Procedure Joins	7-3
	Demonstration: Using the DATA Step to Perform a Match-Merge.....	7-14
	Demonstration: Using a PROC SQL Join to Perform a Match-Merge	7-17
	Exercises.....	7-20
7.2	Using an Index to Combine Data.....	7-25
	Demonstration: Using Multiple SET Statements with KEY= Options	7-43
	Exercises.....	7-48
7.3	Combining Summary and Detail Data	7-53
	Exercises.....	7-66
7.4	Combining Data Conditionally	7-70
	Exercises.....	7-85

7.5	Solutions	7-89
	Solutions to Exercises	7-89
	Solutions to Student Activities (Polls/Quizzes).....	7-125
Chapter 8	User-Defined Functions and Formats	8-1
8.1	User-Defined Functions	8-3
	Demonstration: Creating and Using a User-Defined Function	8-11
	Demonstration: Using a User-Defined Function in an SQL Step	8-14
	Exercises.....	8-16
8.2	User-Defined Formats.....	8-18
	Demonstration: Creating Formats from Functions.....	8-22
	Demonstration: Creating Functions from Formats.....	8-27
	Demonstration: User-Defined Informats.....	8-38
	Exercises.....	8-41
8.3	Solutions	8-43
	Solutions to Exercises	8-43
	Solutions to Student Activities (Polls/Quizzes).....	8-51
Chapter 9	Learning More.....	9-1
9.1	Introduction.....	9-3

Course Description

This course is for SAS programmers who prepare data for analysis. The comparisons of manipulation techniques and resource cost benefits are designed to help programmers choose the most appropriate technique for their data situation.

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 <http://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 the 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 SAS Bookstore on the web at <http://support.sas.com/publishing/> for a complete list of books and a convenient order form.

Prerequisites

This course is **not** appropriate for beginning SAS software users. Before attending this course, you should have at least nine months of SAS programming experience and should have completed the SAS® Programming 2: Data Manipulation Techniques course. Specifically, you should be able to do the following:

- understand your operating system file structures and perform basic operating system tasks
- understand programming logic concepts
- understand the compilation and execution processes of the DATA step
- use different varieties of input to create SAS data sets from external files
- use SAS software to access SAS libraries
- create and use SAS date values
- read, concatenate, merge, match-merge, and interleave SAS data sets
- use the DROP=, KEEP=, and RENAME= data set options
- create multiple output data sets
- use one-dimensional array processing and DO loops to process data iteratively
- use SAS functions to perform data manipulation and transformations
- use the FORMAT procedure to create user-defined formats.

Chapter 1 Introduction

1.1 An Overview of SAS Foundation	1-3
1.2 Course Logistics	1-6
1.3 Creating the Course Data	1-13
Demonstration: Creating Course Data Files	1-13
Exercises	1-14

1.1 An 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.

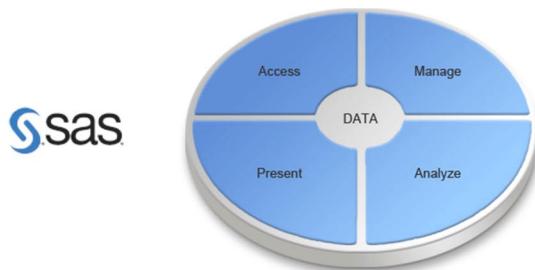


4

What Can You Do with SAS?

SAS software enables you to do the following:

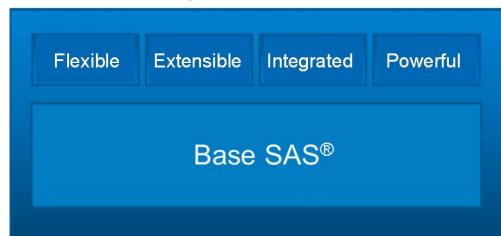
- access data across multiple sources
- manage data
- perform sophisticated analyses
- deliver information across your organization



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. Base SAS provides the following:

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

6

About This Class

This class focuses on programming efficiencies and advanced data manipulation techniques, including the following:

- efficiency considerations
- controlling input and output processing
- controlling data set size
- indexing SAS data sets
- using DATA step arrays
- using DATA step hash and hiter objects
- combining data horizontally



7

1.01 Poll

Have you worked with SAS DATA step programming?

- a. yes, only 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 that are 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

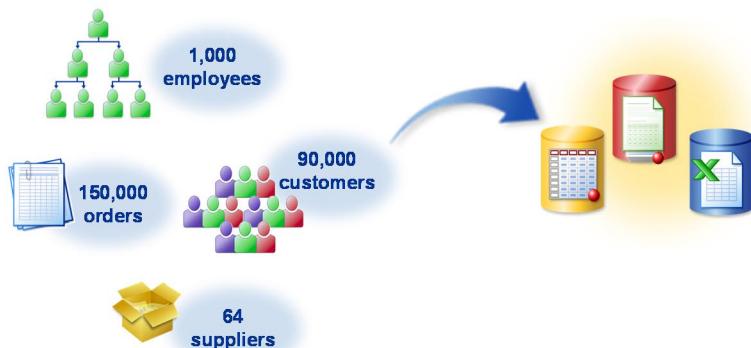
This course focuses on a fictitious global sports and outdoors retailer that has traditional stores, an online store, and a large catalog business.



12

Orion Star Data

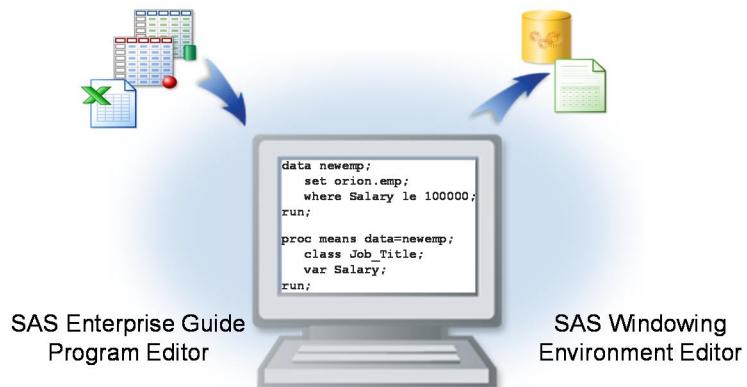
Large amounts of data are stored in transactional systems in various formats.



13

Orion Star Business Scenarios

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



14

	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
Projects	yes	no
Autocomplete	yes	no
Program Flow Analysis	yes	no

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.

SAS Enterprise Guide 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.

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

Running SAS Programs

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



17

Running SAS Programs

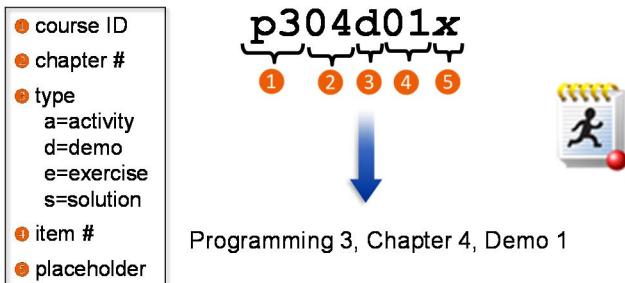
There are other modes for processing SAS programs.

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>

18

Program Naming Conventions

In this course, you retrieve and save SAS programs. SAS program names use the structure shown below.



19

Filename and Library Name References

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

Examples:

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

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

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

20

Three Levels of Exercises

The course is designed to have you complete only **one** set of exercises. Select the level that is 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.
Students might need to use the Help facility.

21

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



22

Extending Your Learning

After class, you 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.

23

1.3 Creating the Course Data

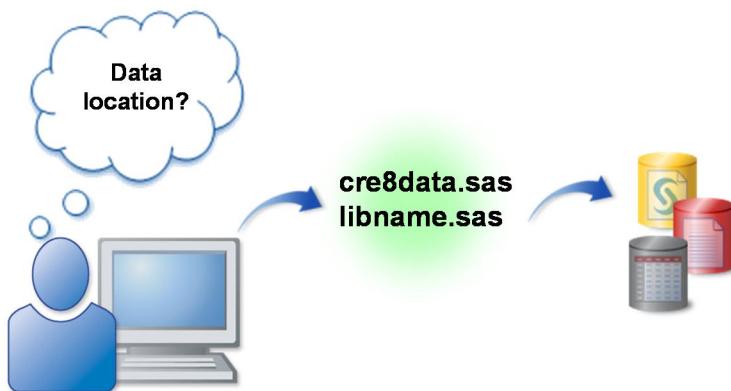
Objectives

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

26

Business Scenario

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



27



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.

- 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 these files.

Create the data files here: _____

- Select **File** ⇒ **Open** ⇒ **Program**.
- If necessary, navigate to the data folder.
- Select **cre8data** and click **Open**. The program is displayed in an editor.

Notice the default value for the %LET statement.

- If your files are to be saved in a location other than **s:\workshop**, change the %LET statement. Modify the value assigned to PATH= to reflect the alternate location from step 1 above.

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

- Press F3 to submit the program.
- View the log and verify that there are no errors.
- View the results 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 SAS or SAS Enterprise Guide session.

- Open the **libname** program.

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

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

- Press F3 to submit the **libname** program.
- View the log 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

- 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 these files

Create the data files here: _____

- Select **File** ⇒ **Open** ⇒ **Program**.

- c. Navigate to the data folder, select **cre8data**, and click **Open**. The program is displayed in an editor.

Observe the default value for the %LET statement.

```
/* Windows/UNIX */

/* STEP 1: Notice the default values for the %LET statements. */

/* STEP 2: If your files are not to be located in S:\workshop */
/* change the value of PATH= %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;
```

- d. If your files are to be saved in a location other than **s:\workshop**, change the %LET statement. Modify the value assigned to PATH= to reflect the alternate location from step **1.a** above.

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

- e. Press F3 to submit the program.
- f. View the log and verify that there are no errors.
- g. View the results and verify that the output contains a list of data files that is 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. View the log and verify that there are no errors or warnings.

Chapter 2 Efficient SAS® Programming

2.1 Identifying Computer Resources Related to Efficiency.....	2-3
Exercises	2-19
2.2 Solutions	2-21
Solutions to Exercises	2-21
Solutions to Student Activities (Polls/Quizzes)	2-22

2.1 Identifying Computer Resources Related to Efficiency

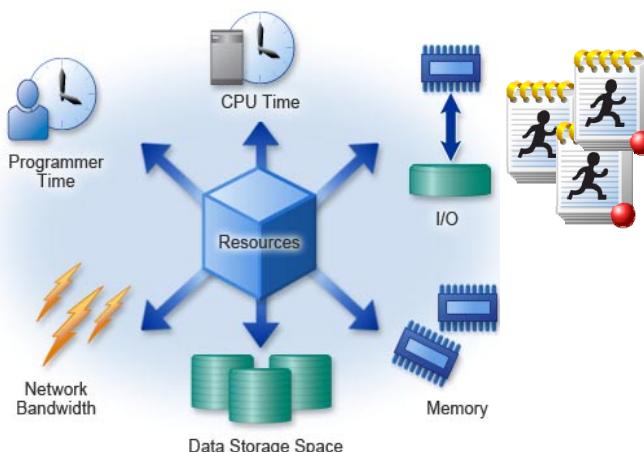
Objectives

- Identify the computer resources related to program efficiency.
- Benchmark resource usage.
- Use SAS system options to report computer resource usage.

2

SAS Program Resources

What resources are required to execute a SAS program?

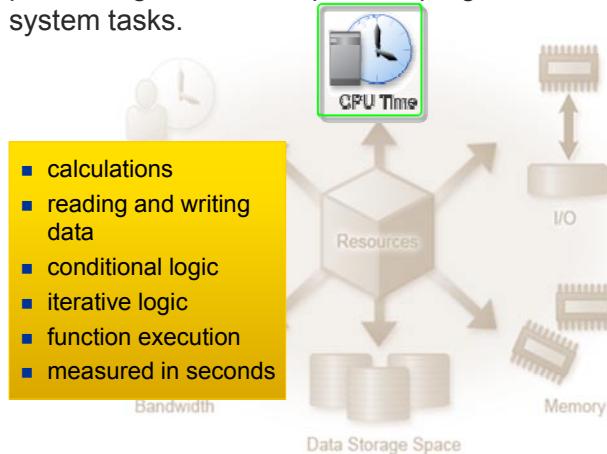


3

CPU	measures the amount of time that the central processing unit uses to perform requested tasks such as calculations, reading and writing data, conditional and iterative logic, and so on. Total CPU time is the combination of system CPU time and user CPU time.
	<i>System CPU time:</i> the CPU time spent to perform operating system tasks (system overhead tasks) that support the execution of SAS code (file system access, network access, memory management)
	<i>User CPU time:</i> the CPU time spent to execute the SAS code as written by the user
I/O	provides a measurement of the Read and Write operations that are performed when data and programs are moved from a storage device to memory (input) or from memory to a storage or display device (output).
	UNIX I/O: recorded as block input and block output operations
	z/OS I/O: recorded as EXCP count
Memory	is the size of the work area that is required to hold executable program modules, data, and buffers.
Data storage space	is the amount of space on a disk or tape that is required to store data.
Programmer time	is the amount of time that is required for the programmer to write and maintain the program. This can be decreased through well documented, logical programming practices.
Network bandwidth	refers to the capacity to share data between devices. Bandwidth represents the amount of data that can pass through a network interface over time. This resource is heavily dependent on network loads.

Computer Resources Related to Efficiency

CPU time is the amount of time that the central processing unit uses to perform program and operating system tasks.

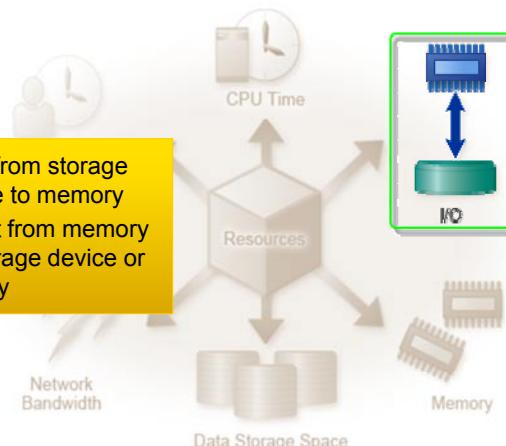


Computer Resources Related to Efficiency

I/O measures Read and Write operations between a storage device and memory.

- input from storage device to memory
- output from memory to storage device or display

5

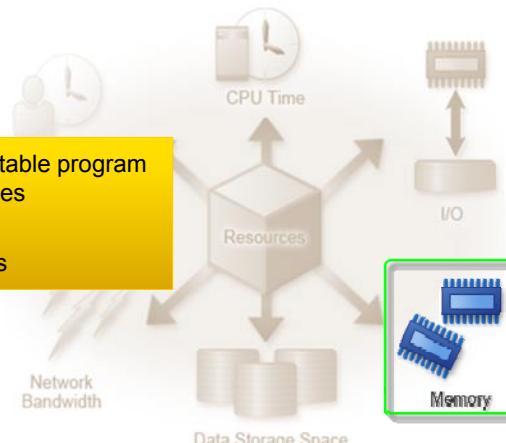


Computer Resources Related to Efficiency

Memory is the size of the work area in volatile memory.

- executable program modules
- data
- buffers

6

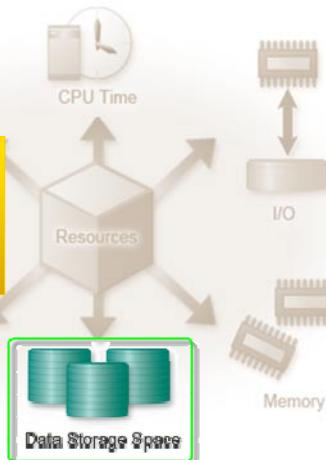


Computer Resources Related to Efficiency

Data storage space is the physical space that is needed to store data on mass storage devices.

- disc drives
- tape drives
- flash drives
- memory cards

7

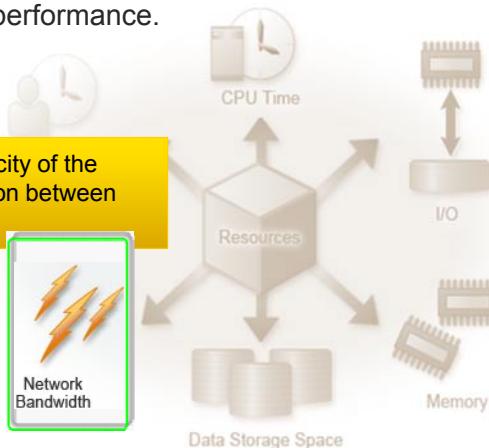


Computer Resources Related to Efficiency

Network bandwidth is the available throughput for data communications. Greater throughput typically results in greater performance.

the capacity of the connection between devices

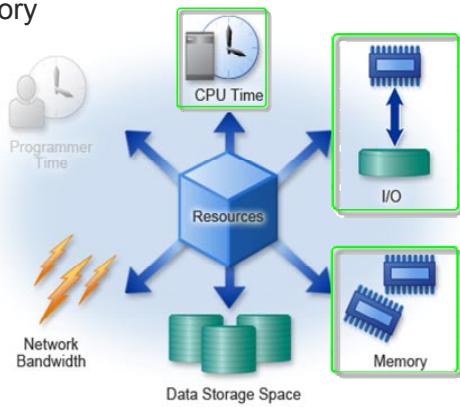
8



Computer Resources Related to Efficiency

CPU time, I/O, and memory are computer resources that the programmer can manage to help SAS programs execute more efficiently.

-  Although important, *programmer time* is not considered a computer resource.



9

2.01 Quiz

Match the definitions to the resources.

Definition

- the physical space needed to store data
- amount of time used for calculations, iterative logic, conditional logic, and reading and writing data
- size of the work area for holding executable program modules, data, and buffers
- measurement of Read and Write operations
- throughput for data communications

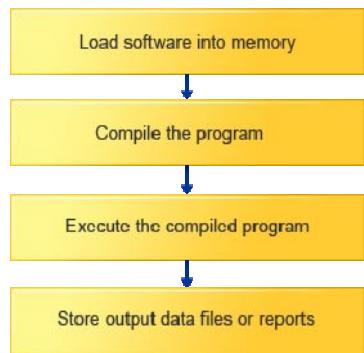
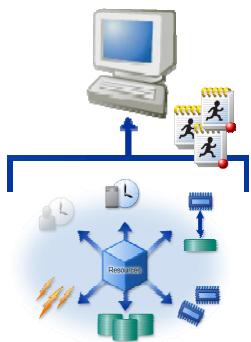
Resources

- a) I/O
- b) memory
- c) CPU time
- d) data storage space
- e) network bandwidth

10

Computer Resources Related to Efficiency

What tasks must the computer perform to execute your SAS program?

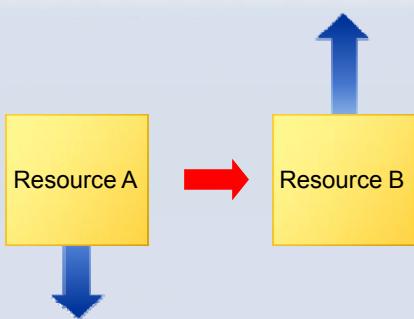


- Computer resources are consumed during each task.

12

Understanding Trade-Offs in Efficiency

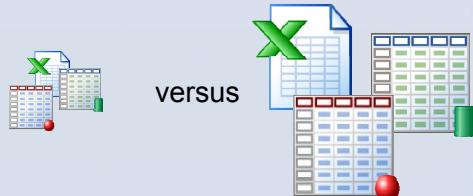
Minimizing resource usage can be challenging.
Decreasing Resource A might require an increase
in Resource B.



13

Understanding Trade-Offs in Efficiency

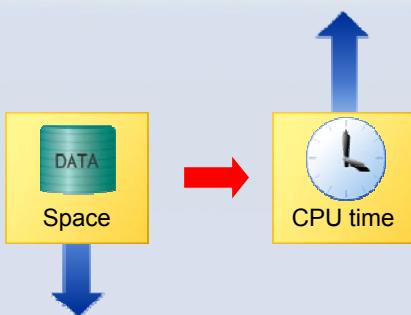
Resource usage is data dependent. Size and type of data can influence resource utilization.



14

Understanding Trade-Offs in Efficiency

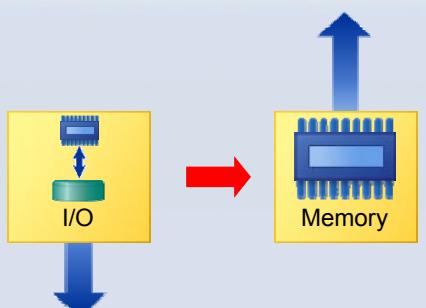
Compressing data saves storage space but requires more CPU time to uncompress.



15

Understanding Trade-Offs in Efficiency

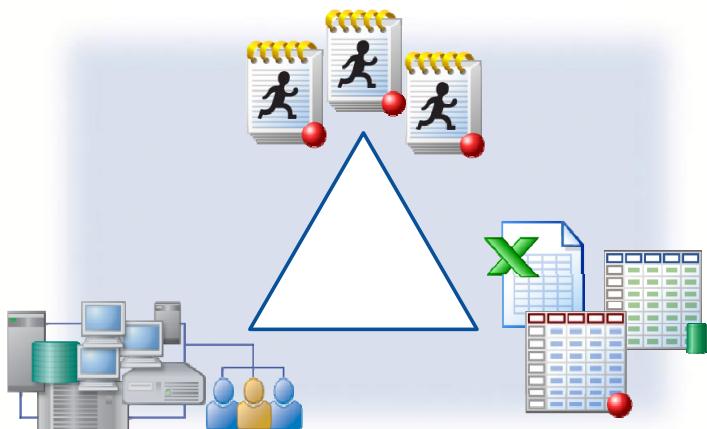
Increasing buffer size decreases the number of I/O operations but requires more available memory.



16

Understanding Trade-Offs in Efficiency

You must decide which factors are the most important for improving resource usage at your site.



17

To make this decision, you must know the following:

- which resources are scarce or costly at your site
- how and when your programs are used
- the type and volume of data that your programs process

2.02 Multiple Choice Poll

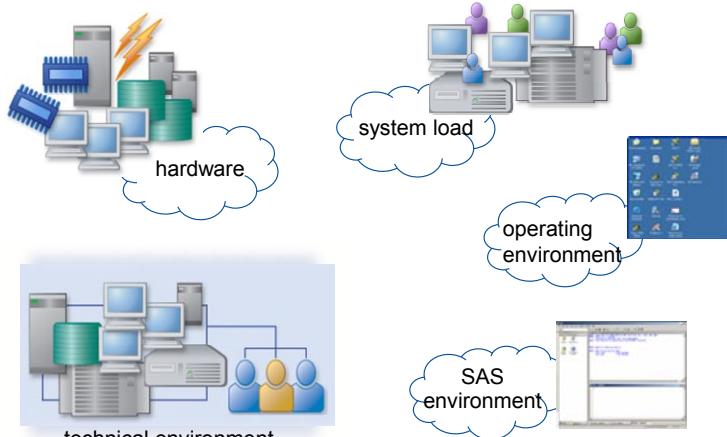
If you decrease disk space usage, which resource might *increase* in usage?

- a. CPU time
- b. I/O
- c. both
- d. neither

18

Assessing Your Technical Environment

You need to understand your site's technical environment and resource constraints.

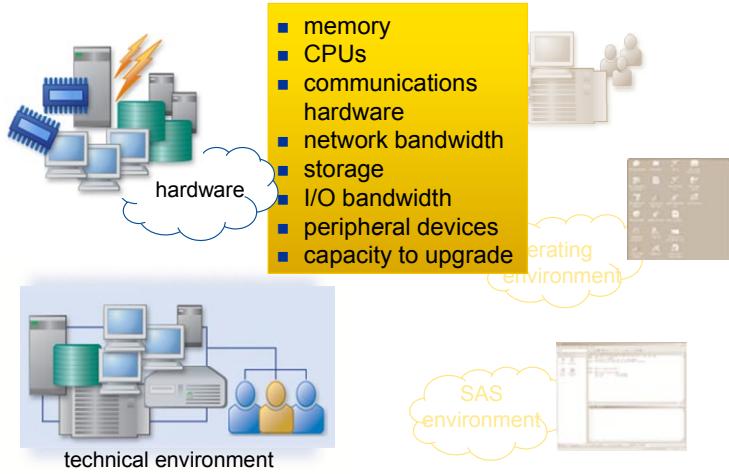


20

...

Assessing Your Technical Environment

You need to understand your site's hardware.

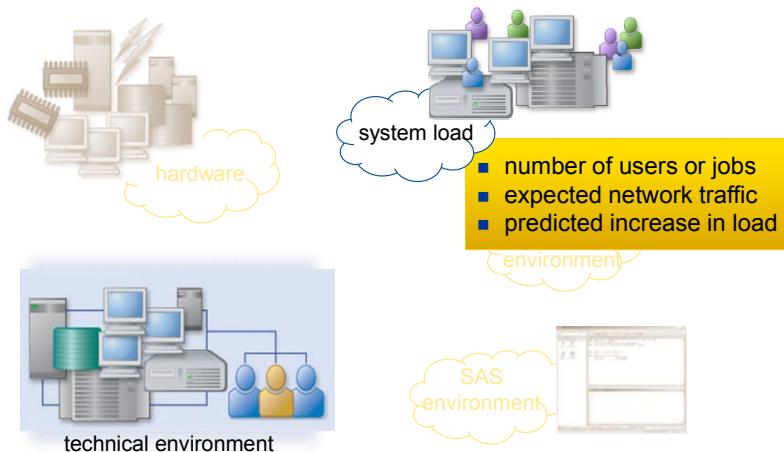


21

...

Assessing Your Technical Environment

You need to understand your site's system load.



22

...

Assessing Your Technical Environment

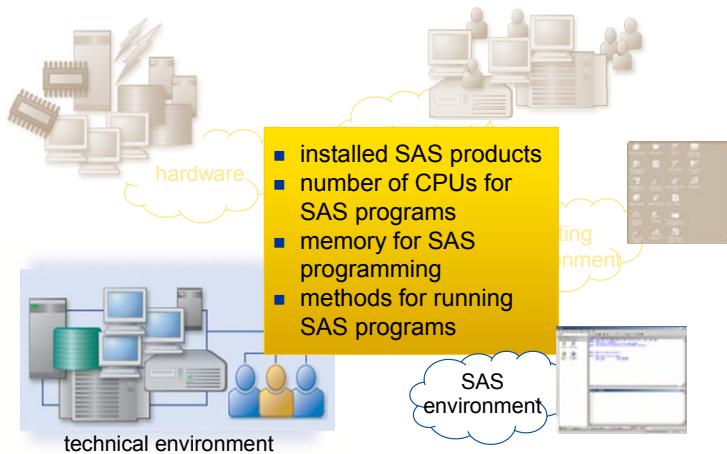
You need to understand your site's operating environment.



23

Assessing Your Technical Environment

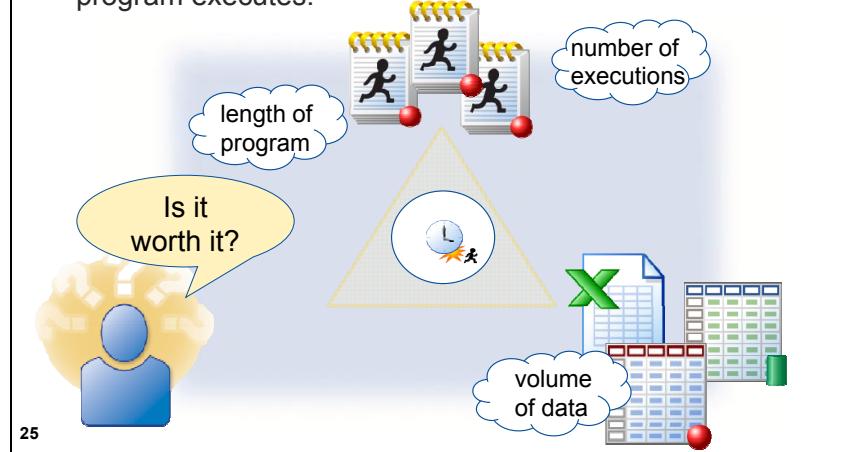
You need to understand your site's SAS environment.



24

Understanding Your Programs and Data

Consider the length and complexity of the program, the size of the data sources, and how often a particular program executes.



Understanding Your Programs and Data

Above all, you need to know your data.

The screenshot shows the SAS Explorer interface. On the left is a properties panel for "Orion.Employeepayroll Properties" with tabs for General, Details, Columns, Indexes, Integrity, and Passwords. The "Columns" tab displays a table of columns with their names, types, lengths, formats, and informat. The main area shows an "Explorer" window titled "Contents of 'Orion'" with a tree view of data sets and tables like "Agesmod", "Budget", "Catalog", "Continent", "Country", "Countryinfo", "Couponpcpt", "Coupons", "Customer", "Customer...", "Customer...", "Customer...", "Customer...", "Emplov..._Emplov...", "Emplov...", and "Results". A green "X" icon is overlaid on the bottom right of the interface. A small number "26" is located at the bottom left corner of the slide.

2.03 Multiple Answer Poll

To improve efficiency, on which of the following should you focus?

- a. programs that read many large data sets
- b. small programs
- c. programs that read small amounts of data
- d. programs that run often

27

Benchmarking SAS Programs

How do you measure and compare specific techniques for efficiency?

IF ...
 THEN ...;
IF ...
 THEN ...;

IF ...
 THEN ...;
ELSE ...;

SELECT ...;
WHEN ...;
END;

real time 0.21 seconds
user cpu time 0.15 seconds
system cpu time 0.01 seconds
Memory 481k
OS Memory 9640k



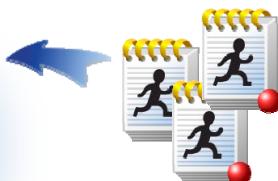
29

Benchmarking

Benchmarking is the process of measuring and comparing resource usage.

Partial SAS Log

real time	0.21 seconds
user cpu time	0.15 seconds
system cpu time	0.01 seconds
Memory	481k
OS Memory	9640k



- Compare performance
- between techniques
 - at different periods of resource availability.

30

Benchmarking SAS Programs

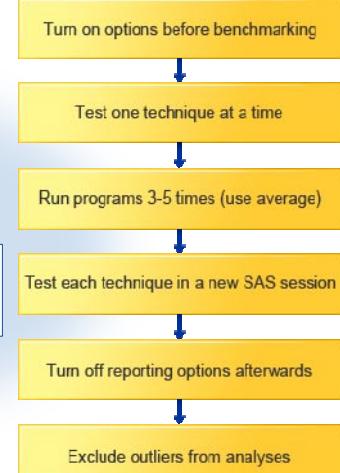
When you benchmark, follow these guidelines.



Actual data



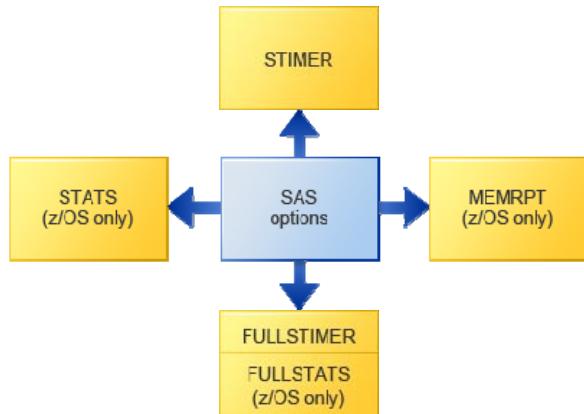
Actual conditions



31

Tracking Resource Usage

SAS system options for tracking resource usage are operating system dependent.



32

There are four SAS system options that you can use to track and report on resource utilization.

STIMER tracks the CPU time that is used to perform a task (DATA or PROC step).

MEMRPT tracks the memory that is used while performing a task.

FULLTIMER tracks the usage of additional resources. This option is ignored unless STIMER or MEMRPT is in effect. It can also be specified by the alias FULLSTATS.

STATS writes information that is tracked by the above options to the SAS log.

 Refer to the *SAS® Companion* for your operating environment to obtain more information about operating-system-dependent options for tracking resource usage.

Tracking Resource Usage: Windows and UNIX

STIMER is the default system option to display real time and CPU time in the SAS log after each step.

OPTIONS STIMER;

NOTE: PROCEDURE PRINT used (Total process time):
 real time 3.48 seconds
 cpu time 0.62 seconds

FULLSTIMER displays all available system performance statistics.

NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.20 seconds
 user cpu time 0.06 seconds
 system cpu time 0.01 seconds
 Memory 340k
 OS Memory 10152k
 Timestamp 7/14/09 10:58:29 AM

OPTIONS FULLSTIMER;

33

Tracking Resource Usage: z/OS

STIMER is the default system option to display real time and CPU time in the SAS log after each step.

OPTIONS STIMER; (invocation only)

NOTE: PROCEDURE PRINT used (Total process time):
 real time 3.48 seconds
 cpu time 0.62 seconds

FULLSTATS displays all available system performance statistics.

CPU time -	00:00:00.02
Elapsed time -	00:00:00.31
Vector affinity time -	00:00:00.00
Vector usage time -	00:00:00.00
RSM Hiperspace time -	00:00:00.00
EXCP count -	122
Task memory -	3436K (58K data, 3378K program)
Total memory -	8350K (2720K data, 5630K program)

OPTIONS FULLSTATS;

34

Tracking Resource Usage: z/OS

Additional options are available in the z/OS environment. The STATS system option specifies whether performance statistics are to be written to the SAS log.

OPTIONS STATS;

The displayed statistics are determined by MEMRPT, STIMER, and FULLSTATS settings.

If both the STATS and MEMRPT options are in effect, then the total memory used by the SAS session is written to the SAS log.

OPTIONS MEMRPT;

35

2.04 Multiple Choice Poll

Which of the following should you **not** do when you are benchmarking?

- Execute the code for each technique in a separate SAS session.
- Use an average of resource usage values rather than the minimums.
- When you benchmark, make sure that the system is idle.
- Include only the SAS code that is essential for performing the task that you are measuring.

36



Exercises

1. Benchmarking

The **p302e01** program reads a 617,000-row raw data file. Investigate the program and notice that it currently subsets observations based on **Order_Date**.

p302e01

```

data _null_;
  infile "&path\BIGorderfact.dat" pad; *Windows and UNIX;
/* infile '.workshop.rawdata(BIGorderfact)' pad; *z/OS; */
  input @37 Order_Date date9. @;
  input @1 Customer_ID 12.
    @13 Employee_ID 12.
    @25 Street_ID 12.
    @46 Delivery_Date Date9.
    @55 Order_ID 12.
    @67 Order_Type 2.
    @69 Product_ID 12.
    @81 Quantity 4.
    @90 Total_Retail_Price 13.
    @105 CostPrice_Per_Unit 10.
    @115 Discount 5.;
  if year(Order_Date)=2010;
  format Customer_ID Employee_ID Street_ID Order_ID
    Product_ID 12. Order_Date Delivery_Date date9.
    Order_Type 2. Quantity 4. Total_Retail_Price dollar13.2
    CostPrice_Per_Unit dollar10.2 Discount Percent. ;
run;

```

Notes about the syntax:

PAD This INFILE statement option controls whether SAS pads the records that are read from an external file with blanks to the length that is specified in the LRECL= option. The LRECL= option specifies the logical record length. It is dependent on the operating environment.

@ The single trailing @ sign holds an input record for the execution of the next INPUT statement within the same iteration of the DATA step.

- Turn on the appropriate option for reporting extended resource statistics in the log.
 - Submit the program and check the log to investigate resource utilization metrics.
 - Modify the program by moving the subsetting IF statement closer to the top of the DATA step.
-  Make sure that you move the subsetting IF statement to a location as early in the program as possible.
- Submit the revised program and re-check the log to investigate resource utilization metrics.
 - Were any resources conserved when you moved the subsetting IF statement?
-
- Turn off the option to report extended resource statistics in the log.

Level 2

2. Investigating the PROC SORT TAGSORT Option

- Using either the SAS Help facility (available from the Help menu or the Help icon in your SAS session) or SAS online documentation (<http://support.sas.com/documentation/index.html>), investigate the TAGSORT option used in PROC SORT.
- What resources are conserved by using the TAGSORT option? _____

2.2 Solutions

Solutions to Exercises

1. Benchmarking

- Turn on the appropriate option for reporting extended resource statistics in the log.

```
options fullstimer;

data _null_;
  <additional SAS code>
  ...

```

- Submit the program and check the log to investigate resource utilization metrics.
- Modify the program by moving the subsetting IF statement closer to the top of the DATA step.

p302s01

```
data _null_;
  infile "&path\BIGorderfact.dat" pad; *Windows and UNIX;
/* infile '.workshop.rawdata(BIGorderfact)' pad; *z/OS; */
  input @37 Order_Date date9. @;
  if year(Order_Date)=2010;
  input  @1 Customer_ID 12.
         @13 Employee_ID 12.
         @25 Street_ID 12.
         @46 Delivery_Date date9.
         @55 Order_ID 12.
         @67 Order_Type 2.
         @69 Product_ID 12.
         @81 Quantity 4.
         @90 Total_Retail_Price 13.
         @105 CostPrice_Per_Unit 10.
         @115 Discount 5.;
  format Customer_ID Employee_ID Street_ID Order_ID Product_ID 12.
         Order_date Delivery_Date date9.
         Order_Type 2. Quantity 4. Total_Retail_Price dollar13.2
         CostPrice_Per_Unit dollar10.2 Discount Percent.;

run;
```

- Submit the revised program and re-check the log to investigate resource utilization metrics.

- e. Were any resources conserved when you moved the IF statement?

You should see less CPU utilization when you move the subsetting IF statement closer to the top of the program. For observations not involved in the subset, numeric values are not converted.

As the subset becomes smaller relative to the number of records in the raw data file, more CPU is conserved. The Real Time that it takes the program to execute should decrease.

- f. Turn off the option to report extended resource statistics in the log

```
options nofullstimer;
```

2. Investigating the PROC SORT TAGSORT Option

- Using either the SAS Help facility (available from the Help menu or the Help icon in your SAS session) or SAS online documentation (<http://support.sas.com/documentation/index.html>), investigate the TAGSORT option used in PROC SORT.
- What resources are conserved by using the TAGSORT option? **The TAGSORT option can be used to conserve disk space that is needed when you sort large files. Only the BY variables and an observation reference are stored on disk.**

Solutions to Student Activities (Polls/Quizzes)

2.01 Quiz – Correct Answers

Match the definitions to the resources.

Definition

- d the physical space needed to store data
- c amount of time used for calculations, iterative logic, conditional logic, and reading and writing data
- b size of the work area for holding executable program modules, data, and buffers
- a measurement of Read and Write operations
- e throughput for data communications

Resources

- a) I/O
- b) memory
- c) CPU time
- d) data storage space
- e) network bandwidth

2.02 Multiple Choice Poll – Correct Answer

If you decrease disk space usage, which resource might *increase* in usage?

- a. CPU time
- b. I/O
- c. both
- d. neither

A compressed data set requires less storage space, but requires more CPU time to uncompress.

19

2.03 Multiple Answer Poll – Correct Answers

To improve efficiency, on which of the following should you focus?

- a. programs that read many large data sets
- b. small programs
- c. programs that read small amounts of data
- d. programs that run often

In general, as the volume of data increases and the frequency of program execution increases, there are greater opportunities for improving program efficiency.

28

2.04 Multiple Choice Poll – Correct Answer

Which of the following should you ***not*** do when you are benchmarking?

- a. Execute the code for each technique in a separate SAS session.
- b. Use an average of resource usage values rather than the minimums.
- c. When you benchmark, make sure that the system is idle.
- d. Include only the SAS code that is essential for performing the task that you are measuring.

Chapter 3 Controlling I/O Processing and Memory

3.1 SAS DATA Step Processing.....	3-3
Demonstration: Viewing the Descriptor Portion of a SAS Data Set.....	3-8
3.2 Controlling I/O.....	3-9
Exercises	3-21
3.3 Reducing the Length of Numeric Variables	3-24
Exercises	3-37
3.4 Compressing SAS Data Sets	3-40
Exercises	3-49
3.5 Using SAS Views	3-51
Demonstration: Creating a SAS DATA View	3-56
Demonstration: Creating and Using a PROC SQL View	3-64
Exercises	3-72
3.6 Solutions	3-76
Solutions to Exercises	3-76
Solutions to Student Activities (Polls/Quizzes)	3-94

3.1 SAS DATA Step Processing

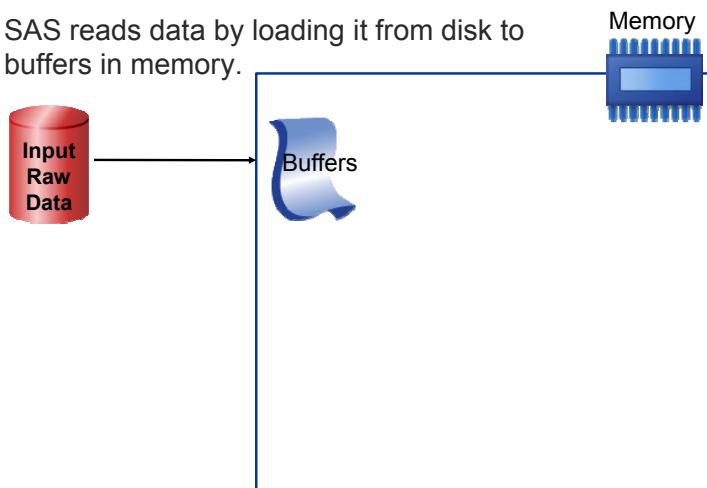
Objectives

- Describe how SAS reads and writes data.
- Explain the concept of data set pages.

3

How SAS Reads Data

SAS reads data by loading it from disk to buffers in memory.

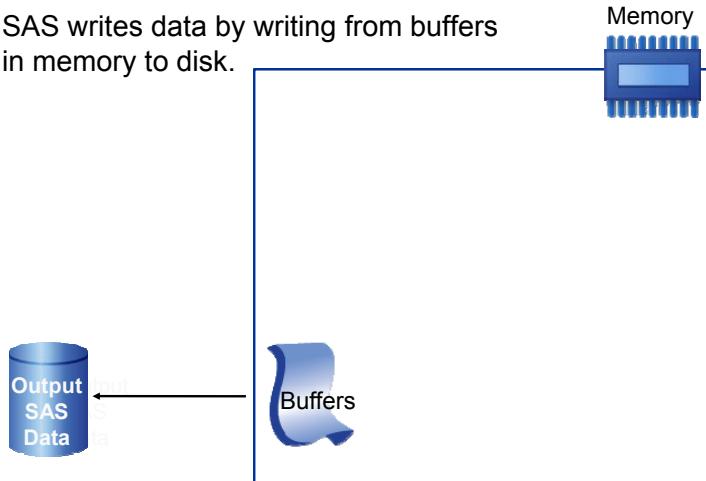


4

...

How SAS Writes Data

SAS writes data by writing from buffers in memory to disk.

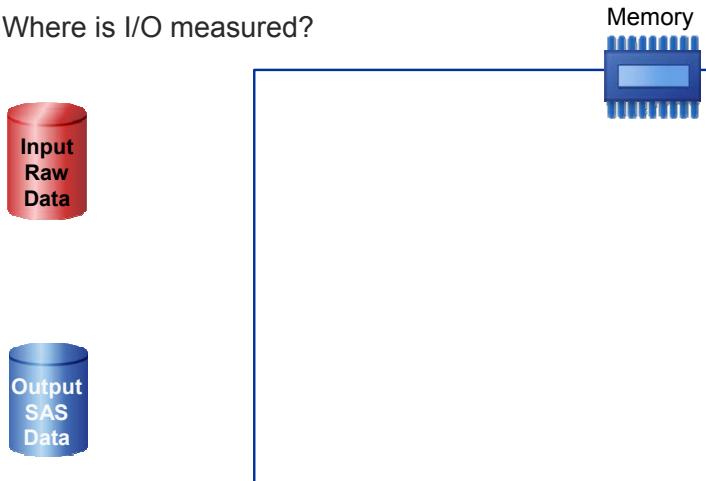


5

...

Reading Raw Data

Where is I/O measured?

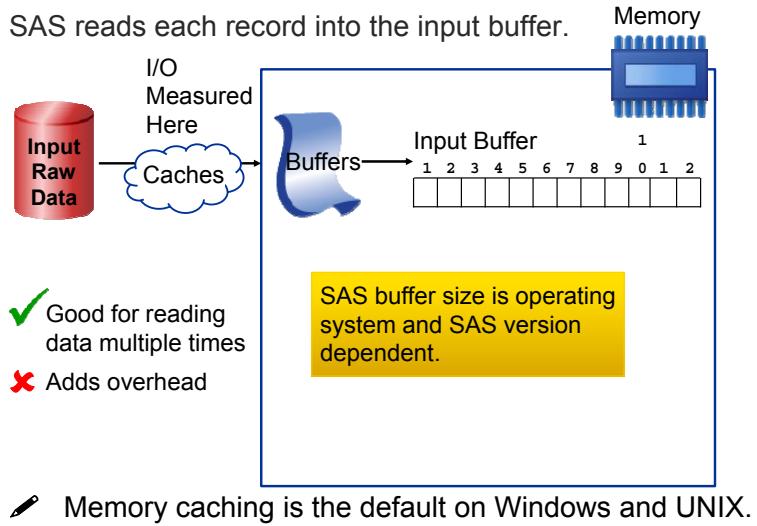


6

...

Reading Raw Data

SAS reads each record into the input buffer.

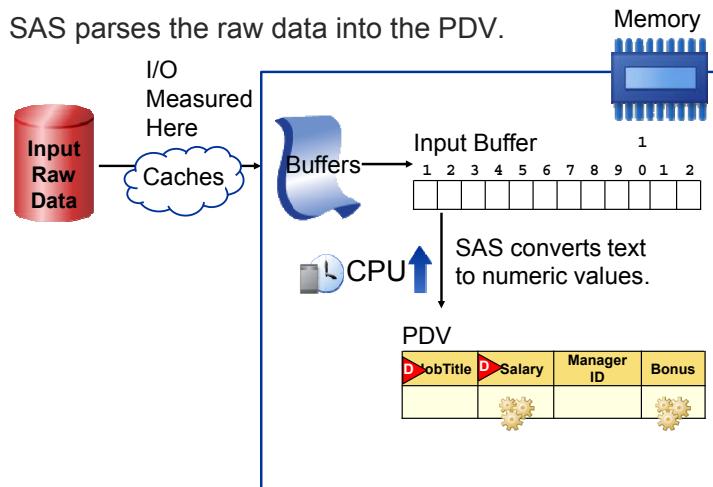


7

...

Reading Raw Data

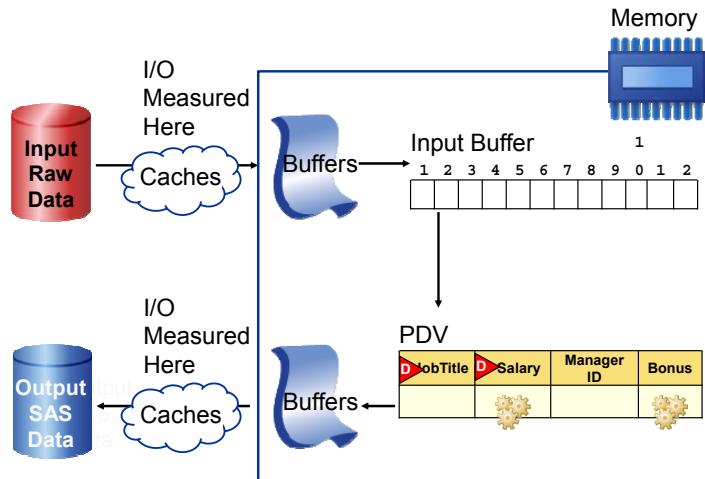
SAS parses the raw data into the PDV.



8

...

Reading Raw Data



9

Reading SAS Data

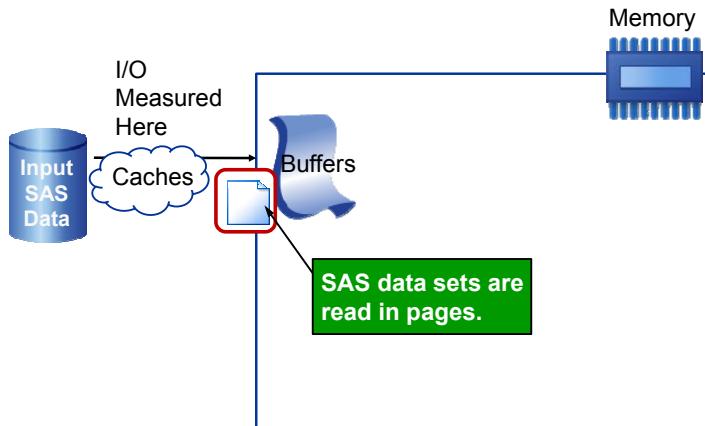
How does the DATA step process input SAS data?



10

...

Reading SAS Data



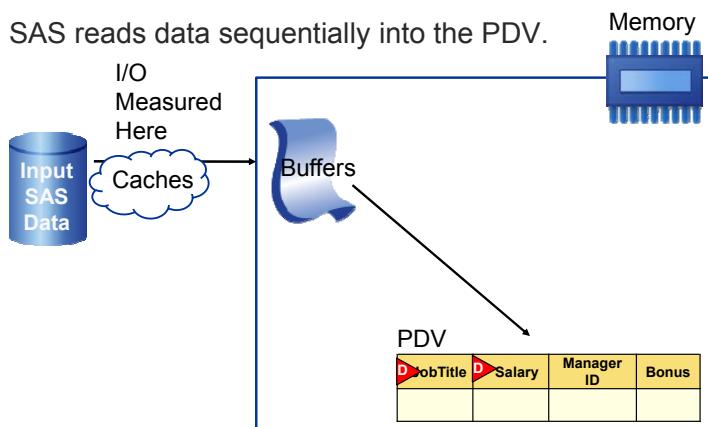
- When SAS creates a data set, it sets the page size permanently to either a default value or a value that you specify.

11

...

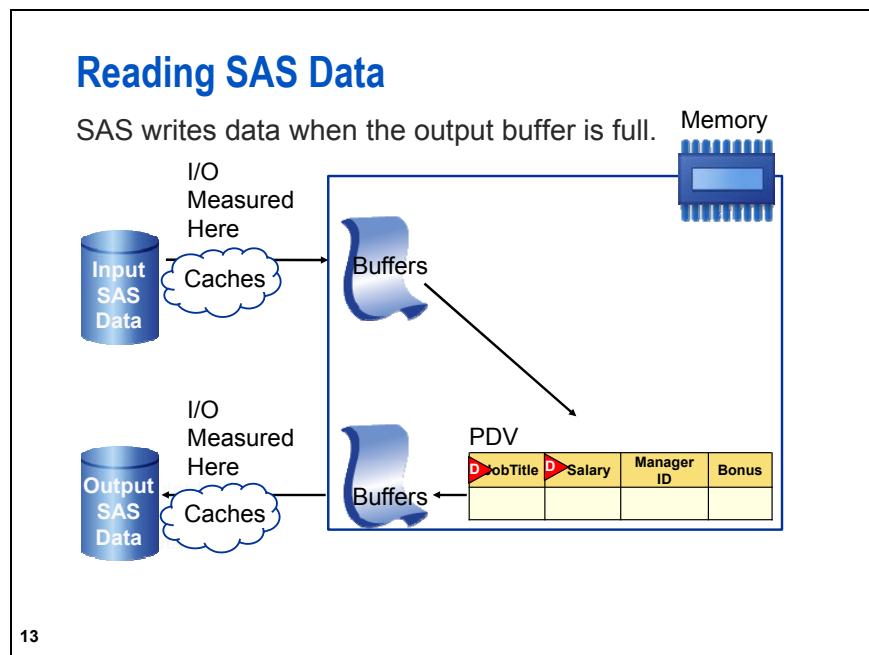
Reading SAS Data

SAS reads data sequentially into the PDV.



12

...



Viewing the Descriptor Portion of a SAS Data Set

p303d01

This demonstration illustrates how to view the descriptor portion of a SAS data set. It concentrates on the Engine/Host Information section.

1. Open the program **p303d01**.
2. Submit the PROC CONTENTS step to display the Engine/Host Dependent Information descriptor portion of **orion.saleshistory**.

p303d01

```
proc contents data=orion.saleshistory;
run;
```

3. Review the results in the Output window or on the Output tab.

Partial PROC CONTENTS Output

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	6
First Data Page	1
Max Obs per Page	355
Obs in First Data Page	336
Index File Page Size	4096
Number of Index File Pages	26
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	s:\workshop\saleshistory.sas7bdat
Release Created	9.0401M0
Host Created	X64_ES08R2

4. The following information can help determine the overall size of the data set:

- Data set page size: **65536 bytes**
- Number of data set pages: **6**
- Index page size: **4096 bytes**
- Number of index pages: **26**

3.2 Controlling I/O

Objectives

- Describe the business scenario.
- Explain the importance of conserving I/O.
- List common techniques for reducing I/O operations.

17

Business Scenario

Human Resources is concerned that large data files consume too much disc space and cause programs to take longer to run. How can performance be improved?

Employee Data Set									
EmployeeID	EmployeeName	StreetID	Street Number	StreetName	City	Postal Code	State	Country	Phone
120101	Lu, Patrick	1600102980	51	Blaney	Blaney				
120101	Lu, Patrick	1600102980		Blaney	Blaney				
120102	Zhou, Tom	1600101750	1	Bourne	Bourne				
120102	Zhou, Tom	1600101750	1	Bourne	Bourne				
120103	Smith, John	1600103074	166	Blaney	Blaney				
1230	Sales Manager	1230	CA	USA	(408) 555-1230				
1230	Sales Manager	1230	CA	USA	(408) 555-1230				
3125	SAU	Sales Manager	Sales Manager	Sales Manager	Sales Manager	3125	CA	USA	(408) 555-1235
3125	SAU	Sales Manager	Sales Manager	Sales Manager	Sales Manager	3125	CA	USA	(408) 555-1235
2119	AU	Sales Manager	Sales Manager	Sales Manager	Sales Manager	2119	AU	Australia	(02) 555-1235
6074	15887	.	S	9	Home	4611215555-3349			
6074	15887	.	S	0	Work	+611215551-0001			
3510	10744	.	O	2	Home	+611315555-9700			
3510	10744	.	O	2	Work	+611315551-0002			
3996	51114	.	M	1	Home	+611215555-3398			

18

Possible Solutions

Considerations for improving performance and conserving resources include the following:

reducing the amount of data processed

reducing the length of numeric variables

compressing data files

using SAS views

19

3.01 Multiple Answer Poll

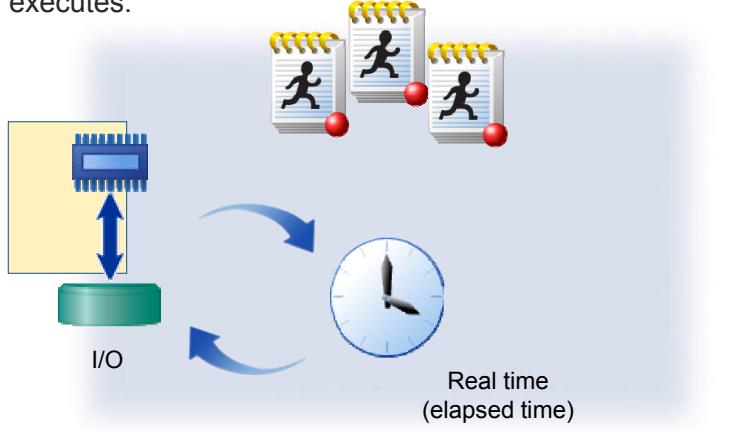
When you read raw data and create a SAS data set using the DATA step, when is I/O measured?

- a. when SAS copies the input data from the source to a buffer in memory
- b. when SAS copies the input data from the input buffer to the program data vector
- c. when SAS copies the output data from the output buffer to the output data set
- d. none of the above

21

Introduction

Real time correlates highly with I/O. Reducing I/O is one of the best ways to influence how quickly a SAS program executes.

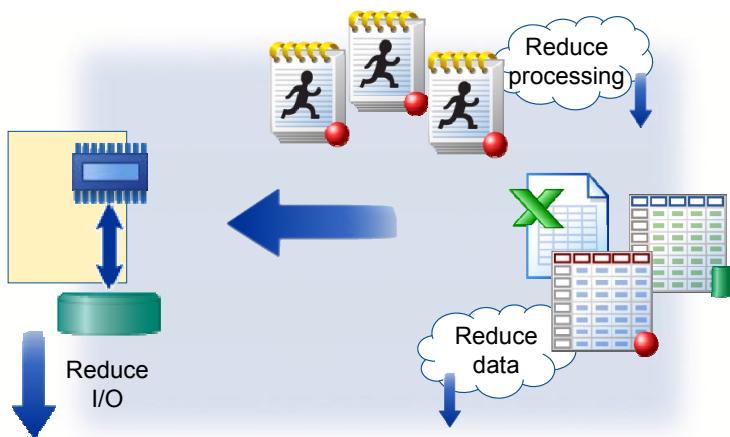


23

...

Introduction

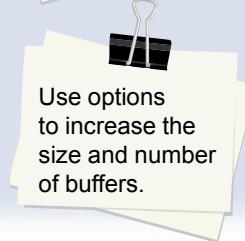
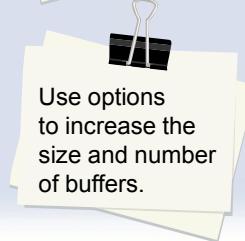
Reduce I/O by reducing the amount of data or the number of times that the data is processed.



24

Reducing I/O

Common techniques for reducing I/O include the following:

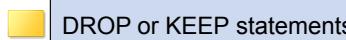
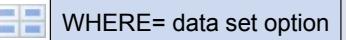
-  Minimize the number of variables and observations that are processed.
-  Use the SASFILE statement.
-  Reduce the number of times that data is processed.
-  Use options to increase the size and number of buffers.

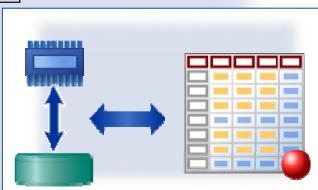


25

Technique 1: Minimize the Number of Variables and Processed Observations

To minimize variables, use the following:

-  DROP or KEEP statements
 -  DROP= or KEEP= data set options
- To minimize observations, use the following:
-  WHERE statement
 -  WHERE= data set option
 -  OBS= and FIRSTOBS= data set options



...

28

Technique 1: Minimize the Number of Variables and Processed Observations

The data set **bonus** contains 11 variables and 424 observations. How can this program's performance be improved?



Program 1

```
data bonus;
  set orion.staff;
  YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
  where JobTitle contains 'Manager';
  class ManagerID;
  var YrEndBonus;
run;
```

29

p303d02

Technique 1: Minimize the Number of Variables and Processed Observations

The data set **bonus** contains 2 variables and 41 observations. The DATA step writes a smaller file and PROC MEANS loads a smaller file.



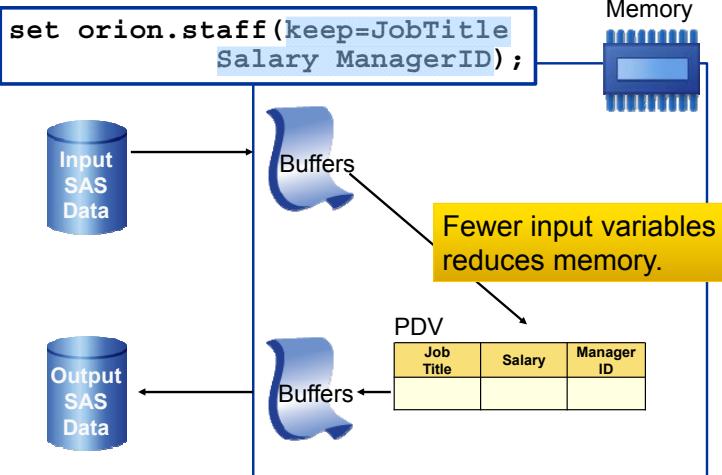
Program 2

```
data bonus(keep=ManagerID YrEndBonus);
  set orion.staff (keep=JobTitle
                  Salary ManagerID);
  where JobTitle contains 'Manager';
  YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
  class ManagerID;
  var YrEndBonus;
run;
```

30

p303d02

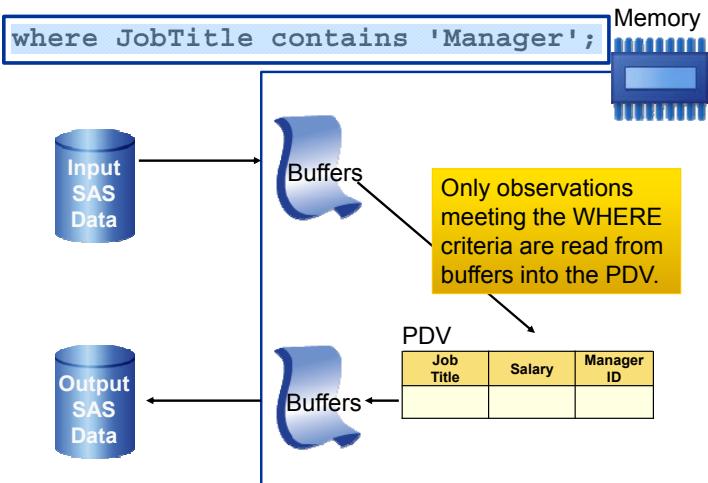
Reading SAS Data



31

...

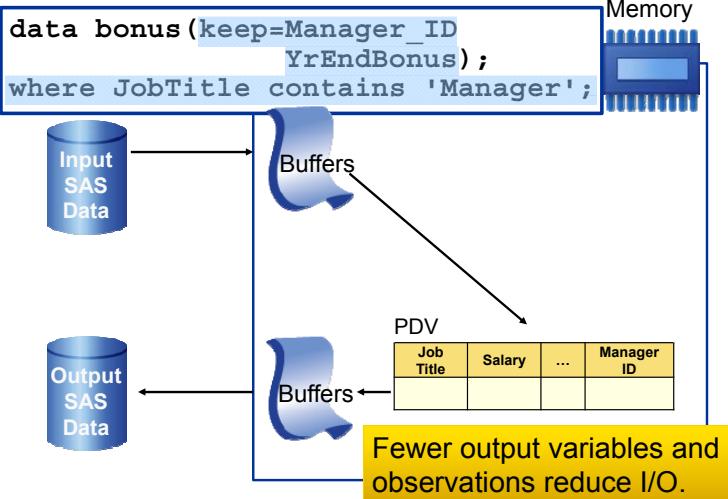
Reading SAS Data



32

...

Writing SAS Data



33



Fewer variables and observations are written to the **bonus** data set, which reduces the number of pages that are required.

Setup for the Poll

Review the following SAS programs:

```

1 data bonus;
   set orion.staff;
   YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
   where JobTitle contains 'Manager';
   class ManagerID;
   var YrEndBonus;
run;

2 data bonus(keep=ManagerID YrEndBonus);
   set orion.staff(keep=JobTitle
                  Salary ManagerID);
   where JobTitle contains 'Manager';
   YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
   class ManagerID;
   var YrEndBonus;
run;

```

34

p303d02

3.02 Multiple Choice Poll

In addition to decreasing I/O when the DATA step creates **bonus**, where else does program 2 decrease I/O?

- a. Program 2 reads fewer variables from **orion.staff** into the PDV.
- b. The PROC MEANS step in program 2 loads a smaller version of **bonus**.
- c. No additional decrease in I/O occurs in program 2.

35

Technique 2: Reduce the Number of Times That Data Is Processed

Subsetting in the DATA step results in a smaller file for PROC MEANS to load. How can this program's performance be improved?



Program 1

```
data bigsalaries;
  set orion.staff;
  where Salary > 50000;
run;

proc means data=bigsalaries mean sum;
  class ManagerID;
  var Salary;
run;
```

38

p303d03

Technique 2: Reduce the Number of Times That Data Is Processed

Reduce I/O and data storage space by subsetting in the PROC step. No intermediate data set is needed.



Program 2

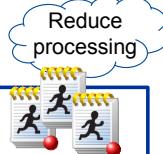
```
proc means data=orion.staff mean sum;
  class ManagerID;
  var Salary;
  where Salary > 50000;
run;
```

39

p303d03

Technique 3: Use the SASFILE Statement

The data set **orion.customerdim** is read from disk four times, and incurs I/O each time.



```
proc freq data=orion.customerdim;
  tables CustomerCountry CustomerType;
run;
proc print data=orion.customerdim noobs;
  where CustomerType='Orion Club Gold members high activity';
  var CustomerID CustomerName CustomerAgeGroup;
run;
proc means data=orion.customerdim mean median max min;
  var CustomerAge;
  class CustomerGroup;
run;
proc tabulate data=orion.customerdim format=8.;
  class CustomerAgeGroup CustomerType;
  table CustomerType All=Total,
    CustomerAgeGroup*n=' ' All=Total*n=' '/rts=45;
run;
```

41

p303d04

Technique 3: Use the SASFILE Statement

The SASFILE global statement loads an entire SAS data set into memory for subsequent DATA and PROC steps. Loading the file only once reduces I/O.

```
SASFILE SAS-data-set OPEN|LOAD|CLOSE;
```



- ✍ This technique requires that the data set fit entirely into memory.

42

OPEN

Opens the file and allocates the buffers, but defers reading the data into memory until a procedure or a statement that references the file is executed.

LOAD

Opens the file, allocates the buffers, and reads the data into memory.

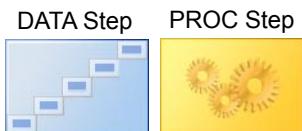
CLOSE

Releases the buffers and closes the file.

Technique 3: Use the SASFILE Statement

Two SASFILE statements are used together. The first allocates buffers and loads the SAS data set into memory in its entirety. The in-memory version is used when the file is requested.

```
SASFILE SAS-data-set LOAD;
```



```
SASFILE SAS-data-set CLOSE;
```

The second SASFILE statement closes the data set and releases the SAS buffers.

43

Technique 3: Use the SASFILE Statement

```
sasfile orion.customerdim load;
proc freq data=orion.customerdim;
  tables CustomerCountry CustomerType;
run;
proc print data=orion.customerdim noobs;
  where CustomerType='Orion Club Gold members high active';
  var CustomerID CustomerName CustomerAgeGroup;
run;
proc means data=orion.customerdim mean median max min;
  var CustomerAge;
  class CustomerGroup;
run;
proc tabulate data=orion.customerdim format=8.;
  class CustomerAgeGroup CustomerType;
  table CustomerType All>Total;
  Cus*SASFILE SAS-data-set CLOSE; *n=1 /rts=45;
run;
sasfile orion.customerdim close;
```



44

p303d04

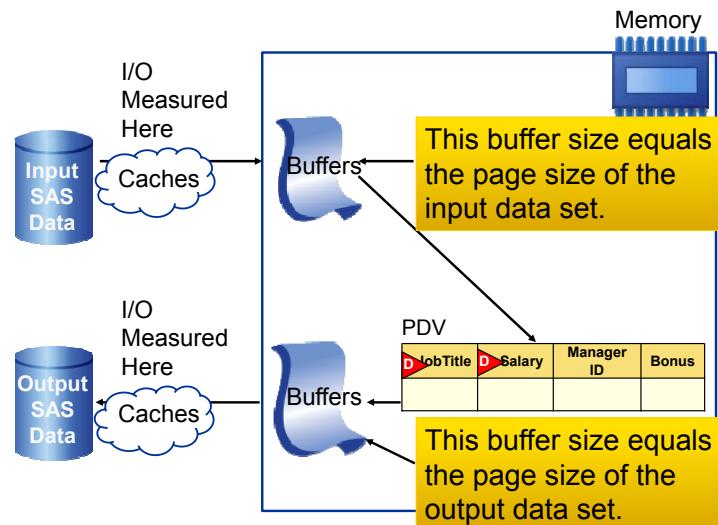
3.03 Multiple Answer Poll

Which of the following does the SASFILE LOAD statement do?

- a. loads the specified data set into memory
- b. defers reading the data into memory until a procedure or statement is executed
- c. closes the data set and releases the SAS buffers
- d. allocates buffers

45

Technique 4: Use BUFSIZE= and BUFNO=



48

Technique 4: Use BUFSIZE= and BUFNO=

Increasing buffer size can decrease I/O because more data can be processed for each I/O operation.

- As a data set option, BUFSIZE= controls the output buffer size for a **specific** data set.

```
SAS-data-set (BUFSIZE=n | nK | nM | nG | nT | hexX | MIN | MAX)
```

- As a system option, BUFSIZE= controls the output buffer size for **all** data sets.

```
OPTIONS BUFSIZE=n | nK | nM | nG | nT | hexX | MIN | MAX;
```

49

Technique 4: Use BUFSIZE= and BUFNO=

Increasing the number of buffers can decrease I/O because more data can be processed for each I/O operation.

- As a data set option, BUFNO= controls the number of buffers that are available for **specific** data sets.

SAS-data-set (BUFNO=n)

- As a system option, BUFNO= controls the number of buffers that are available for **all** data sets.

OPTIONS BUFNO=n;

50

Technique 4: Use BUFSIZE= and BUFNO=

The reduction in I/O comes at the cost of increased memory consumption.

```
data prices(bufsize=2K bufno=4);
  infile "&path/prices.dat" dlm='*';
  input ProductID:12. StartDate:date9.
    EndDate:date9.
    UnitCostPrice:dollar7.2
    UnitSalesPrice:dollar7.2; bytes per I/O
run;
```

-  The product of BUFNO= and BUFSIZE= determines how much data can be transferred in a single I/O operation.

51

p303d05



Exercises

Level 1

1. Using the SASFILE Statement

- Open the program **p303e01** and review the code.

Partial p303e01

```

proc freq data=orion.customerdim;
  tables customertype;
run;

proc freq data=orion.orderfact;
  tables ordertype;
run;

proc freq data=orion.productdim;
  tables productcategory;
run;

proc freq data=orion.employeepayroll;
  tables employeegender;
run;

```

- b. Modify the program by adding a SASFILE statement that opens and allocates buffers for **orion.employeepayroll**, but does not load it into memory. Answer the following questions:
- 1) At what point in the program is **orion.employeepayroll** loaded into memory? _____
 - 2) Is PROC MEANS reading the data from disk or memory? _____
 - c. Execute the program. Was the PROC SORT step successful? _____
 - d. Correct the PROC SORT step by adding the appropriate option to write the data to a temporary file named **work.employeepayroll**.
 - e. Write a SASFILE statement that closes the file and releases the buffers.

Level 2**2. Using Multiple SASFILE Statements**

- a. Open the program **p303e02** and submit it.

p303e02

```

options fullstimer;
proc sql;
  select EmployeeName,
    sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalContribution,
    Recipients
  from orion.employeeaddresses as a,
    orion.employeedonations as d
  where a.EmployeeID=d.EmployeeID;
quit;
options nofullstimer;

```

- b. Add the appropriate statement or statements to open and load both the **orion.employeeaddresses** and **orion.employeedonations** data sets into memory. At the end of the program, unload both data sets from memory.
- c. Submit the revised program and check the log to make sure that it ran successfully.

Challenge

3. Using the APPEND Procedure with the SASFILE Statement

- a. Open the program **p303e03**, which contains a PROC COPY step. Copy the data sets **orion.sales** and **orion.nonsales** to your **Work** library so that you can maintain the integrity for those original data sets. Submit the program.

p303e03

```
proc copy in=orion out=work;
  select sales nonsales;
run;
```

- b. Add the appropriate statement or statements to open and load the entire **work.sales** data set into memory, a PROC APPEND step to append the temporary **work.nonsales** data set to the temporary **work.sales** data set, and a PROC PRINT step to print observations 164 through 168 of the enlarged **work.sales** data set. Print only the variables **EmployeeID**, **FirstName**, **LastName**, and **JobTitle**. At the end of the program, unload the **work.sales** data set.
- c. Submit the revised program and examine the SAS log and output.

PROC PRINT Output

Work.Sales Observations 164-168				
Obs	EmployeeID	First Name	Last Name	Job Title
164	121144	Renee	Capachietti	Sales Manager
165	121145	Dennis	Lansberry	Sales Manager
166	120101	Patrick	Lu	Director
167	120104	Kareen	Billington	Administration Manager
168	120105	Liz	Povey	Secretary I

3.3 Reducing the Length of Numeric Variables

Objectives

- List techniques to reduce data set size.
- Describe how SAS stores numeric values.
- Describe how to safely reduce the space that is required to store numeric values in SAS data sets.

55

Business Scenario

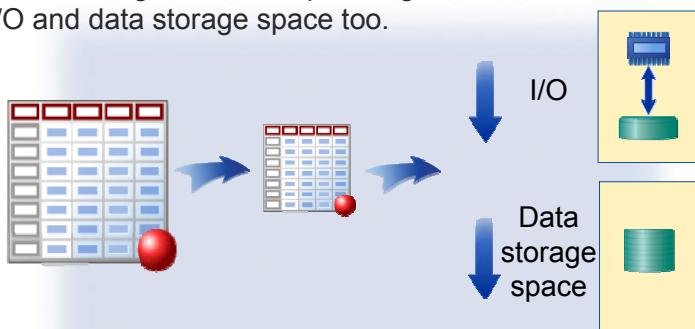
Human Resources needs to improve program performance and conserve resources.

- ✓ reducing the amount of data processed
- reducing the length of numeric variables
- compressing data files
- using SAS views

56

Techniques to Reduce Data File Size

It is not always possible to remove variables or observations to reduce file size. However, reducing variable lengths and compressing data files reduces I/O and data storage space too.



57

Techniques to Reduce Data File Size

Integers only
Reduce the length of numeric variables.

Compress the data set.

Reduce data.

58

How Variables Are Stored

SAS variables are stored as either character or numeric.

ProductName



ProductID



59

How Variables Are Stored

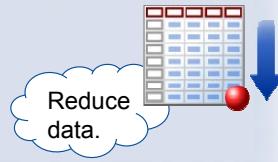
Character data is stored as one character per byte.

ProductName



= tent

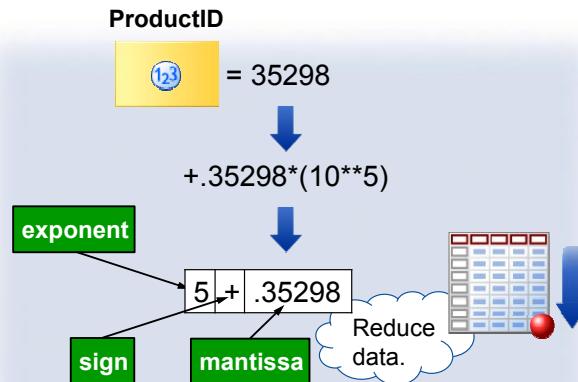
t | e | n | t



60

How Variables Are Stored

Numeric data is stored as floating-point numbers in real binary representation.



61

How Variables Are Stored

Summary of floating-point numbers stored in 8 bytes

Representation	Base	Exponent Bits	Maximum Mantissa Bits
IBM Mainframe	16	7	56
IEEE (UNIX, Windows, Open VMS Alpha)	2	11	52

- ✍ The sign occupies one bit.

62

Reducing the Length of Numeric Variables

Reduce the length of numeric variables with a LENGTH statement.

```
LENGTH variable(s) <$> length;
```

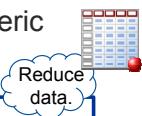


Advantages	Disadvantages
Data storage space ↓	CPU time ↑
I/O ↓	Can alter large or high-precision values

63

Reducing the Length of Numeric Variables

The reduced length applies to the output DATA set. Determining an appropriate length for numeric variables is more involved.



```
data empsshort;
length StreetID 6
      EmployeeID ManagerID
      StreetNumber EmployeeHireDate
      EmployeeTermDate BirthDate 4
      Dependents 3;
merge employeeaddresses
      employeeorganization
      employeepayroll
      employeephones;
by EmployeeID;
run;
```

p303d06

64

Reducing the Length of Numeric Variables

Numeric Variables: Windows and UNIX

Length in Bytes	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992



65

Reducing the Length of Numeric Variables

Numeric Variables: z/OS

Length in Bytes	Largest Integer Represented Exactly
2	256
3	65,536
4	16,777,216
5	4,294,967,296
6	1,099,511,627,776
7	281,474,946,710,656
8	72,057,594,037,927,936



66

Reducing the Length of Numeric Variables

StreetID is a 10-digit integer.

```
data empsshort;
length StreetID 6
EmployeeID ManagerID
StreetNumber EmployeeHireDate
EmployeeTermDate BirthDate 4
Dependents 3;
```



Length in Bytes	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

p303d06

67

Reducing the Length of Numeric Variables

These variables can be up to six digits.

```
data empsshort;
length StreetID 6
EmployeeID ManagerID
StreetNumber EmployeeHireDate
EmployeeTermDate BirthDate 4
Dependents 3;
```



Length in Bytes	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

p303d06

68

Reducing the Length of Numeric Variables

Date variables typically fit into four bytes.

```
data empsshort;
  length StreetID 6
    EmployeeID ManagerID
    StreetNumber EmployeeHireDate
    EmployeeTermDate BirthDate 4
    Dependents 3;
    Calendar Date
```

The diagram illustrates the range of SAS Date values. A horizontal axis at the bottom is labeled "SAS Date Value" and shows three points: -365, 0, and 365,243. Above this, another horizontal axis is labeled "Calendar Date" and shows three points: 01JAN1959, 01JAN1960, and 01JAN2960. Arrows point from the numerical values to their corresponding dates. A callout bubble says "Reduce data." pointing to the code.

69

p303d06

Reducing the Length of Numeric Variables

Number of dependents most likely does not exceed 8,192.

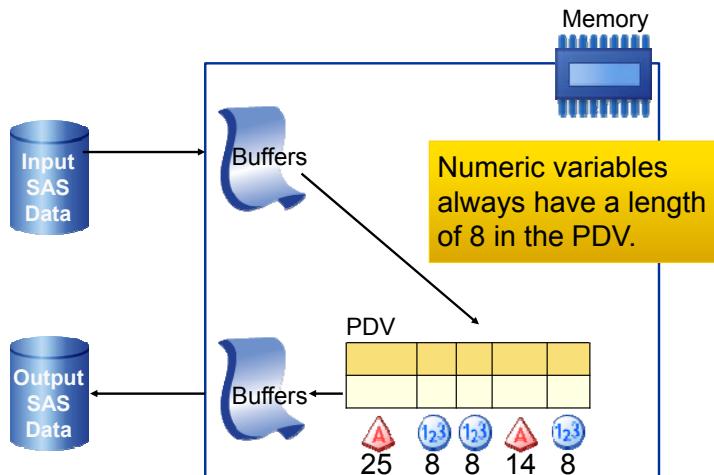
```
data empsshort;
  length StreetID 6
    EmployeeID ManagerID
    StreetNumber EmployeeHireDate
    EmployeeTermDate BirthDate 4
    Dependents 3;
```

Length in Bytes	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

70

p303d06

Reducing the Length of Numeric Variables

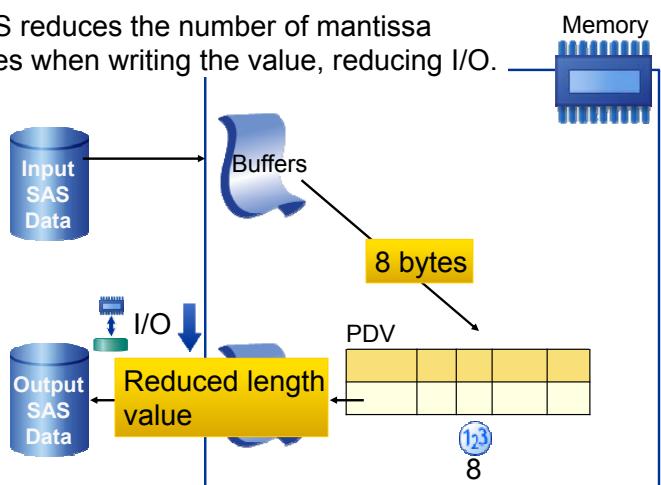


71

...

Reducing the Length of Numeric Variables

SAS reduces the number of mantissa bytes when writing the value, reducing I/O.

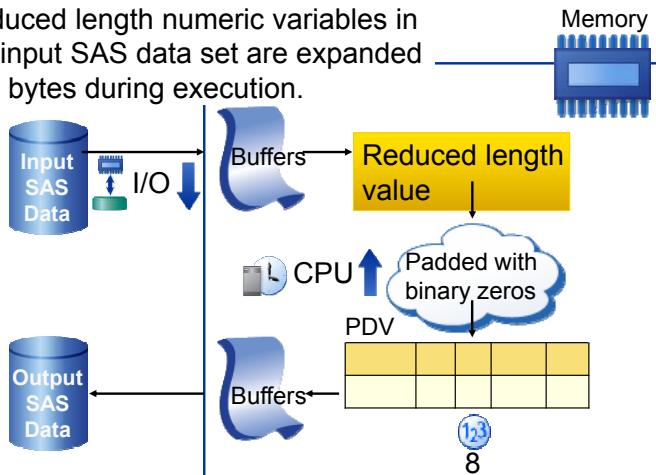


72

...

Reducing the Length of Numeric Variables

Reduced length numeric variables in the input SAS data set are expanded to 8 bytes during execution.



73

Idea Exchange

A data set contains 200 numeric variables with integer values ranging from 1 to 10. If you reduce the length of these numeric variables to the smallest appropriate byte size, how many bytes are you saving per observation? How many bytes are saved per 1,000 observations?



74

Reducing the Length of Numeric Variables

To decrease the length of all numeric variables, use the DEFAULT= option in the LENGTH statement.



Reduce
data.

```
data empsshort;
  length default=4;
  <additional SAS code>
run;
```

76

Dangers of Reducing the Length of Numeric Variables: Nonintegers



```
data test;
  length x 4;
  x=1/10;
  y=1/10;
  put x=;
  put y=;
run;

data _null_;
  set test;
  put x=;
  put y=;
run;
```

77

p303a01

3.04 Poll

Open and submit the program **p303a01**.

Examine the log.

After the second DATA step executes, are the values of X and Y equal?

- Yes
- No

78

Dangers of Reducing the Length of Nonintegers

During the first DATA step, the PUT statements write values while they are in the PDV, before length is reduced. X and Y are equal at this time.

Partial SAS Log

```

177  data test;
178  length x 4;
179  x=1/10;
180  y=1/10;
181  put x=;
182  put y=;
183  run;
x=0.1
y=0.1
NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

```

80

p303a01

Dangers of Reducing the Length of Nonintegers

During the second DATA step, padding the variable X with binary zeros when it is read into the PDV modifies the value.

Partial SAS Log

```
172 data _null_;
173   set test;
174   put x=;
175   put y=;
176 run;
x=0.0999999642
y=0.1
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
```

81

p303a01



SAS procedures also expand numbers to 8 bytes in memory.

Dangers of Reducing the Length Too Much

```
data test;
  length x 3;
  x=72345;
  put x=;
run;

data _null_;
  set test;
  put x=;
run;
```

A length of 3 is not large enough to precisely store this value.

Partial SAS Log

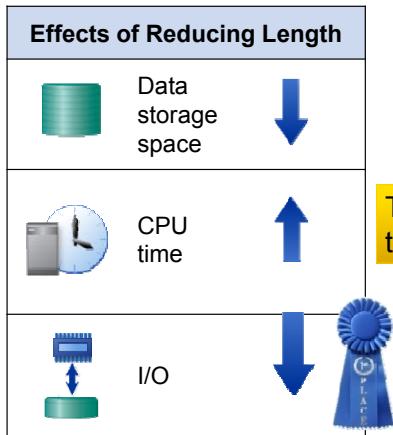
```
45 data _null_;
46   set test;
47   put x=;
48 run;
x=72336
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

Length in Bytes	Largest Integer Represented Exactly
3	8,192

82

p303d07

Reduced Length Numeric Variables Trade-Offs



The savings in I/O outweigh the cost of extra CPU time.

83



Exercises

Level 1

4. Creating Reduced Length Numerics

Purchase data is stored in the following three data sets whose names indicate the purchase methods: **orion.internet**, **orion.retail**, and **orion.catalog**.

- Use PROC CONTENTS to determine the names and types of the variables in each of the purchase data sets.
- Open the program **p303e04**. Edit the DATA step that creates **allcustomers** to appropriately reduce the length of the numeric variables based on their values.



Do not reduce the length of the variables **TotalRetailPrice**, **CostPricePerUnit**, or **Discount**. They are not integer values.

p303e04

```
data allcustomers;
  set orion.catalog orion.internet orion.retail;
run;
```

- Submit the altered program and examine the SAS log.
- Use PROC CONTENTS to check the length of the numeric variables in **allcustomers**.

Level 2

5. Creating Reduced Length Numerics and Precision

- Open the program **p303e05** and submit it.

p303e05

```
data five;
  length Num5 5 Num8 8;
  do Num8=1e10 to 1e13 by 1e11;
    Num5=Num8;
    output;
  end;
run;

proc print data=five;
  title 'Reducing the length of numeric data to 5';
  format Num5 Num8 20. ;
run;
```

- At what point in the sequence of numbers does the length of the variable **Num5** lose precision?
-

Challenge

6. Determining the Minimum Number of Bytes for Reduced Length Numerics

The program below from the Help facility (**p303e06**) finds the minimum length of bytes (**MinLen**) that are needed for numbers stored in a native SAS data set named **numbers**. The **numbers** data set contains the variable **Value**. The variable **Value** contains a range of numbers. In this example, the range is 8191 to 8194 for Windows and UNIX, and 269 to 272 for z/OS.

p303e06

```
/* Windows and UNIX DATA step */
data numbers;
  input Value;
  datalines;
8191
8192
8193
8194
;

/* z/OS DATA step */
/*
data numbers;
  input Value;
  datalines;
269
270
271
```

```

272
;
*/
data temp;
  set numbers;
  X=Value;
  do L=8 to 1 by -1;
    if X NE trunc(X,L) then
      do;
        MinLen=L+1;
        output;
        return;
      end;
    end;
  run;

title;
proc print data=temp noobs;
  var Value MinLen;
run;

```

- a. Run the program that is stored in **p303e06** and examine the output.

Windows and UNIX Output

	Value	Min Len
	8191	3
	8192	3
	8193	4
	8194	3

- b. Investigate the Help facility to determine why the minimum length for the number 8194 is less than that of the number 8193 (Windows and UNIX) or why 272 is less than that of the number 271 (z/OS). The information can be found by following this path:

SAS Products \Rightarrow **Base SAS** \Rightarrow **SAS 9.4 Language References: Concepts** \Rightarrow **SAS System Concepts** \Rightarrow **SAS Variables** \Rightarrow **Numeric Precision in SAS Software**

3.4 Compressing SAS Data Sets

Objectives

- Define the structure of a compressed SAS data file.
- Create a compressed SAS data file.
- List the advantages and disadvantages of compression.

87

Business Scenario

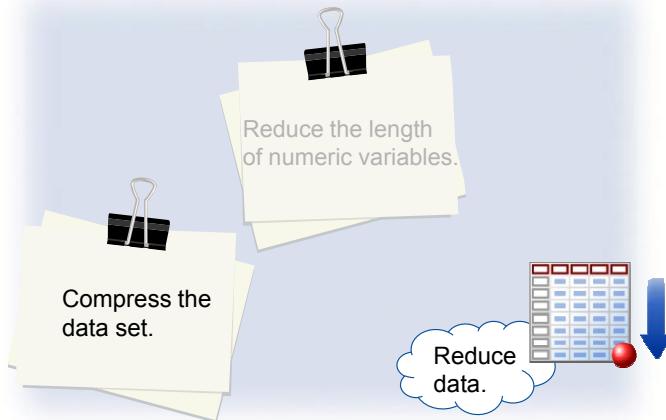
Human Resources needs to improve program performance and conserve resources.

- ✓ reducing the amount of data processed
- ✓ reducing the length of numeric variables
- ➡ compressing data files
- using SAS views

88

Techniques to Reduce Data File Size

SAS data compression can solve both storage space and I/O concerns.



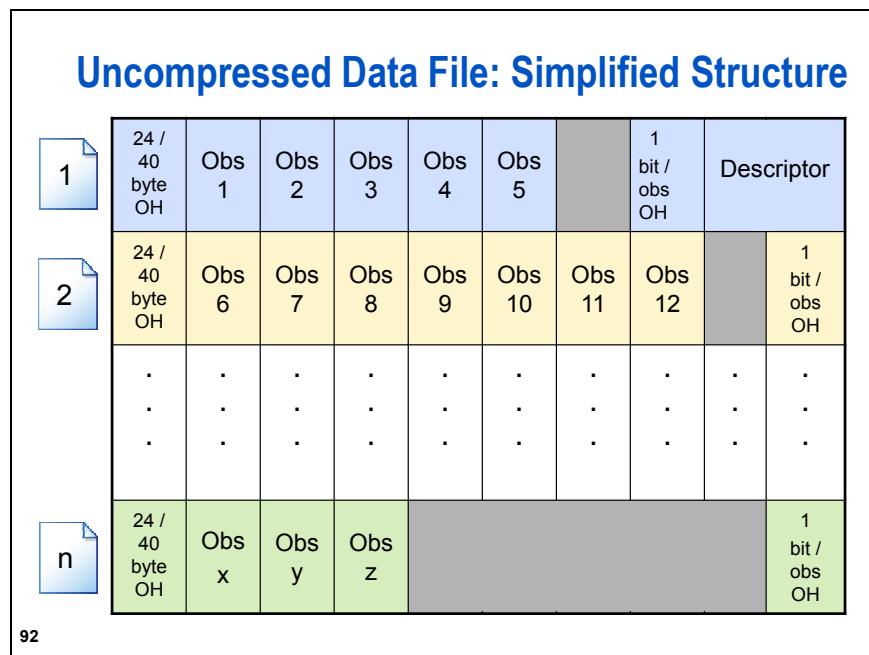
89

3.05 Poll

By default, the observations in a SAS data file have varying lengths.

- Yes
- No

90

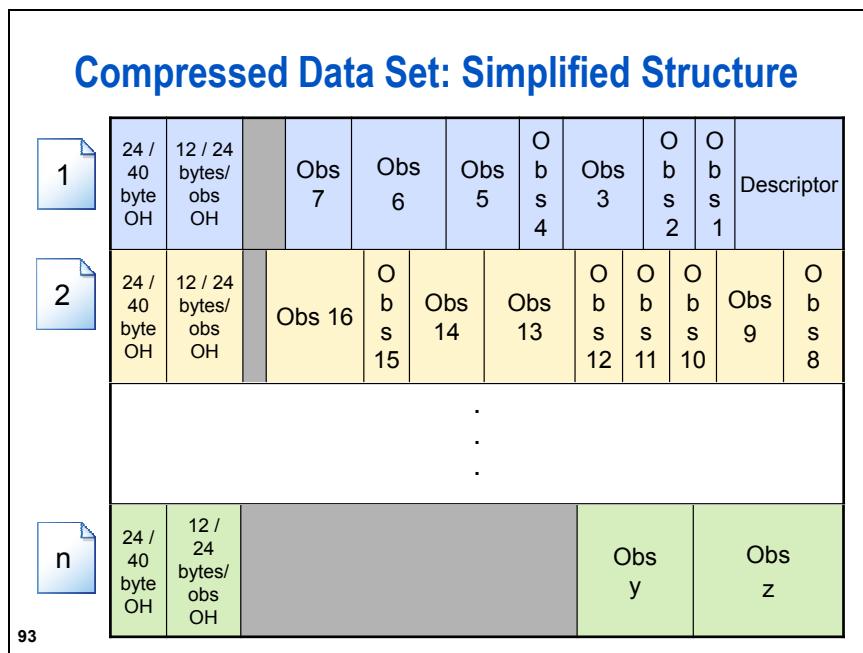


This is a visualization tool to help you understand how SAS data files are structured. SAS data files are not actually stored in exactly this manner.

The following are features of uncompressed SAS data files:

- All observations use the same number of bytes.
- Each variable occupies the same number of bytes in every observation.
- Character values are padded with blanks.
- Numeric values are padded with binary zeros.
- The descriptor portion of the data set uses part of the first data set page.
- There is a 24-byte overhead at the beginning of each page on 32-bit systems.
- There is a 40-byte overhead at the beginning of each page on 64-bit systems.
- There is a 1-bit per observation overhead, rounded up to the nearest byte.
- New observations are added at the end of the file. If a new page is needed for a new observation, a whole data set page is added.
- Deleted observation space is never reused, unless the entire data file is rebuilt.

In an uncompressed SAS data file, each observation is a fixed-length record.



This is a visual depiction of the storage used for a compressed SAS data file.

The following are features of compressed SAS data files:

- Each observation is a single string of bytes. Variable types and boundaries are ignored.
- Each observation can have a different length.
- Consecutive, repeating characters and numbers are collapsed into fewer bytes.
- If an updated observation is larger than its original size, it is stored on either the same data set page or on a different page with a pointer to the original page.
- The descriptor portion of the data set is stored at the end of the first data set page.
- There is a 24-byte overhead at the beginning of each page on 32-bit systems.
- There is a 40-byte overhead at the beginning of each page on 64-bit systems.
- There is a 12-byte-per-observation overhead on 32-bit systems.
- There is a 24-byte-per-observation overhead on 64-bit systems.
- Deleted observation space can be reused if the REUSE=YES data set or system option was turned on when the SAS data file was compressed.

SAS data files, but not views, can be stored in compressed form.

Compressing a file reduces the number of bytes that are required to represent each observation. In a compressed file, each observation is a variable-length record.

Creating Compressed SAS Data Files

There are two standard compression algorithms.
The optimal algorithm depends on the data.

- ✓ Character data with repeated characters



RLE (Run-Length Encoding)

COMPRESS=CHAR | YES

- ✓ Numeric data

- ✓ Character data with patterns

- ✓ Observations > 1000 bytes



RDC (Ross Data Compression)

COMPRESS=BINARY



CPU Time

94

Creating Compressed SAS Data Files

As a data set option, COMPRESS= compresses only the associated output data set.

```
SAS-data-set(COMPRESS=NO | YES | CHAR | BINARY);
```

As a system option, COMPRESS= compresses **all** output data sets that are created during the SAS session.

```
OPTIONS COMPRESS=NO | YES | CHAR | BINARY;
```

95

p303d08

Creating Compressed SAS Data Files

```
data empchar(compress=char pointobs=yes);
  merge employee_ SAS-data-set (POINTOBS=YES | NO)
        employee_
        employee_
        employee_phones;
  by Employee_ID;
run;
```

`POINTOBS=YES` enables SAS to process a compressed data set with random access. This is the default setting. It might be beneficial to set `POINTOBS=NO` to sequentially process a compressed file.

96

p303d09

How SAS Compresses Data

In uncompressed form, all observations use 35 bytes for these two variables.

Name	Type	Length
LastName	character	20
FirstName	character	15

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	
A	D	A	M	S												B	I	L	L						
LastName																									
					FirstName																				
Last																									
					First																				
Name																									
					Last																				
					First																				
Name																									

97

How SAS Compresses Data

With RLE compression, only 13 bytes are needed for this observation.

Name	Type	Length
LastName	character	20
FirstName	character	15



COMPRESS=CHAR

LastName FirstName

- @ indicates how many uncompressed characters follow.
 - # indicates the number of blanks repeated at this point in the observation.

98

How SAS Compresses Data

In uncompressed form, every observation uses 1600 bytes for these 200 numeric variables.

Name	Type	Length
Answer1	numeric	8
...		
Answer200	numeric	8

repeated values

pattern of values
(1-2-1-2-1-2)

Answer1	Answer2	Answer3	Answer4	Answer5	Answer6	...	Answer200
1	2	1	2	1	2		2
1	1	1	1	1	1		1
2	2	2	2	2	2		2

101

How SAS Compresses Data

 Ross Data Compression

1 2 3 4 5 6 7 8 9

@	+1	1	#	@	+1	2	#	%
---	----	---	---	---	----	---	---	---

1600 bytes
9 or 11 bytes

COMPRESS=BINARY

Answer1	Answer2	Answer3	Answer4	Answer5	Answer6	Answer200
1	2	1	2	1	2	2
...						



102

- @ indicates how many uncompressed characters follow.
- +1 indicates the sign and exponent.
- # indicates the number of binary zeros repeated at this point in the observation.
- % indicates how many times these values are repeated.

Dependencies and Trade-Offs in Compression

Compressing SAS data files is **always** an experiment.

Some SAS data files do **not** compress well.

- few repeated characters
- small physical size
- few missing values
- short text strings

SAS Log (Windows)

```

55 data orders(compress=binary);
56 set orion.orders;
57 run;

NOTE: There were 490 observations read from the data set ORION.ORDERS.
NOTE: The data set WORK.ORDERS has 490 observations and 6 variables.
NOTE: Compressing data set WORK.ORDERS increased size by 100.00 percent.
      Compressed is 2 pages; un-compressed would require 1 pages.
NOTE: DATA statement used (Total process time):
      real time          0.29 seconds
      cpu time           0.12 seconds
  
```

103 p303d10

Dependencies and Trade-Offs in Compression

Some SAS data files generally *do* compress well.

- many missing values
- many observations with few bytes in long character variables

SAS Log (Windows)

```
48 data CustDimComp(compress=binary);
49 set orion.Customerdimmore;
50 run;

NOTE: There were 1500 observations read from the data set
      ORION.CUSTOMERDIMMORE.
NOTE: The data set WORK.CUSTDIMCOMP has 1500 observations and 11 variables.
NOTE: Compressing data set WORK.CUSTDIMCOMP decreased size by 20.00 percent.
      Compressed is 4 pages; un-compressed would require 5 pages.
NOTE: DATA statement used (Total process time):
      real time          0.07 seconds
      cpu time           0.01 seconds
```

104

p303d10

Dependencies and Trade-Offs in Compression

If the maximum size of the observation is less than the overhead introduced by compression, then compression is disabled.

1	24 / 40 byte OH	12 / 24 bytes per obs OH		Obs 5	Obs 4	Obs 3	Obs 2	Obs 1	Descriptor
2	24 / 40 byte OH	12 / 24 bytes per obs OH		Obs 11	Obs 10	Obs 9	Obs 8	Obs 7	Obs 6
n	24 / 40 byte OH	12 / 24 bytes per obs OH							

```
18 data test(compress=yes);
19 x=1;
20 run;
```

NOTE: Compression was disabled for data set WORK.TEST because compression overhead would increase the size of the data set.

105

p303d11

- When you use the COMPRESS= option, SAS knows the size of the overhead introduced by compression and the maximum size of an observation

If the maximum size of the observation is less than the 12-byte (32-bit systems) or 24-byte (64-bit systems) overhead introduced by compression, SAS does the following:

- disables compression
- creates an uncompressed data set
- issues a note stating the file was not compressed

Compressing SAS Data Files

Resource Trade-Offs

Resource	Uncompressed	Compressed
 Data storage space	↑	↓
 CPU time	↓	↑
 I/O	↑	↓



106



Exercises

Level 1

7. Compressing SAS Data Files

- a. Open the program **p303e07**.

Partial p303e07

```
proc sort data=orion.employeeaddresses
           out=employeeaddresses;
   by EmployeeID;
run;
proc sort data=orion.employeeorganization
           out=employeeorganization;
   by EmployeeID;
run;
proc sort data=orion.employeepayroll out=employeepayroll;
   by EmployeeID;
run;
```

- b. After the DATA step, insert a PROC CONTENTS step to examine the descriptor portion of the **work.employees** file.

How many data set pages are needed for this file in its uncompressed form? _____

- c. Add the appropriate data set option to compress the **work.employees** file using Run-Length Encoding.

How many data set pages are needed for this file in its RLE compressed form? _____

What is the value of the COMPRESSED attribute in the output? _____

- d. Modify the data set option to compress the **work.employees** file using Ross Data Compression.

How many data set pages are needed for this file in its RDC compressed form? _____

What is the value of the COMPRESSED attribute in the output? _____

Which attributes of the file are added to the PROC CONTENTS output when compression is turned on?

Which attributes of the file are removed when compression is turned on?

Level 2

8. Compressing SAS Data Files

The following is the list of variables in the data set **orion.productlist**:

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
1	ProductID	Num	8	12.	Product ID
4	ProductLevel	Num	8	12.	Product Level
2	ProductName	Char	45		Product Name
5	ProductRefID	Num	8	12.	Product Reference ID
3	SupplierID	Num	8	12.	Supplier ID

The following is the list of variables in the data set **orion.supplier**:

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
6	Country	Char	2		Country
3	StreetID	Num	8	12.	Street ID
5	SupStreetNumber	Char	8		Supplier Street Number
4	SupplierAddress	Char	45		Supplier Address
1	SupplierID	Num	8	12.	Supplier ID
2	SupplierName	Char	30		Supplier Name

- a. If you merge these two files by **SupplierID** and compress the output data, which method of compression do you think is the most appropriate?

 If you use SAS Enterprise Guide, you need to scroll down in the **orion.supplier** data set to see the nonmissing values of **SupplierName**.

- b. Merge **orion.productlist** and **orion.supplier** by **SupplierID** to create a data set named **suppliernames1**. Compress the date set. Use the method that you predicted in part a.

 Sort the data by **SupplierID** to prepare for the merge.

- c. Merge **orion.productlist** and **orion.supplier** by **SupplierID** to create a data set named **suppliernames2**. Compress it and use the alternative method.

- 1) Which method was better? _____
- 2) Why was that method better? _____

 Your results might vary from the suggested solutions depending on the operating platform and method that you used to create the data on that platform.

3.5 Using SAS Views

Objectives

- Create a SAS DATA step view.
- Investigate when to use a SAS view.

Business Scenario

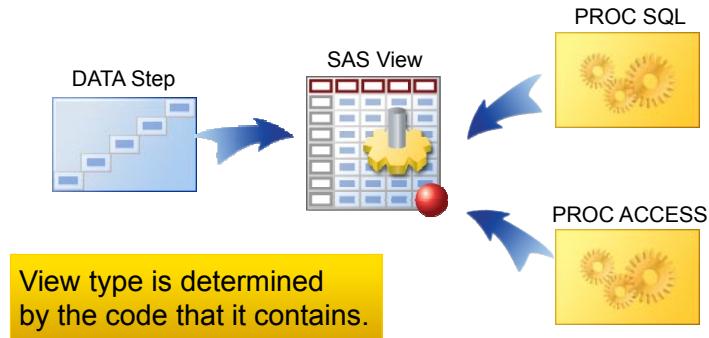
Human Resources needs to improve program performance and conserve resources.

- ✓ reducing the amount of data processed
- ✓ reducing the length of numeric variables
- ✓ compressing data files
- ➡ using SAS views

111

SAS Data Views

A view is a generalized database concept for a data file that contains a stored program and no data.



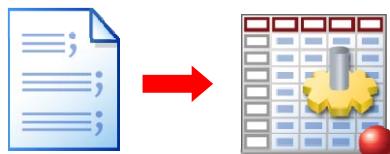
112

What Is a SAS View?

Views are dynamic, space efficient, and convenient.

A view has the following characteristics:

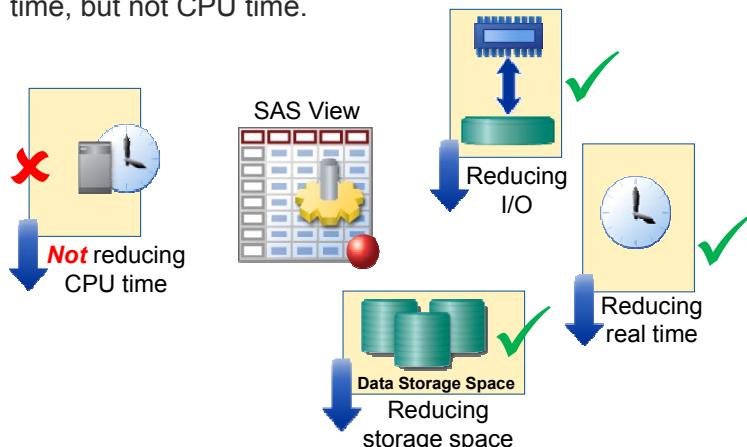
- is stored SAS code
- contains no actual data
- extracts underlying data each time it is used and accesses the most current data
- prevents users from modifying the code



113

SAS Views: Advantages

Using a view saves I/O, data storage space, and real time, but not CPU time.



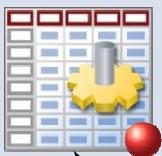
114

SAS Views: Advantages

The difference between SAS data files and SAS data views is in what is stored on disk.



data stored on disk



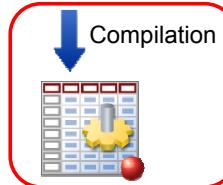
instructions stored on disk

115

SAS Views: Advantages

When you create a view, SAS compiles and stores the DATA step code as a view.

```
data orion.quarterv/view=orion.quarterv;
  infile MON dlm=',';
  <additional SAS statements>
run;
```



```
filename MON 'ext-file';
proc print data=orion.quarterv;
  <additional SAS statements>
run;
```



Execution

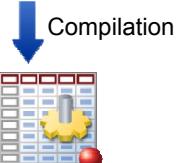
continued...

116

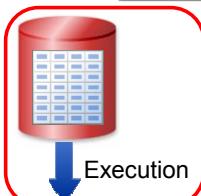
SAS Views: Advantages

The code executes only when you reference the view.

```
data orion.quarterv/view=orion.quarterv;
  infile MON dlm=',';
  <additional SAS statements>
run;
```



```
filename MON 'ext-file';
proc print data=orion.quarterv;
  <additional SAS statements>
run;
```



117

Creating a SAS DATA Step View

The view name must match one of the data set names.



```
data orion.quarterv / view=orion.quarterv;
  infile MON dlm=',';
  input CustomerID OrderID OrderType
    OrderDate : date9.
    DeliveryDate : date9..;
  time=time();
  format time ti
run;
```

```
DATA SAS-data-set(s) / VIEW=view-name;
  <INFILE fileref;>
  <INPUT variable(s);>
  <SET or MERGE statement(s);>
RUN;
```

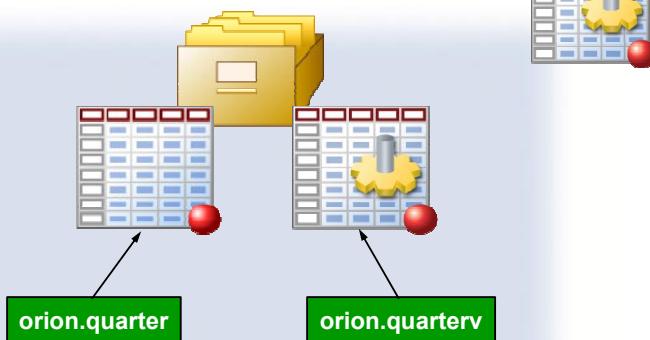
can contain any DATA step statements

p303d12

121

Using a SAS View

The view name cannot be the same as an existing table name in the same library.



122



Creating a SAS DATA View

p303d12

This demonstration illustrates how to generate a report based on data coming from three raw data files without creating a stored SAS data set.

1. Open the program **p303d12**.
2. **Part A:** Highlight and submit the FILENAME statement and DATA step that create the data file named **orion.quarter**. A new variable, **time**, is created and formatted to show the current time of day.

Partial p303d12

```

filename MON ("&path/mon3.dat"
              "&path/mon2.dat"
              "&path/mon1.dat"); * Windows and UNIX;
data orion.quarter;
  infile MON dlm=',';
  input CustomerID OrderID OrderType
        OrderDate : date9. DeliveryDate : date9. ;
  time=time();
  format time timeAMPM.;
run;

```

- a. Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```

73   filename MON ("&path\mon3.dat"
                     "&path\mon2.dat"
                     "&path\mon1.dat"); * Windows and UNIX;
76   data orion.quarter;
77     infile MON dlm=',';
78     input CustomerID OrderID OrderType
           OrderDate : date9. DeliveryDate : date9. ;
79     time=time();
80     format time timeAMPM.;
82   run;
NOTE: The infile MON is:
      Filename=S:\workshop\mon3.dat,
      File List=( 'S:\workshop\mon3.dat' 'S:\workshop\mon2.dat' 'S:\workshop\mon1.dat' ),
      RECFM=V,LRECL=32767

NOTE: The infile MON is:
      Filename=S:\workshop\mon2.dat,
      File List=( 'S:\workshop\mon3.dat' 'S:\workshop\mon2.dat' 'S:\workshop\mon1.dat' ),
      RECFM=V,LRECL=32767

NOTE: The infile MON is:
      Filename=S:\workshop\mon1.dat,
      File List=( 'S:\workshop\mon3.dat' 'S:\workshop\mon2.dat' 'S:\workshop\mon1.dat' ),
      RECFM=V,LRECL=32767

NOTE: 9 records were read from the infile MON.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: 7 records were read from the infile MON.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: 4 records were read from the infile MON.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: The data set ORION.QUARTER has 20 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          0.04 seconds
      cpu time          0.03 seconds

```

- b. Highlight and submit the PROC PRINT step.

Partial p303d12

```

proc print data=orion.quarter;
  title 'orion.quarter';
  format OrderDate DeliveryDate date9. ;
run;

```

- c. Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```

84   proc print data=orion.quarter;
85     title 'orion.quarter';
86     format OrderDate DeliveryDate date9. ;
87   run;

```

```
NOTE: There were 20 observations read from the data set ORION.QUARTER.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

- d. Review the PROC PRINT results in the Output window or on the Output tab. Focus on the new **time** variable.

Partial PROC PRINT Output (First five of 20 observations)

orion.quarter						
Obs	Customer ID	OrderID	Order Type	OrderDate	Delivery Date	time
1	4	1232410925	3	02MAR2011	03MAR2011	12:32:06 PM
2	4	1232455720	1	09MAR2011	09MAR2011	12:32:06 PM
3	19	1232478868	1	13MAR2011	13MAR2011	12:32:06 PM
4	70201	1232517885	3	19MAR2011	24MAR2011	12:32:06 PM
5	4	1232530384	3	20MAR2011	21MAR2011	12:32:06 PM

- e. Ask the question: If you resubmit the PROC PRINT step to re-create the report, do you think that the values of **time** change? The answer is **No**. You simply reprint a static data file.

Highlight and submit the PROC PRINT step a second time.

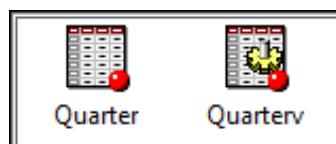
- f. Review the PROC PRINT results in the Output and Log windows. Verify that the **time** values did not change.

Partial PROC PRINT Output (First five of 20 observations)

orion.quarter						
Obs	Customer ID	OrderID	Order Type	OrderDate	Delivery Date	time
1	4	1232410925	3	02MAR2011	03MAR2011	12:32:06 PM
2	4	1232455720	1	09MAR2011	09MAR2011	12:32:06 PM
3	19	1232478868	1	13MAR2011	13MAR2011	12:32:06 PM
4	70201	1232517885	3	19MAR2011	24MAR2011	12:32:06 PM
5	4	1232530384	3	20MAR2011	21MAR2011	12:32:06 PM

3. **Part B:** Highlight and submit the DATA step that creates the data view named **orion.quarterv**. The Explorer window shows the **orion.quarter** data file and the **orion.quarterv** view. Notice that the view has a slightly different icon.

```
data orion.quarterv/view=orion.quarterv;
  infile MON dlm=',';
  input CustomerID OrderID OrderType
        OrderDate : date9. DeliveryDate : date9.;
  time=time();
  format time timeAMPM.;
run;
```



- a. Examine the SAS log. Focus on the DATA step that created the view. Notice that there are no execution time messages regarding number of observations and variables. The view contains the stored, compiled DATA step code but no data.

The stored code did not execute when the view was created. How do you cause the code to execute?

 Benefit: Because views do not store data, they occupy very little space on disk. They are extremely space efficient.

- b. Highlight and submit the PROC PRINT step.

Partial p303d12

```
proc print data=orion.quarterv;
  title 'orion.quarterv view';
  format OrderDate DeliveryDate date9. ;
run;
```

- c. A reference to the view in a SAS program causes the view to execute. Examine the SAS log. Focus on the PROC PRINT step that read the view to generate the report. Notice that all of the execution time messages regarding data being read and observations being created now appear in the log after the PROC PRINT step.

Partial SAS Log

```
90  proc print data=orion.quarterv;
91    title 'work.quarter view';
92    format OrderDate DeliveryDate date9. ;
93  run;
```

NOTE: The infile MON is:
 Filename=S:\workshop\mon3.dat,

 File List=('S:\workshop\mon3.dat'
 'S:\workshop\mon2.dat'
 'S:\workshop\mon1.dat'),
 RECFM=V,LRECL=32767

NOTE: The infile MON is:
 Filename=S:\workshop\mon2.dat,

 File List=('S:\workshop\mon3.dat'
 'S:\workshop\mon2.dat'
 'S:\workshop\mon1.dat'),
 RECFM=V,LRECL=32767

NOTE: The infile MON is:
 Filename=S:\workshop\mon1.dat,

 File List=('S:\workshop\mon3.dat'
 'S:\workshop\mon2.dat'
 'S:\workshop\mon1.dat'),
 RECFM=V,LRECL=32767

NOTE: 9 records were read from the infile MON.
 The minimum record length was 80.
 The maximum record length was 80.

```

NOTE: 7 records were read from the infile MON.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: 4 records were read from the infile MON.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: View ORION.QUARTERV.VIEW used (Total process time):
      real time          0.57 seconds
      cpu time           0.00 seconds

NOTE: There were 20 observations read from the data set ORION.QUARTERV.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.60 seconds
      cpu time           0.01 seconds

```

- d. Notice that the view delivers data to PROC PRINT, in the same way as the DATA step. Notice that the values of **time** are updated.

Partial PROC PRINT Output (First five of 20 observations)

orion.quarterv view						
Obs	Customer ID	OrderID	Order Type	OrderDate	Delivery Date	time
1	4	1232410925	3	02MAR2011	03MAR2011	12:34:12 PM
2	4	1232455720	1	09MAR2011	09MAR2011	12:34:12 PM
3	19	1232478868	1	13MAR2011	13MAR2011	12:34:12 PM
4	70201	1232517885	3	19MAR2011	24MAR2011	12:34:12 PM
5	4	1232530384	3	20MAR2011	21MAR2011	12:34:12 PM

- e. Ask the following question: If you resubmit the PROC PRINT step to re-create the report, do you think the values of **time** change? The answer is **Yes**. You re-execute the stored view code that contains a compiled DATA step when you resubmit PROC PRINT.

Highlight and submit the PROC PRINT step a second time.

- f. Review the PROC PRINT results in the Output window and verify that the **time** values changed. Why do they change?

Partial PROC PRINT Output (First five of 20 observations)

orion.quarterv view						
Obs	Customer ID	OrderID	Order Type	OrderDate	Delivery Date	time
1	4	1232410925	3	02MAR2011	03MAR2011	12:35:14 PM
2	4	1232455720	1	09MAR2011	09MAR2011	12:35:14 PM
3	19	1232478868	1	13MAR2011	13MAR2011	12:35:14 PM
4	70201	1232517885	3	19MAR2011	24MAR2011	12:35:14 PM
5	4	1232530384	3	20MAR2011	21MAR2011	12:35:14 PM



Benefit: Views are dynamic. They deliver data in real time.

4. **Part C:** Highlight and submit the PROC CONTENTS step.

Partial p303d12

```

proc contents data=orion.quarterv;
run;

```

- a. Review the PROC CONTENTS results in the Output window. Notice that SAS considers the view a data set with a member type of *VIEW*. The engine name of SASDSV is an acronym for SAS DATA step view.

Notice also that the view has variables but no observations. If the view does not store observations, what does it store?

Partial PROC CONTENTS Output

The CONTENTS Procedure		
Data Set Name	ORION.QUARTERV	Observations .
Member Type	VIEW	Variables 6
Engine	SASDSV	Indexes 0
Created	02/17/2014 13:23:49	Observation Length 48
Last Modified	02/17/2014 13:23:49	Deleted Observations 0
Protection		Compressed NO

3.06 Multiple Choice Poll

Which of the following are advantages of using a DATA step view as opposed to using a static table?

- a. storing complex code for reuse
- b. reducing programming errors
- c. accessing the most current data in changing files
- d. avoiding storing a SAS copy of a large data file
- e. all of the above

Using a SAS View

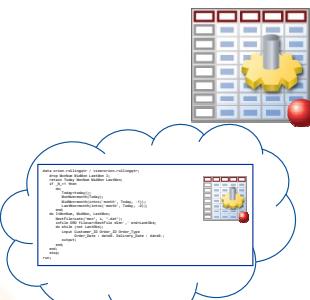
After a view is created, how do you know what code it contains?

orion.quarterv			
Obs	CustomerID	OrderID	OrderType
1	4	1232410925	3
2	4	1232455720	1
3	19	1232478868	1
4	70201	1232517885	3
5		1232530384	3
6		1232530393	1

SAS - [VIEWTABLE:Orion.Quarterv]

CustomerID	OrderID	OrderType
1	4	1232410925
2	4	1232455720
3	19	1232478868
4	70201	1232517885
5	4	1232530384
6	49	1232530393

126



Using a SAS View

Use the DESCRIBE statement to retrieve program source code from a DATA step view.

```
data view=orion.quarterv;
  describe;
run;
```

```
DATA VIEW=view-name;
  DESCRIBE;
  RUN;
```



Partial SAS Log

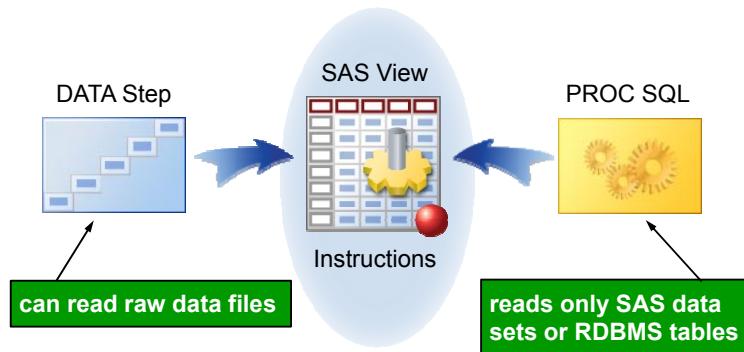
```
NOTE: DATA step view ORION.QUARTERV is defined as:
data orion.quarterv / view=orion.quarterv;
  infile MON dlm=',';
  input CustomerID OrderID OrderType OrderDate : date9.
    DeliveryDate : date9.;
  time=time();
  format time timeAMPM.;
```

127

p303d12

Creating Views Using PROC SQL

Views can also be created using PROC SQL.



128

Creating Views Using PROC SQL

Create PROC SQL views using the CREATE VIEW statement.

```
proc sql;
  create view orion.namesview as
    select e.EmployeeID, e.EmployeeName,
           s.ManagerID,
           m.EmployeeName as ManagerName
      from orion.staff as s,
           orion.employeeaddresses as e,
           orion.employeeaddresses as m
     where e.EmployeeID=s.EmployeeID
       and m.EmployeeID=s.ManagerID;
quit;
```

CREATE VIEW PROC-SQL-view AS query expression;

129

p303d13

Creating Views Using PROC SQL

Embed a LIBNAME statement in a PROC SQL view by specifying the USING clause.

```
proc sql;
  create view orion.namesview as
    select e.EmployeeID, e.EmployeeName,
           s.ManagerID,
           m.EmployeeName as ManagerName
      from orion.staff as s,
           orion.employeeaddresses as e,
           orion.employeeaddresses as m
     where e.EmployeeID=s.EmployeeID
       and m.EmployeeID=s.ManagerID
   using libname orion "&path";
quit;
  USING LIBNAME-clause;
```



130

p303d14



Creating and Using a PROC SQL View

p303d14

This demonstration illustrates creating and using a PROC SQL view.

1. Open the program **p303d14**.
2. Highlight and submit the first PROC SQL step to create the SQL view named **orion.namesview**.

Partial **p303d14**

```
proc sql;
  create view orion.namesview as
    select e.EmployeeID,
           e.EmployeeName,
           s.ManagerID,
           m.EmployeeName as ManagerName
      from orion.staff,
           orion.employeeaddresses as e,
           orion.employeeaddresses as m
     where e.EmployeeID=staff.EmployeeID
       and m.EmployeeID=staff.ManagerID
   using libname orion "&path";
quit;
```

3. Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```

79  proc sql;
80      create view orion.namesview as
81          select e.EmployeeID,
82                  e.EmployeeName,
83                  s.ManagerID,
84                  m.EmployeeName as ManagerName
85          from orion.staff,
86                  orion.employeeaddresses as e,
87                  orion.employeeaddresses as m
88          where e.EmployeeID=staff.EmployeeID
89          and m.EmployeeID=staff.ManagerID
90          using libname orion "&path";
91  quit;
NOTE: SQL view ORION.NAMESVIEW has been defined.
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.32 seconds
      cpu time           0.04 seconds

```

- Highlight and submit the second PROC SQL step that uses the view to create a report of Fred Benyami's employees.

Partial p303d14

```

proc sql;
title "Fred Benyami's Employees";
select EmployeeID,
       EmployeeName
  from orion.namesview
 where ManagerName='Benyami, Fred';
quit;

```

- Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```

134  proc sql;
135  title "Fred Benyami's Employees";
136  select EmployeeID,
137          EmployeeName
138  from orion.namesview
139  where ManagerName='Benyami, Fred';
140  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.14 seconds
      cpu time           0.04 seconds

```

- Review the SELECT statement results in the Output window or on the Output tab.

SELECT Statement Output

Fred Benyami's Employees	
EmployeeID	EmployeeName
120794	Cross, Samantha
120798	Ardskin, Elizabeth
120799	Stefandonovan, Jeffery
120814	Scroggin, Victor

Avoid Intermediate Copies of Data

What is the advantage of this SAS program?

```
data bonusview(keep=ManagerID YrEndBonus)
  / view=bonusview;
  set orion.staff;
  YrEndBonus=Salary * 0.05;
  where JobTitle contains 'Manager';
run;

proc means data=bonusview mean sum;
  class ManagerID;
  var YrEndBonus;
run;
```

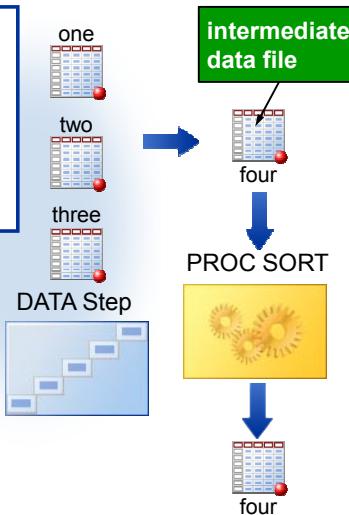
132

p303d15

Avoid Intermediate Copies of Data

```
data four;
  set one two three;
run;

proc sort data=four;
  by X;
run;
```

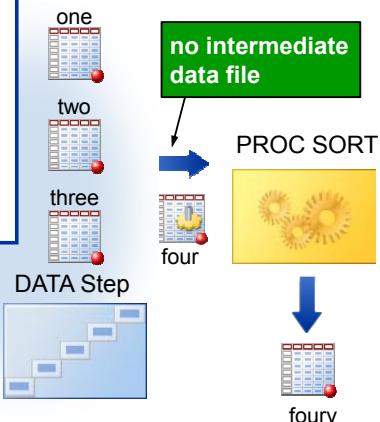


133

Avoid Intermediate Copies of Data

```
data four/view=four;
  set one two three;
run;

proc sort data=four
           out=fourv;
  by X;
run;
```



134

Choosing between a View and a File

If you process the same data multiple times in a single program, is it efficient to use a view?

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

proc freq data=orion.orders_view;
  tables Order_Type;
run;

proc means data=orion.orders_view;
run;
```

135

Choosing between a View and a File

Task	Technique
1 Reference the same file many times in one program	SAS data file

A static data file is best because the view code executes every time the view is referenced.

```
data order;
   set orion.orders_view;
run;

proc print data=order;
run;

proc freq data=order;
   tables Order_Type;
run;

proc means data=order;
run;
```

136

Choosing between a View and a File

If you are reading a file whose structure often changes, which should you use?

File1-Today
1 1 2 2 1---5---0---5---0---5
John Smith 21

File1-Tomorrow
1 1 2 2 1---5---0---5---0---5
John Smith Ted Jones

File1-Next Week
1 1 2 2 1---5---0---5---0---5
21 John Smith

```
filename rawdata 'file1';
proc print data=orion.sview;
run;

filename rawdata 'file1'
proc freq data=orion.sview;
   tables JobCode;
run;

filename rawdata 'file1'
proc means data=orion.sview;
run;
```

137

Choosing between a View and a File

2	Task	Technique
	Read files whose structures often change	SAS data file

File1-Today

1	1	2	2
1---5----0---	5----0---	5	
John Smith	21		

File1-Tomorrow

1	1	2	2
1---5----0---	5----0---	5	
John Smith	Ted Jones		

File1-Next Week

1	1	2	2
1---5----0---	5----0---	5	
21	John Smith		

```
filename rawdata 'file1';
proc print data=orion.sview;
run;
```

```
filename rawdata 'file1'
proc freq data=orion.sview;
tables JobCode;
run;
```

```
filename rawdata 'file1';
proc freq data=orion.sview;
tables JobCode;
run;
```

The view code needs to be redefined to read the new file structure.

138

Choosing between a View and a File

If the data you process is time sensitive, which should you use?

Employee ID	Employee Birth Date	Employee Hire Date	Employee Age
120101	18AUG1980	01JUL2007	34
120102	11AUG1973	01JUN1993	41
120103	22JAN1954	01JAN1978	60
120104	11MAY1958	01JAN1985	56
120105	21DEC1978	01MAY2003	36

139

Choosing between a View and a File

Task	Technique
Process time-sensitive data	SAS data view

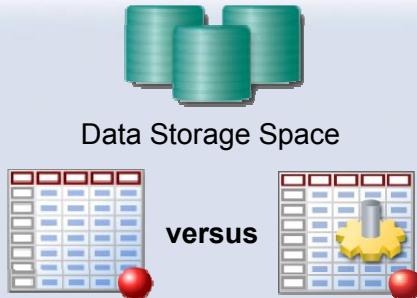
Employee ID	Employee Birth Date	Employee Hire Date	Employee Age
120101	18AUG1980	01JUL2007	34
120102	11AUG1973	01JUN1993	41
120103	22JAN1954	01JAN1978	60
120104	11MAY1958	01JAN1985	56
120105	21DEC1978	01MAY2003	36

The view executes and recalculates time-sensitive data each time it is referenced.

140

Choosing between a View and a File

If data storage space is limited, which should you use?

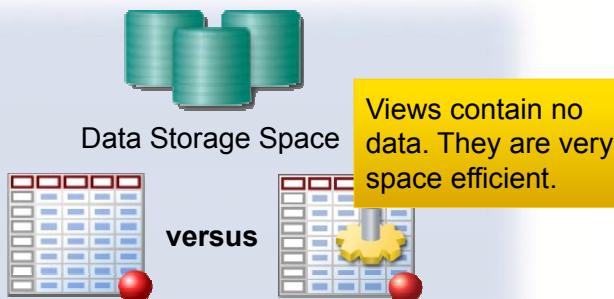


141

Choosing between a View and a File

If data storage space is limited, which should you use?

Task	Technique
4 If storage space is limited	SAS data view



142

Choosing between DATA Step Views and PROC SQL Views

Listed below are common programming tasks and the recommended view types.

Task	View Type
Perform complex conditional processing	DATA step view
Update data in place	PROC SQL view
Read data in various file formats	DATA step view
Subset data before processing	PROC SQL view
Assign a libref in the view	PROC SQL view
Use subqueries in WHERE processing	PROC SQL view
Read data from a DBMS	Either type of view



143

3.07 Quiz

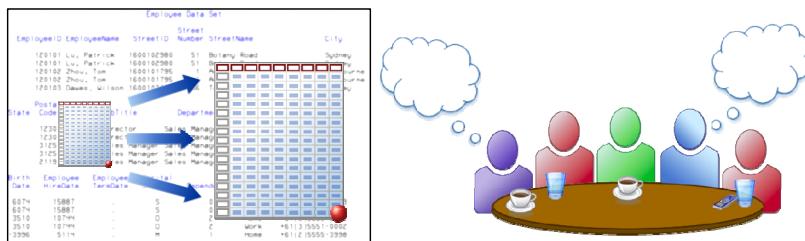
Match the following tasks with the correct solution:

- a) only DATA step view
 - b) only SQL view
 - c) either DATA step or SQL view
- perform complex conditional processing
 - update data in place
 - read data in various file formats
 - subset data before processing it
 - assign a libref in the view
 - use subqueries in WHERE processing

144

Idea Exchange

Which of the techniques to save I/O, data storage space, and processing time do you think Human Resources should implement to improve program performance and conserve resources?



146



Exercises

Level 1

9. Creating a DATA Step View

- a. Create a DATA step view that is named **ccdonations**.

- Read only the observations from the data set **orion.employeedonations** where the value of the variable **PaidBy** is *Credit Card*.
- Create a variable named **TotalDonations** as the total of the variable values for **Qtr1**, **Qtr2**, **Qtr3**, and **Qtr4**.
- Create a new variable named **DonationCategory** with the following values:

Value of TotalDonations	DonationCategory
Less than 100	Less than \$100
Greater than or equal to 100	\$100 or More

- b. Submit the DATA step view program and examine the SAS log.

Partial SAS Log

```
NOTE: DATA STEP view saved on file WORK.CCDONATIONS.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
      real time          0.12 seconds
      cpu time           0.06 seconds
```

- c. Open and submit the program **p303e09** to create a report from the view **ccdonations**. Use the variable **DonationCategory** as a class variable and the variable **TotalDonations** as an analysis variable. Verify that the view was created correctly.

p303e09

```
proc means data=ccdonations sum n nonobs maxdec=2;
  class DonationCategory;
  var TotalDonations;
run;
```

PROC MEANS Output

The MEANS Procedure		
Analysis Variable : TotalDonations		
Donation Category	Sum	N
\$100 or more	200.00	2
Less than \$100	1475.00	32

Level 2

10. Creating a View and a File in One DATA Step

- a. In one DATA step, create a view named **youngerthan60** and a file named **olderthan60**.
- Read from the data set **orion.employeepayroll**.

Partial **orion.employeepayroll**

Obs	EmployeeID	Gender	Employee Salary	Birth Date	Employee HireDate	Employee TermDate	Marital Status	Dependents
1	120101	M	163040	7535	17348	.	S	0
2	120102	M	108255	4971	12205	.	O	2
3	120103	M	87975	-2535	6575	.	M	1
4	120104	F	46230	-600	9132	.	M	1
5	120105	F	27110	6929	15826	.	S	0

- Use the variable **BirthDate** to calculate each employee's age as of today.
- The view should contain the employees who are younger than 60.
- The file should contain the employees who are 60 or older.

b. Submit the DATA step program and examine the SAS log.

Partial SAS Log

```
NOTE: DATA STEP view saved on file WORK.YOUNGERTHAN60.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
```

c. Attempt to print the first five observations of the file **olderthan60**. (This attempt will be unsuccessful.)

Partial SAS Log

```
70 proc print data=olderthan60(obs=5) noobs;
ERROR: File WORK.OLDERTHAN60.DATA does not exist.
71   title 'Older60 Data Set';
72   run;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
```

d. Print the first five observations of the view **youngerthan60**.

PROC PRINT Output

Youngerthan60 Data Set								
Employee	ID	Gender	Employee Salary	BirthDate	HireDate	Employee TermDate	Employee Status	Marital Dependents
120101	M	163040	18AUG1980	01JUL2007	.	S	0	33
120102	M	108255	11AUG1973	01JUN1993	.	O	2	40
120104	F	46230	11MAY1958	01JAN1985	.	M	1	56
120105	F	27110	21DEC1978	01MAY2003	.	S	0	35
120108	F	27660	23FEB1988	01AUG2010	.	S	0	26



Age calculations are based on the current date. Your results might differ from what is shown above.

e. Examine the SAS log.

Partial SAS Log

```

76 proc print data=youngerthan60(obs=5) noobs;
77   title 'Youngerthan60 Data Set';
78 run;
NOTE: View WORK.YOUNGERTHAN60.VIEW used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
NOTE: There were 424 observations read from the data set ORION.EMPLOYEEPAYROLL.
NOTE: The data set WORK.OLDERTHAN60 has 47 observations and 9 variables.
NOTE: There were 5 observations read from the data set WORK.YOUNGERTHAN60.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds

```

- f. Print the first five observations of the file **olderthan60** successfully.

PROC PRINT Output

Olderthan60 Data Set									
Employee		Employee			Employee			Employee	
ID	Gender	Salary	BirthDate	HireDate	TermDate	Status	Dependents	Marital	Age
120103	M	87975	22JAN1953	01JAN1978	.	M	1		61
120106	M	26960	23DEC1948	01JAN1978	.	M	2		65
120107	F	30475	21JAN1953	01FEB1978	.	M	2		61
120113	F	26870	10MAY1948	01JAN1978	.	S	0		60
120114	F	31285	08FEB1948	01JAN1978	.	M	3		60

- g. Why are you unable to print **olderthan60** in part c?
-
-

Challenge

11. Creating and Using an SQL View with the USING Clause

The starter program **p303e11** contains a PROC SQL step that creates a view.

p303e11

```

proc sql;
  create view orion.payrolldonations as
    select EmployeeID, Qtr1, Qtr2, Qtr3, Qtr4,
           sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonations
    from orion.employeedonations
    where PaidBy='Payroll Deduction';
quit;

proc print data=orion.payrolldonations(obs=5) noobs;
  title 'payrolldonations View';
run;

```

- a. Open the program **p303e11**. Edit it to assign the libref with the USING clause. The appropriate library definition is listed below.

Windows and UNIX	“&path”
z/OS	“&path..sasdata”

- b. Submit the modified PROC SQL step.
- c. Write and submit a LIBNAME statement to clear the **orion** libref.
- d. Write and submit a LIBNAME statement to assign a libref of **sasdata** to the library that is specified in the table above.
- e. Submit the PROC PRINT step to print the first five observations of the SQL view **sasdata.payrolldonations**.

PROC PRINT Output

payrolldonations View						
Obs	EmployeeID	Qtr1	Qtr2	Qtr3	Qtr4	Total Donations
1	120267	15	15	15	15	60
2	120269	20	20	20	20	80
3	120271	20	20	20	20	80
4	120272	10	10	10	10	40
5	120669	15	15	15	15	60

- f. Write and submit a LIBNAME statement to clear the **sasdata** libref.
- g. Reassign the **orion** libref.

3.6 Solutions

Solutions to Exercises

1. Using the SASFILE Statement

- a. Open the program **p303e01** and review the code.
- b. Modify the program by adding a SASFILE statement that opens and allocates buffers for **orion.employeepayroll**, but does not load it into memory. Answer the following questions:


```
sasfile orion.employeepayroll open;
```

 - 1) At what point in the program is **orion.employeepayroll** loaded into memory? **The PROC FREQ step that analyzes employeegender loads the data into memory.**
 - 2) Is PROC MEANS reading the data from disk or memory? **At this point in the program, the data resides in memory.**
 - c. Execute the program. Was the PROC SORT step successful? **No, the file cannot be overwritten on disk because it is open in memory.**
 - d. Correct the PROC SORT step by adding the appropriate option to write the data to a temporary file named **payroll**.

- e. Write a SASFILE statement that closes the file and releases the buffers.

```
sasfile orion.employee payroll close;
```

2. Using Multiple SASFILE Statements

- a. Open the program **p303e02** and submit it.

- 1) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
139 options fullstimer;
140
141 proc sql;
142   select EmployeeName,
143         sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalContribution,
144         Recipients
145   from orion.employeeaddresses as a,
146        orion.employeedonations as d
147   where a.EmployeeID=d.EmployeeID;
148 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.07 seconds
      user cpu time     0.00 seconds
      system cpu time   0.00 seconds
      memory            493.00k
      OS Memory         16620.00k
      Timestamp         07/23/2014 04:15:22 PM
      Step Count         22  Switch Count  0

149
150 options nofullstimer;
```

- b. Add the appropriate statement or statements to open and load both the **orion.employeeaddresses** and **orion.employeedonations** data sets into memory. At the end of the program, unload both data sets from memory.

Partial **p303s02**

```
options fullstimer;

sasfile orion.employeeaddresses load;
sasfile orion.employeedonations load;

proc sql;
  select EmployeeName,
         sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalContribution,
         Recipients
  from orion.employeeaddresses as a,
       orion.employeedonations as d
  where a.EmployeeID=d.EmployeeID;
quit;

sasfile orion.employeeaddresses close;
```

```
sasfile orion.employeeaddresses close;
options nofullstimer;
```

- c. Submit the revised program.
 - 1) Submit the revised SAS program.
 - 2) Examine the SAS log to verify that the program executed without errors and confirm that the SASFILE statements executed properly.

SAS Log

```
151 options fullstimer;
152
153 sasfile orion.employeeaddresses load;
NOTE: The file ORION.EMPLOYEEADDRESSES.DATA has been loaded into memory by the
SASFILE statement.
154 sasfile orion.employeedonations load;
NOTE: The file ORION.EMPLOYEEDONATIONS.DATA has been loaded into memory by the
SASFILE statement.
155
156 proc sql;
157   select EmployeeName,
158     sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalContribution,
159     Recipients
160   from orion.employeeaddresses as a,
161       orion.employeedonations as d
162   where a.EmployeeID=d.EmployeeID;
163 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.01 seconds
      user cpu time      0.01 seconds
      system cpu time    0.00 seconds
      memory             427.21k
      OS Memory          16620.00k
      Timestamp           07/23/2014 04:18:41 PM
      Step Count          23  Switch Count  0

164
165 sasfile orion.employeeaddresses close;
NOTE: The file ORION.EMPLOYEEADDRESSES.DATA has been closed by the SASFILE statement.
166 sasfile orion.employeedonations close;
NOTE: The file ORION.EMPLOYEEDONATIONS.DATA has been closed by the SASFILE statement.
167
168 options nofullstimer;
```

3. Using the APPEND Procedure with the SASFILE Statement

- a. Open the program **p303e03**.
- b. Add the appropriate statement or statements to open and load the entire **work.sales** data set into memory. Include a PROC APPEND step to append the temporary **work.nonsales** data set to the temporary **work.sales** data set, and a PROC PRINT step to print observations 164 through 168 of the enlarged **work.sales** data set. Print only the variables **EmployeeID**, **FirstName**, **LastName**, and **JobTitle**. At the end of the program, unload the **work.sales** data set.

Enter the following SAS statements in the Editor window:

Partial p303s03

```

proc copy in=orion out=work;
  select Sales NonSales;
run;

sasfile sales load;

proc append base=sales data=nonsales(rename=(First=FirstName
  Last=LastName)) force;
run;

proc print data=sales(firstobs=164 obs=168);
  var EmployeeID FirstName LastName JobTitle;
  title "Work.Sales";
  title2 "Observations 164-168";
run;

sasfile sales close;

```

- c. Submit the revised program and examine the SAS log and output.
 - 1) Submit the revised SAS program.
 - 2) Examine the SAS log to verify that the program executed without errors and confirm that the SASFILE statements executed properly.

SAS Log

```

137 proc copy in=orion out=work;
138   select Sales NonSales;
139 run;
NOTE: Copying ORION.SALES to WORK.SALES (memtype=DATA).
NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.SALES has 165 observations and 9 variables.
NOTE: Copying ORION.NONSALES to WORK.NONSALES (memtype=DATA).
NOTE: There were 235 observations read from the data set ORION.NONSALES.
NOTE: The data set WORK.NONSALES has 235 observations and 9 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
140
141 sasfile sales load;
NOTE: The file WORK.SALES.DATA has been loaded into memory by the SASFILE statement.
142
143 proc append base=sales data=nonsales(rename=(First=FirstName
144   Last=LastName)) force;
145 run;
NOTE: Appending WORK.NONSALES to WORK.SALES.
NOTE: There were 235 observations read from the data set WORK.NONSALES.
NOTE: 235 observations added.
NOTE: The data set WORK.SALES has 400 observations and 9 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds

```

```

146
147 proc print data=sales(firstobs=164 obs=168);
148   var EmployeeID FirstName LastName JobTitle;
149   title "Work.Sales";
150   title2 "Observations 164-168";
151 run;
NOTE: There were 5 observations read from the data set WORK.SALES.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
152
153 sasfile sales close;
NOTE: The file WORK.SALES.DATA has been closed by the SASFILE statement.

```

- 3) View the output to verify that it matches the expected output.

4. Creating Reduced Length Numerics

- a. Use PROC CONTENTS to determine the names and types of the variables in each of the purchase data sets.
 - 1) Open a SAS session.
 - 2) Enter the following PROC CONTENTS steps in the Editor window:

Partial p303s04

```

proc contents data=orion.catalog;
  title 'orion.catalog Contents';
run;

proc contents data=orion.internet;
  title 'orion.internet Contents';
run;

proc contents data=orion.retail;
  title 'orion.retail Contents';
run;

```

- 3) Submit the PROC CONTENTS steps.
- 4) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

154 proc contents data=orion.catalog;
155   title 'orion.catalog Contents';
156 run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time          0.01 seconds
      cpu time          0.03 seconds

157
158 proc contents data=orion.internet;
159   title 'orion.internet Contents';
160 run;

```

```

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time            0.03 seconds
      cpu time             0.00 seconds

161
162 proc contents data=orion.retail;
163   title 'orion.retail Contents';
164 run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time            0.01 seconds
      cpu time             0.00 seconds

```

- 5) Examine the output.

Partial PROC CONTENTS Output

orion.catalog Contents					
Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
10	CostPricePerUnit	Num	8	DOLLAR13.2	Cost Price Per Unit
1	CustomerID	Num	8	12.	Customer ID
5	DeliveryDate	Num	8	DATE9.	Date Order was Delivered
11	Discount	Num	8	PERCENT.	Discount in percent of Normal Total Retail Price
2	EmployeeID	Num	8	12.	Employee ID
4	OrderDate	Num	8	DATE9.	Date Order was placed by Customer
6	OrderID	Num	8	12.	Order ID
7	ProductID	Num	8	12.	Product ID
8	Quantity	Num	8		Quantity Ordered
3	StreetID	Num	8	12.	Street ID
9	TotalRetailPrice	Num	8	DOLLAR13.2	Total Retail Price for This Product
orion.internet Contents					
Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
10	CostPricePerUnit	Num	8	DOLLAR13.2	Cost Price Per Unit
1	CustomerID	Num	8	12.	Customer ID
5	DeliveryDate	Num	8	DATE9.	Date Order was Delivered
11	Discount	Num	8	PERCENT.	Discount in percent of Normal Total Retail Price
2	EmployeeID	Num	8	12.	Employee ID
4	OrderDate	Num	8	DATE9.	Date Order was placed by Customer
6	OrderID	Num	8	12.	Order ID
7	ProductID	Num	8	12.	Product ID
8	Quantity	Num	8		Quantity Ordered
3	StreetID	Num	8	12.	Street ID
9	TotalRetailPrice	Num	8	DOLLAR13.2	Total Retail Price for This Product
orion.retail Contents					
Alphabetic List of Variables and Attributes					
10	CostPricePerUnit	Num	8	DOLLAR13.2	Cost Price Per Unit
1	CustomerID	Num	8	12.	Customer ID

5	DeliveryDate	Num	8	DATE9.	Date Order was Delivered
11	Discount	Num	8	PERCENT.	Discount in percent of Normal Total Retail Price
2	EmployeeID	Num	8	12.	Employee ID
4	OrderDate	Num	8	DATE9.	Date Order was placed by Customer
6	OrderID	Num	8	12.	Order ID
7	ProductID	Num	8	12.	Product ID
8	Quantity	Num	8		Quantity Ordered
3	StreetID	Num	8	12.	Street ID
9	TotalRetailPrice	Num	8	DOLLAR13.2	Total Retail Price for This Product

- b. Open the program **p303e04**. Edit the DATA step that creates **allcustomers** to appropriately reduce the length of the numeric variables based on their values.

Enter the following LENGTH statement in the DATA step, which creates **allcustomers**:

Partial **p303s04**

```
data allcustomers;
length Quantity 3
      CustomerID OrderDate DeliveryDate 4
      EmployeeID 5
      StreetID OrderID 6
      ProductID 7;
set orion.catalog orion.internet orion.retail;
run;
```

 Be sure to place the LENGTH statement before the SET statement.

- c. Submit the altered program and examine the SAS log.

- 1) Submit the PROC CONTENTS steps.
- 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
20  data allcustomers;
21    length Quantity 3
22      CustomerID OrderDate DeliveryDate 4
23      EmployeeID 5
24      StreetID OrderID 6
25      ProductID 7;
26    set orion.catalog orion.internet orion.retail;
27  run;

WARNING: Multiple lengths were specified for the variable CustomerID by input data
          set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable EmployeeID by input data
          set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable StreetID by input data
          set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable OrderDate by input data
          set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable DeliveryDate by input data
          set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable OrderID by input data
          set(s). This may cause truncation of data.
```

```

WARNING: Multiple lengths were specified for the variable ProductID by input data
         set(s). This may cause truncation of data.
WARNING: Multiple lengths were specified for the variable Quantity by input data
         set(s). This may cause truncation of data.
NOTE: There were 38 observations read from the data set ORION.CATALOG.
NOTE: There were 123 observations read from the data set ORION.INTERNET.
NOTE: There were 324 observations read from the data set ORION.RETAIL.
NOTE: The data set WORK.ALLCUSTOMERS has 485 observations and 11 variables.
NOTE: DATA statement used (Total process time):
      real time          0.10 seconds
      cpu time          0.07 seconds

```

- d. Use PROC CONTENTS to check the length of the numeric variables in **allcustomers**.

- 1) Enter the following PROC CONTENTS step in the Editor window:

Partial p303s04

```

proc contents data=allcustomers;
  title 'allcustomers Contents';
run;

```

- 2) Submit the PROC CONTENTS step.
- 3) Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```

174 proc contents data=allcustomers;
175   title 'allcustomers Contents';
176 run;
NOTE: PROCEDURE CONTENTS used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds

```

- 4) Examine the output.

Partial PROC CONTENTS Output

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
10	CostPricePerUnit	Num	8	DOLLAR13.2	Cost Price Per Unit
2	CustomerID	Num	4	12.	Customer ID
4	DeliveryDate	Num	4	DATE9.	Date Order was Delivered
11	Discount	Num	8	PERCENT.	Discount in percent of Normal Total Retail Price
5	EmployeeID	Num	5	12.	Employee ID
3	OrderDate	Num	4	DATE9.	Date Order was placed by Customer
7	OrderID	Num	6	12.	Order ID
8	ProductID	Num	7	12.	Product ID
1	Quantity	Num	3		Quantity Ordered
6	StreetID	Num	6	12.	Street ID
TotalRetailPrice Num 8 DOLLAR13.2 Total Retail Price for This Product					

5. Creating Reduced Length Numerics and Precision

- a. Open the program **p303e05** and submit it.

- 1) Submit the program.
- 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

17  data five;
18    length Num5 5 Num8 8;
19    do Num8=1e10 to 1e13 by 1e11;
20      Num5=Num8;
21      output;
22    end;
23  run;
NOTE: The data set WORK.FIVE has 100 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
24
25  proc print data=five;
26    title 'Reducing the length of numeric data to 5';
27    format Num5 Num8 20. ;
28  run;
NOTE: There were 100 observations read from the data set WORK.FIVE.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

```

- 3) View the output to verify that it matches the expected output.
- a. At what point in the sequence of numbers, does the length of the variable **Num5** lose precision?

Partial Output

Reducing the length of numeric data to 5

Obs	Num5	Num8
1	10000000000	10000000000
2	11000000000	11000000000
3	21000000000	21000000000
4	31000000000	31000000000
5	41000000000	41000000000
6	51000000000	51000000000
7	60999998976	61000000000

The variable **Num5** loses precision at the number **610000000000** on Windows and UNIX and at **1010000000000** on z/OS.

6. Determining the Minimum Number of Bytes for Reduced Length Numerics

- a. Run the program that is stored in **p303e06** and examine the output.
 - 1) Submit the program.
 - 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

29  /* Windows and UNIX DATA step */
30  data numbers;
```

```

31      input Value;
32      datalines;
NOTE: The data set WORK.NUMBERS has 4 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
37  ;
38
39  /* z/OS DATA step */
40  /*
41  data numbers;
42    input Value;
43    datalines;
44  269
45  270
46  271
47  272
48  ;
49  */
50
51  data temp;
52    set numbers;
53    X=Value;
54    do L=8 to 1 by -1;
55      if X NE trunc(X,L) then
56        do;
57          MinLen=L+1;
58          output;
59          return;
60        end;
61      end;
62    run;
NOTE: Argument 2 to function TRUNC(8191,2) at line 674 column 15 is invalid.
Value=8191 X=8191 L=2 MinLen=3 _ERROR_=1 _N_=1
NOTE: Argument 2 to function TRUNC(8192,2) at line 674 column 15 is invalid.
Value=8192 X=8192 L=2 MinLen=3 _ERROR_=1 _N_=2
NOTE: Argument 2 to function TRUNC(8194,2) at line 674 column 15 is invalid.
Value=8194 X=8194 L=2 MinLen=3 _ERROR_=1 _N_=4
NOTE: Mathematical operations could not be performed at the following places. The
      results of the operations have been set to missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      3 at 674:15
NOTE: There were 4 observations read from the data set WORK.NUMBERS.
NOTE: The data set WORK.TEMP has 4 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time          0.00 seconds
63
64  title;
65
66  proc print data=temp noobs;
67    var Value MinLen;
68  run;
NOTE: There were 4 observations read from the data set WORK.TEMP.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds

```

- 3) View the output to verify that it matches the expected output.
- b. Investigate the Help facility to determine why the minimum length for the number 8194 is less than that of the number 8193 (Windows and UNIX) or why 272 is less than that of the number 271 (z/OS).
- 1) Click the **Help** button.
 - 2) Navigate to **Using SAS Software in Your Operating Environment** \Rightarrow **The 9.4 Companion for Windows** (or UNIX or z/OS) \Rightarrow **Features of the SAS Language for Windows** (or UNIX or z/OS) \Rightarrow **Length and Precision of Variables Under Windows** (or UNIX or z/OS) \Rightarrow **Numeric Variables**.

The minimum length required for the value 271 (or 8193) is greater than the minimum required for the value 272 (or 8194). This fact illustrates that it is possible for the largest number in a range of numbers to require fewer bytes of storage than a smaller number. If precision is needed for all numbers in a range, you should use the minimum length for all the numbers, not only the largest one.

7. Compressing SAS Data Files

- a. Open the program **p303e07**.
- b. After the DATA step, insert a PROC CONTENTS step to examine the descriptor portion of the **work.employees** file.

How many data set pages are needed for this file in its uncompressed form? **7**

- c. Add the appropriate data set option to compress the **work.employees** file using Run-Length Encoding.

How many data set pages are needed for this file in its RLE compressed form? **4**

What is the value of the COMPRESSED attribute in the output? **CHAR**

- d. Modify the data set option to compress the **work.employees** file using Ross Data Compression.

How many data set pages are needed for this file in its RDC compressed form? **4**

What is the value of the COMPRESSED attribute in the output? **BINARY**

Which attributes of the file are added to the PROC CONTENTS output when compression is turned on? **Reuse Space, Point To Observations**

Which attributes of the file are removed when compression is turned on? **First Data Page, Max Obs per Page, Obs in First Data Page**

8. Compressing SAS Data Files

- a. If you merge these two files by **SupplierID** and compress the output data, which method of compression do you think is the most appropriate?

Because the data set supplier is heavily character data, CHAR is probably most appropriate.

- b. Merge **orion.productlist** and **orion.supplier** by **SupplierID** to create a data set named **suppliernames1**. Compress the data set. Use the method that you predicted in part a.

 Sort the data by **SupplierID** to prepare for the merge.

- 1) Enter the following SAS program in the Editor window:

Partial p303s08

```
proc sort data=orion.productlist out=productlist;
  by SupplierID;
run;
proc sort data=orion.supplier out=supplier;
  by SupplierID;
run;

data suppliernames1(compress=char);
  merge supplier productlist;
  by SupplierID;
run;
```

- 2) Submit the program.
- 3) Examine the SAS log to determine the percentage of space that was saved.

Partial SAS Log

```
51 proc sort data=orion.productlist out=productlist;
52   by SupplierID;
53 run;

NOTE: There were 556 observations read from the data set ORION.PRODUCTLIST.
NOTE: The data set WORK.PRODUCTLIST has 556 observations and 5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.11 seconds
      cpu time          0.01 seconds

54 proc sort data=orion.supplier out=supplier;
55   by SupplierID;
56 run;

NOTE: There were 52 observations read from the data set ORION.SUPPLIER.
NOTE: The data set WORK.SUPPLIER has 52 observations and 6 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.03 seconds
      cpu time          0.00 seconds

57
58 data suppliernames1(compress=char);
59   merge supplier productlist;
60   by SupplierID;
61 run;

NOTE: There were 52 observations read from the data set WORK.SUPPLIER.
NOTE: There were 556 observations read from the data set WORK.PRODUCTLIST.
NOTE: The data set WORK.SUPPLIERNAMES1 has 556 observations and 10 variables.
NOTE: Compressing data set WORK.SUPPLIERNAMES1 decreased size by 0.00 percent.
      Compressed is 2 pages; un-compressed would require 2 pages.
```

```
NOTE: DATA statement used (Total process time):
      real time            0.00 seconds
      cpu time             0.01 seconds
```

- c. Merge **orion.productlist** and **orion.supplier** by **SupplierID** to create a data set named **suppliernames2**. Compress it and use the alternative method.

- Enter the following SAS program in the Editor window:

Partial p303s08

```
data suppliernames2 (compress=binary);
  merge supplier productlist;
  by SupplierID;
run;
```

- Submit the program.
- Examine the SAS log to verify that the program executed without errors.

Partial SAS Log

```
263  data suppliernames2(compress=binary);
264    merge supplier productlist;
265    by SupplierID;
266  run;
NOTE: There were 52 observations read from the data set WORK.SUPPLIER.
NOTE: There were 556 observations read from the data set WORK.PRODUCTLIST.
NOTE: The data set WORK.SUPPLIERNAMES2 has 556 observations and 10 variables.
NOTE: Compressing data set WORK.SUPPLIERNAMES2 increased size by 50.00 percent.
      Compressed is 3 pages; un-compressed would require 2 pages.
NOTE: DATA statement used (Total process time):
      real time            0.00 seconds
      cpu time             0.00 seconds
```

- Which method was better? **Char**
- Why was that method better? **The Sales data set consists heavily of character data.**

9. Creating a DATA Step View

- a. Create a DATA step view that is named **ccdonations**.

Enter the following SAS statements in the Editor window:

Partial p303s09

```
data ccdonations / view=ccdonations;
  set orion.employeedonations;
  length DonationCategory $15;
  where PaidBy='Credit Card';
  TotalDonations=sum(of Qtr1 - Qtr4);
  if TotalDonations >= 100 then DonationCategory='$100 or more';
  else DonationCategory='Less than $100';
run;
```

- b. Submit the DATA step view program and examine the SAS log.

- Submit the revised SAS program.

- 2) Examine the SAS log to verify that the program executed without errors. Confirm that the DATA step view was created properly.

SAS Log

```

21  data ccdonations / view=ccdonations;
22  set orion.employeedonations;
23  length DonationCategory $15;
24  where PaidBy='Credit Card';
25  TotalDonations=sum(of Qtr1 - Qtr4);
26  if TotalDonations >= 100 then DonationCategory='$100 or more';
27  else DonationCategory='Less than $100';
28 run;

```

NOTE: DATA STEP view saved on file WORK.CCDONATIONS.

NOTE: A stored DATA STEP view cannot run under a different operating system.

NOTE: DATA statement used (Total process time):

real time	0.03 seconds
cpu time	0.03 seconds

- c. Open and submit the program **p303e09** to create a report from the view **ccdonations**. Use the variable **DonationCategory** as a class variable and the variable **TotalDonations** as an analysis variable. Verify that the view was created correctly.

- 1) Submit the **p303e09** program.

p303s09

```

proc means data=ccdonations sum n nonobs maxdec=2;
  class DonationCategory;
  var TotalDonations;
run;

```

- 2) View the output to verify that it matches the expected output.

PROC MEANS Output

The MEANS Procedure		
Analysis Variable : TotalDonations		
Donation Category	Sum	N
\$100 or more	200.00	2
Less than \$100	1475.00	32

10. Creating a View and a File in One DATA Step

- a. In one DATA step, create a view named **youngerthan60** and a file named **olderthan60**.

Enter the following SAS statements in the Editor window:

Partial p303s10

```

data olderthan60 youngerthan60 / view=youngerthan60;
  set orion.employeepayroll;
  Age=int(yrdif(BirthDate, today(), 'act/act'));
  if Age >= 60 then output olderthan60;
  else output youngerthan60;

```

```
format BirthDate EmployeeHireDate EmployeeTermDate date9. ;
run;
```

- b. Submit the DATA step program and examine the SAS log.
- 1) Submit the SAS program.
 - 2) Examine the SAS log to verify that the program executed without errors. Confirm that the DATA step view was created properly.

SAS Log

```
29  data olderthan60 youngerthan60 / view=youngerthan60;
30  set orion.employeepayroll;
31  Age=int(yrdif(BirthDate, today(), 'act/act'));
32  if Age >= 60 then output olderthan60;
33  else output youngerthan60;
34  format BirthDate EmployeeHireDate EmployeeTermDate date9. ;
35  run;
```

```
NOTE: DATA STEP view saved on file WORK.YOUNGERTHAN60.
NOTE: A stored DATA STEP view cannot run under a different operating system.
NOTE: DATA statement used (Total process time):
      real time          0.06 seconds
      cpu time          0.01 seconds
```

- c. Attempt to print the first five observations of the file **olderthan60**. (This attempt will be unsuccessful.)
- 1) Enter the following SAS statements in the Editor window:

Partial p303s10

```
proc print data=olderthan60(obs=5);
  title 'Olderthan60 Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Examine the SAS log to identify the error.

Partial SAS Log

```
15  proc print data=olderthan60(obs=5) noobs;
ERROR: File WORK.OLDERTHAN60.DATA does not exist.
16  title 'Olderthan60 Data Set';
17  run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.07 seconds
      cpu time          0.04 seconds
```

- d. Print the first five observations of the view **youngerthan60**.

- 1) Enter the following SAS statements in the Editor window:

Partial p303s10

```
proc print data=youngerthan60(obs=5);
```

```
title 'Youngerthan60 Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- e. Examine the SAS log.

Partial SAS Log

```
76 proc print data=youngerthan60(obs=5) noobs;
77   title 'Youngerthan60 Data Set';
78 run;
NOTE: View WORK.YOUNGERTHAN60.VIEW used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
NOTE: There were 424 observations read from the data set ORION.EMPLOYEEPAYROLL.
NOTE: The data set WORK.OLDERTHAN60 has 70 observations and 9 variables.
NOTE: There were 5 observations read from the data set WORK.YOUNGERTHAN60.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
```

- f. Print the first five observations of the file **olderthan60** successfully.

- 1) Enter the following SAS statements in the Editor window:

Partial p303s10

```
proc print data=olderthan60(obs=5);
  title 'Olderthan60 Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Examine the SAS log.

Partial SAS Log

```
23 proc print data=olderthan60(obs=5) noobs;
24   title 'Olderthan60 Data Set';
25 run;
NOTE: There were 5 observations read from the data set WORK.OLDERTHAN60.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
```

- g. Why are you unable to print **olderthan60** in part c?

The file olderthan60 was not created until the step that created the view youngerthan60 was executed.

11. Creating and Using an SQL View with the USING Clause

- a. Open the program **p303e11**. Edit it to assign the libref with the USING clause.
- 1) Add the highlighted USING clause to the existing CREATE VIEW statement in the Editor window:

Partial p303s11

```

proc sql;
  create view orion.payrolldonations as
    select EmployeeID, Qtr1, Qtr2, Qtr3, Qtr4,
           sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonations
      from orion.employeedonations
     where PaidBy='Payroll Deduction'
           using libname orion "&path";
quit;

```

- b. Submit the modified PROC SQL step.

Examine the SAS log.

Partial SAS Log

```

26 proc sql;
27   create view orion.payrolldonations as
28     select EmployeeID, Qtr1, Qtr2, Qtr3, Qtr4,
29       sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonations
30     from orion.employeedonations
31   where PaidBy='Payroll Deduction'
32     using libname orion "&path";
NOTE: SQL view ORION.PAYROLLDONATIONS has been defined.
33 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.06 seconds
      cpu time           0.04 seconds

```

- c. Write and submit a LIBNAME statement to clear the **orion** libref

- 1) Enter the following LIBNAME statement in the Editor window:

Partial p303s11

```
libname orion clear;
```

- 2) Submit the LIBNAME statement.
- 3) Examine the SAS log to verify that the libref **orion** was deassigned correctly.

Partial SAS Log

```

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

```

- b. Write and submit a LIBNAME statement to assign a libref of **sasdata** to the library that is specified in the table above.

- 1) Enter the following LIBNAME statement in the Editor window:

Partial p303s11

```
libname sasdata "&path"; * Windows/UNIX LIBNAME;
```

- 2) Submit the LIBNAME statement.
- 3) Examine the SAS log to verify that the libref **sasdata** was assigned correctly.

Partial SAS Log

```
35 libname sasdata "&path";
NOTE: Libref SASDATA was successfully assigned as follows:
      Engine:      V9
      Physical Name: S:\Workshop
34 !                                     * Windows/UNIX LIBNAME;
```

- c. Submit the PROC PRINT step to print the first five observations of the SQL view **sasdata.payrolldonations**.

Partial p303s11

```
proc print data=sasdata.payrolldonations(obs=5);
run;
```

Partial SAS Log

```
35 proc print data=sasdata.payrolldonations(obs=5);
36 run;

NOTE: There were 5 observations read from the data set ORION.EMPLOYEEDONATIONS.
      WHERE PaidBy='Payroll Deduction';
NOTE: There were 5 observations read from the data set SASDATA.PAYROLLDONATIONS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.21 seconds
      cpu time           0.07 seconds
```

- 1) Examine the PROC PRINT output to verify that it matches the expected output.

- d. Write and submit a LIBNAME statement to clear the **sasdata** libref.

- 1) Enter the following LIBNAME statement in the Editor window:

Partial p303s11

```
libname sasdata clear;
```

- 2) Submit the LIBNAME statement.

- 3) Examine the SAS log to verify that the libref **sasdata** was deassigned correctly.

Partial SAS Log

```
37 libname sasdata clear;
NOTE: Libref SASDATA has been deassigned.
```

- e. Reassign the **orion** libref.

- 1) Enter the following LIBNAME statement in the Editor window:

Partial p303s11

```
libname orion "&path";
```

- 2) Submit the LIBNAME statement.

- 3) Examine the SAS log to verify that the libref **orion** was assigned correctly.

Partial SAS Log

```
35 libname orion "&path";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:      V9
      Physical Name:  S:\Workshop
```

Solutions to Student Activities (Polls/Quizzes)

3.01 Multiple Answer Poll – Correct Answers

When you read raw data and create a SAS data set using the DATA step, when is I/O measured?

- a. when SAS copies the input data from the source to a buffer in memory
- b. when SAS copies the input data from the input buffer to the program data vector
- c. when SAS copies the output data from the output buffer to the output data set
- d. none of the above

22

3.02 Multiple Choice Poll – Correct Answer

In addition to decreasing I/O when the DATA step creates **bonus**, where else does program 2 decrease I/O?

- a. Program 2 reads fewer variables from **orion.staff** into the PDV.
- b. The PROC MEANS step in program 2 loads a smaller version of **bonus**.
- c. No additional decrease in I/O occurs in program 2.

36

3.03 Multiple Answer Poll – Correct Answers

Which of the following does the SASFILE LOAD statement do?

- a. loads the specified data set into memory
- b. defers reading the data into memory until a procedure or statement is executed
- c. closes the data set and releases the SAS buffers
- d. allocates buffers

46

Idea Exchange

A data set contains 200 numeric variables with integer values ranging from 1 to 10. If you reduce the length of these numeric variables to the smallest appropriate byte size, how many bytes are you saving per observation? How many bytes are saved per 1,000 observations?

These integer values fit into 3 bytes on a Windows PC/Server. Saving 5 bytes per variable equates to saving 1,000 bytes per observation. That results in a savings of 1,000,000 bytes (1MB) for every 1,000 observations.



75

3.04 Poll – Correct Answer

Open and submit the program **p303a01**.

Examine the log.

After the second DATA step executes, are the values of X and Y equal?

- Yes
- No

79

3.05 Poll – Correct Answer

By default, the observations in a SAS data file have varying lengths.

- Yes
- No

By default, SAS observations have fixed lengths.

91

3.06 Multiple Choice Poll – Correct Answers

Which of the following are advantages of using a DATA step view as opposed to using a static table?

- a. storing complex code for reuse
- b. reducing programming errors
- c. accessing the most current data in changing files
- d. avoiding storing a SAS copy of a large data file
- e. all of the above

125

3.07 Quiz – Correct Answer

Match the following tasks with the correct solution:

- a) only DATA step view
 - b) only SQL view
 - c) either DATA step or SQL view
-
- a perform complex conditional processing
 - b update data in place
 - a read data in various file formats
 - b subset data before processing it
 - b assign a libref in the view
 - b use subqueries in WHERE processing

145

Chapter 4 Accessing Observations

4.1 Access Methods	4-3
4.2 Accessing Observations by Number.....	4-5
Exercises	4-26
4.3 Creating an Index	4-27
Exercises	4-54
4.4 Using an Index	4-56
Exercises	4-70
4.5 Solutions	4-74
Solutions to Exercises	4-74
Solutions to Student Activities (Polls/Quizzes)	4-89

4.1 Access Methods

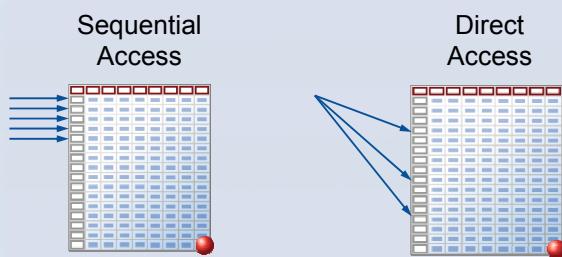
Objectives

- Define access methods.
- Identify reasons to use direct access.
- List techniques that you can use to directly access observations in a SAS data set.

3

Accessing Observations

Observations in SAS data sets can be accessed sequentially or directly.



Sequential access is the default.

4

Accessing Observations

When sequential access is used to subset a SAS data set, every observation is read to extract the desired subset.

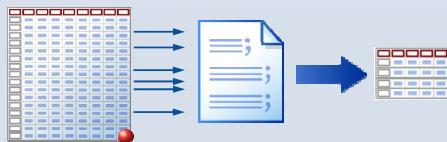


For large files, this can be very resource intensive.

5

Accessing Observations

When direct access is used to subset a SAS data set, observations are read directly from their physical location to extract the desired subset.



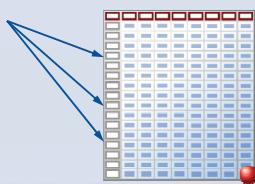
6

Accessing Observations

In this chapter, you focus on direct access techniques to perform these specific tasks.

- Subset a SAS data set based on observation number.
- Subset a large SAS data set based on a variable value.

Direct
Access



7

4.2 Accessing Observations by Number

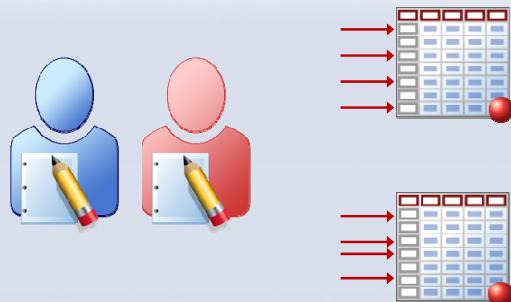
Objectives

- Use the DATA step to retrieve observations by number.
- Use the DATA step to create a systematic sample.

9

Business Scenario

The Marketing Department wants to send satisfaction questionnaires to a sample of the customers. The customers are selected using a systematic sampling technique without reading the entire SAS data set.



10

Systematic Samples

A *systematic sample* is a sample that contains observations that are chosen at regular intervals.

Partial `orion.orderfact`

CustomerID	EmployeeID	StreetID	OrderDate	Delivery Date	OrderID	...
63	121039	9260125492	11JAN2007	11JAN2007	1230058123	...
5	99999999	9260114570	15JAN2007	19JAN2007	1230080101	...
45	99999999	9260104847	20JAN2007	22JAN2007	1230106883	...
41	120174	1600101527	28JAN2007	28JAN2007	1230147441	...
183	120134	1600100760	27FEB2007	27FEB2007	1230315085	...
.
.

11

Using the DATA Step with the NOBS= Option

The NOBS= option assigns the number of observations in the SAS data set to a temporary variable.

```
data subset;
  do PickIt=1 to TotObs by 50;
    set orion.orderfact
      (keep=CustomerID
       EmployeeID
       StreetID
       OrderID) point=PickIt
                  nobs=TotObs;
    output;
  end;
  stop;      SET data-set-name NOBS=observation-number;
run;
```

12

p304d01

...

Using the DATA Step with the POINT= Option

The POINT= option specifies a temporary variable whose numeric value determines which observation is read.

```
data subset;
  do PickIt=1 to TotObs by 50;
    set orion.orderfact
      (keep=CustomerID
       EmployeeID
       StreetID
       OrderID) point=PickIt
                  nobs=TotObs;
    output;
  end;
  stop;      SET data-set-name POINT=point-variable;
run;
```

13

p304d01

...

Using the STOP Statement

The STOP statement prevents the continuous processing of the DATA step.

```
data subset;
  do PickIt=1 to TotObs by 50;
    set orion.orderfact
      (keep=CustomerID
       EmployeeID
       StreetID
       OrderID) point=PickIt
      nobs=TotObs;
    output;
  end;
  stop;
run;
```

14

p304d01

Compilation Phase

```
data subset;
  do PickIt=1 to TotObs by 50;
    set orion.orderfact
      (keep=CustomerID
       EmployeeID
       StreetID
       OrderID)
      point=PickIt
      nobs=TotObs;
    output;
  end;
  stop;
run;
```

PDV

► PickIt	► Tot Obs	Customer ID	Employee ID	StreetID	OrderID	► N_
	617					

15

p304d01

...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
1	617	1

16

p304d01

...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
1	617	63	121039	9260125492	1230058123	1

17

p304d01

...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

Output current observation.

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
1	617	63	121039	9260125492	1230058123	1

18

p304d01
...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
51	617	63	121039	9260125492	1230058123	1

19

p304d01
...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
51	617	63	121039	9260125492	1230058123	1

20

p304d01

...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.order_act
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
51	617	17023	99999999	2600100021	1230931366	1

21

p304d01

...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

Output current observation.

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
51	617	17023	99999999	2600100021	1230931366	1

22

p304d01
...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...

50	17023	99999999	...
51	17023	99999999	...


```
data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;
```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
101	617	17023	99999999	2600100021	1230931366	1

23

p304d01
...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...
50
50	17023	99999999	...
51	17023	99999999	...
51

```

data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop;
run;

```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
651	617	215	120175	1600102721	1243963366	1

24 p304d01 ...

Execution Phase

Partial orion.orderfact

obs	Customer ID	Employee ID	...
1	63	121039	...
2	5	99999999	...
50
50	17023	99999999	...
51	17023	99999999	...
51

```

data subset;
do PickIt=1 to TotObs by 50;
  set orion.orderfact
    (keep=CustomerID
     EmployeeID
     StreetID
     OrderID)
  point=PickIt
  nobs=TotObs;
  output;
end;
stop; Execution stops.
run;

```

PDV

D▶ PickIt	D▶ Tot Obs	Customer ID	Employee ID	StreetID	OrderID	D▶ N_
651	617	215	120175	1600102721	1243963366	1

25 p304d01

Resulting Data Set

Partial `work.subset`

Systematic Sample				
Obs	CustomerID	EmployeeID	StreetID	OrderID
1	63	121039	9260125492	1230058123
2	17023	99999999	2600100021	1230931366
3	17	121037	9260123306	1231757107
4	195	120160	1600101663	1232590052
5	41	120134	1600101527	1233545775

26

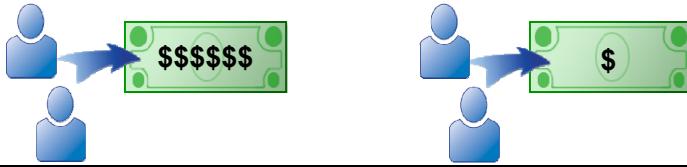
p304d01

Business Scenario

The Marketing Department wants to identify which two customers ordered the most expensive and least expensive items.

Partial `orion.orderfact`

Customer ID	ProductID	Quantity	Total Retail Price	CostPrice PerUnit	Discount
63	220101300017	1	\$16.50	\$7.45	.
5	230100500026	1	\$247.50	\$109.55	.
45	240600100080	1	\$28.30	\$8.55	.
41	240600100010	2	\$32.00	\$6.50	.
183	240200200039	3	\$63.60	\$8.80	.



27

Sorting the Data

The data set **orion.orderfact** is not stored in sorted order by **TotalRetailPrice**. Before selecting the customers, sort the data.

```
proc sort data=orion.orderfact
           out=SortedOrderFact;
   by descending TotalRetailPrice;
run;
```

28

p304d02

Selecting Observations

The first two observations in the data set are written to the data set **top**. The last two observations are written to the data set **bottom**.

Partial work.SortedOrderFact

Customer ID	Product ID	Quantity	Total Retail Price	CostPrice PerUnit	Discount	...
70100	240200100173	2	\$1937.20	\$247.70
79	240200100076	1	\$1796.00	\$246.55
.
11171	240200100021	1	\$2.70	\$1.20
79	230100500045	2	\$2.60	\$0.60

29

Idea Exchange

How would you select the observations with the lowest **TotalRetailPrice** using the DATA step without reading the entire SAS data set?



30

Compilation Phase

```
data top bottom;
  set SortedOrderFact(obs=2) end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS= (TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs point=ReadOBS;
      output bottom;
    end;
run;
```

Partial PDV

onlastrow	ReadOBS	TotOBS	CustomerID	Total Retail Price	_N_
.	.	617	.	.	.

31

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
set SortedOrderFact(obs=2)
  end=onlastrow;
output top;
if onlastrow then
  do ReadOBS=(TotOBS) to
    (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Partial PDV

D	onlastrow	D	ReadOBS	D	TotOBS	CustomerID	Total Retail Price	D	_N_
	0		.		617	70100	1937.20		1

32

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
set SortedOrderFact(obs=2)
  end=onlastrow;
output top;
if onlastrow then
  do ReadOBS=(TotOBS) to
    (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Output current observation.

Partial PDV

D	onlastrow	D	ReadOBS	D	TotOBS	CustomerID	Total Retail Price	D	_N_
	0		.		617	70100	1937.20		1

33

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

False

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
0	.	617	70100	1937.20	1

34

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Implicit RETURN;

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
0	.	617	70100	1937.20	1

35

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
set SortedOrderFact(obs=2)
  end=onlastrow;
output top;
if onlastrow then
  do ReadOBS=(TotOBS) to
    (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	.	617	79	1796.00	2

36

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
set SortedOrderFact(obs=2)
  end=onlastrow;
output top;
if onlastrow then
  do ReadOBS=(TotOBS) to
    (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Output current observation.

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	.	617	79	1796.00	2

37

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	.	617	79	1796.00	2

38

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
    set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
    output bottom;
  end;
run;

```

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	617	617	79	1796.00	2

39

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;

```

Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	617	617	79	2.60	2

40

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;

```

Output current observation.

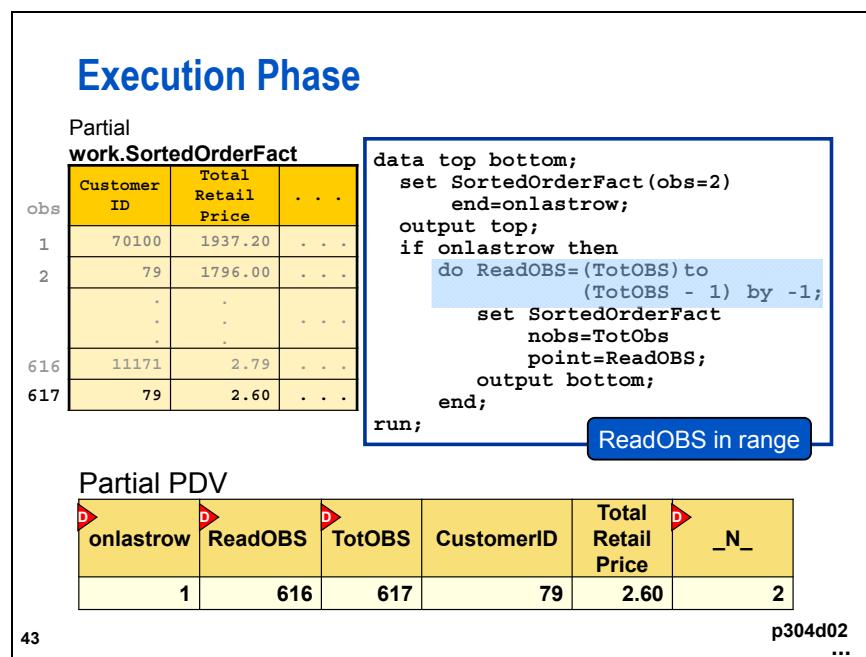
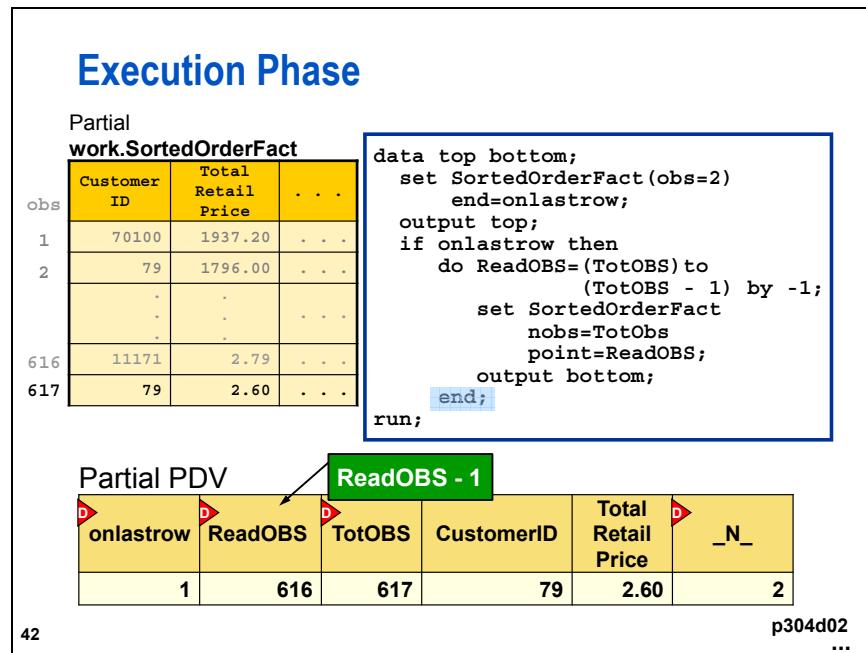
Partial PDV

D onlastrow	D ReadOBS	D TotOBS	CustomerID	Total Retail Price	D _N_
1	617	617	79	2.60	2

41

p304d02

...



Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;

```

Partial PDV

onlastrow	ReadOBS	TotOBS	CustomerID	Total Retail Price	_N_
1	616	617	11171	2.79	2

44

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2)
    end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;

```

Output current observation.

Partial PDV

onlastrow	ReadOBS	TotOBS	CustomerID	Total Retail Price	_N_
1	616	617	11171	2.79	2

45

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2);
  end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
      output bottom;
    end;
run;

```

Implicit RETURN;

Partial PDV

onlastrow	ReadOBS	TotOBS	CustomerID	Total Retail Price	_N_
1	616	617	11171	2.79	2

46

p304d02

...

Execution Phase

Partial

work.SortedOrderFact

obs	Customer ID	Total Retail Price	...
1	70100	1937.20	...
2	79	1796.00	...

616	11171	2.79	...
617	79	2.60	...

```

data top bottom;
  set SortedOrderFact(obs=2);
  end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
      nobs=TotObs
      point=ReadOBS;
      output bottom;
    end;
run;

```

EOF

Partial PDV

onlastrow	ReadOBS	TotOBS	CustomerID	Total Retail Price	_N_
1	616	617	11171	2.79	3

47

p304d02

...

Resulting Data Sets

Partial work.top

Top 2 Big Spenders	
CustomerID	TotalRetail Price
70100	\$1,937.20
79	\$1,796.00

Partial work.bottom

Bottom 2 Frugal Spenders	
CustomerID	TotalRetail Price
79	\$2.60
11171	\$2.70

48

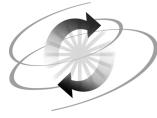
p304d02

4.01 Short Answer Poll

If the OBS=2 data set option was omitted in the first SET statement, how many times would the DATA step execute?

```
data top bottom;
  set SortedOrderFact end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;
```

49



Exercises

Level 1

1. Generating a Systematic Sample

- Generate a systematic sample. Select every 10th supplier. Start with observation 10 from the data set **orion.productdim**. The sample should have the following characteristics:
 - The sample should contain only the variables **ProductLine**, **ProductID**, **ProductName**, and **SupplierName**.
 - Name the output data set **productsample**.
- Print the first five observations of **productsample**. Omit the observation numbers from the output.

PROC PRINT Output

Systematic Sample of Products			
ProductID	ProductLine	ProductName	SupplierName
210200500002	Children	Children's Mitten	AllSeasons Outdoor Clothing
210200900033	Children	Osprey France Nylon Shorts	Triple Sportswear Inc
220100100044	Clothes & Shoes	Sports glasses Satin Alumin.	Eclipse Inc
220100100241	Clothes & Shoes	Big Guy Men's Santos Shorts Dri Fit	Eclipse Inc
220100100513	Clothes & Shoes	Woman's Deception Dress	Eclipse Inc

Level 2

2. Retrieving the Top Five and Bottom Five Observations

- Create two data sets, **highest** and **lowest**. **Work.highest** contains the five managers with the highest combined salary groups. **Work.lowest** contains the five managers with the lowest combined salary groups. Use **orion.totalsalaries** to retrieve the observations. Each observation in **orion.totalsalaries** represents one manager's salaries for all of his or her employees.
- Print all observations from **highest** and **lowest**.

Hint: Sort **orion.totalsalaries** before you select the desired observations.

Top 5 Salary Groups		
ManagerID	Numemps	DeptSal
121143	51	\$1,410,530
120102	48	\$1,344,595
121145	45	\$1,216,055
120259	6	\$941,155
120103	30	\$793,835

Bottom 5 Salary Groups		
ManagerID	Numemps	DeptSal
120748	1	\$26,545
120714	1	\$28,535
120666	1	\$29,980
120712	1	\$31,630
120672	1	\$35,935

Challenge

3. Generating a Random Sample with Replacement

- a. Generate a random sample *with* replacement of 50 customers from **orion.customerdim**. If the customer age is less than 40, place those customers in a data set named **underforty**. If the customer age is greater than or equal to 40, place the customers in a data set named **fortyplus**.

Hint: Use the RANUNI function. If you obtain zero observations in one of the data sets, run the program again. It is possible that the selected observations might all be 40 or older or all could be less than 40. If you used a constant seed in the RANUNI function, change the seed value and resubmit the program.

- b. Print both data sets. Suppress the observation number. Show only the **CustomerID** and **CustomerName** variables, and print only the first five observations from each data set.

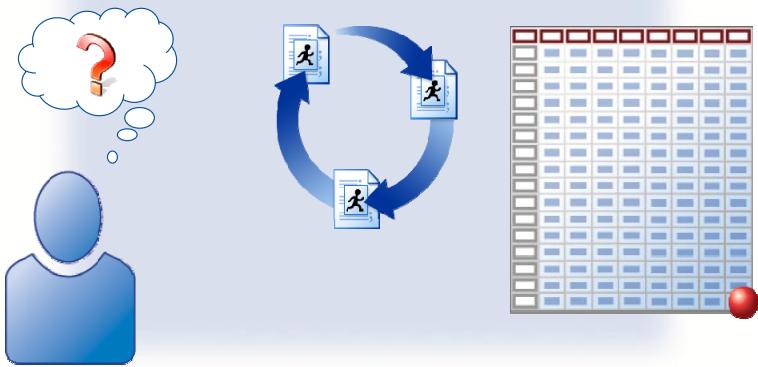
4.3 Creating an Index

Objectives

- Define indexes.
- List the uses of indexes.
- Use the DATA step to create indexes.
- Use PROC DATASETS to create and maintain indexes.
- Use PROC SQL to create and maintain indexes.

Reading Large SAS Data Sets

Very large SAS data sets can take a long time to process when they are read sequentially.

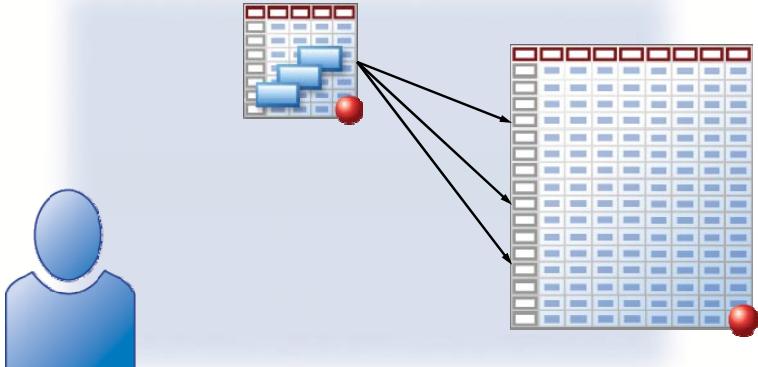


55

...

Reading Large SAS Data Sets

Use an index to locate specific observations directly.

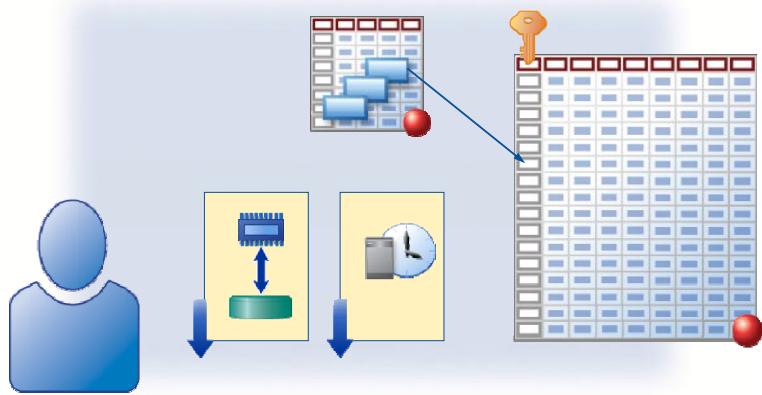


56

...

Reading Large SAS Data Sets

Using an index can save I/O and CPU time.



57

4.02 Multiple Answer Poll

Do any of the data files that you use have indexes?

- a. Yes, my SAS data sets have indexes.
- b. Yes, I use data from an RDBMS (such as Oracle, Teradata, Sybase, or DB2) that has indexes.
- c. No, none of the data that I use has indexes.

58

Overview of SAS Indexes

Indexes are useful in several ways.

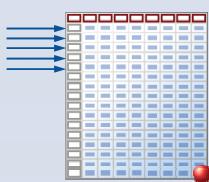
Benefit	Related SAS Element
Access small subsets more quickly	WHERE statement
Return observations in sorted order	BY statement
Perform table lookups	SET statement with KEY= option
Join observations	PROC SQL
Modify observations	MODIFY statement with KEY= option

64

SAS Processing without Indexes

How is data processed if the input data is *not* indexed?

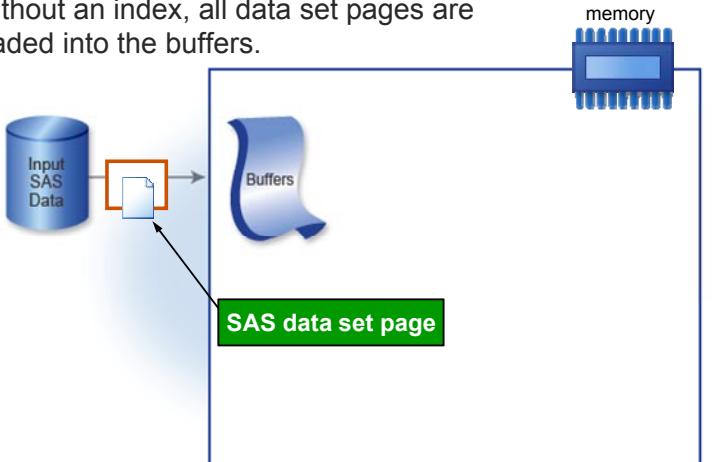
```
data customer14958;
  set orion.saleshistory;
  where CustomerID=14958;
run;
```



65

SAS Processing without Indexes

Without an index, all data set pages are loaded into the buffers.

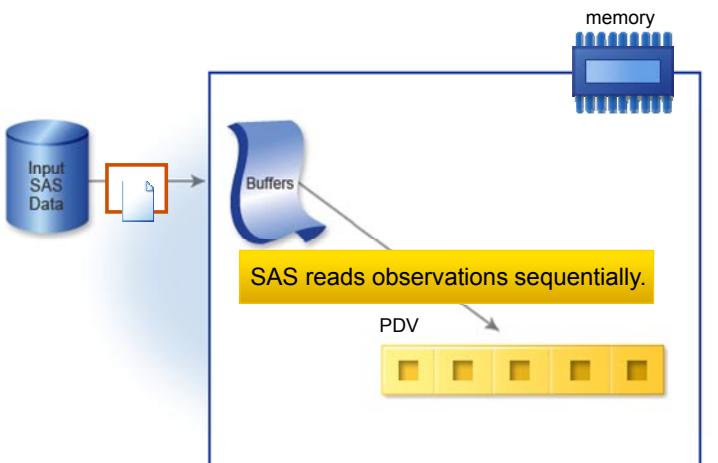


66

...

SAS Processing without Indexes

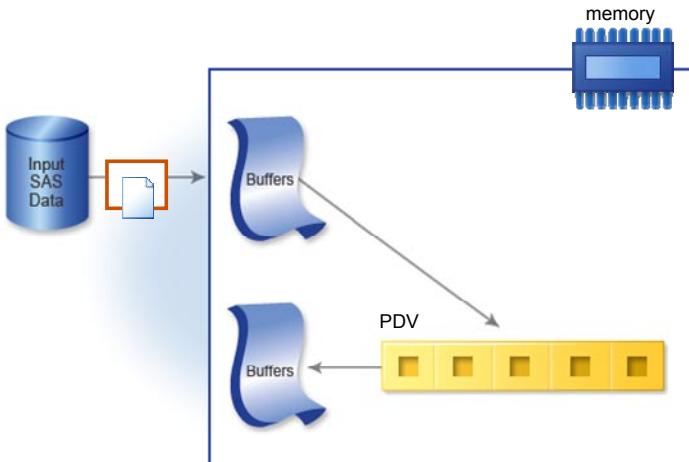
SAS reads observations sequentially.



67

...

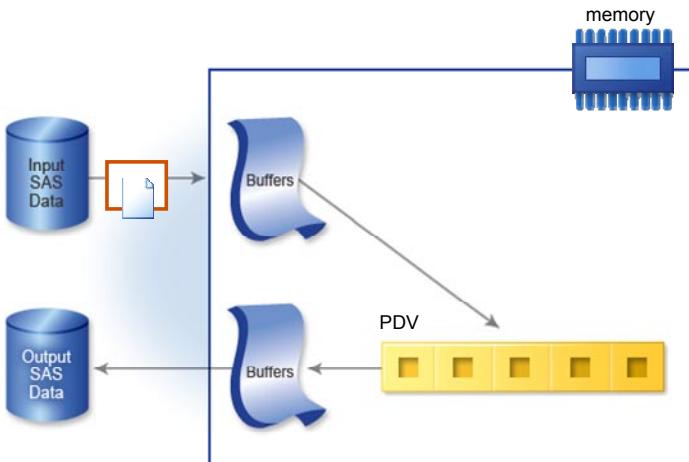
SAS Processing without Indexes



68

...

SAS Processing without Indexes

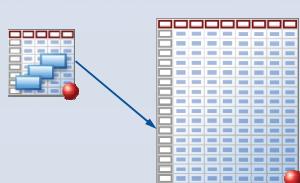


69

SAS Processing with Indexes

How is data processed if the input data *is* indexed?

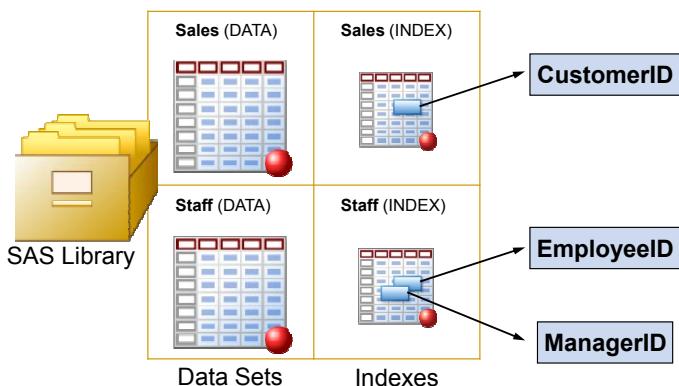
```
data customer14958;
  set orion.saleshistory;
  where CustomerID=14958;
run;
```



70

Location of Indexes

Indexes and the data sets that they index are stored in the same SAS library.



71

Simplified Index File

The index file consists of entries that are stored in order by the key variables.

Partial `orion.saleshistory`

CustomerID	EmployeeID	...
14958	121031	...
14844	121042	...
14864	99999999	...
14909	120436	...
14862	120481	...
14853	120454	...
14838	121039	...
14842	121051	...
14815	99999999	...
14797	120604	...
...

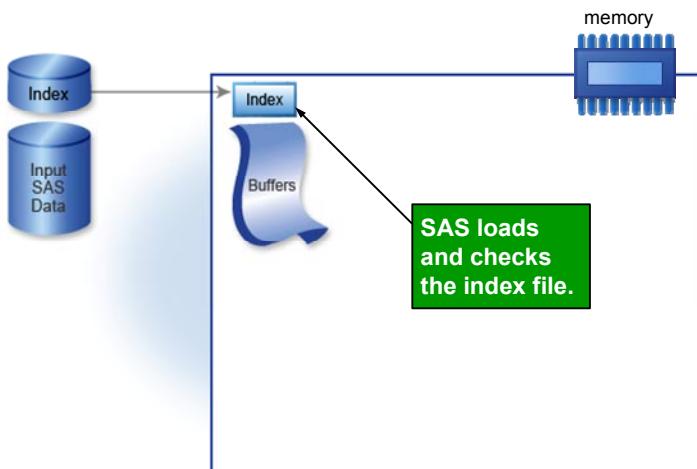
Simplified Index

CustomerID	Record ID (RID)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
...	...
14958	1(1, 24)
14972	1(14)
...	...

- The index is stored with the key values in ascending sorted order.

72

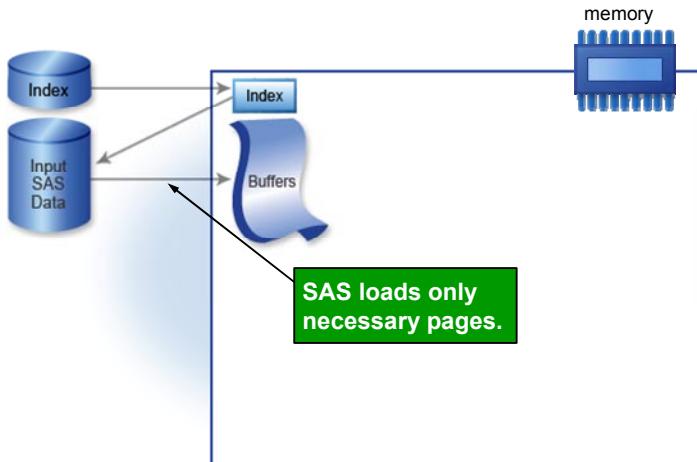
SAS Processing with Indexes



73

...

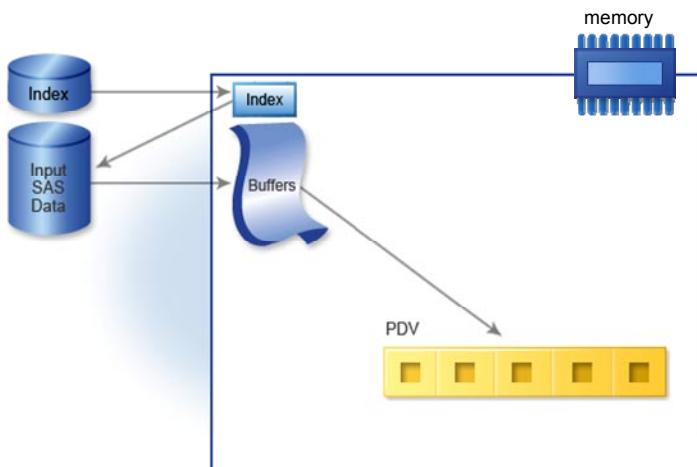
SAS Processing with Indexes



74

...

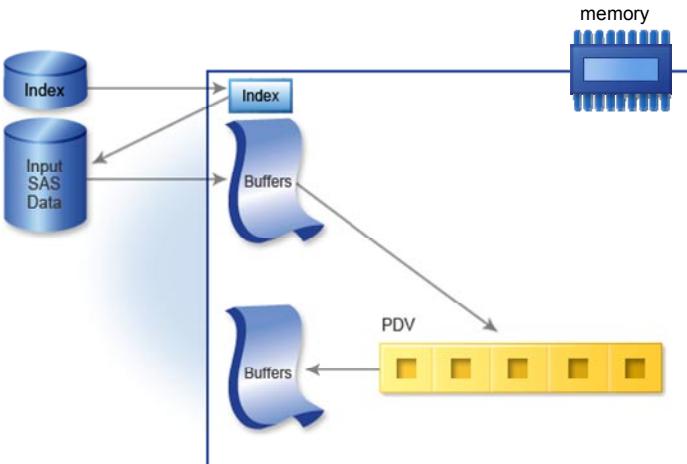
SAS Processing with Indexes



75

...

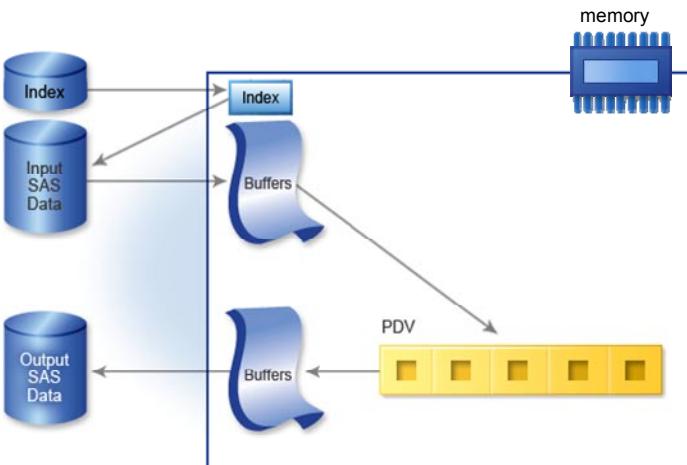
SAS Processing with Indexes



76

...

SAS Processing with Indexes



77

SAS Processing: Checking the Index File

A SAS program contains this WHERE statement.

```
where CustomerID=14958;
```



Simplified Index

	CustomerID	Record Identifier
	4006	17(85)
	4021	17(89)
	4059	17(90)
	4063	17(80, 86)
:	:	:
	14958	1(1, 24)
	14972	1(14)

78

...

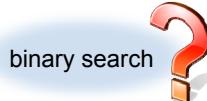
SAS Processing: Checking the Index File

SAS performs a binary search on the index file.



Simplified Index

	CustomerID	Record Identifier
	4006	17(85)
	4021	17(89)
	4059	17(90)
	4063	17(80, 86)
:	:	:
	14958	1(1, 24)
	14972	1(14)



79

...

SAS Processing: Checking the Index File

SAS divides the index into two halves.

```
where CustomerID=14958;
```



Simplified Index

CustomerID	Record Identifier
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
:	:
14958	1(1, 24)
14972	1(14)

80

...

SAS Processing: Checking the Index File

```
where CustomerID=14958;
```

Is 14958 in the top half or the bottom half?



Simplified Index

CustomerID	Record Identifier
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
:	:
14958	1(1, 24)
14972	1(14)

81

...

SAS Processing: Checking the Index File

```
where CustomerID=14958;
```



Simplified Index

CustomerID	Record Identifier
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
:	:
14958	1(1, 24)
14972	1(14)

82

...

SAS Processing: Checking the Index File

```
where CustomerID=14958;
```



Simplified Index

CustomerID	Record Identifier
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
:	:
14958	1(1, 24)
14972	1(14)

83

...

SAS Processing: Checking the Index File

```
where CustomerID=14958;
```

Simplified Index	
CustomerID	Record Identifier
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
:	:
14958	1(1, 24)
14972	1(14)



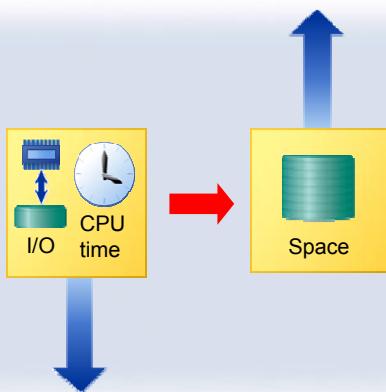
Page

Obs, obs

84

Resource Trade-Offs with Indexes

Using indexes involves trade-offs.



85

4.03 Multiple Choice Poll

If a WHERE statement uses an index to retrieve a small subset of data, which of these resources is conserved?

- a. I/O
- b. disk space
- c. memory
- d. programmer time

86

Types of SAS Indexes

There are two types of indexes.

- simple
- composite

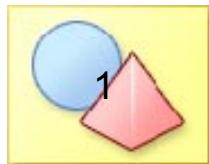
Index Type	Key Variables	Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID ProductID	SaleID

88

Index Options

There are two options for indexes.

- UNIQUE
- NOMISS



89

Index Options: UNIQUE

The concatenation of the values for **OrderID** and **ProductID** forms a unique identifier for a row of data.



Partial `orion.saleshistory`

Customer ID	...	Order ID	Order Type	Product ID	...	Product Group	...
14958	...	1230016296		1	210200600078	...	N.D. Gear, Kids
14844	...	1230096476		1
14864	...	1230028104		2	240600100115	...	Bathing Suits
14909	...	1230044374	SaleID		240100200001	...	Darts

90

Index Options: NOMISS

The missing value of **ProductID** is not included in the index.



Partial orion.saleshistory

Customer ID	...	Order ID	Order Type	Product ID	...	Product Group	...
14958	...	1230016296	1	210200600078	...	N.D. Gear, Kids	...
14844	...	1230096476	1	STOP SIGN
14864	...	1230028104	2	240600100115	...	Bathing Suits	...
14909	...	1230044374	1	240100200001	...	Darts	...

91

4.04 Multiple Answer Poll

On which of these indexed variables can you use the UNIQUE option?

- CustomerID** in an **orders** data set where a customer can place multiple orders
- OrderDate** in an **orders** data set
- EmployeeID** in a data set containing each individual employee and the family members' names stored in the variables **Dependent1** through **Dependent10**
- ProductID** in a data set containing the product identifier and the product description

92

Business Scenario

The SAS data set **orion.saleshistory** is often queried with a WHERE statement. Three indexes are needed to speed up the queries on **orion.saleshistory**.

Partial **orion.saleshistory**

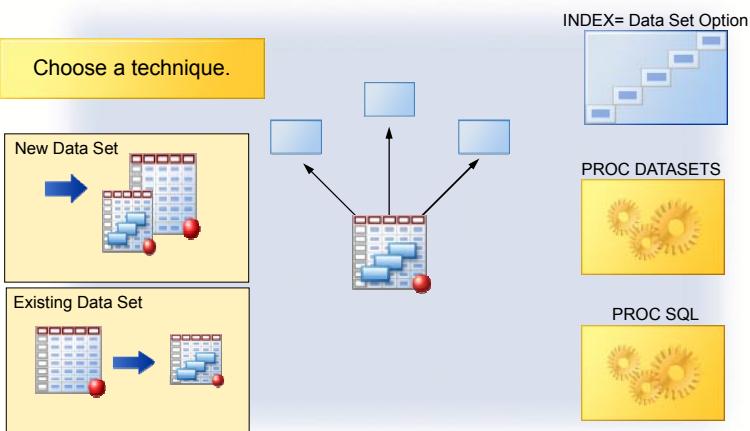
Customer ID	...	Order ID	Order Type	Product ID	...	Product Group	...
14958	...	1230016296	1	210200600078	...	N.D. Gear, Kids	...
14844	...	1230096476	1	220100100354	...	Eclipse Clothing	...
14864	...	1230028104	2	240600100115	...	Bathing Suits	...

Index Type	Key Variables	Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID, ProductID	SaleID

94

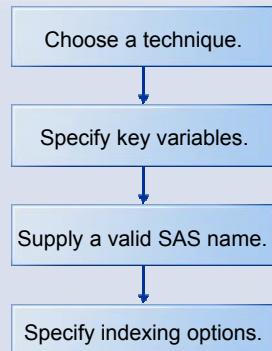
Creating an Index

Choose a technique for creating the index.



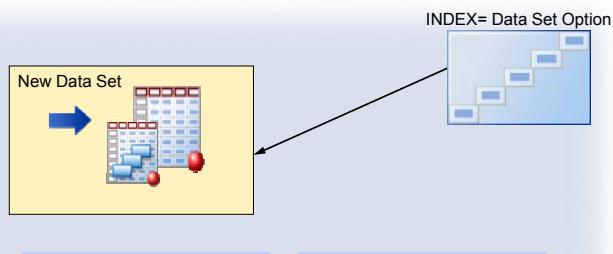
98

Creating an Index



101

Using the INDEX= Data Set Option



Advantages

- ✓ can create the index and data set in one step

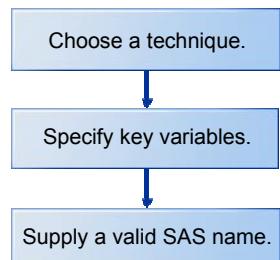
Disadvantages

- ✗ have to plan indexes in advance

102

Using the INDEX= Data Set Option

You already determined the key variables and names for any composite indexes.



Index Type	Key Variables	Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID ProductID	SaleID

103

Using the INDEX= Data Set Option

```

options msglevel=i;
data orion.saleshistory(index=
                           (CustomerID ProductID
                            SaleID=(OrderID ProductID) /
                            unique));
set orion;
ValueCost=YearMonth;
INDEX=(index-specification-1 ...<index-specification-n>)
      input(YearID,4.);
format ValueCost dollar12.
      YearMonth monyy7.;
label ValueCost="Value Cost"
      YearMonth="Month/Year";
run;
  
```

104

p304d03

...

Using the MSGLEVEL=I SAS System Option

```

options msglevel=i;
data orion.sale;
  OPTIONS MSGLEVEL=N||;
  SaleID=(OrderID ProductID) /
CustomerID=YearMonth monyy/.
  label ValueCost="Value Cost"
        YearMonth="Month/Year";
run;

```

Partial SAS Log

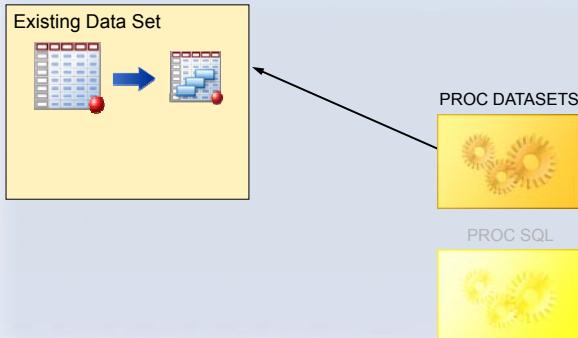
NOTE: There were 1500 observations read from the data set ORION.HISTORY.
 NOTE: The data set ORION.SALESHISTORY has 1500 observations and 22 variables.
 NOTE: Composite index SaleID has been defined.
 NOTE: Simple index ProductID has been defined.
 NOTE: Simple index CustomerID has been defined.
 NOTE: DATA statement used (Total process time):
 real time 1.57 seconds
 cpu time 0.17 seconds

105

p304d03

Using PROC DATASETS

To create or delete an index on an existing data set,
 you can use PROC DATASETS.



106

Using PROC DATASETS

Here is an example of how to create an index with PROC DATASETS.

```
options msglevel=n;
proc datasets library=orion nolist;
  modify saleshistory;
    index create CustomerID;
    index create ProductID;
    index create SaleID=(OrderID
      ProductID) /unique;
quit;

PROC DATASETS LIBRARY=libref NOLIST;
  MODIFY SAS-data-set;
    INDEX CREATE index-specification-1;
    INDEX CREATE index-specification-2;
    INDEX CREATE index-specification-3;
QUIT;
```

107

p304d04

Using PROC DATASETS

PROC DATASETS writes messages about index creation and deletion. The MSGLEVEL= option is not needed.

```
options msglevel=n;
proc datasets library=orion nolist;
  modify saleshistory;
    index create CustomerID;
    index create ProductID;
    index create SaleID=(OrderID
      ProductID) /unique;
quit;
```

108

p304d04

...

Using PROC DATASETS

This PROC DATASETS step deletes the three specified indexes on **orion.saleshistory**.

```
proc datasets library=orion nolist;
  modify saleshistory;
    index delete CustomerID
          ProductID SaleID;
  quit;
PROC DATASETS LIBRARY=/libref NOLIST;
  MODIFY SAS-data-set;
    INDEX DELETE index-1 <...index-n> | _ALL_;
  QUIT;
```

109

p304d05

4.05 Quiz

Open and submit the program **p304a01**.

What error messages are in the log?

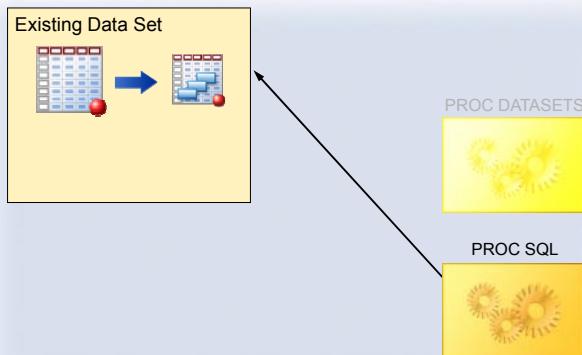
p304a01

```
options msglevel=n;
proc datasets library=orion nolist;
  modify saleshistory;
    index create CustomerID;
    index create ProductGroup;
    index create SaleID=(OrderID
                      ProductID) /unique;
  quit;
```

110

Using PROC SQL

To create or drop an index onto an existing data set, you can use PROC SQL.



112

Using PROC SQL

This shows how to create an index using PROC SQL.

```

proc sql;
  create index CustomerID
    on orion.saleshistory(CustomerID);
  create index ProductGroup
    on orion.saleshistory(ProductGroup);
  create unique index SaleID
    on orion.saleshistory(OrderID,
    ProductID);
quit;

PROC SQL;
  CREATE <UNIQUE> INDEX index-name
    ON table-name(column-name-1,...  

      column-name-n);
  QUIT;
  
```

113

p304d06

...

Using PROC SQL

```
proc sql;
  create index CustomerID
    on orion.saleshistory(CustomerID);
  create index ProductGroup
    on orion.saleshistory(ProductGroup);
  create unique index SaleID
    on orion.saleshistory(OrderID,
      ProductID);
quit;
```

```
PROC SQL;
  CREATE <UNIQUE> INDEX index-name
    ON table-name(column-name-1, ...
      column-name-n);
QUIT;
```

114

p304d06

...

Using PROC SQL

```
proc sql;
  create index CustomerID
    on orion.saleshistory(CustomerID);
  create index ProductGroup
    on orion.saleshistory(ProductGroup);
  create unique index SaleID
    on orion.saleshistory(OrderID,
      ProductID);
quit;
```

```
PROC SQL;
  CREATE <UNIQUE> INDEX index-name
    ON table-name(column-name-1, ...
      column-name-n);
QUIT;
```

115

p304d06

...

Using PROC SQL

```

proc sql;
  create index CustomerID
    on orion.saleshistory(CustomerID);
  create index ProductGroup
    on orion.saleshistory(ProductGroup);
  create unique index SaleID
    on orion.saleshistory(OrderID,
quit; 39  proc sql;
40    create index CustomerID
41      on orion.saleshistory(CustomerID);
NOTE: Simple index CustomerID has been defined.
42 create index ProductGroup
43      on orion.saleshistory(ProductGroup);
NOTE: Simple index ProductGroup has been defined.
44   create unique index SaleID
45     on orion.saleshistory(OrderID, ProductID);
NOTE: Composite index SaleID has been defined.
46 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.15 seconds
      cpu time           0.01 seconds

```

116

p304d06

Using PROC SQL

```

proc sql;
  drop index CustomerID, ProductGroup, SaleID
    from orion.saleshistory;
quit;

```

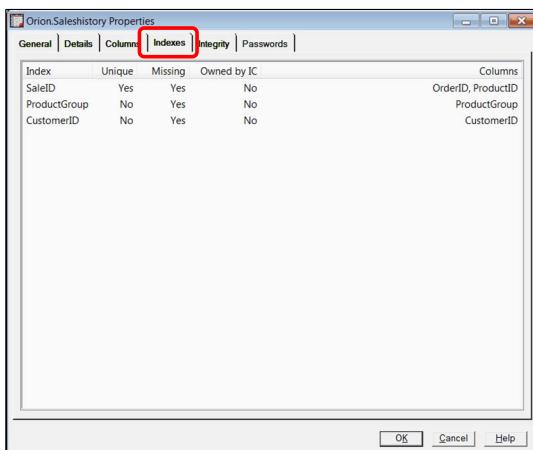
PROC SQL;
DROP INDEX *index-name-1*, ... *index-name-n*
FROM *table-name*;
QUIT;

117

p304d07

Documenting Indexes

The Indexes tab in the data set Properties window can be used to document indexes.



118

Documenting Indexes

The following can also be used to document indexes:

- PROC CONTENTS

```
proc contents data=orion.saleshistory;
run;
```
- PROC DATASETS

```
proc datasets lib=orion nolist;
contents data=saleshistory;
quit;
```

Partial Output

Alphabetic List of Indexes and Attributes

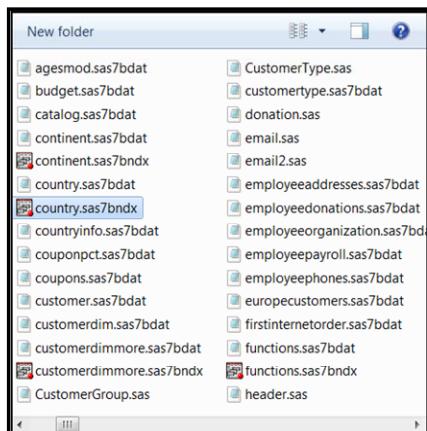
#	Index	Unique Option	# of Unique Values	Variables
1	CustomerID		1046	
2	ProductGroup		56	
3	SaleID	YES	1500	OrderID ProductID

119

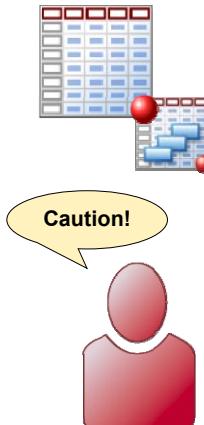
p304d08

Avoiding Damage to Indexes and Data Sets

Deleting an index file using operating system commands damages the data set.



120



4.06 Multiple Answer Poll

If you manage an index using operating environment features, what problems might occur?

- The index file might be damaged.
- Files created outside SAS might be damaged.
- The data set associated with the index file might be damaged.

121



Exercises

Level 1

4. Creating Indexes

- Open the program **p304e04**, and add the INDEX= option to create two indexes.

- a simple index **CustomerID**, based on the variable **CustomerID**
 - a unique index **OrderID**, based on the variable **OrderID**
- b. Add the SAS system option that enables you to verify that your indexes were created.
 - c. Submit the program and examine the SAS log.
 - d. Use PROC SQL to delete the **OrderID** index from the **orders** data set.
 - e. Submit the PROC SQL step and examine the SAS log.
 - f. Use PROC DATASETS to create a composite index named **OrDate** based on the **OrderID** and **OrderDate** variables for the **orders** data set.
 - g. Submit the PROC DATASETS step and examine the SAS log.
 - h. Use PROC CONTENTS or PROC DATASETS to look at the index information.
 - i. Turn off the SAS system option that enables you to verify that your indexes were created.

Level 2

5. Updating Indexes

- a. Use the **orion.pricelist** SAS data set to create a temporary data set named **pricelist** that contains a new variable named **UnitProfit**. The new variable is the difference between the variables **UnitSalesPrice** and **UnitCostPrice**. Create a unique index on the **ProductID** variable.

 Be sure to turn on the SAS system option that enables you to see that your unique index is created.
 - b. Submit your program and examine the log.
 - c. Open the program **p304e05** and submit it.
 - d. View the log and determine whether the new observation was added.
 - e. Why or why not?
-
-

Challenge

6. Creating Indexes on New Variables

- a. Create a temporary SAS data set named **allstaff** by concatenating the data sets **orion.sales** and **orion.nonsales**. **Work.allstaff** has the following characteristics:
 - Create a new variable named **AgeHired**. Use the INTCK function and the optional function argument that is the number of years between the variables **HireDate** and **BirthDate**.
 - Index the **allstaff** data set on the variable **AgeHired**.

Hint: Rename the variables **First** and **Last** in **orion.nonsales** so that they are consistent with the variables' names **FirstName** and **LastName** in **orion.sales**.

- b. Submit the program and examine the SAS log.
- c. Print the observations in **work.allstaff** where **AgeHired** is greater than 30. Print only the variables **EmployeeID**, **BirthDate**, **HireDate**, and **AgeHired**. Verify that the **AgeHired** index was used to retrieve the selected observations.

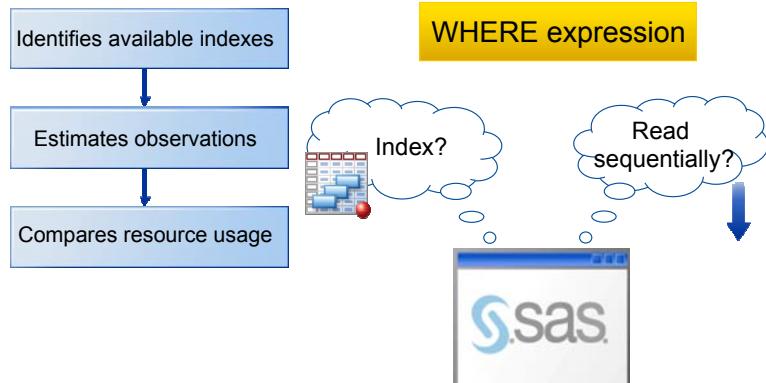
4.4 Using an Index

Objectives

- Describe when an index is used for WHERE statement processing.
- Control index usage with WHERE statement processing using data set options.
- Use index maintenance techniques.

Determining Index Usage for WHERE Processing

SAS follows these steps to decide whether to use an index.



127

Determining Index Usage for WHERE Processing

SAS determines whether there is an available index or indexes that can be used.



128

Identify Available Index

SAS might use an index when a WHERE expression references either a simple index key variable ...

Index Type	Key Variables	Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID, ProductID	SaleID

```
where CustomerID < 4100;
```

129

Identify Available Index

... or the primary key variable of a composite index.

Index Type	Key Variables	Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID, ProductID	SaleID

```
where OrderID < 1230063235;
```

130

Identify Available Index

Only one index is used, even if the WHERE expression contains multiple conditions specifying different variables.

		Index Name
Simple	CustomerID	CustomerID
Simple	ProductID	ProductID
Composite	OrderID, ProductID	SaleID

```
where CustomerID < 4100 and OrderID < 1230063235;
```

131

Setup for the Poll

The following indexes were created on the **orion.saleshistory** data set.

Partial PROC DATASETS Output

Alphabetic List of Indexes and Attributes					
#	Index	Unique Option	# of Unique Values	Variables	
1	CUSTOMERID		1046		
2	ProductGroup		56		
3	SaleID	YES	1500	OrderID ProductID	

132

4.07 Multiple Answer Poll

Which of the following WHERE conditions could possibly use an index?

- a. `where ProductID=230100700004;`
- b. `where CustomerID=15020 or CustomerID=14853;`
- c. `where OrderID=1230036183;`

133

Identify Available Index: Compound Optimization

Compound optimization is the process of optimizing multiple WHERE expression conditions using a single composite index.



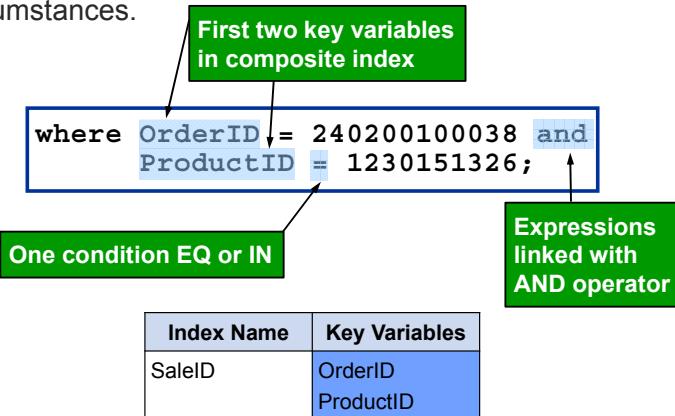
`where OrderID=240200100038 and ProductID=1230151326;`

Index Name	Key Variables
SaleID	OrderID ProductID

135

Identify Available Index: Compound Optimization

Compound optimization can occur only in certain circumstances.



136

When SAS Does Not Use Indexes

- No single index can supply all required observations.

```
where ProductGroup=ProductCategory;
```

- Any function other than TRIM or SUBSTR appears in the WHERE expression.

```
where scan(ProductGroup,1)="Eclipse";
```

- The SUBSTR function searches a string beginning at a position other than the first position.

```
where substr(ProductGroup, 9) = "Clothes";
```

137

continued...

When SAS Does Not Use Indexes

- The SOUNDS-LIKE operator (=*) appears in the WHERE expression.

```
where ProductGroup=*"Eclipse";
```

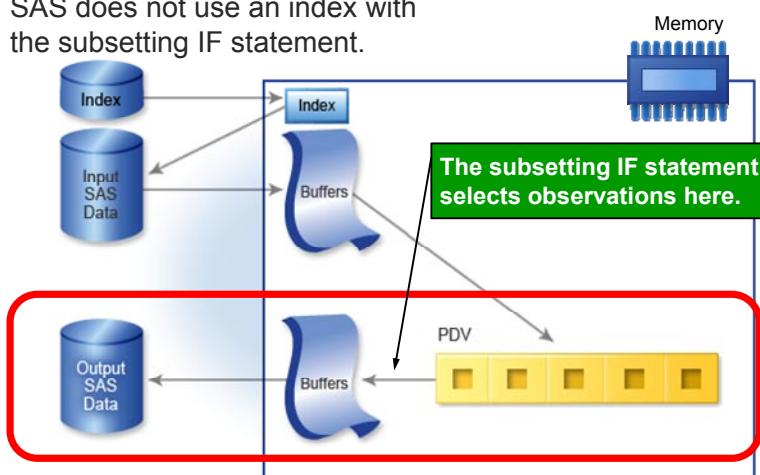
- A subsetting IF statement is used.

```
if ProductGroup="Eclipse";
```

138

When SAS Does Not Use Indexes

SAS does not use an index with the subsetting IF statement.



143

4.08 Multiple Choice Poll

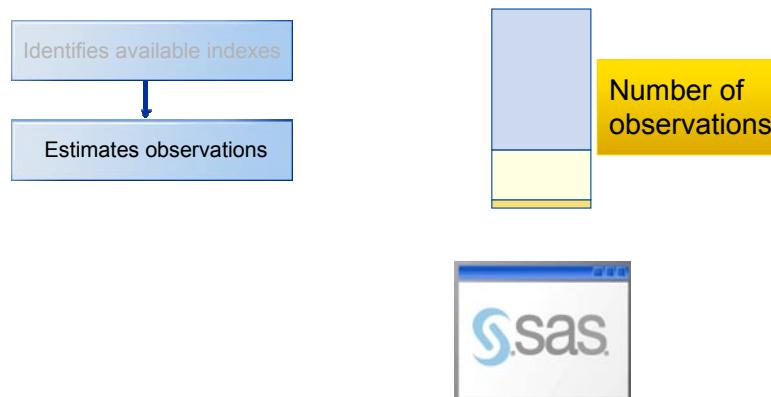
Which of the following WHERE statements can use the composite index **SaleID** for compound optimization?

- a. `where OrderID=240200100038 or ProductID=1230151326;`
- b. `where OrderID=. and ProductID=1230151326;`
- c. `where int(OrderID/1000000000)=240 and ProductID=1230151326;`
- d. `where OrderID>240000000000 and ProductID<1240000000;`

144

Determining Index Usage for WHERE Processing

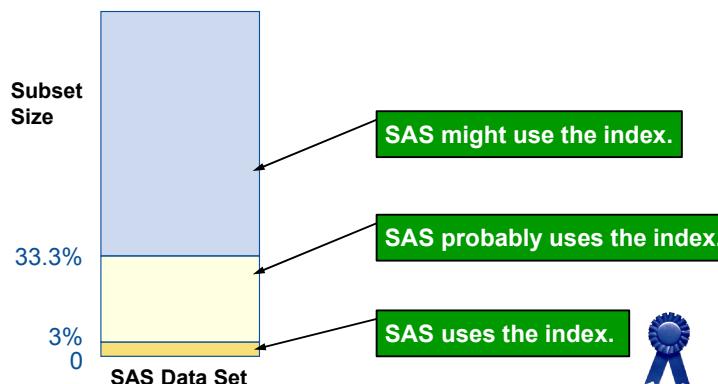
SAS estimates the number of qualified observations.



146

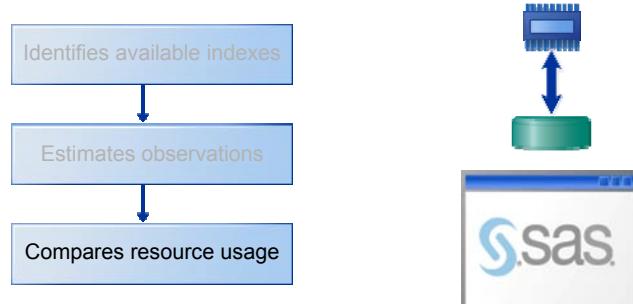
Compare Resource Usage

Subset size determines the probability of using an index.



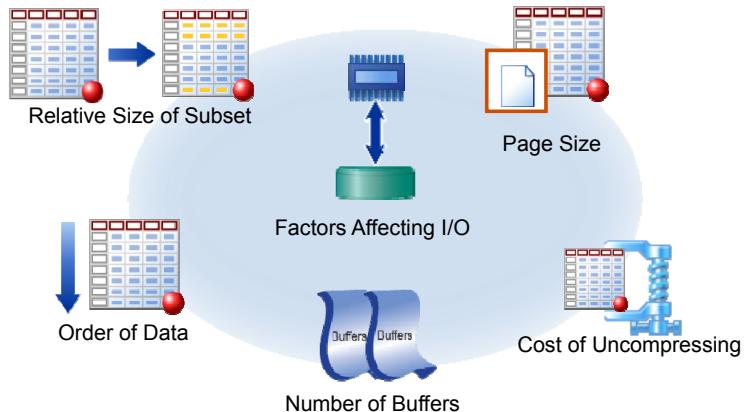
Determining Index Usage for WHERE Processing

SAS predicts the I/O usage.



Compare Resource Usage

Here are factors that affect I/O.



149

Compare Resource Usage

If the data is stored in sorted order, a smaller number of I/Os are needed to use the index because fewer data set pages are read.

```
where CustomerID=13337;
```

Sorted Data Set

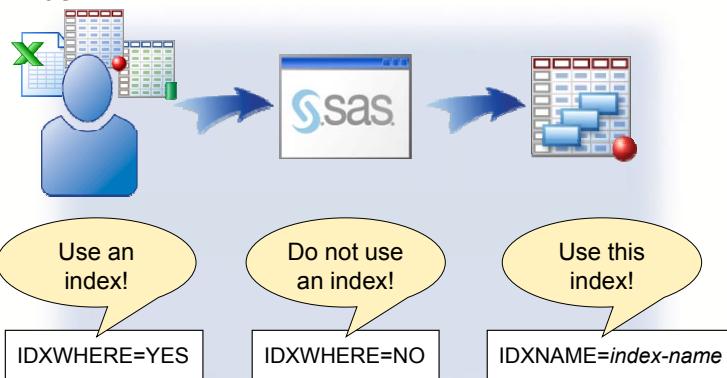
Obs	Customer_ID
1286	13337
1287	13337
1288	13337
1291	13350
1292	13350



150

Controlling Index Usage for WHERE Processing

Data set options enable you to direct SAS when to use an index.



156

Controlling Index Usage for WHERE Processing

IDXWHERE=YES ensures that SAS uses an index.

```
options msglevel=i;
proc print data=orion.saleshistory
  (idxwhere=yes);
  where CustomerID in (14844,4983,5862,10032)
    and ProductGroup contains 'Shoes';
  var CustomerID ProductID ProductGroup ;
  title 'With an Index';
run;
```



157

p304d09

Controlling Index Usage for WHERE Processing

SAS decides which index is the best.

Partial SAS Log

```
1669 options msglevel=i;
1670 proc print data=orion.saleshistory(idxwhere=yes);
1671   where CustomerID in (14844,4983,5862,10032)
1672     and ProductGroup contains 'Shoes';
INFO: Data set option (IDXWHERE=YES) forced an index to be used rather
      than a sequential pass for where-clause processing.
INFO: Index CustomerID selected for WHERE clause optimization.
1673   var CustomerID ProductID ProductGroup ;
1674   title 'With an Index';
1675 run;
```

Use an index!

IDXWHERE=YES

158

p304d09

Controlling Index Usage for WHERE Processing

IDXNAME= forces SAS to use a specific index. SAS disregards the possibility that a sequential search of the data set might be more resource efficient.

```
options msglevel=i;
proc print data=orion.saleshistory
            (idxname=CustomerID);
  where CustomerID in (14844,4983,
                        5862,10032)
    and ProductGroup contains 'Shoes';
  var CustomerID ProductID ProductGroup;
  title 'With an Index';
run;
```

Use this index!

IDXNAME=index-name

159

p304d10

Controlling Index Usage for WHERE Processing

Partial SAS Log

```

92 options msglevel=I;
193 proc print data=orion.sales_history(idxname=CustomerID);
194   where CustomerID in (14844,4983,5862,10032)
195     and ProductGroup contains 'Shoes';
INFO: Index CustomerID selected for WHERE clause optimization.
196   var CustomerID Product_ID ProductGroup ;
197   title 'With an Index';
198 run;

NOTE: There were 3 observations read from the data set ORION.SALES_HISTORY.
      WHERE CustomerID in (4983, 5862, 10032, 14844) and
            ProductGroup contains 'Shoes';
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.15 seconds
      cpu time           0.01 seconds

```

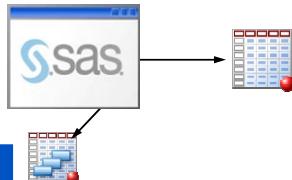
- IDXWHERE= and IDXNAME= are mutually exclusive.

160

p304d10

Maintaining Indexes

SAS maintains the index as long as you use a method that updates in place.



Update Data in Place Methods

PROC APPEND

INSERT INTO statement
in PROC SQL

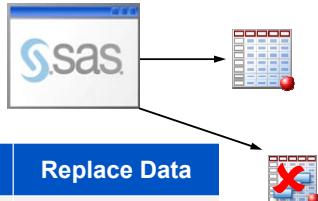
MODIFY statement
in the DATA step

161

...

Maintaining Indexes

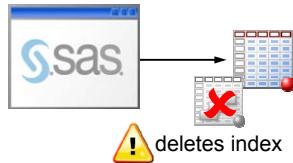
The SET and MERGE statements replace data, so SAS does not maintain the index.



Update Data in Place Methods	Replace Data
PROC APPEND	SET statement
INSERT INTO statement in PROC SQL	MERGE statement
MODIFY statement in the DATA step	

162

Maintaining Indexes



Task	Example
Delete a data set	<code>proc datasets lib=work;</code> <code> delete a;</code> <code>run;</code>
Rebuild a data set with a DATA step	<code>data a;</code> <code> set a;</code> <code>run;</code>
Sort the data set in place using the FORCE option in PROC SORT	<code>proc sort data=a force;</code> <code> by var;</code> <code>run;</code>

163

Maintaining Indexes

To create a sorted copy of an indexed data set and maintain its index, use the INDEX= option on the output file.

Must use OUT=

Must use INDEX= to index

```
options msglevel=i;
proc sort data=orion.saleshistory
            out=salessorted(index=(CustomerID));
by CustomerID ProductID;
run;
```

164

p304d11

Guidelines for Indexing

DO This	DO NOT Do This
Index to retrieve a small subset of observations from a large data set.	Index a data set with fewer than three data file pages.
Index on variables that are discriminating; make the first key variable the most discriminating.	Index on variables with few unique values.
Create and maintain indexes that are used frequently.	Index frequently changing data unless it is worth the resources.
Sort data before indexing.	Create indexes on rarely used variables.
Minimize the number of indexes.	Try to create a single index for all queries.

165



Exercises

Level 1

7. Using an Index

- Open the program p304e07.

p304e07

```

options msglevel=I;
*** Example 1;
data rdu;
  set orion.saleshistory;
  if OrderID=1230166613;
run;
*** Example 2;
proc print data=orion.saleshistory;
  where OrderID=1230166613 or ProductID=220200100100;
run;
*** Example 3;
proc print data=orion.saleshistory;
  where ProductGroup ne 'Shoes';
run;
*** Example 4;
proc print data=orion.saleshistory;
  where CustomerID=12727;
run;
**** Example 5;
data saleshistorycopy;
  set orion.saleshistory;
run;
options msglevel=n;

```

- b. Submit the SAS code. If **orion.saleshistory** does not have the indexes **orderid**, **productgroup**, and **saleid**, use the program code in the comments to create them.
- c. Examine the SAS log.
- d. Answer the following questions:

- 1) Does Example 1 use an index? Why or why not?

Replace the IF statement with a WHERE statement, and resubmit the program. Does the example now use an index? Why or why not?

- 2) Does Example 2 use an index? Why or why not?

Replace the OR operator with the AND operator, and resubmit the program. Does the example now use an index? Why or why not?

- 3) Does Example 3 use an index? Why or why not?
-
-

Replace the NE operator with the EQ operator, and resubmit the program. Does the example now use an index? Why or why not?

- 4) Does Example 4 use an index? Why or why not?
-
-

Add the IDXWHERE=NO data set option and resubmit the program. Is the output from the PROC PRINT step with an index different from the output from the PROC PRINT step without an index? _____

What message do you see in the log?

- 5) In Example 5, does the data set **saleshistorycopy** have an index? Why or why not?
-
-

Level 2

8. Suppressing Index Usage

Create a detail report from the **orion.supplier** SAS data set that lists all of the variables and observations where **SupplierID** is greater than 1000. Ensure that the data is processed sequentially.

Partial PROC PRINT Output

SupplierID	SupplierName	StreetID	SupplierAddress	Street Number	Country
1280	British Sports Ltd	9250100844	85 Station Street	85	GB
1303	Eclipse Inc	9260107621	1218 Carriole Ct	1218	US
1684	Magnifico Sports	7350100062	Rua Costa Pinto 2	2	PT
1747	Pro Sportswear Inc	9260109782	2434 Edgebrook Dr	2434	US
2963	3Top Sports	9260111024	5033 Charles B Root Wynd	5033	US
2995	Van Dammeren International	6300100165	Transistorstraat 6	6	NL

Hint: There should be 46 observations read from **orion.supplier**.

Challenge

9. Updating Centile Information for an Index

- Submit program **p304e09**.

p304e09

```
data orders(index=(OrderID/unique CustomerID));
  set orion.orders;
run;
proc contents data=orders centiles;
run;
```

- Using the DATASETS procedure, specify that the centile information about the **CustomerID** index should be updated when more than 1% of the data changes.
- Refresh the **CustomerID** index.

Hint: REFRESH is an option for the INDEX CENTILES statement in PROC DATASETS.

- Submit the program **p304e09c**, which adds new observations to the orders data set.

p304e09c

```
data neworders;
  set orion.orders(obs=5);
  month=month(orderdate);
  day=day(orderdate);
  orderdate=mdy(month,day,2008);
  month=month(deliverydate);
  day=day(deliverydate);
  deliverydate=mdy(month,day,2008);
  orderid+14239000;
  drop month day;
run;
proc append base=orders data=neworders;
run;
```

- Submit a PROC CONTENTS step to view the contents of **orders**. Compare the centile information for the **CustomerID** index from step 9.a. to the current **CustomerID** index centile information. Were the centiles updated?

Why or why not?

4.5 Solutions

Solutions to Exercises

1. Generating a Systematic Sample

- a. Generate a systematic sample. Select every 10th supplier. Start with observation 10 from the data set **orion.productdim**.

- 1) Enter the following DATA step in the editor:

Partial p304s01

```
data productsample;
  do i=10 to TotObs by 10;
    set orion.productdim(keep=ProductLine ProductID
                           ProductName SupplierName)
        nobs=TotObs
        point=i;
    output;
  end;
  stop;
run;
```

- 2) Submit the SAS code in the editor.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
122 data productsample;
123   do i=10 to TotObs by 10;
124     set orion.productdim(keep=ProductLine ProductID
125                           ProductName SupplierName)
126       nobs=TotObs
127       point=i;
128     output;
129   end;
130   stop;
131 run;

NOTE: The data set WORK.PRODUCTSAMPLE has 48 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.15 seconds
      cpu time           0.01 seconds
```

- b. Print the first five observations of **productssample**. Omit the observation numbers.

- 1) Enter the following PROC PRINT step in the editor:

Partial p304s01

```
proc print data=productssample(obs=5) noobs;
  title "Systematic Sample of Products";
run;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.

PROC PRINT Output

Systematic Sample of Products			
ProductID	ProductLine	ProductName	SupplierName
210200500002	Children	Children's Mitten	AllSeasons Outdoor Clothing
210200900033	Children	Osprey France Nylon Shorts	Triple Sportswear Inc
220100100044	Clothes & Shoes	Sports glasses Satin Alumin.	Eclipse Inc
220100100241	Clothes & Shoes	Big Guy Men's Santos Shorts Dri Fit	Eclipse Inc
220100100513	Clothes & Shoes	Woman's Deception Dress	Eclipse Inc

2. Retrieving the Top Five and Bottom Five Observations

- a. Create two data sets, **highest** and **lowest**. **Work.highest** contains the five managers with the highest combined salary groups. **Work.lowest** contains the five managers with the lowest combined salary groups. Use **orion.totalsalaries** to retrieve the observations. Each observation in **orion.totalsalaries** represents one manager's salaries for all of his or her employees.

```

proc sort data=orion.totalsalaries
           out=SortedTotalSalaries;
   by DeptSal;
run;

data lowest highest;
  do ReadOBS=1 to 5;
    set SortedTotalSalaries;
    output lowest;
  end;
  do ReadOBS=(TotOBS) to (TotOBS - 4) by -1;
    set SortedTotalSalaries nobs=TotObs point=ReadOBS;
    output highest;
  end;
  stop;
run;

proc print data=highest noobs;
  title 'Top 5 Salary Groups';
run;

proc print data=lowest noobs;
  title 'Bottom 5 Salary Groups';
run;

title;

```

- b. Print all observations from **highest** and **lowest**.

Top 5 Salary Groups

	ManagerID	Numemps	DeptSal
	121143	51	\$1,410,530
	120102	48	\$1,344,595
	121145	45	\$1,216,055
	120259	6	\$941,155
	120103	30	\$793,835

Bottom 5 Salary Groups

	ManagerID	Numemps	DeptSal
	120748	1	\$26,545
	120714	1	\$28,535
	120666	1	\$29,980
	120712	1	\$31,630
	120672	1	\$35,935

3. Generating a Random Sample with Replacement

- a. Generate a random sample with replacement of 50 customers from **orion.customerdim**. If the customer age is less than 40, place those customers in a data set named **underforty**. If the customer age is greater than or equal to 40, place the customers in a data set named **fortyplus**.

- 1) Enter the following DATA step in the editor:

Partial p304s03

```
data underforty fortyplus;
  drop i SampSize;
  SampSize=50;
  do i=1 to SampSize;
    PickIt=ceil(ranuni(0) * TotObs);
    set orion.customer_dim point=PickIt nobs=TotObs;
    if CustomerAge < 40 then output underforty;
    else output fortyplus;
  end;
  stop;
run;
```

- 2) Submit the SAS code in the editor.

- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
143 data underforty fortyplus;
144   drop i SampSize;
145   SampSize=50;
146   do i=1 to SampSize;
147     PickIt=ceil(ranuni(0) *TotObs);
148     set orion.customerdim point=PickIt nobs=TotObs;
149     if CustomerAge < 40 then output underforty;
```

```

150      else output fortyplus;
151      end;
152      stop;
153 run;

NOTE: The data set WORK.UNDERFORTY has 27 observations and 11 variables.
NOTE: The data set WORK.FORTYPLUS has 23 observations and 11 variables.
NOTE: DATA statement used (Total process time):
      real time          0.23 seconds
      cpu time           0.06 seconds

```



The number of observations in your data sets probably varies depending on the seed that you provide.

- b.** Print both data sets. Suppress the observation number, show only the **CustomerID** and **CustomerName** variables, and print only the first five observations from each data set.

- 1) Enter the following two PRINT steps in the editor:

Partial **p304s03**

```

proc print data=fortyplus noobs;
  title 'Customers Forty or Older';
  var CustomerID CustomerName;
run;

proc print data=underforty noobs;
  title 'Customers under Forty';
  var CustomerID CustomerName;
run;

```

- 2) Submit the two PROC PRINT steps.
- 3) View the output to verify it matches the expected output.

PROC PRINT Output

Customers Forty or Older

CustomerID	CustomerName
183	Duncan Robertshawe
17	Jimmie Evans
17	Jimmie Evans
56	Roy Siferd
2618	Theunis Brazier

Customers under Forty

CustomerID	CustomerName
79	Najma Hicks
70100	Wilma Yeargan
12	David Black
9	Cornelia Krahel
39	Alphone Greenwald



The first five observations in your data sets probably vary from the observations displayed above.

4. Creating Indexes

- a. Open the program **p304e04**, and add the INDEX= option to create two indexes:

- a simple index **Customer_ID**, based on the variable **Customer_ID**
- a unique index **Order_ID**, based on the variable **Order_ID**

1) Open **p304e01**.

2) Enter the following INDEX= data set option to modify the **work.orders** data set:

(index=(CustomerID OrderID/unique))

- b. Add the SAS system option that enables you to verify that your indexes were created.

Enter **options msglevel=i;** before the DATA step.

Partial **p304s04**

```
options msglevel=i;
data orders(index=(CustomerID OrderID/unique));
  set orion.orders;
  DaysToDelivery=DeliveryDate-OrderDate;
run;
```

- c. Submit the program and examine the SAS log.

1) Submit the SAS code in the Editor window.

2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
45  options msglevel=i;
46  data orders(index=(CustomerID OrderID /  unique));
47    set orion.orders;
48    DaysToDeliver=DeliveryDate-OrderDate;
49  run;

NOTE: There were 490 observations read from the data set ORION.ORDERS.
NOTE: The data set WORK.ORDERS has 490 observations and 7 variables.
NOTE: Simple index OrderID has been defined.
NOTE: Simple index CustomerID has been defined.
NOTE: DATA statement used (Total process time):
      real time          0.02 seconds
      cpu time           0.01 seconds
```

- d. Use PROC SQL to delete the **OrderID** index from the **orders** data set.

Enter the following PROC SQL step in the editor:

Partial **p304s04**

```
proc sql;
  drop index OrderID
  from orders;
```

```
quit;
```

- e. Submit the PROC SQL step and examine the SAS log.
- 1) Submit the PROC SQL step in the Editor window.
 - 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
53  proc sql;
54    drop index OrderID
55      from orders;
NOTE: Index OrderID has been dropped.
56  quit;
```

- f. Use PROC DATASETS to create a composite index named **OrDate** based on the **OrderID** and **OrderDate** variables for the **orders** data set.

Enter the following PROC DATASETS code in the editor.

Partial p304s04

```
proc datasets library=work nolist;
  modify orders;
  index create OrDate=(OrderID OrderDate);
quit;
```

- g. Submit the PROC DATASETS step and examine the SAS log.
- 1) Submit the PROC DATASETS step in the Editor window.
 - 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
57  proc datasets library=work nolist;
58    modify orders;
59    index create OrDate=(OrderID OrderDate);
NOTE: Composite index OrDate has been defined.
60  quit;
```

- h. Use PROC CONTENTS or PROC DATASETS to look at the index information.

- 1) Enter one of the following steps in the editor.

Partial p304s04

```
/* CONTENTS solution */
proc contents data=orders;
run;
/* DATASETS solution */
proc datasets library=work nolist;
  contents data=orders;
quit;
```

- 2) Submit the PROC step and examine the output.
- 3) View the output to verify that it matches the expected output.

- 4) Scroll to the **Alphabetic List of Index and Attributes** section.

Partial PROC CONTENTS or PROC DATASETS Output

Alphabetic List of Indexes and Attributes			
#	Index	# of Unique Values	Variables
1	CustomerID	75	
2	OrDate	490	OrderID OrderDate

- i. Turn off the SAS system option that enables you to verify that your indexes were created.

- 1) Enter the following OPTIONS statement in the Editor window:

Partial p304s04

```
options msglevel=n;
```

- 2) Submit the OPTIONS statement.

5. Updating Indexes

- a. Use the **orion.pricelist** SAS data set to create a temporary data set named **pricelist** that contains a new variable named **UnitProfit**. **UnitProfit** is the difference between the variables **UnitSalesPrice** and **UnitCostPrice**. Create a unique index on the **ProductID** variable.

 Be sure to turn on the SAS system option that enables you to see that your unique index is created.

Enter the following SAS code in the editor:

p304s05

```
options msglevel=i;

data pricelist(index=(ProductID/unique));
  set orion.pricelist;
  UnitProfit=UnitSalesPrice-UnitCostPrice;
run;

options msglevel=n;
```

- b. Submit the SAS code in the editor.

Examine the SAS log to verify that the program executed without errors.

SAS Log

```
5  options msglevel=i;
6  data pricelist(index=(ProductID/unique));
7    set orion.pricelist;
8    UnitProfit=UnitSalesPrice-UnitCostPrice;
9  run;

NOTE: There were 259 observations read from the data set ORION.PRICELIST.
NOTE: The data set WORK.PRICELIST has 259 observations and 7 variables.
NOTE: Simple index ProductID has been defined.
```

```

NOTE: DATA statement used (Total process time):
      real time          0.04 seconds
      cpu time           0.01 seconds

10   options msglevel=n;

```

- c. Open the program **p304e05** and submit it.
 - 1) Open program **p304e05**.
 - 2) Submit the program.
- d. View the log and determine whether the new observation was added.

Partial SAS Log

```

31   proc sql;
32     insert into pricelist(ProductID,
33                           StartDate,
34                           EndDate,
35                           UnitCostPrice,
36                           UnitSalesPrice,
37                           Factor,
38                           UnitProfit)
39     values (210200100009, '15FEB2007'd, '31DEC9999'd, 15.50, 34.70, 1.00, 19.20);
ERROR: Duplicate values not allowed on index ProductID for file PRICELIST.
NOTE: This insert failed while attempting to add data from VALUES clause 1 to the data
      set.
NOTE: Deleting the successful inserts before error noted above to restore table to a
      consistent state.
40   quit;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.15 seconds
      cpu time           0.04 seconds

```

- e. Why or why not?

The new observation was not added because an observation with the value 210200100009 for ProductID already exists in the data set pricelist. The unique index on ProductID prevents the second observation from being added.

6. Creating Indexes on New Variables

- a. Create a temporary SAS data set named **allstaff** by concatenating the data sets **orion.sales** and **orion.nonsales**.

Enter the following SAS code in the Editor window:

Partial p304s06

```

options msglevel=i;
data allstaff(index=(AgeHired));
  set orion.sales orion.nonsales(rename=(First=FirstName
                                         Last=LastName));
  AgeHired=intck('year',BirthDate,Hiredate,'c');
run;
options msglevel=n;

```

- b.** Submit the program and examine the SAS log.
- 1) Submit the contents of the Editor window.
 - 2) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

33  options msglevel=i;
34  data allstaff(index=(AgeHired));
35    set orion.sales orion.nonsales(rename=(First=FirstName
36                                Last=LastName));
37    AgeHired=intck('year',BirthDate,Hiredate,'c');
38  run;

NOTE: Missing values were generated as a result of performing an operation on missing
      values.
      Each place is given by: (Number of times) at (Line):(Column).
      1 at 37:13
NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: There were 235 observations read from the data set ORION.NONSALES.
NOTE: The data set WORK.ALLSTAFF has 400 observations and 10 variables.
NOTE: Simple index AgeHired has been defined.
NOTE: DATA statement used (Total process time):
      real time          0.07 seconds
      cpu time          0.01 seconds

```

- c.** Print the observations in **work.allstaff** where **AgeHired** is greater than 30. Print only the variables **EmployeeID**, **BirthDate**, **HireDate**, and **AgeHired**. Verify that the **AgeHired** index was used to retrieve the selected observations.

- 1) Enter the following PROC PRINT step in the Editor window.

Partial p304s06

```

proc print data=allstaff;
  var EmployeeID BirthDate HireDate AgeHired;
  where AgeHired>30;
run;
options msglevel=n;

```

- 2) Submit the SAS code in the Editor window.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

48  proc print data=allstaff;
49    var EmployeeID BirthDate HireDate AgeHired;
50    where AgeHired>30;
INFO: Index AgeHired selected for WHERE clause optimization.
51  run;

NOTE: There were 7 observations read from the data set WORK.ALLSTAFF.
      WHERE AgeHired>30;
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.04 seconds
      cpu time          0.01 seconds
52  options msglevel=n;

```

- 4) View the output to verify that it matches the expected output.

7. Using an Index

- a. Open the program **p304e07**.
- b. Submit the SAS code. If **orion.saleshistory** does not have the indexes **orderid**, **productgroup**, and **saleid**, use the program code in the comments to create them.
- c. Examine the SAS log.
- d. Answer the following questions:

- 1) Does Example 1 use an index? Why or why not?

No, the code uses an IF statement, not a WHERE statement.

Replace the IF statement with a WHERE statement, and resubmit the program.

Partial **p304s07**

```
data rdu;
  set orion.saleshistory;
  where OrderID=1230166613;
run;
```

Does the example now use an index? Why or why not?

Yes, the WHERE statement uses the SaleID index.

- 2) Does Example 2 use an index? Why or why not?

No, the WHERE statement uses the OR operator, which prohibits index use.

Replace the OR operator with the AND operator, and resubmit the program.

Partial **p304s07**

```
proc print data=orion.saleshistory;
  where OrderID=1230166613 and ProductID=220200100100;
run;
```

Does the example now use an index? Why or why not?

Yes, the WHERE statement uses both of the variables in the composite SaleID index with the AND operator.

- 3) Does Example 3 use an index? Why or why not?

No, too much of the data satisfied the WHERE statement criteria.

Replace the NE operator with the EQ operator, and resubmit the program.

Partial **p304s07**

```
proc print data=orion.saleshistory;
  where ProductGroup='Shoes';
run;
```

Does the example now use an index? Why or why not?

No, the data is too randomly distributed. The following INFO message is in the log:

```
INFO: Index ProductGroup not used. Sorting into index order may help.
```

- 4) Does Example 4 use an index? Why or why not?

Yes, only a small subset of the data that satisfied the WHERE statement criteria was returned.

Add the IDXWHERE=NO data set option and resubmit the program.

Partial p304s07

```
proc print data=orion.saleshistory(idxwhere=no);
  where CustomerID=12727;
run;
```

Is the output from the PROC PRINT step with an index different from the output from the PROC PRINT step without an index?

No, the results are the same.

What message do you see in the log?

```
INFO: Data set option (IDXWHERE=NO) forced a sequential pass of the data rather than
use of an index for where-clause processing.
```

- 5) In Example 5, does the data set **SalesHistoryCopy** have an index?

No, the DATA step re-creates orion.saleshistory as work.saleshistorycopy. The index is not copied when the data set is created in the DATA step.

8. Suppressing Index Usage

Create a detail report from the **orion.supplier** SAS data set that lists all of the variables and observations where **SupplierID** is greater than 1000. Ensure that the data is processed sequentially.

- Enter the following PROC PRINT step in the Editor:

p304s08

```
options msglevel=i;

proc print data=orion.supplier(idxwhere=no) noobs;
  where SupplierID > 1000;
run;

options msglevel=n;
```

- Submit the SAS code in the editor.
- View the output.

Partial PROC PRINT Output

SupplierID	SupplierName	StreetID	SupplierAddress	Street Number	Country
1280	British Sports Ltd	9250100844	85 Station Street	85	GB
1303	Eclipse Inc	9260107621	1218 Carriole Ct	1218	US
1684	Magnifico Sports	7350100062	Rua Costa Pinto 2	2	PT
1747	Pro Sportswear Inc	9260109782	2434 Edgebrook Dr	2434	US
2963	3Top Sports	9260111024	5033 Charles B Root Wynd	5033	US

- d. Examine the SAS log to verify that the program executed without errors.

SAS Log

```

92  options msglevel=i;
93  proc print data=orion.supplier(idxwhere=no obs=5) noobs;
94    where SupplierID > 1000;
INFO: Data set option (IDXWHERE=NO) forced a sequential pass of the data rather than use
      of an index for where-clause processing.
95  run;

NOTE: There were 5 observations read from the data set ORION.SUPPLIER.
      WHERE SupplierID>1000;
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.09 seconds
      cpu time           0.01 seconds

96  options msglevel=n;

```

9. Updating Centile Information for an Index

- a. Submit the program p304e09.

- 1) Open program p304e09.
- 2) Submit the SAS code in the editor.
- 3) View the output.

Partial PROC CONTENTS Output

Alphabetic List of Indexes and Attributes						
#	Index	Unique Option	Update Centiles	Current Percent	# of Unique Values	# of Variables
<i>... Lines Removed ...</i>						
2	OrderID	YES		5	0	490
					1230058123	
					1230699509	
					1231135703	
					1231544990	
					1231956902	
					1232601472	
					1233078086	
					1233920786	
					1234588648	

1236055696
1237478988
1238353296
1238846184
1239408849
1240137702
1240692950
1241652707
1242265757
1242923327
1243568955
1244296274

- b. Using the DATASETS procedure, specify that the centile information about the **CustomerID** index should be updated when more than 1% of the data changes.
- c. Refresh the **CustomerID** index.

- 1) Enter the following PROC DATASETS step in the editor:

p304s09

```
proc datasets lib=work nolist;
  modify orders;
    index centiles CustomerID/updatecentiles=1;
    index centiles CustomerID/refresh;
quit;
```

- 2) Submit the SAS code.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
16  proc datasets lib=work nolist;
17    modify orders;
18      index centiles CustomerID/updatecentiles=1;
NOTE: Index CustomerID centiles update percent changed to 1.
19      index centiles CustomerID/refresh;
NOTE: Index CustomerID centiles refreshed.
20  quit;

NOTE: MODIFY was successful for WORK.ORDERS.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

- d. Submit the program **p304e09c**, which adds new observations to the work.orders data set.
- 1) Open program **p304e09c**.
 - 2) Submit the SAS code.
 - 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

108  data neworders;
109  set orion.orders(obs=5);
110  month=month(orderdate);
111  day=day(orderdate);
112  orderdate=mdy(month,day,2008);
113  month=month(deliverydate);
114  day=day(deliverydate);
115  deliverydate=mdy(month,day,2008);
116  orderid+14239000;
117  drop month day;
118  run;

NOTE: There were 5 observations read from the data set ORION.ORDERS.
NOTE: The data set WORK.NEWORDERS has 5 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          0.09 seconds
      cpu time           0.04 seconds
119
120  proc append base=orders data=neworders;
121  run;

NOTE: Appending WORK.NEWORDERS to WORK.ORDERS.
NOTE: There were 5 observations read from the data set WORK.NEWORDERS.
NOTE: 5 observations added.
NOTE: The data set WORK.ORDERS has 495 observations and 6 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

```

- e. Submit a PROC CONTENTS step to view the contents of **orders**. Compare the centile information from part **9.a.** to the current centile information.

- 1) Enter the following PROC CONTENTS step in the editor:

Partial p304s09

```

proc contents data=orders centiles;
run;

```

- 2) Submit the SAS code.
- 3) View the output to verify that it matches the expected output.
- 4) Scroll down to the **Alphabetic List of Indexes and Attributes** output object.
- 5) Were the centiles updated?

Yes

Partial PROC CONTENTS Output

# Index	Option	The CONTENTS Procedure			
		Alphabetic List of Indexes and Attributes			
		Unique	Update	Current # of Centiles	Unique Percent Values Variables
1	CustomerID	1	0	81	
				4	
				9	
				10	
				16	
				19	
				27	
				34	
				41	
				49	
				52	
				63	
				79	
				89	
				111	
				171	
				195	
				544	
				2806	
				17023	
				70165	
				14239183	

Why or why not?

More than 1% of the data was changed when you added five observations.

Solutions to Student Activities (Polls/Quizzes)

4.01 Short Answer Poll – Correct Answer

If the OBS=2 data set option was omitted in the first SET statement, how many times would the DATA step execute? **617**

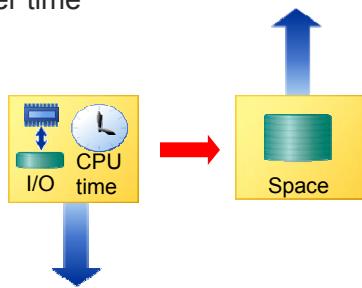
```
data top bottom;
  set SortedOrderFact end=onlastrow;
  output top;
  if onlastrow then
    do ReadOBS=(TotOBS) to
      (TotOBS - 1) by -1;
      set SortedOrderFact
        nobs=TotObs
        point=ReadOBS;
      output bottom;
    end;
run;
```

50

4.03 Multiple Choice Poll – Correct Answer

If a WHERE statement uses an index to retrieve a small subset of data, which of these resources is conserved?

- a. I/O
- b. disk space
- c. memory
- d. programmer time



87

4.04 Multiple Answer Poll – Correct Answers

On which of these indexed variables can you use the UNIQUE option?

- CustomerID** in an **orders** data set where a customer can place multiple orders
- OrderDate** in an **orders** data set
- EmployeeID** in a data set containing each individual employee and the family members' names stored in the variables **Dependent1** through **Dependent10**
- ProductID** in a data set containing the product identifier and the product description

93

4.05 Quiz – Correct Answer

Open and submit the program **p304a01**.

What error messages are in the log?

```

1 options msglevel=n;
2 proc datasets library=orion nolist;
3   modify saleshistory;
4   index create CustomerID;
ERROR: An index named CustomerID with the same definition already exists for
      file ORION.SALESHISTORY.DATA.
5   index create ProductGroup;
6   index create SaleID=(OrderID
7           ProductID)/unique;
8 quit;

NOTE: Statements not processed because of errors noted above.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time          0.48 seconds
      cpu time           0.09 seconds

```



If an index exists, it must be deleted before it can be re-created.

111

4.06 Multiple Answer Poll – Correct Answers

If you manage an index using operating environment features, what problems might occur?

- a. The index file might be damaged.
- b. Files created outside SAS might be damaged.
- c. The data set associated with the index file might be damaged.

122

4.07 Multiple Answer Poll – Correct Answers

Which of the following WHERE conditions could possibly use an index?

- a. `where ProductID=230100700004;`
- b. `where CustomerID=15020 or CustomerID=14853;`
- c. `where OrderID=1230036183;`

134

Chapter 5 DATA Step Arrays

5.1	Introduction to Lookup Techniques.....	5-3
5.2	One-Dimensional Arrays.....	5-9
	Exercises	5-26
5.3	Multidimensional Arrays.....	5-30
	Exercises	5-44
5.4	Loading a Multidimensional Array from a SAS Data Set	5-48
	Exercises	5-69
5.5	Solutions	5-74
	Solutions to Exercises	5-74
	Solutions to Student Activities (Polls/Quizzes)	5-87

5.1 Introduction to Lookup Techniques

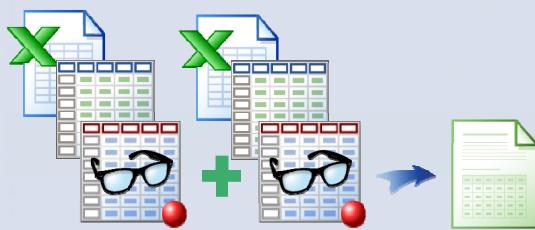
Objectives

- Define table lookup.
- List table lookup techniques.

3

Introduction

There is often a need to combine or look up data from multiple sources to create meaningful reports.



4

Introduction

When data sources do not share a common structure,
a lookup table can match them.



5

Lookup Table Data

You can use several techniques to look up data.

Continent ID	Continent Name
91	North America
93	Europe
94	Africa
95	Asia
96	Australia/Pacific

6

IF-THEN/ELSE Statements

Lookup tables can be SAS programming statements.

```
data countryinfo;
  set orion.country;
  if ContinentID=91
    then Continent='North America';
  else if ContinentID=93
    then Continent='Europe';
  else if ContinentID=94
    then Continent='Africa';
  else if ContinentID=95
    then Continent='Asia';
  else if ContinentID=96
    then Continent='Australia/Pacific';
run;
```



7

DATA Step Merge

Lookup tables can be SAS data sets that are accessed during a DATA step merge.

```
data countryinfo;
  merge orion.country
        orion.Continent;
  by ContinentID;
run;
```



8

PROC SQL Join

Lookup tables can be SAS data sets that are accessed during a PROC SQL join.

```
proc sql;
  create table countryinfo as
    select * from
      orion.country a, orion.Continent b
    where a.ContinentID=b.ContinentID;
quit;
```



9

User-Defined Formats

Lookup tables can be user-defined formats that are accessed with a FORMAT statement or PUT function.

```
proc format;
  value ContName
    91='North America'  93='Europe'
    94='Africa'          95='Asia'
    96='Australia/Pacific';
run;

proc print data=orion.country;
  format ContinentID ContName.;
run;

data countryinfo;
  set orion.country;
  Continent=put(ContinentID,ContName.);
run;
```



10

Arrays

Lookup tables can be arrays.

```
data countryinfo;
  array ContName{91:96} $ 30 _temporary_
    ('North America',
     '',
     'Europe',
     'Africa',
     'Asia',
     'Australia/Pacific');
  set orion.country;
  Continent=ContName{ContinentID};
run;
```



11

Table Lookup Techniques

The technique that is used to perform a table lookup depends on the data.



Table Location	Technique	Lookup Table
DATA step	IF-THEN/ELSE statements	SAS programming statements
Disk	SQL join merge SET/SET KEY=	SAS data set
Memory	FORMAT statement PUT statement PUT function	user-defined format
	array reference	array
	FIND method	hash object

12

Arrays as Lookup Tables

This chapter focuses on arrays as lookup tables.



Table Location	Technique	Lookup Table
DATA step	IF-THEN/ELSE statements	SAS programming statements
Disk	SQL join merge SET/SET KEY=	SAS data set
Memory	FORMAT statement PUT statement PUT function	user-defined format
	array reference	array
	FIND method	hash object

13

5.01 Multiple Choice Poll

Which of these is an example of a table lookup?

- You have the data for January sales in one data set, February sales in a second data set, and March sales in a third. You need to create a report for the entire first quarter.
- You want to send birthday cards to employees. The employees' names and addresses are in one data set and their birthdates are in another.
- You need to calculate the amount that each customer owes for his purchases. The price per item and the number of items purchased are stored in the same data set.

14

5.02 Multiple Answer Poll

Which techniques do you currently use when you perform table lookups with a single data set?

- a. arrays
- b. hash object
- c. formats
- d. none of the above

16

5.2 One-Dimensional Arrays

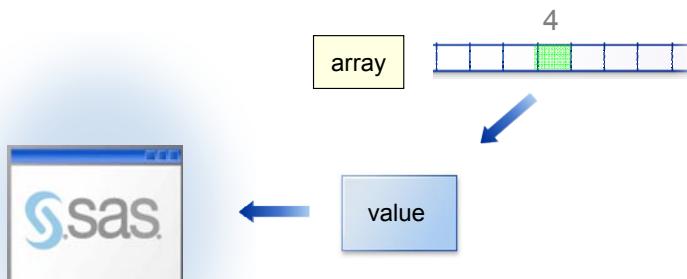
Objectives

- Describe array references as a lookup technique.
- Define one-dimensional arrays and multidimensional arrays.
- Use a one-dimensional array as a lookup table.

19

Using One-Dimensional Arrays as Lookup Tables

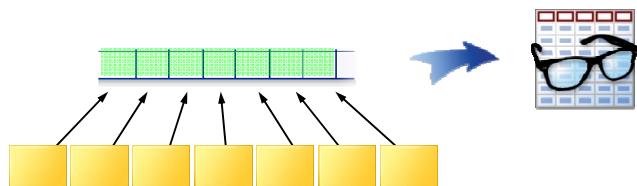
SAS retrieves a data value based on the value's position in the array when arrays are used.



20

Using One-Dimensional Arrays as Lookup Tables

In SAS, an array is simply a way to refer to a group of variables with a single name.



21

Using One-Dimensional Arrays as Lookup Tables

Think of the elements of a one-dimensional array as a numbered row.

1	2	3	4
N. America	Europe	Africa	Asia

22

...

Using One-Dimensional Arrays as Lookup Tables

If you specify position number 3, the array returns the value *Africa*.



1	2	3	4
N. America	Europe	Africa	Asia

23

Reviewing SAS Arrays

An array is declared with an ARRAY statement. Array elements can be SAS variables or temporary data elements.

```
array ContName{91:96} $ 30 _temporary_
  ('North America',
   '',
   'Europe',
   'Africa',
   'Asia',
   'Australia/Pacific');
```

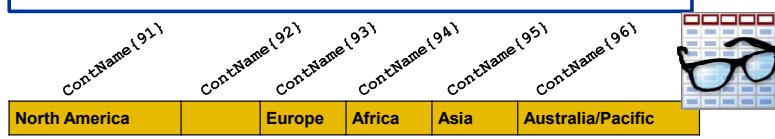
ARRAY array-name {number-of-elements} <\$> <length>
<_temporary_> <list-of-variables> <(initial-values)>;

24

Reviewing SAS Arrays

This ARRAY statement assigns an initial value to each corresponding element. SAS matches the elements and values by position, so the values must be listed in the order of the array elements.

```
array ContName{91:96} $ 30 _temporary_
  ('North America',
   '',
   'Europe',
   'Africa',
   'Asia',
   'Australia/Pacific');
```



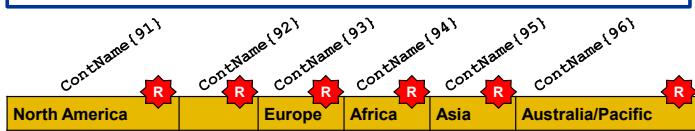
25

...

Reviewing SAS Arrays

When an initial value list is specified, all array elements behave as if they were named in a RETAIN statement.

```
array ContName{91:96} $ 30 _temporary_
  ('North America',
   '',
   'Europe',
   'Africa',
   'Asia',
   'Australia/Pacific');
```



26

5.03 Quiz

The ARRAY statement below creates temporary array elements.

What is created if `_temporary_` is omitted?

```
array ContName{91:96} $ 30 _temporary_
  ('North America',
   '',
   'Europe',
   'Africa',
   'Asia',
   'Australia/Pacific');
```

27

Example: Looking Up Continent Names Using an Array

A one-dimensional array can serve as a lookup table with the names of the continents.

Partial `orion.country`

Country	Country Name	Population	ContinentID
AU	Australia	20,000,000	96
CA	Canada	.	91
DE	Germany	80,000,000	93
IL	Israel	5,000,000	95
TR	Turkey	70,000,000	95

North America	Europe	Africa	Asia	Australia/Pacific
---------------	--------	--------	------	-------------------

29

Example: Looking Up Continent Names Using an Array

```
data countryinfo;
array ContName{91:96} $ 30 _temporary_
  ('North America',
   '',
   'Europe',
   'Africa',
   'Asia',
   'Australia/Pacific');
set orion.country;
Continent=ContName{ContinentID};
run;
```



30

Business Scenario

Calculate the difference between the current salary of each employee hired in a specific year and the average salary of all employees hired that year.

year of hire=????



31

Using a Table Lookup

Orion.salarystats contains the salary statistics for all Orion Star employees who were hired in the years 1978 through 2011, including average salaries.

Partial **orion.salarystats**

Statistic	Yr1978	Yr1979	Yr1980	...
MedianSalary	30025	29442.5	30020	...
StdSalary	28551.9	9918.35	22356.91	...
SumSalary	2393860	132150	235030	...
AvgSalary	39243.61	33037.5	39171.67	...

32

Using a Table Lookup

Orion.employeepayroll contains each Orion Star employee's hire date and current salary.

Partial **orion.employeepayroll**

EmployeeID	Salary	BirthDate	EmployeeHireDate
120101	163040	18AUG1980	01JUL2007
120102	108255	11AUG1973	01JUN1993
120103	87975	22JAN1953	01JAN1978
120104	46230	11MAY1958	01JAN1985
120105	27110	21DEC1978	01MAY2003

33

5.04 Multiple Choice Poll

Which data set can be used to populate the array?

- a. **orion.employeepayroll**
- b. **orion.salarystats**

34

Using a Table Lookup

Combine the data with a *table lookup*.

Partial orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980
MedianSalary	30025	29442.5	30020
StdSalary	28551.9	9918.35	22356.91
SumSalary	2393860	132150	235030
AvgSalary	39243.61	33037.5	39171.67

Partial orion.employeepayroll

YEAR(SAS-date-value)

EmployeeID	Salary	BirthDate	EmployeeHireDate
120101	163040	18AUG1980	01JUL2007
120102	108255	11AUG1973	01JUN1993
120103	87975	22JAN1953	01JAN1978
120104	46230	11MAY1958	01JAN1985
120105	27110	21DEC1978	01MAY2003

36

Using a One-Dimensional Array

```

data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if _n_=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;

```

37

p305d01

...

Using a One-Dimensional Array

```

data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if _n_=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;


```

38

p305d01

Compilation

```

data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif
    dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if _n_=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
      Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;

```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	EmployeeHireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif

39

p305d01

...

Compilation

```
data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif
    dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
      Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011

40

...

Compilation

```
data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif
    dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
      Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic

41

p305d01
...

Compilation

```
data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif
    dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
      Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate

42 p305d01
...

Compilation

```
data compare;
  keep EmployeeID YearHired Salary
    Average SalaryDif;
  format Salary Average SalaryDif
    dollar12.2;
  array yr{1978:2011} Yr1978-Yr2011;
  if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
  set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
      Salary);
  YearHired=year(EmployeeHireDate);
  Average=yr{YearHired};
  SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired

43 p305d01
...

Compilation

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired	_N_

p305d01

44

...

Compilation

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	D Yr1978	D Yr1979	D Yr1980	D Yr1981	D Yr1982	...

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

D Yr2007	D Yr2008	D Yr2009	D Yr2010	D Yr2011	D Statistic	EmployeeID	D Employee HireDate	Year Hired	D _N_

p305d01

45

...

Execution

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	yr1978	yr1979	yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
.

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

D_Yr2007	D_Yr2008	D_Yr2009	D_Yr2010	D_Yr2011	D_Statistic	EmployeeID	D_Employee HireDate	Year Hired	D_N_
.	1

46

p305d01

...

Execution

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
.	.	.	39243.61	33037.5	39171.67	34170	37506.25	.

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

D_Yr2007	D_Yr2008	D_Yr2009	D_Yr2010	D_Yr2011	D_Statistic	EmployeeID	D_Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	.	.	.	1

47

p305d01

...

Execution

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
163040	.	.	39243.61	33037.5	39171.67	34170	37506.25	

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	.	1

48

p305d01

...

Execution

```
data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;
```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
163040	.	.	39243.61	33037.5	39171.67	34170	37506.25	

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	2007	1

49

p305d01

...

Execution

```

data compare;
keep EmployeeID YearHired Salary
      Average SalaryDif;
format Salary Average SalaryDif
      dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
  (where=(Statistic='AvgSalary'));
set orion.employeepayroll
  (keep=EmployeeID EmployeeHireDate
   Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;

```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
163040	35082.5	.	39243.61	33037.5	39171.67	34170	37506.25	

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	2007	1

51

p305d01

...

Execution

```

data compare;
keep EmployeeID YearHired Salary
      Average SalaryDif;
format Salary Average SalaryDif
      dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
  (where=(Statistic='AvgSalary'));
set orion.employeepayroll
  (keep=EmployeeID EmployeeHireDate
   Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;

```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	Yr1978	Yr1979	Yr1980	Yr1981	Yr1982	...
163040	35082.5	127957.5	39243.61	33037.5	39171.67	34170	37506.25	

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011	Statistic	EmployeeID	Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	2007	1

52

p305d01

...

Execution

```

data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;

```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	D>Yr1978	D>Yr1979	D>Yr1980	D>Yr1981	D>Yr1982	...
163040	35082.5	127957.5	39243.61	33037.5	39171.67	34170	37506.25	

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Statistic	EmployeeID	D>Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	2007	1

53

p305d01

...

Execution

```

data compare;
keep EmployeeID YearHired Salary
    Average SalaryDif;
format Salary Average SalaryDif
    dollar12.2;
array yr{1978:2011} Yr1978-Yr2011;
if n=1 then set orion.salarystats
    (where=(Statistic='AvgSalary'));
set orion.employeepayroll
    (keep=EmployeeID EmployeeHireDate
        Salary);
YearHired=year(EmployeeHireDate);
Average=yr{YearHired};
SalaryDif=Salary-Average;
run;

```

Partial Listing of orion.salarystats

Statistic	Yr1978	Yr1979	Yr1980	...
AvgSalary	39243.61	33037.5	39171.67	...

Partial Listing of orion.employeepayroll

EmployeeID	Salary	Employee HireDate
120101	163040	01JUL2007
120102	108255	01JUN1993
120103	87975	01JAN1978
120104	46230	01JAN1985
120105	27110	01MAY2003

Partial PDV

Salary	Average	SalaryDif	D>Yr1978	D>Yr1979	D>Yr1980	D>Yr1981	D>Yr1982	...
163040	35082.5	127957.5	39243.61	33037.5	39171.67	34170	37506.25	

yr{2007} yr{2008} yr{2009} yr{2010} yr{2011}

D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Statistic	EmployeeID	D>Employee HireDate	Year Hired	D_N_
35082.5	29904.44	30576.36	27883.71	28861.67	AvgSalary	120101	01JUL2007	2007	1

54

p305d01

...

Resulting Data

```
proc print data=compare(obs=5);
  var EmployeeID YearHired Salary Average SalaryDif;
  title 'Using One Dimensional Arrays';
run;
```

PROC PRINT Output

Using One Dimensional Arrays					
Obs	EmployeeID	Year Hired	Salary	Average	SalaryDif
1	120101	2007	\$163,040.00	\$35,082.50	\$127,957.50
2	120102	1993	\$108,255.00	\$88,588.75	\$19,666.25
3	120103	1978	\$87,975.00	\$39,243.61	\$48,731.39
4	120104	1985	\$46,230.00	\$36,436.67	\$9,793.33
5	120105	2003	\$27,110.00	\$36,533.75	\$-9,423.75

55

p305d01

5.05 Quiz

Which of the following ARRAY statements are similar to the statement

```
array yr{1978:2011} Yr1978-Yr2011;
```

and compile without errors?

- a. array yr{34} Yr1978-Yr2011;
- b. array yr{1978-2011} Yr1978-Yr2011;
- c. array yr{78:11} Yr1978-Yr2011;
- d. array yr{78-011} Yr1978-Yr2011;

56



Exercises

Level 1

1. Using a One-Dimensional Array to Combine Data

The data set **orion.retail** has information about retail sales.

Partial orion.retail

Partial orion.retail Data Set						
Obs	CustomerID	EmployeeID	StreetID	Order Date	Delivery Date	OrderID
1	63	121039	9260125492	11JAN2007	11JAN2007	1230058123
2	41	120174	1600101527	28JAN2007	28JAN2007	1230147441
3	183	120134	1600100760	27FEB2007	27FEB2007	1230315085
4	56	121059	9260111871	15MAR2007	15MAR2007	1230404278
5	183	120149	1600100760	22MAR2007	22MAR2007	1230440481
Obs	ProductID	Quantity	TotalRetail Price	CostPrice PerUnit	Discount	
1	220101300017	1	\$16.50	\$7.45	.	
2	240600100010	2	\$32.00	\$6.50	.	
3	240200200039	3	\$63.60	\$8.80	.	
4	220200300002	2	\$75.00	\$17.05	.	
5	230100600005	1	\$129.80	\$63.20	.	

The data set **orion.retailinformation** has statistics about those retail sales.

Partial orion.retailinformation

Partial orion.retailinformation Data Set					
Obs	Statistic	Month1	Month2	Month3	
1	SumRetailPrice	\$1,599.80	\$1,160.80	\$113.70	
2	MeanRetailPrice	\$228.54	\$193.47	\$28.43	
3	MedianRetailPrice	\$258.20	\$123.30	\$27.65	
Obs	Month4	Month5	Month6	Month7	Month8
1	\$671.10	\$520.70	\$561.99	\$288.29	\$1,033.40
2	\$83.89	\$86.78	\$70.25	\$48.05	\$103.34
3	\$49.10	\$68.30	\$52.20	\$44.90	\$54.25
Obs	Month9	Month10	Month11	Month12	
1	\$425.70	\$736.10	\$2,399.30	\$1,347.58	
2	\$70.95	\$105.16	\$218.12	\$134.76	
3	\$65.35	\$101.50	\$69.40	\$61.70	

- a. Combine the two data sets to create a data set named **compare**. Use the month from **OrderDate** to determine which value of **MedianRetailPrice** to retrieve. The **compare** data set should contain all the variables from **orion.retail** and variables named **Month** and **MedianRetailPrice**.
- b. Print the first five observations of the resulting data set.

Partial PROC PRINT Output

Partial Compare Data Set						
Obs	CustomerID	EmployeeID	StreetID	Order Date	Delivery Date	OrderID
1	63	121039	9260125492	11JAN2007	11JAN2007	1230058123

2	41	120174	1600101527	28JAN2007	28JAN2007	1230147441
3	183	120134	1600100760	27FEB2007	27FEB2007	1230315085
4	56	121059	9260111871	15MAR2007	15MAR2007	1230404278
5	183	120149	1600100760	22MAR2007	22MAR2007	1230440481
Obs	ProductID	Quantity	TotalRetail Price	CostPrice PerUnit	Discount	Median Retail Price
1	220101300017	1	\$16.50	\$7.45	.	1 \$258.20
2	240600100010	2	\$32.00	\$6.50	.	1 \$258.20
3	240200200039	3	\$68.60	\$8.80	.	2 \$123.30
4	220200300002	2	\$75.00	\$17.05	.	3 \$27.65
5	230100600005	1	\$129.80	\$63.20	.	3 \$27.65

Level 2

2. Using a One-Dimensional Array as a Lookup Table

The data set **orion.shoestats** contains statistics for the shoe product lines.

orion.shoestats

orion.shoestats Data Set					
Obs	Stat	Product21	Product22	Product23	Product24
1	Frequency	66.000	277.000	.	18.000
2	MfgSuggestedRetailPriceMean	70.788	174.292	.	173.056
3	MfgSuggestedRetailPriceMin	17.000	13.000	.	5.000
4	MfgSuggestedRetailPriceMax	130.000	385.000	.	398.000
5	MfgSuggestedRetailPriceMedian	68.000	164.000	.	190.500
6	MfgSuggestedRetailPriceStdDev	21.731	71.703	.	141.389

- a. Use an array to create a data set that is named **trans** that has 24 observations.
- b. Print the first five observations of the **trans** data set.

PROC PRINT Output

The TRANS data set			
Obs	Stat	Product Line	Value
1	Frequency	21	66.000
2	Frequency	22	277.000
3	Frequency	23	.
4	Frequency	24	18.000
5	MfgSuggestedRetailPriceMean	21	70.788

Challenge

3. Using a One-Dimensional Array

- a. Use the program **p305e03** to create a temporary data set that is named **orderfact** for the year 2011 and customer IDs 89 and 2550, sorted by **OrderType**.

p305e03

```

proc sort data=orion.orderfact
    out=orderfact(keep=CustomerID OrderType OrderDate
                  DeliveryDate Quantity);
    where CustomerID in (89, 2550)
        and year(OrderDate)=2011;
    by OrderType;
run;

proc sql;
    title 'Count by Order Type';
    select OrderType,
           count(*) as count
    from orderfact
    group by OrderType;
quit;

```

orderfact

orderfact					
Obs	CustomerID	Order Type	Order Date	Delivery Date	Quantity
1	89	1	03JAN2011	04JAN2011	6
2	89	1	01OCT2011	01OCT2011	1
3	89	1	01OCT2011	01OCT2011	1
4	89	1	15DEC2011	15DEC2011	4
5	89	2	17JUN2011	21JUN2011	2
6	2550	3	04MAY2011	09MAY2011	3
7	2550	3	04MAY2011	09MAY2011	1

The PROC SQL step in the starter code creates a report that shows the number of observations for each value of the **OrderType** variable for the **CustomerID** values 89 and 2550.

Count by Order Type

Order Type	Count
1	4
2	1
3	2

- b. Create a data set named **all** that has one observation for each **OrderType**, where there are a varying number of observations for each **OrderType** in the original **orderfact** data set. Use the maximum number of observations for all order types as the array dimension. Create three arrays that create variables to hold the order dates, the delivery dates, and the quantity.
- c. Print the **all** data set.

The ALL Data Set						
Obs	Ordered Date1	Ordered Date2	Ordered Date3	Ordered Date4	Delivery Date1	Delivery Date2
1	03JAN2011	01OCT2011	01OCT2011	15DEC2011	04JAN2011	01OCT2011
2	17JUN2011	.	.	.	21JUN2011	.
3	04MAY2011	04MAY2011	.	.	09MAY2011	09MAY2011

Obs	Delivery		Delivery			
	Date3	Date4	Quantity1	Quantity2	Quantity3	Quantity4
1	01OCT2011	15DEC2011	6	1	1	4
2	.	.	2	.	.	.
3	.	.	3	1	.	.

Obs	n	CustomerID	Order	Delivery	Order	Quantity
			Date	Date	Type	
1	4	89	15DEC2011	15DEC2011	1	4
2	1	89	17JUN2011	21JUN2011	2	2
3	2	2550	04MAY2011	09MAY2011	3	1

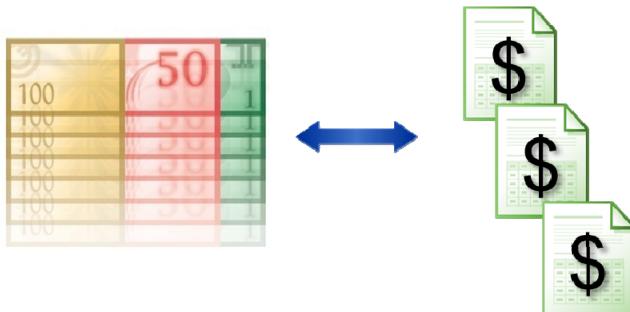
5.3 Multidimensional Arrays

Objectives

- Define a multidimensional array.
- Explain the differences between a one-dimensional array and a multidimensional array.
- Use a multidimensional array as a lookup table.

Business Scenario

Compare the actual profit values for each Orion Star company to the budgeted profit values.



62

Business Scenario

The SAS data set **orion.profit** contains data for the years 2007 through 2011, separated by month.

Partial **orion.profit**

Company	YYMM	Sales	Cost	Salaries	Profit
Logistics	07M01	\$457,809	\$210,914	\$127,525	\$119,370
Logistics	07M02	\$325,138	\$149,718	\$127,525	\$47,895
Logistics	07M03	\$276,805	\$127,827	\$134,198	\$14,780
Logistics	07M04	\$558,806	\$264,868	\$134,198	\$159,741

63

5.06 Quiz

Examine the descriptor portion of the data set **orion.profit**. What is the type of the variable **YYMM**?

64

Business Scenario

This table contains the budgeted amounts for each of those months and years. Each **row** represents a month, and each **column** represents a year.

	Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
Month 1	\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
Month 2	\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000
	\$1,160,000	\$1,380,000	\$1,640,000	\$1,410,000	\$1,440,000
	\$1,710,000	\$2,100,000	\$2,420,000	\$2,130,000	\$2,270,000
	\$1,990,000	\$2,350,000	\$2,840,000	\$2,480,000	\$2,670,000
	\$2,560,000	\$3,020,000	\$3,580,000	\$3,070,000	\$3,410,000
	\$2,590,000	\$2,890,000	\$3,550,000	\$3,010,000	\$3,490,000
	\$2,550,000	\$2,840,000	\$3,580,000	\$3,030,000	\$3,500,000
	\$1,070,000	\$1,180,000	\$1,550,000	\$1,260,000	\$1,520,000
	\$1,160,000	\$1,270,000	\$1,600,000	\$1,360,000	\$1,700,000
	\$1,260,000	\$1,470,000	\$1,780,000	\$1,540,000	\$1,950,000
	\$2,870,000	\$3,120,000	\$3,760,000	\$3,210,000	\$4,370,000

66

Business Scenario

Combine the budget amounts in the table with the actual amount in the SAS data set.

Partial orion.profit

Company	YYMM	Sales	Cost	Salaries	Profit
Logistics	07M01	\$457,809	\$210,914	\$127,525	\$119,370
Logistics	07M02	\$325,138	\$149,718	\$127,525	\$47,895
Logistics	07M03	\$276,805	\$127,827	\$134,198	\$14,780

Partial Lookup Table

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000
\$1,160,000	\$1,380,000	\$1,640,000	\$1,410,000	\$1,440,000

67

5.07 Quiz

What do the data set **orion.profit** and the lookup table have in common?

Partial orion.profit

Company	YYMM	Sales	Cost	Salaries	Profit
Logistics	07M01	\$457,809	\$210,914	\$127,525	\$119,370
Logistics	07M02	\$325,138	\$149,718	\$127,525	\$47,895
Logistics	07M03	\$276,805	\$127,827	\$134,198	\$14,780

Partial Lookup Table

Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000
\$1,160,000	\$1,380,000	\$1,640,000	\$1,410,000	\$1,440,000

68

Business Scenario

Find the budgeted amounts for each company, year, and month.

Partial `orion.profit`

Company	YYMM	Sales	Cost	Salaries	Profit
Logistics	07M01	\$457,809	\$210,914	\$127,525	\$119,370
Logistics	07M02	\$325,138	\$149,718	\$127,525	\$47,895



Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000



Partial `budgetamt`

Company	YYMM	Sales	Cost	Salaries	Profit	BudgetAmt
Logistics	07M01	\$457,809	\$210,914	\$127,525	\$119,370	\$1,590,000
Logistics	07M02	\$325,138	\$149,718	\$127,525	\$47,895	\$1,290,000

70

Multidimensional Arrays

Multidimensional arrays can have two or more dimensions.

To combine the table of budgeted values with the data set containing the actual profit, use a ***two-dimensional*** array. Values are retrieved based on two dimensions.

	2007	2008
Month 1	1,1	1,2
Month 2	2,1	2,2

71

Using Multidimensional Arrays

In this example, the first dimension is 2 rows and the second dimension is 5 columns, so the array has a total of 10 elements.

```
array B{2,5} _temporary_
(1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
```

ARRAY *array-name* {*...,rows,cols*} <\$> <*length>*<*elements*><(*initial values*)>;

B{1,1}	B{1,2}	B{1,3}	B{1,4}	B{1,5}
B{2,1}	B{2,2}	B{2,3}	B{2,4}	B{2,5}

- ✍ In this example, only the first two months of data are required.

72

- ✍ For this example, only two rows and five columns are included in the array.

The keyword `_TEMPORARY_` can be used instead of *elements* to create temporary data elements.

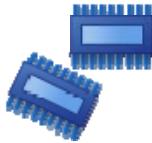
When you use a multidimensional array, the following statements are true:

- You must supply a subscript value for each dimension to process a specific array element.
- You can use a DO loop to process elements in a given dimension.
- You can use nested DO loops to process elements in multiple dimensions.

Using Multidimensional Arrays

The entire array must fit in memory at one time.

```
array B{2,5} _temporary_
(1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
```



73

5.08 Quiz

Which of the answers would be equivalent to the following ARRAY statement?

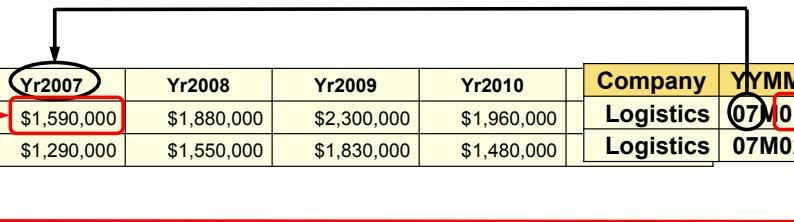
```
array B{2,5} B1-B10 (1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
```

- a. array B{2,2003:2007} B1-B10
(1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
- b. array B{2,5}
(1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
- c. array B{2,5} _temporary_
(1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);

74

Using Multidimensional Arrays

Find the budgeted amounts for each company, year, and month.



Yr2007	Yr2008	Yr2009	Yr2010	Company	YYMM
\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	Logistics	07M01
\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	Logistics	07M02

76

Using Multidimensional Arrays

```
data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000, 1960000, 1970000,
     1290000, 1550000, 1830000, 1480000, 1640000);
  set orion.profit(where=(Sales ne .)
                    obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;
```



- ✍ The budget data does not exist in a SAS data set.

77

p305d02

...

Using Multidimensional Arrays

```
data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000, 1960000, 1970000,
     1290000, 1550000, 1830000, 1480000, 1640000);
  set orion.profit(where=(Sales ne .)
                    obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;
```

78

p305d02

...

Using Multidimensional Arrays

```
data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000, 1960000, 1970000,
     1290000, 1550000, 1830000, 1480000, 1640000);
  set orion.profit(where=(Sales ne .)
                   obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;
```

79

p305d02

...

Using Multidimensional Arrays

```
data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000, 1960000, 1970000,
     1290000, 1550000, 1830000, 1480000, 1640000);
  set orion.profit(where=(Sales ne .)
                   obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
```



80

p305d02

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
1960000, 1970000, 1290000,
1550000, 1830000, 1480000,
1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

	B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000	

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
	1

81 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
1960000, 1970000, 1290000,
1550000, 1830000, 1480000,
1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

	B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000	

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M01	457809	210914	127525	119370	.	.	.	1

82 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M01	457809	210914	127525	119370	2007	1	.	1

83

p305d02

...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year(YYMM);
BudgetAmt=B{1,2007};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M01	457809	210914	127525	119370	2007	1	1590000	1

85

p305d02

...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .))
Y=year(M);
M=month(M);
BudgetAmt=B{N};
run;

```

Implicit OUTPUT; Implicit RETURN;

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M01	457809	210914	127525	119370	2007	1	1590000	1

86 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{N};
obs=2;
run;

```

Reinitialize PDV

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M01	457809	210914	127525	119370	.	.	.	2

87 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000,
     1960000, 1970000, 1290000,
     1550000, 1830000, 1480000,
     1640000);
  set orion.profit(where=(Sales ne .)
                    obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D Y	D M	BudgetAmt	D N_
Logistics	07M02	325138	149718	127525	47895			.	2

88

p305d02

...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
  drop Y M;
  array B{2,2007:2011} _temporary_
    (1590000, 1880000, 2300000,
     1960000, 1970000, 1290000,
     1550000, 1830000, 1480000,
     1640000);
  set orion.profit(where=(Sales ne .)
                    obs=2);
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D Y	D M	BudgetAmt	D N_
Logistics	07M02	325138	149718	127525	47895	2007	2	.	2

89

p305d02

...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{2,2007};
run;

```

BudgetAmt=1290000

	B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000	

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M02	325138	149718	127525	47895	2007	2	1290000	2

91 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
 1960000, 1970000, 1290000,
 1550000, 1830000, 1480000,
 1640000);
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{2,2007};
run;

```

Implicit OUTPUT;
Implicit RETURN;

	B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000	

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M02	325138	149718	127525	47895	2007	2	1290000	2

92 p305d02 ...

Execution

Partial orion.profit

Company	YYMM	Sales	Cost	...
Logistics	07M01	457809	210914	...
Logistics	07M02	325138	149718	...

```

data budgetamt;
drop Y M;
array B{2,2007:2011} _temporary_
(1590000, 1880000, 2300000,
1960000, 1970000, 1290000,
1550000, 1830000, 1480000,
1640000);
set orion.profit(where=(Sales ne .)
obs=2);
Y=year (YYMM);
M=month (YYMM);
BudgetAmt=B{M,Y};

```

Execution stops.

B1/2007	B1/2008	B1/2009	B1/2010	B1/2011	B2/2007	B2/2008	B2/2009	B2/2010	B2/2011
1590000	1880000	2300000	1960000	1970000	1290000	1550000	1830000	1480000	1640000

PDV

Company	YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	BudgetAmt	D _N
Logistics	07M02	325138	149718	127525	47895	2007	2	1290000	2

93 p305d02



Exercises

Level 1

4. Using a Two-Dimensional Array

Orion Star wants to send discount coupons to their customers. The amounts of the discounts are given in the following table:

Previous Quantity Ordered

OrderType	1	2	3	4	5	6
1	10	10	15	20	20	25
2	10	15	20	25	25	30
3	10	15	15	20	25	25

The data set **orion.orderfact** contains the variables **CustomerID**, **Quantity**, and **OrderType**.

Partial orion.orderfact

Obs	CustomerID	Order	
		Type	Quantity
1	63	1	1
2	5	2	1
3	45	2	1
4	41	1	2
5	183	1	3
6	79	2	1

7	23	2	1
8	23	2	2
9	45	2	2
10	45	2	1

- a. Use a two-dimensional array to combine the data set with the table of values to create a data set that is named **customercoupons** with a variable named **CouponValue**.
- b. Print the first five observations of the **customercoupons** data set.

PROC PRINT Output

The Coupon Value				
Obs	CustomerID	Order Type	Quantity	Coupon Value
1	63	1	1	10
2	5	2	1	10
3	45	2	1	10
4	41	1	2	10
5	183	1	3	15

Level 2

5. Using a Two-Dimensional Array

The following table shows the average manufacturer's suggested retail price for shoes, based on the product line and product category:

Product Category		
Product Line	1	2
21	.	70.79
22	173.79	174.40
23	.	.
24	29.65	287.8

The data set **orion.shoesales** contains **ProductID**, **ProductName**, and **TotalRetailPrice** for all of the shoes sold by Orion Star.

Partial **orion.shoesales**

ProductID	ProductName	TotalRetail Price
220200200024	Pro Fit Gel Gt 2030 Women's Running Shoes	\$178.50
220200100092	Big Guy Men's Air Terra Sebec Shoes	\$83.00
240200100043	Bretagne Performance Tg Men's Golf Shoes L.	\$282.40
220100700024	Armadillo Road Dmx Women's Running Shoes	\$99.70
220200300157	Hardcore Men's Street Shoes Large	\$220.20

- a. Create a data set named **combine**. Use a two-dimensional array to combine the table of values with the product line and the product category ID.
- The product line is indicated by the first two digits of the **ProductID** variable.
 - The product category is indicated by the third and fourth digits of the **ProductID** variable.

 The variables **ProductLine** and **ProductCatID** must be numeric.

- b. Print the first five observations of the **combine** data set.

PROC PRINT Output

Obs	ProductID	ProductName	TotalRetail Price
1	220200200024	Pro Fit Gel Gt 2030 Women's Running Shoes	\$178.50
2	220200100092	Big Guy Men's Air Terra Sebec Shoes	\$83.00
3	240200100043	Bretagne Performance Tg Men's Golf Shoes L.	\$282.40
4	220100700024	Armadillo Road Dmx Women's Running Shoes	\$99.70
5	220200300157	Hardcore Men's Street Shoes Large	\$220.20

Obs	ProductID	Manufacturer		
		Product Line	Product Category	Suggested Price
1	220200200024	22	2	174.40
2	220200100092	22	2	174.40
3	240200100043	24	2	287.80
4	220100700024	22	1	173.79
5	220200300157	22	2	174.40

Challenge

6. Using a Three-Dimensional Array

The warehouse location for the products in the **orion.productlist** data set is given in the following table:

Warehouse Locations

ProductLine	ProductCategory	ProductLocID	Warehouse
21	0	0	A2100
21	0	1	A2101
21	1	0	A2110
21	1	1	A2111
21	2	0	A2120
21	2	1	A2121
22	0	0	B2200
22	0	1	B2201

22	1	0	B2210
22	1	1	B2211
22	2	0	B2220
22	2	1	B2221

Open the program **p305e06** that retrieves the Level 1 products from the **orion.productlist** data set.

p305e06

```
data warehouses;
  set orion.productlist(keep=ProductID ProductName ProductLevel
                        where=(ProductLevel=1));
  ProdID=put(ProductID,12.);
  ProductLine=input(substr(ProdID,1,2),2.);
  ProductCategory=input(substr(ProdID,3,2),2.);
  ProductLocID=input(substr(ProdID,12,1),1.);
  if ProductLine in (21,22) and ProductCategory<=2
    and ProductLocID<2;
run;
```

Modify **p305e06** to obtain the desired results.

- Enter the values of the **Warehouse** column into a three-dimensional array. Use the values of **ProductLine**, **ProductCategory**, and **ProductLocID** as the dimensions. Create a data set named **warehouses** that includes the warehouse for each product.
- Print the first five observations of the **warehouses** data set.

PROC PRINT Output

Warehouses Data					
Obs	ProductID	ProductName	Product Level		
1	210200400020	Kids Baby Edge Max Shoes			1
2	210200400070	Tony's Children's Deschutz (Bg) Shoes			1
3	210201000050	Kid Children's T-Shirt			1
4	220100100101	Big Guy Men's Chaser Poplin Pants			1
5	220100100241	Big Guy Men's Santos Shorts Dri Fit			1
Obs	ProdID	Product Line	Product Category	Product LocID	Warehouse
1	210200400020	21	2	0	A2120
2	210200400070	21	2	0	A2120
3	210201000050	21	2	0	A2120
4	220100100101	22	1	1	B2211
5	220100100241	22	1	1	B2211

5.4 Loading a Multidimensional Array from a SAS Data Set

Objectives

- Load a multidimensional array from a SAS data set.
- Identify the advantages of an array as a lookup table.

97

Business Scenario

Compare the actual profit values for each Orion Star company to the budgeted profit values.



98

Business Scenario

This time, the budget values are stored in the **orion.budget** data set.

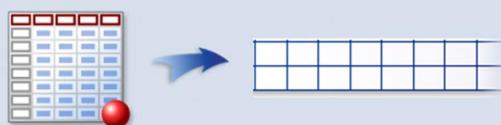
Partial **orion.budget**

Month	Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
1	\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
2	\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000
3	\$1,160,000	\$1,380,000	\$1,640,000	\$1,410,000	\$1,440,000
4	\$1,710,000	\$2,100,000	\$2,420,000	\$2,130,000	\$2,270,000
5	\$1,990,000	\$2,350,000	\$2,840,000	\$2,480,000	\$2,670,000
.					
.					
12	\$2,870,000	\$3,120,000	\$3,760,000	\$3,210,000	\$4,370,000

99

Loading an Array from a Data Set

When there are too many values to easily hardcode in an array, an array can be loaded from a SAS data set.



100

Using Multidimensional Arrays

```

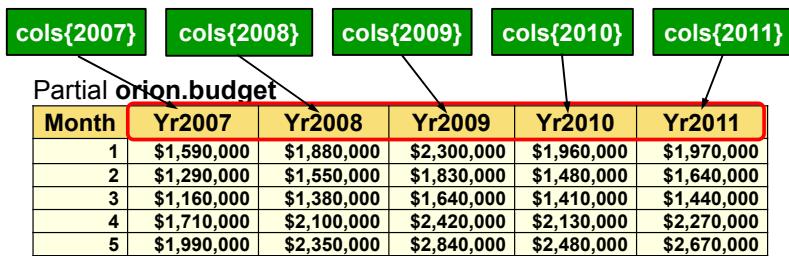
data budgetamt;
  drop Yr2007-Yr2011 Month Mon Yr Y M;
  array B{12,2007:2011} _temporary_;
  if _N_=1 then do Mon=1 to 12;
    set orion.budget;
    array cols{2007:2011} Yr2007-Yr2011;
    do Yr=2007 to 2011;
      B{Mon,Yr}=cols{Yr};
    end;
  end;
  set orion.profit(where=(Sales ne .));
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;

```

101

p305d03

Loading the Array



102

Using Multidimensional Arrays

```

data budgetamt;
  drop Yr2007-Yr2011 Month Mon Yr Y M;
  array B{12,2007:2011} _temporary_;
  if _N_=1 then do Mon=1 to 12;
    set orion.budget;
    array cols {2007:2011} Yr2007-Yr2011;
    do Yr=2007 to 2011;
      B{Mon,Yr}=cols{Yr};
    end;
  end;
  set orion.profit(where=(Sales ne .));
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;

```

103

p305d03

Loading the Array

cols{2007}

Partial orion.budget

Month	Yr2007	Yr2008	Yr2009	Yr2010	Yr2011
1	\$1,590,000	\$1,880,000	\$2,300,000	\$1,960,000	\$1,970,000
2	\$1,290,000	\$1,550,000	\$1,830,000	\$1,480,000	\$1,640,000
3	\$1,160,000	\$1,380,000	\$1,640,000	\$1,410,000	\$1,440,000
4	\$1,710,000	\$2,100,000	\$2,420,000	\$2,130,000	\$2,270,000
5	\$1,990,000	\$2,350,000	\$2,840,000	\$2,480,000	\$2,670,000

B{1,2007}

104

5.09 Quiz

How many elements are in the array that is defined by the following ARRAY statement?

```
array B{12,2007:2011} _temporary_;
```

- a. 0
- b. 24
- c. 48
- d. 60

105

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
.	1	...
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N	-	1	...	
.	1	...	

107

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
  drop Yr2007-Yr2011 Month Mon Yr Y M;
  array B{12,2007:2011} _temporary_;
  if _N_=1 then do Mon=1 to 12;
    set orion.budget;
    array cols{2007:2011} Yr2007-Yr2011;
    do Yr=2007 to 2011;
      B{Mon,Yr}=cols{Yr};
    end;
  end;
  set orion.profit(where=(Sales ne .));
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;
```

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
1
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _{N_}
.	1

108 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
  drop Yr2007-Yr2011 Month Mon Yr Y M;
  array B{12,2007:2011} _temporary_;
  if _N_=1 then do Mon=1 to 12;
    set orion.budget;
    array cols{2007:2011} Yr2007-Yr2011;
    do Yr=2007 to 2011;
      B{Mon,Yr}=cols{Yr};
    end;
  end;
  set orion.profit(where=(Sales ne .));
  Y=year(YYMM);
  M=month(YYMM);
  BudgetAmt=B{M,Y};
run;
```

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
1	1	1590000	1880000	2300000	1960000	1970000	.	.
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _{N_}
.	1

109 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
1		1	1590000	1880000	2300000	1960000	1970000	2007								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1				...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{1,2007}=cols{2007};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
1		1	1590000	1880000	2300000	1960000	1970000	2007								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1				...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

1590000

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
1	1	1590000	1880000	2300000	1960000	1970000	2008	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

113

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

1590000

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
1	1	1590000	1880000	2300000	1960000	1970000	2008	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

114

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do yr=2007 to 2011;
    B{1,2008}=cols{2008};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000							...	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
1		1	1590000	1880000	2300000	1960000	1970000	2008								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N
.	1

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Yr out
of bounds

Continue until Yr=2012

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000				...	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
1		1	1590000	1880000	2300000	1960000	1970000	2012								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N
.	1

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

Partial PDV

cols{2007}	cols{2008}	cols{2009}	cols{2010}	cols{2011}				
D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
2	1	1590000	1880000	2300000	1960000	1970000	2012	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

117 ...

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

Partial PDV

cols{2007}	cols{2008}	cols{2009}	cols{2010}	cols{2011}				
D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
2	2	1290000	1550000	1830000	1480000	1640000	2012	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

118 ...

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000				..	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
2		2	1290000	1550000	1830000	1480000	1640000	2007								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1

119

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{2,2007}=cols{2007};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000			..	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
2		2	1290000	1550000	1830000	1480000	1640000	2007								
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1

121

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

data budgetamt;
 drop Yr2007-Yr2011 Month Mon Yr Y M;
 array B{12,2007:2011} _temporary_;
 if N=1 then do Mon=1 to 12;
 set orion.budget;
 array cols{2007:2011} Yr2007-Yr2011;
 do Yr=2007 to 2011;
 B{Mon,Yr}=cols{Yr};
 end;
 end;
 set orion.profit(where=(Sales ne .));
 Y=year(YYMM);
 M=month(YYMM);
 BudgetAmt=B{M,Y};
run;

B[1,2007]	B[1,2008]	B[1,2009]	B[1,2010]	B[1,2011]	B[2,2007]	B[2,2008]	B[2,2009]	B[2,2010]	B[2,2011]
1590000	1880000	2300000	1960000	1970000	1290000			...	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
2			2		1290000		1550000		1830000		1480000		1640000		2008	

YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget Amt	D	N
.	1

122

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

data budgetamt;
 drop Yr2007-Yr2011 Month Mon Yr Y M;
 array B{12,2007:2011} _temporary_;
 if N=1 then do Mon=1 to 12;
 set orion.budget;
 array cols{2007:2011} Yr2007-Yr2011;
 do Yr=2007 to 2011;
 B{2,2008}=cols{2008};
 end;
 end;
 set orion.profit(where=(Sales ne .));
 Y=year(YYMM);
 M=month(YYMM);
 BudgetAmt=B{M,Y};
run;

B[1,2007]	B[1,2008]	B[1,2009]	B[1,2010]	B[1,2011]	B[2,2007]	B[2,2008]	B[2,2009]	B[2,2010]	B[2,2011]
1590000	1880000	2300000	1960000	1970000	1290000	1550000		...	

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
2			2		1290000		1550000		1830000		1480000		1640000		2008	

YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget Amt	D	N
.	1

123

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

B{1,2007} B{1,2008} B{1,2009} B{1,2010} B{1,2011} B{2,2007} B{2,2008} B{2,2009} B{2,2010} B{2,2011}

1590000	1880000	2300000	1960000	1970000	1290000	1550000
---------	---------	---------	---------	---------	---------	---------	-----	-----	-----

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

Don	DMonth	DYr2007	DYr2008	DYr2009	DYr2010	DYr2011	DYr	Company
2	2	1290000	1550000	1830000	1480000	1640000	2009	

YYMM	Sales	Cost	Salaries	Profit	D Y	D M	Budget Amt	D N_
.	1

124 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

B{1,2007} B{1,2008} B{1,2009} B{1,2010} B{1,2011} B{2,2007} B{2,2008} B{2,2009} B{2,2010} B{2,2011}

1590000	1880000	2300000	1960000	1970000	1290000	1550000
---------	---------	---------	---------	---------	---------	---------	-----	-----	-----

Eventually, Mon=12 and Yr=2010

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

Partial PDV

Don	DMonth	DYr2007	DYr2008	DYr2009	DYr2010	DYr2011	DYr	Company
12	12	2870000	3120000	3760000	3210000	4370000	2010	

YYMM	Sales	Cost	Salaries	Profit	D Y	D M	Budget Amt	D N_
.	1

125 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
set orion.budget;
array cols{2007:2011} Yr2007-Yr2011;
do Y=2007 to 2011;
B{12,2010}=cols{2010};
else,
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
12	12	2870000	3120000	3760000	3210000	4370000	2010	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

127 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
set orion.budget;
array cols{2007:2011} Yr2007-Yr2011;
do Y=2007 to 2011;
B{Mon,Yr}=cols{Yr};
end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	

Partial PDV

D _{on}	D _{Month}	D _{Yr2007}	D _{Yr2008}	D _{Yr2009}	D _{Yr2010}	D _{Yr2011}	D _{Yr}	Company
12	12	2870000	3120000	3760000	3210000	4370000	2011	
YYMM	Sales	Cost	Salaries	Profit	D _Y	D _M	Budget Amt	D _N
.	1

128 ...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{12,2011}=cols{2011};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
12		12	2870000	3120000	3760000	3210000	4370000	4370000	2011							
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1

130

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company
12		12	2870000	3120000	3760000	3210000	4370000	4370000	2012							
YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget	Amt	D	N				
.	1

131

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

B{1,2007} B{1,2008} B{1,2009} B{1,2010} B{1,2011} B{2,2007} B{2,2008} B{2,2009} B{2,2010} B{2,2011}

1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000
---------	---------	---------	---------	---------	---------	---------	-----	---------	---------

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company		
			13		12		2870000		3120000		3760000		3210000		4370000		2012	
132																		

YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget Amt	D	N
.	1

...

Execution

Partial orion.budget

Month	Yr2007	Yr2008	...
1	1590000	1880000	...
2	1290000	1550000	...
3	1160000	1380000	...
4	1710000	2100000	...
.
.

B{1,2007} B{1,2008} B{1,2009} B{1,2010} B{1,2011} B{2,2007} B{2,2008} B{2,2009} B{2,2010} B{2,2011}

1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000
---------	---------	---------	---------	---------	---------	---------	-----	---------	---------

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Partial PDV

D	on	D	Month	D	Yr2007	D	Yr2008	D	Yr2009	D	Yr2010	D	Yr2011	D	Yr	Company		
			13		12		2870000		3120000		3760000		3210000		4370000		2012	
133																		

YYMM	Sales	Cost	Salaries	Profit	D	Y	D	M	Budget Amt	D	N
.	1

...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

cols{2007} cols{2008} cols{2009} cols{2010} cols{2011}

D>on	D>Month	D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Yr	Company
13	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M01	457809	210914	127525	119370	.	.	.	1

134

...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```
data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;
```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

cols{2007} cols{2008} cols{2009} cols{2010} cols{2011}

D>on	D>Month	D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Yr	Company
13	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M01	457809	210914	127525	119370	2007	1	.	1

135

...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;


```

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D→on	D→Month	D→Yr2007	D→Yr2008	D→Yr2009	D→Yr2010	D→Yr2011	D→Yr	Company
13	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D→Y	D→M	Budget Amt	D→N_
07M01	457809	210914	127525	119370	2007	1	.	1

136

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
BudgetAmt=B{1,2007};
run;

```

B{1,2007}	B{1,2008}	B{1,2009}	B{1,2010}	B{1,2011}	B{2,2007}	B{2,2008}	B{2,2009}	B{2,2010}	B{2,2011}
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D→on	D→Month	D→Yr2007	D→Yr2008	D→Yr2009	D→Yr2010	D→Yr2011	D→Yr	Company
13	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D→Y	D→M	Budget Amt	D→N_
07M01	457809	210914	127525	119370	2007	1	1590000	1

137

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.budget;
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Implicit OUTPUT;
Implicit RETURN;

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

cols{2007} cols{2008} cols{2009} cols{2010} cols{2011}

D>on	D>Month	D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Yr	Company
13	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M01	457809	210914	127525	119370	2007	1	1590000	1

138

...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.budget;
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Reinitialize PDV

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

cols{2007} cols{2008} cols{2009} cols{2010} cols{2011}

D>on	D>Month	D>Yr2007	D>Yr2008	D>Yr2009	D>Yr2010	D>Yr2011	D>Yr	Company
.	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M01	457809	210914	127525	119370	.	.	.	2

139

...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

False

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D→on	D→Month	D→Yr2007	D→Yr2008	D→Yr2009	D→Yr2010	D→Yr2011	D→Yr	Company
.	12	2870000	3120000	3760000	3210000	4370000	2012	Logistics
YYMM	Sales	Cost	Salaries	Profit	D→Y	D→M	Budget Amt	D→N_
07M02	457809	210914	127525	119370	.	.	.	2

140 ...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if N=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D→on	D→Month	D→Yr2007	D→Yr2008	D→Yr2009	D→Yr2010	D→Yr2011	D→Yr	Company
.	12	2870000	3120000	3760000	3210000	4370000	.	Logistics
YYMM	Sales	Cost	Salaries	Profit	D→Y	D→M	Budget Amt	D→N_
07M02	325138	149718	127525	47895	.	.	.	2

141 ...

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D>on D>Month D>Yr2007 D>Yr2008 D>Yr2009 D>Yr2010 D>Yr2011 D>Yr Company
. 12 2870000 3120000 3760000 3210000 4370000 . Logistics

YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M02	325138	149718	127525	47895	2007	2	1290000	2

142

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	325138	...
Logistics	07M03	276805	...
Logistics	07M04	558806	...
.

```

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011} _temporary_;
if _N_=1 then do Mon=1 to 12;
  set orion.budget;
  array cols{2007:2011} Yr2007-Yr2011;
  do Yr=2007 to 2011;
    B{Mon,Yr}=cols{Yr};
  end;
end;
set orion.profit(where=(Sales ne .));
Y=year(YYMM);
M=month(YYMM);
BudgetAmt=B{M,Y};
run;

```

Implicit OUTPUT;
Implicit RETURN;

B(1,2007)	B(1,2008)	B(1,2009)	B(1,2010)	B(1,2011)	B(2,2007)	B(2,2008)	B(2,2009)	B(2,2010)	B(2,2011)
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000

Partial PDV

D>on D>Month D>Yr2007 D>Yr2008 D>Yr2009 D>Yr2010 D>Yr2011 D>Yr Company
. 12 2870000 3120000 3760000 3210000 4370000 . Logistics

YYMM	Sales	Cost	Salaries	Profit	D>Y	D>M	Budget Amt	D>N_
07M02	325138	149718	127525	47895	2007	2	1290000	2

143

Execution

Partial orion.profit

Company	YYMM	Sales	...
Logistics	07M01	457809	...
Logistics	07M02	200100	
Logistics	07M03		
Logistics	07M04	558806	...
.

data budgetamt;
drop Yr2007-Yr2011 Month Mon Yr Y M;
array B{12,2007:2011}_temporary_;
if _N_=1 then do Mon=1 to 12;
set orion.budget;
array cols{2007:2011} Yr2007-Yr2011,
do Yr=2007 to 2011;
ls{Yr};
run;

Continue until EOF in orion.profit.

Yr	2007	2008	2009	2010	2011	2007	2008	2009	2010	2011
1590000	1880000	2300000	1960000	1970000	1290000	1550000	...	3210000	4370000	

Partial PDV

cols{2007}	cols{2008}	cols{2009}	cols{2010}	cols{2011}	D_Yr	D_Month	D_Yr2007	D_Yr2008	D_Yr2009	D_Yr2010	D_Yr2011	D_Yr	Company
.	12	2870000	3120000	3760000	3210000	4370000	.	Logistics					

YYMM	Sales	Cost	Salaries	Profit	D_Y	D_M	Budget Amt	D_N_
07M02	325138	149718	127525	47895	2007	2	1290000	2

144

Using Arrays for Table Lookups

Advantages	Disadvantages
Faster than a hash object or format, if you can use it	A contiguous chunk of memory requested at compile time
Use of positional order	Memory requirements to load the entire array
Use of multiple values to determine the array element to be returned	Must have a numeric value as a pointer to the array elements
Ability to use a non-sorted and non-indexed base data set	Can return only a single value from the lookup operation
Use of numeric expressions to determine which element of the array is to be looked up; exact match not required	Must supply dimensions at compile time by either hardcoding or macro variables



Exercises

Level 1

7. Using a Two-Dimensional Array

Orion Star wants to send discount coupons to the customers. The amounts of the discounts are given in the data set **orion.coupons**.

orion.coupons

orion.coupons Data Set							
Obs	OT	Quantity1	Quantity2	Quantity3	Quantity4	Quantity5	Quantity6
1	1	10	10	15	20	20	25
2	2	10	15	20	25	25	30
3	3	10	15	15	20	25	25

The data set **orion.orderfact** contains the variables **CustomerID**, **OrderType**, and **Quantity**.

Partial **orion.orderfact**

Obs	CustomerID	Order	
		Type	Quantity
1	63	1	1
2	5	2	1
3	45	2	1
4	41	1	2
5	183	1	3

- Create a two-dimensional array with the values from **orion.coupons**. Use values from **orion.orderfact** and the array to create a new variable named **CouponValue**. Name the new data set **customercoupons**.
- Print the first five observations of the **customercoupons** data set.

customercoupons Data Set			
CustomerID	Order		Coupon
	Type	Quantity	Value
63	1	1	10
5	2	1	10
45	2	1	10
41	1	2	10
183	1	3	15

8. Using a Two-Dimensional Array

Orion Star wants to send discount coupons to their customers. The amounts of the discounts are stored in the data set **orion.couponpct**.

orion.couponpct

orion.couponpct			
Obs	OT	Quant	Value
1	1	1	10
2	1	2	10
3	1	3	15
4	1	4	20
5	1	5	20
6	1	6	25
7	2	1	10
8	2	2	15
9	2	3	20

10	2	4	25
11	2	5	25
12	2	6	30
13	3	1	10
14	3	2	15
15	3	3	15
16	3	4	20
17	3	5	25
18	3	6	25

The data set **orion.orderfact** contains the variables **CustomerID**, **OrderType**, and **Quantity**.

Partial **orion.orderfact**

Obs	CustomerID	Order Type	Quantity
1	63	1	1
2	5	2	1
3	45	2	1
4	41	1	2
5	183	1	3

- a. Create a two-dimensional array with the values from **orion.couponpct**. Use values from **orion.orderfact** and the array to create a new variable named **CouponValue**. Name the new data set **customercoupons**.
- b. Print the first five observations of the **customercoupons** data set.

PROC PRINT Output

The Coupon Percentage Value			
CustomerID	Order Type	Quantity	Coupon Value
63	1	1	10
5	2	1	10
45	2	1	10
41	1	2	10
183	1	3	15

Level 2

9. Using a Two-Dimensional Array

The data set **orion.msp** contains the average manufacturer's suggested retail price for shoes, based on the product line and the product category. The product group ID is the last two digits of **ProdCatID**.

orion.msp

Obs	Prod Line	Prod CatID	AvgSuggested RetailPrice
1	21	2101	.
2	21	2102	70.79

	3	22	2201	173.79
	4	22	2202	174.40
	5	23	2301	.
	6	23	2302	.
	7	24	2401	29.63
	8	24	2402	287.80

The data set **orion.shoesales** contains the variables **ProductID**, **ProductName**, and **TotalRetailPrice** for all of the shoes sold by Orion Star.

Partial **orion.shoesales**

ProductID	ProductName	TotalRetail Price
220200200024	Pro Fit Gel Gt 2030 Women's Running Shoes	\$178.50
220200100092	Big Guy Men's Air Terra Sebec Shoes	\$83.00
240200100043	Bretagne Performance Tg Men's Golf Shoes L.	\$282.40
220100700024	Armadillo Road Dmx Women's Running Shoes	\$99.70
220200300157	Hardcore Men's Street Shoes Large	\$220.20

- Create a data set named **combine**. Use a two-dimensional array to combine the table of values from **orion.msp** with **orion.shoesales**.
 - Create a new variable named **ManufacturerSuggestedPrice** based on the values of the product line and product category.
 - The product line is indicated by the first two digits of the **ProductID** variable.
 - The product category ID is indicated by the third and fourth digits of the **ProductID** variable.
 - Keep only the **ProductID**, **ProductName**, **TotalRetailPrice**, and **ManufacturerSuggestedPrice** variables.
- Print the first five observations of the **combine** data set.

PROC PRINT Output

combine Data Set				
Manufacturer Suggested Price	ProductID	ProductName	TotalRetail Price	
\$174.40	220200200024	Pro Fit Gel Gt 2030 Women's Running Shoes		\$178.50
\$174.40	220200100092	Big Guy Men's Air Terra Sebec Shoes		\$83.00
\$287.80	240200100043	Bretagne Performance Tg Men's Golf Shoes L.		\$282.40
\$173.79	220100700024	Armadillo Road Dmx Women's Running Shoes		\$99.70
\$174.40	220200300157	Hardcore Men's Street Shoes Large		\$220.20

Challenge

10. Using a Three-Dimensional Array

The data set **orion.warehouses** contains the warehouse location for all products. The **ProductLine** variable has values from 21 to 24 inclusive. The **ProductCatID** variable has values from 0 to 8 inclusive. The variable **ProductLocID** has values from 0 to 9 inclusive.

Partial **orion.warehouses**

orion.warehouses				
Obs	Product Line	Product CatID	Product LocID	Warehouse
1	21	0	0	A2100
2	21	0	1	A2101
3	21	1	0	A2110
4	21	1	1	A2111
5	21	2	0	A2120
6	21	2	2	A2122
7	21	2	3	A2123
8	21	2	4	A2124
9	21	2	5	A2125
10	21	2	6	A2126

- Write a DATA step to create a data set named **warehouses**.
- Load the values from **orion.warehouses** into a three-dimensional array.
- Read only the variables **ProductID**, **ProductName**, and **ProductLevel** and all of the observations from **orion.productlist** where **ProductLevel** = 1. Use the **ProductID** variable to determine the values of **ProductLine**, **ProductCatID**, and **ProductLocID**.
 - The product line is indicated by the first two digits of the **ProductID** variable.
 - The product category ID is indicated by the third and fourth digits of the **ProductID** variable.
 - The product location ID identifies the location of the product within a warehouse and is the last digit of the **ProductID** variable.
- Use **ProductLine**, **ProductCatID**, and **ProductLocID** to retrieve the value from the array and create a variable named **Warehouse**.
- Keep only the variables **ProductID**, **ProductName**, and **Warehouse**.
- Print the first five observations of the **warehouses** data set.

PROC PRINT Output

warehouses		
Warehouse	ProductID	ProductName
A2129	210200100009	Kids Sweat Round Neck, Large Logo
A2127	210200100017	Sweatshirt Children's O-Neck
A2122	210200200022	Sunfit Slow Swimming Trunks
A2123	210200200023	Sunfit Stockton Swimming Trunks Jr.
A2126	210200300006	Fleece Cuff Pant Kid'S

5.5 Solutions

Solutions to Exercises

1. Using a One-Dimensional Array to Combine Data

- a. Combine the **orion.retail** and **orion.retailinformation** data sets to create a data set named **compare**. The data set should contain the variables from **orion.retail** and variables named **Month** and **MedianRetailPrice**, where **Month** is the month of the date on which the product was ordered.

- 1) Enter the following DATA step in the editor:

Partial p305s01

```
data compare;
  drop Month1-Month12 Statistic;
  array mon{12} Month1-Month12;
  if _N_=1 then
    set orion.retailinformation
      (where=(Statistic='MedianRetailPrice'));
  set orion.retail;
  Month=month(OrderDate);
  MedianRetailPrice=mon{Month};
  format MedianRetailPrice dollar8.2;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
4  data compare;
5    drop Month1-Month12 Statistic;
6    array mon{12} Month1-Month12;
7    if _N_=1 then
8      set orion.retailinformation(where=(Statistic='MedianRetailPrice'));
9    set orion.retail;
10   Month=month(OrderDate);
11   MedianRetailPrice=mon{Month};
12   format MedianRetailPrice dollar8.2;
13 run;
NOTE: There were 1 observations read from the data set ORION.RETAILINFORMATION.
      WHERE Statistic='MedianRetailPrice';
NOTE: There were 324 observations read from the data set ORION.RETAIL.
NOTE: The data set WORK.COMPARE has 324 observations and 13 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds
```

- b. Print the first five observations of the resulting data set.

- 1) Enter the following PROC PRINT step in the editor:

Partial p305s01

```
proc print data=compare(obs=5);
  title 'Partial Compare Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
14  proc print data=compare(obs=5);
15    title 'Partial Compare Data Set';
16  run;
NOTE: There were 5 observations read from the data set WORK.COMPARE.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.03 seconds
      cpu time          0.03 seconds
```

2. Using a One-Dimensional Array as a Lookup Table

- a. Use an array to create a data set that is named **trans** that has 24 observations.

- 1) Enter the following DATA step in the editor:

Partial p305s02

```
data trans;
  drop Product21-Product24;
  array prod{21:24} Product21-Product24;
  set orion.shoestats;
  do ProductLine=21 to 24;
    Value=prod{ProductLine};
    output;
  end;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
19  data trans;
20    drop Product21-Product24;
21    array prod{21:24} Product21-Product24;
22    set orion.shoestats;
23    do ProductLine=21 to 24;
24      Value=prod{ProductLine};
25      output;
26    end;
27  run;
NOTE: There were 6 observations read from the data set ORION.SHOESTATS.
NOTE: The data set WORK.TRANS has 24 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
```

- b. Print the first five observations of the **trans** data set.

- 1) Enter the following PROC PRINT step in the editor:

p305s02

```
proc print data=trans(obs=5);
  title 'The TRANS data set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
28 proc print data=trans(obs=5);
29   title 'The TRANS data set';
30 run;
NOTE: There were 5 observations read from the data set WORK.TRANS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
```

3. Using a One-Dimensional Array

- a. Use the program p305e03 to create a temporary data set that is named orderfact for the year 2011 and customer IDs 89 and 2550, sorted by OrderType.
- b. Create the data set named all that has one observation for each OrderType, where there are a varying number of observations for each OrderType in the original orderfact data set. Use the maximum number of observations for all order types as the array dimension. Create three arrays that create variables to hold the order dates, the delivery dates, and the quantity.

- 1) Enter the following DATA step in the editor after the PROC SORT and PROC SQL steps:

Partial **p305s03**

```
data all;
  array ordt{*} OrderedDate1-OrderedDate4;
  array deldt{*} DeliveryDate1-DeliveryDate4;
  array q{*} Quantity1 - Quantity4;
  format OrderedDate1-OrderedDate4 DeliveryDate1-
    DeliveryDate4 date9.;

N=0;
do until (last.OrderType);
  set orderfact;
  by OrderType;
  N+1;
  ordt{N}=OrderDate;
  deldt{N}=DeliveryDate;
  q{N}=Quantity;
end;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```

45  data all;
46    array ordt{*} OrderedDate1-OrderedDate4;
47    array deldt{*} DeliveryDate1-DeliveryDate4;
48    array q{*} Quantity1 - Quantity4;
49    format OrderedDate1-OrderedDate4 DeliveryDate1-DeliveryDate4 date9. ;
50    N=0;
51    do until (last.OrderType);
52      set orderfact;
53      by OrderType;
54      N+1;
55      ordt{N}=OrderDate;
56      deldt{N}=DeliveryDate;
57      q{N}=Quantity;
58    end;
59  run;
NOTE: There were 7 observations read from the data set WORK.ORDERFACT.
NOTE: The data set WORK.ALL has 3 observations and 18 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time          0.03 seconds

```

- c. Print the all data set.
- 1) Enter the following PROC PRINT step in the editor:
- Partial p305s03**
- ```

proc print data=all;
 title 'The ALL Data Set';
run;

```
- 2) Submit the PROC PRINT step.
  - 3) Verify that the output from your PROC PRINT step matches the expected output.
  - 4) Examine the SAS log to verify that the program executed without errors.

#### SAS Log

```

60 proc print data=all;
61 title 'The ALL Data Set';
62 run;
NOTE: There were 3 observations read from the data set WORK.ALL.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 4. Using a Two-Dimensional Array

- a. Use a two-dimensional array to combine the data set with the table of values to create a data set that is named **customercoupons** with a variable named **CouponValue**.

- 1) Enter the following DATA step in the editor:

## Partial p305s04

```

data customercoupons;
array pct{3,6} _temporary_ (10, 10, 15, 20, 20, 25,
 10, 15, 20, 25, 25, 30,
 10, 15, 15, 20, 25, 25);
set orion.orderfact(keep=CustomerID OrderType Quantity);
CouponValue=pct{OrderType,Quantity};
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

## SAS Log

```

65 data customercoupons;
66 array pct{3,6} _temporary_ (10, 10, 15, 20, 20, 25,
67 10, 15, 20, 25, 25, 30,
68 10, 15, 15, 20, 25, 25);
69 set orion.orderfact(keep=CustomerID OrderType Quantity);
70 CouponValue=pct{OrderType,Quantity};
71 run;

NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.CUSTOMERCOUPONS has 617 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.03 seconds

```

- c. Print the first five observations of the **customercoupons** data set.

- 1) Enter the following PROC PRINT step in the editor:

## Partial p305s04

```

proc print data=customercoupons(obs=5);
 title 'The Coupon Value';
run;

```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

## SAS Log

```

72 proc print data=customercoupons(obs=5);
73 title 'The Coupon Value';
74 run;

NOTE: There were 5 observations read from the data set WORK.CUSTOMERCOUPONS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

## 5. Using a Two-Dimensional Array

- a. Create a data set named **combine**. Use a two-dimensional array to combine the table of values with the product line and the product category ID.

- 1) Enter the following DATA step in the editor:

**Partial p305s05**

```
data combine (drop=ProdID);
array msp{21:24,2} _temporary_
(.,70.79,173.79,174.40,.,.,29.65,287.80);
set orion.shoesales;
ProdID=put(ProductID, 12.);
ProductLine=input(substr(ProdID,1,2),2.);
ProductCategory=input(substr(ProdID,3,2),2.);
ManufacturerSuggestedPrice=msp{ProductLine,
 ProductCategory};
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

**SAS Log**

```
86 data combine (drop=ProdID);
87 array msp{21:24,2} _temporary_ (., 70.79,
88 173.79,174.40,
89 ., .,
90 29.65,287.80);
91 set orion.shoesales;
92 ProdID=put(ProductID, 12.);
93 ProductLine=input(substr(ProdID,1,2),2.);
94 ProductCategory=input(substr(ProdID,3,2),2.);
95 ManufacturerSuggestedPrice=msp{ProductLine, ProductCategory};
96 run;
NOTE: There were 58 observations read from the data set ORION.SHOESALES.
NOTE: The data set WORK.COMBINE has 58 observations and 6 variables.
NOTE: DATA statement used (Total process time):
 real time 0.06 seconds
 cpu time 0.01 seconds
```

- b. Print the first five observations of the **combine** data set.

- 1) Enter the following PROC PRINT step in the editor:

**Partial p305s05**

```
proc print data=combine(obs=5);
 title1 "The Coupon Value";
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

**SAS Log**

```
104 proc print data=combine(obs=5);
105 title1 "The Coupon Value";
106 run;
```

```
NOTE: There were 5 observations read from the data set WORK.COMBINE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
```

## 6. Using a Three-Dimensional Array

Open the program **p305e06** that retrieves the Level 1 products from the **orion.productlist** data set.

Modify **p305e06** to obtain the desired results.

- Enter the values of the **Warehouse** column into a three-dimensional array. Use the values of **ProductLine**, **ProductCategory**, and **ProductLocID** as the dimensions. Create a data set named **warehouses** that includes the warehouse for each product.

- Enter the following DATA step in the editor:

Partial **p305s06**

```
data warehouses;
array W{21:22,0:2,0:1} $ 5 _temporary_
 ('A2100','A2101','A2110','A2111','A2120',
 'A2121','B2200','B2201','B2210','B2211',
 'B2220','B2221');
set orion.productlist(keep=ProductID ProductName
 ProductLevel
 where=(ProductLevel=1));
ProdID=put(ProductID,12.);
ProductLine=input(substr(ProdID,1,2),2.);
ProductCategory=input(substr(ProdID,3,2),2.);
ProductLocID=input(substr(ProdID,12,1),1.);
if ProductLine in (21,22) and ProductCategory<=2
 and ProductLocID<2;
Warehouse=W{ProductLine, ProductCategory, ProductLocID};
run;
```

- Submit the DATA step.
- Examine the SAS log to verify that the program executed without errors.

SAS Log

```
109 data warehouses;
110 array W{21:22,0:2,0:1} $ 5 _temporary_ ('A2100','A2101','A2110','A2111',
111 'A2120','A2121','B2200','B2201','B2210','B2211','B2220','B2221');
112 set orion.productlist(keep=ProductID ProductName ProductLevel
113 where=(ProductLevel=1));
114 ProdID=put(ProductID,12.);
115 ProductLine=input(substr(ProdID,1,2),2.);
116 ProductCategory=input(substr(ProdID,3,2),2.);
117 ProductLocID=input(substr(ProdID,12,1),1.);
118 if ProductLine in (21,22) and ProductCategory<=2
119 and ProductLocID<2;
120 Warehouse=W{ProductLine, ProductCategory, ProductLocID};
121 run;
NOTE: There were 481 observations read from the data set ORION.PRODUCTLIST.
 WHERE ProductLevel=1;
NOTE: The data set WORK.WAREHOUSES has 30 observations and 8 variables.
```

```
NOTE: DATA statement used (Total process time):
 real time 0.04 seconds
 cpu time 0.01 seconds
```

- b. Print the first five observations of the **warehouses** data set.

- 1) Enter the following PROC PRINT step in the editor:

Partial p305s06

```
proc print data=warehouses(obs=5);
 title 'Warehouses Data';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
123 proc print data=warehouses(obs=5);
124 title 'Warehouses Data';
125 run;
NOTE: There were 5 observations read from the data set WORK.WAREHOUSES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

## 7. Using a Two-Dimensional Array

- a. Create a two-dimensional array with the values from **orion.coupons**. Use values from **orion.orderfact** and the array to create a new variable named **CouponValue**. Name the new data set **customercoupons**.

- 1) Enter the following DATA step in the editor:

Partial p305s07

```
data customercoupons;
 drop OT i j Quantity1-Quantity6;
 array pct{3,6} _temporary_;
 if _n_=1 then do i=1 to 3;
 set orion.coupons;
 array quan{6} Quantity1-Quantity6;
 do j=1 to 6;
 pct{i,j}=quan{j};
 end;
 end;
 set orion.orderfact(keep=CustomerID OrderType Quantity);
 CouponValue=pct{OrderType,Quantity};
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

## SAS Log

```

23 data customercoupons;
24 drop OT i j Quantity1-Quantity6;
25 array pct{3,6} _temporary_;
26 if _n_=1 then do i=1 to 3;
27 set orion.coupons;
28 array quan{6} Quantity1-Quantity6;
29 do j=1 to 6;
30 pct{i,j}=quan{j};
31 end;
32 end;
33 set orion.orderfact(keep=CustomerID OrderType Quantity);
34 CouponValue=pct{OrderType,Quantity};
35 run;

NOTE: There were 3 observations read from the data set ORION.COUPONS.
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.CUSTOMERCOUPONS has 617 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

- b. Print the first five observations of the **customercoupons** data set.

- 1) Enter the following PROC PRINT step in the editor:

Partial p305s07

```

proc print data=customercoupons(obs=5) noobs;
 title 'customercoupons Data Set';
run;

```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

## SAS Log

```

37 proc print data=customercoupons(obs=5) noobs;
38 title 'customercoupons Data Set';
39 run;

NOTE: There were 5 observations read from the data set WORK.CUSTOMERCOUPONS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 8. Using a Two-Dimensional Array

- a. Create a two-dimensional array with the values from **orion.couponpct**. Use values from **orion.orderfact** and the array to create a new variable named **CouponValue**. Name the new data set **customercoupons**.

- 1) Enter the following DATA step in the editor:

Partial p305s08

```

data customercoupons;
 drop OT Quant Value i;
 array pct{3,6} _temporary_;

```

```

if _N_=1 then do i=1 to all;
 set orion.couponpct nobs=all;
 pct{OT,Quant}=Value;
end;
set orion.orderfact(keep=CustomerID OrderType Quantity);
CouponValue=pct{OrderType,Quantity};
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

#### SAS Log

```

58 data customercoupons;
59 drop OT Quant Value i;
60 array pct{3,6} _temporary_;
61 if _N_=1 then do i=1 to all;
62 set orion.couponpct nobs=all;
63 pct{OT,Quant}=Value;
64 end;
65 set orion.orderfact(keep=CustomerID OrderType Quantity);
66 CouponValue=pct{OrderType,Quantity};
67 run;
NOTE: There were 18 observations read from the data set ORION.COUPONPCT.
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.CUSTOMERCOUPONS has 617 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

- b. Print the first five observations of the **customercoupons** data set.

- 1) Enter the following PROC PRINT step in the editor:

#### Partial p305s08

```

proc print data=customercoupons(obs=5) noobs;
 title 'The Coupon Value Percentage Value';
run;

```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

#### SAS Log

```

69 proc print data=customercoupons(obs=5) noobs;
70 title 'The Coupon Percentage Value';
71 run;

NOTE: There were 5 observations read from the data set WORK.CUSTOMERCOUPONS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 9. Using a Two-Dimensional Array

- a. Create a data set named **combine**. Use a two-dimensional array to combine the table of values from **orion.msp** with **orion.shoesales**.
- Create a new variable named **ManufacturerSuggestedPrice** based on the values of the product line and product category.
  - The product line is indicated by the first two digits of the **ProductID** variable.
  - The product category ID is indicated by the third and fourth digits of the **ProductID** variable.
  - Keep only the **ProductID**, **ProductName**, **TotalRetailPrice**, and **ManufacturerSuggestedPrice** variables.

- 1) Enter the following DATA step in the editor:

Partial p305s09

```
data combine;
array msp{21:24,2} _temporary_;
keep ProductID ProductName TotalRetailPrice
 ManufacturerSuggestedPrice;
format ManufacturerSuggestedPrice dollar8.2;
if _N_= 1 then do i=1 to All;
 set orion.msp nobs=All;
 msp{ProdLine,input(substr(put(ProdCatID,4.),3,2),2.)}
 =AvgSuggestedRetailPrice;
end;
set orion.shoesales;
ProdID=put(ProductID,12.);
ProductLine=input(substr(ProdID,1,2),2.);
ProductCatID=input(substr(ProdID,3,2),2.);
ManufacturerSuggestedPrice= msp{ProductLine, ProductCatID};
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
94 data combine;
95 array msp{21:24,2} _temporary_ ;
96 keep ProductID ProductName TotalRetailPrice ManufacturerSuggestedPrice;
97 format ManufacturerSuggestedPrice dollar8.2;
98 if _N_= 1 then do i=1 to All;
99 set orion.msp nobs=All;
100 msp{ProdLine,input(substr(put(ProdCatID,4.),3,2),2.)}
101 =AvgSuggestedRetailPrice;
102 end;
103 set orion.shoesales;
104 ProdID=put(ProductID,12.);
105 ProductLine=input(substr(ProdID,1,2),2.);
106 ProductCatID=input(substr(ProdID,3,2),2.);
107 ManufacturerSuggestedPrice=msp{ProductLine, ProductCatID};
108 run;
NOTE: There were 8 observations read from the data set ORION.MSP.
NOTE: There were 58 observations read from the data set ORION.SHOESALES.
```

```
NOTE: The data set WORK.COMBINE has 58 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

- b. Print the first five observations of the **combine** data set.

- 1) Enter the following PROC PRINT step in the editor:

Partial p305s09

```
proc print data=combine(obs=5) noobs;
 title1 'combine Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output from your PROC PRINT step matches the expected output.
- 4) Examine the SAS log to verify that the program executed without errors.

SAS Log

```
110 proc print data=combine(obs=5) noobs;
111 title1 'combine Data Set';
112 run;
NOTE: There were 5 observations read from the data set WORK.COMBINE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

## 10. Using a Three-Dimensional Array

- a. Write a DATA step to create a data set named **warehouses**.
- b. Load the values from **orion.warehouses** into a three-dimensional array.
- c. Read only the variables **ProductID**, **ProductName**, and **ProductLevel** and all of the observations from **orion.productlist** where **ProductLevel** = 1. Use the **ProductID** variable to determine the values of **ProductLine**, **ProductCatID**, and **ProductLocID**.
  - The product line is indicated by the first two digits of the **ProductID** variable.
  - The product category ID is indicated by the third and fourth digits of the **ProductID** variable.
  - The product location ID identifies the location of the product within a warehouse and is the last digit of the **ProductID** variable.
- d. Use **ProductLine**, **ProductCatID**, and **ProductLocID** to retrieve the value from the array and create a variable named **Warehouse**.
- e. Keep only the variables **ProductID**, **ProductName**, and **Warehouse**.

- 1) Enter the following DATA step in the editor:

Partial p305s10

```
data warehouses;
 keep ProductID ProductName Warehouse;
 array w{21:24,0:8,0:9} $ 5 _temporary_;
 if _n_=1 then do i=1 to all;
```

```

 set orion.warehouses nobs=all;
 W{ProductLine, ProductCatID, ProductLocID}=Warehouse;
 end;
 set orion.productlist(keep=ProductID ProductName
 ProductLevel
 where=(ProductLevel=1));
 ProdID=put(ProductID,12.);
 ProductLine=input(substr(ProdID,1,2),2.);
 ProductCatID=input(substr(ProdID,3,2),2.);
 ProductLocID=input(substr(ProdID,12,1),1.);
 Warehouse=w{ProductLine, ProductCatID, ProductLocID};
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the program executed without errors.

#### SAS Log

```

115 data warehouses;
116 keep ProductID ProductName Warehouse;
117 array w{21:24,0:8,0:9} $ 5 _temporary_ ;
118 if _n_=1 then do i=1 to all;
119 set orion.warehouses nobs=all;
120 W{ProductLine, ProductCatID, ProductLocID}=Warehouse;
121 end;
122 set orion.productlist(keep=ProductID ProductName
123 ProductLevel
124 where=(ProductLevel=1));
125 ProdID=put(ProductID,12.);
126 ProductLine=input(substr(ProdID,1,2),2.);
127 ProductCatID=input(substr(ProdID,3,2),2.);
128 ProductLocID=input(substr(ProdID,12,1),1.);
129 Warehouse=w{ProductLine, ProductCatID, ProductLocID};
130 run;
NOTE: There were 116 observations read from the data set ORION.WAREHOUSES.
NOTE: There were 481 observations read from the data set ORION.PRODUCTLIST.
 WHERE ProductLevel=1;
NOTE: The data set WORK.WAREHOUSES has 481 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.00 seconds

```

- f. Print the first five observations of the **warehouses** data set.
  - 1) Enter the following PROC PRINT step in the editor:

Partial p305s10

```

proc print data=warehouses(obs=5) noobs;
 title 'warehouses';
run;

```

  - 2) Submit the PROC PRINT step.
  - 3) Verify that the output from your PROC PRINT step matches the expected output.
  - 4) Examine the SAS log to verify that the program executed without errors.

**SAS Log**

```

132 proc print data=warehouses(obs=5) noobs;
133 title 'warehouses';
134 run;
NOTE: There were 5 observations read from the data set WORK.WAREHOUSES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

**Solutions to Student Activities (Polls/Quizzes)****5.01 Multiple Choice Poll – Correct Answer**

Which of these is an example of a table lookup?

- You have the data for January sales in one data set, February sales in a second data set, and March sales in a third. You need to create a report for the entire first quarter.
- b.** You want to send birthday cards to employees. The employees' names and addresses are in one data set and their birthdates are in another.
- You need to calculate the amount that each customer owes for his purchases. The price per item and the number of items purchased are stored in the same data set.

## 5.03 Quiz – Correct Answer

The ARRAY statement below creates temporary array elements.

What is created if \_temporary\_ is omitted?

```
array ContName{91:96} $ 30 _temporary_
 ('North America',
 '',
 'Europe',
 'Africa',
 'Asia',
 'Australia/Pacific');
```

SAS variables, ContName1 – ContName6, and array elements are created.

28

## 5.04 Multiple Choice Poll – Correct Answer

Which data set can be used to populate the array?

- a. orion.employeepayroll
- b. orion.salarystats**

Orion.salarystats contains the average salaries that need to be looked up.

35

## 5.05 Quiz – Correct Answer

Which of the following ARRAY statements are similar to the statement

```
array yr{1978:2011} Yr1978-Yr2011;
```

and compile without errors?

- a. array yr{34} Yr1978-Yr2011;
- b. array yr{1978-2011} Yr1978-Yr2011;
- c. array yr{78:11} Yr1978-Yr2011;
- d. array yr{78-011} Yr1978-Yr2011;

57

## 5.06 Quiz – Correct Answer

Examine the descriptor portion of the data set **orion.profit**. What is the type of the variable **YYMM**?

Use PROC CONTENTS to see the variable type.

```
proc contents data=orion.profit;
run;
```

YYMM is a numeric variable. It represents a SAS date value.

65

## 5.07 Quiz – Correct Answer

What do the data set **orion.profit** and the lookup table have in common?

**They have the month and year in common. In the data set orion.profit, the month and year can be extracted from the variable YYMM. In the table, each row represents a month and each column represents a year.**

69

## 5.08 Quiz – Correct Answer

Which of the answers would be equivalent to the following ARRAY statement?

```
array B{2,5} B1-B10 (1590000, 1880000, 2300000, 1960000, 1970000,
1290000, 1550000, 1830000, 1480000, 1640000);
```

- a. array B{2,2003:2007} B1-B10  
(1590000, 1880000, 2300000, 1960000, 1970000,  
1290000, 1550000, 1830000, 1480000, 1640000);
- b. array B{2,5}  
(1590000, 1880000, 2300000, 1960000, 1970000,  
1290000, 1550000, 1830000, 1480000, 1640000);
- c. array B{2,5} \_temporary\_  
(1590000, 1880000, 2300000, 1960000, 1970000,  
1290000, 1550000, 1830000, 1480000, 1640000);

75

## 5.09 Quiz – Correct Answer

How many elements are in the array that is defined by the following ARRAY statement?

```
array B{12,2007:2011} _temporary_;
```

- a. 0
- b. 24
- c. 48
- d. 60



# Chapter 6 DATA Step Hash and Hiter Objects

|            |                                                       |             |
|------------|-------------------------------------------------------|-------------|
| <b>6.1</b> | <b>Introduction.....</b>                              | <b>6-3</b>  |
| <b>6.2</b> | <b>Hash Object Methods.....</b>                       | <b>6-8</b>  |
|            | Exercises .....                                       | 6-28        |
| <b>6.3</b> | <b>Loading a Hash Object from a SAS Data Set.....</b> | <b>6-30</b> |
|            | Exercises .....                                       | 6-48        |
| <b>6.4</b> | <b>DATA Step Hiter Object .....</b>                   | <b>6-52</b> |
|            | Exercises .....                                       | 6-70        |
| <b>6.5</b> | <b>Solutions .....</b>                                | <b>6-73</b> |
|            | Solutions to Exercises .....                          | 6-73        |
|            | Solutions to Student Activities (Polls/Quizzes) ..... | 6-87        |



# 6.1 Introduction

## Objectives

- Define the DATA step hash object.

3

## Table Lookup Techniques

The technique that is used to perform a table lookup depends on the data.



| Table Location | Lookup Table               | Technique                                         |
|----------------|----------------------------|---------------------------------------------------|
| DATA step      | SAS programming statements | IF-THEN/ELSE statements                           |
| Disk           | SAS data set               | SQL join<br>MERGE<br>SET/SET KEY=                 |
| Memory         | User-defined format        | FORMAT statement<br>PUT statement<br>PUT function |
|                | Array                      | Array reference                                   |
|                | Hash object                | FIND method                                       |

4

## Table Lookup Techniques

This chapter focuses on the hash object, another in-memory lookup table accessible from the DATA step.

| Table Location | Lookup Table               | Technique                                         |
|----------------|----------------------------|---------------------------------------------------|
| DATA step      | SAS programming statements | IF-THEN/ELSE statements                           |
| Disk           | SAS data set               | SQL join<br>MERGE<br>SET/SET KEY=                 |
| Memory         | User-defined format        | FORMAT statement<br>PUT statement<br>PUT function |
|                | Array                      | Array reference                                   |
|                | Hash object                | FIND method                                       |

5

## 6.01 Poll

Have you used hash objects in SAS or other computer languages?

- Yes
- No

6

## Structure of Hash Objects

A hash object resembles a table with rows and columns, and contains a **key** component and a **data** component.

**Hash Object**

| Key | Data |
|-----|------|
| A   | 1    |
| B   | 2    |

Values are retrieved from the data component based on the values in the key component.

7

## Structure of Hash Objects

The key and data components can be numeric or character.

**Hash Object**

| Key | Key | Data | Data |
|-----|-----|------|------|
| A   | 100 | 1    | dog  |
| B   | 101 | 2    | cat  |



The key component can be composite.

The data component can contain multiple data values per key value.

8

## Structure of Hash Objects

Both key and data components must also be defined as DATA step variables.

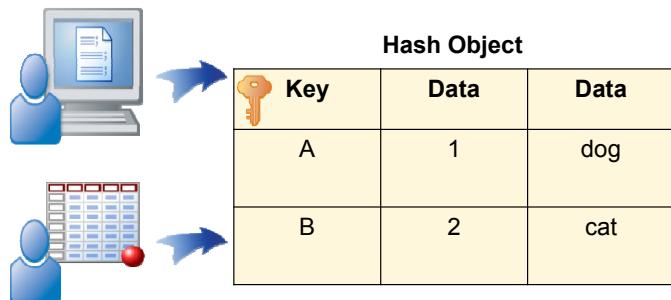
| Key | Data | Data |
|-----|------|------|
| A   | 1    | dog  |
| B   | 2    | cat  |



9

## Structure of Hash Objects

A hash object can be loaded with hardcoded values or from a SAS data set.

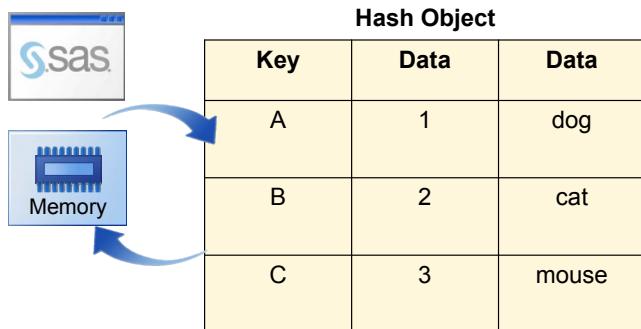


Data loaded into a hash object does not need to be sorted or indexed.

10

## Structure of Hash Objects

The hash object is sized dynamically.



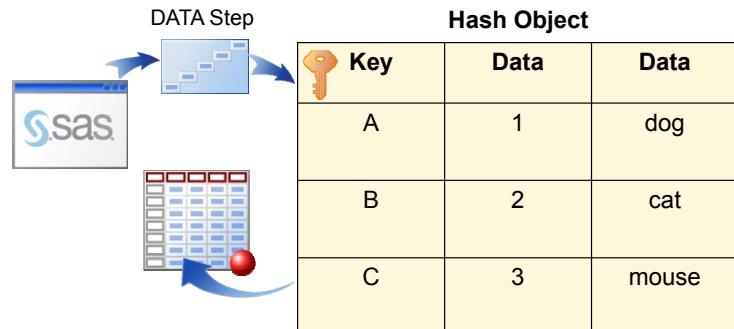
The hash object exists for the duration of the DATA step.

11

...

## Structure of Hash Objects

Data can be downloaded from a hash object to a SAS data set.



12

## 6.2 Hash Object Methods

### Objectives

- Investigate hash object syntax.
- Use hash object methods to load data into a hash object.
- Use a hash object method to look up data.

15

### Business Scenario

Orion Star needs to match continent codes to the code descriptions. The SAS data set **orion.country** has the **ContinentID** variable that contains the code, but no description.

Partial **orion.country**

| Country | CountryName   | Population | ContinentID |
|---------|---------------|------------|-------------|
| AU      | Australia     | 20000000   | 96          |
| CA      | Canada        | .          | 91          |
| DE      | Germany       | 80000000   | 93          |
| IL      | Israel        | 5000000    | 95          |
| TR      | Turkey        | 70000000   | 95          |
| US      | United States | 280000000  | 91          |
| ZA      | South Africa  | 43000000   | 94          |

16

## Business Scenario

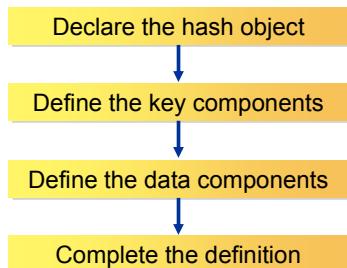
The lookup data is not stored in a data set.

| Continent ID | Continent Name    |
|--------------|-------------------|
| 91           | North America     |
| 93           | Europe            |
| 94           | Africa            |
| 95           | Asia              |
| 96           | Australia/Pacific |

17

## Defining a Hash Object

There are four steps to define a hash object.



18

## Partial Program

This program represents the first attempt at a solution.

```
data country;
 drop rc;
 if _n_=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
 end;
 set orion.country(keep=ContinentID Country
 ContinentName);
 rc=ContName.find();
run;
```

19

p306d01

## Defining a Hash Object: Partial Program

```
data country;
 drop rc; Declare the hash object
 if _n_=1 then do;
 declare hash ContName();
 DECLARE object object-reference(<arg_tag-1:value-1
 <,...arg_tag-n: value-n>);
```

ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country
 ContinentName);
rc=ContName.find();
run;

20

p306d01

## Defining a Hash Object: Partial Program

```

data country;
drop rc; Define the key components
if _n_=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 object.DEFINEKEY (<arg_tag-1: value-1
 <,...arg_tag-n: value-n>>); Africa');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country
 CountryName);
rc=ContName.find();
run;

```

21

p306d01

## Defining a Hash Object: Partial Program

```

data country;
drop rc; Define the data components
if _n_=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 object.DEFINEDATA (<arg_tag-1: value-1
 <,...arg_tag-n: value-n>>); Africa');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country
 CountryName);
rc=ContName.find();
run;

```

22

p306d01

## Defining a Hash Object: Partial Program

```

data country;
drop rc;
if n=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.defineDone();
 object.DEFINEDONE()
 data:'North America';
 data:'Europe';
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country
 CountryName);
rc=ContName.find();
run;

```

Complete the definition

23

p306d01

## Loading Key and Data Values: Partial Program

Next, the hash object is loaded with key and data values.

```

data country;
drop rc;
if n=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.defineDone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.co object.ADD(<arg_tag-1: value-1
 <,...arg_tag-n: value-n>);
rc=ContName.find();
run;

```

The ADD method adds the key and data values to the hash object.

24

p306d01

## Retrieving Matching Data: Partial Program

```

data country;
drop rc;
if _n_=1 then do;
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
run;

```

The FIND method is used to retrieve data from the hash object.

`rc=object.FIND(<KEY: keyvalue-1,..., KEY: keyvalue-n>);`

`rc=ContName.find();`  `run;`

25

p306d01

## Retrieving Matching Data

The FIND method retrieves the value of **ContinentName** from the hash object based on the value of **ContinentID**.

| Hash Object <b>ContName</b> |                               |
|-----------------------------|-------------------------------|
| Key:<br><b>ContinentID</b>  | Data:<br><b>ContinentName</b> |
| 91                          | North America                 |
| 93                          | Europe                        |
| 94                          | Africa                        |
| 95                          | Asia                          |
| 96                          | Australia/Pacific             |

`rc=ContName.find();`

Partial **orion.country**

| Country | ContinentID |
|---------|-------------|
| AU      | 96          |
| CA      | 91          |
| DE      | 93          |
| IL      | 95          |
| TR      | 95          |

Partial PDV

| ContinentName<br>\$ 30 | ContinentID<br>8 | Country<br>\$ 2 |
|------------------------|------------------|-----------------|
|                        | 96               | AU              |

26

p306d01

...

## Retrieving Matching Data

The FIND method returns *Australia/Pacific* as the continent name.

0=success

Hash Object **ContName**

| Key:<br>ContinentID | Data:<br>ContinentName |
|---------------------|------------------------|
| 91                  | North America          |
| 93                  | Europe                 |
| 94                  | Africa                 |
| 95                  | Asia                   |
| 96                  | Australia/Pacific      |

rc=ContName.find();

| Partial orion.country |             |
|-----------------------|-------------|
| Country               | ContinentID |
| AU                    | 96          |
| CA                    | 91          |
| DE                    | 93          |
| IL                    | 95          |
| TR                    | 95          |

Partial PDV

| ContinentName<br>\$ 30 | ContinentID<br>8 | Country<br>\$ 2 |
|------------------------|------------------|-----------------|
| Australia/Pacific      | 96               | AU              |

27

p306d01

## 6.02 Short Answer Poll

Submit the program **p306a01** and examine the SAS log.

You should receive an error. Why?

28

## 6.03 Short Answer Poll

Submit the program **p306a02** and examine the SAS log.

You should receive a note about **ContinentName**.

Why do you get that note?

30

## Completed Program

This program represents the final solution.

```
data country;
 drop rc;
 length ContinentName $ 30;
 if n =1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
 end;
 set orion.country(keep=ContinentID Country
 CountryName);
 rc=ContName.find();
run;
```

32

p306d02



The CALL MISSING routine assigns missing values to the specified character or numeric variables.

## Compilation

```
data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;
```

### PDV

| ContinentName | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ |
|---------------|---------|-------------|-------------|-------|--------|
|               |         |             |             |       |        |

33

p306d02

## 6.04 Short Answer Poll

Why are the statements in the DO group programmed to execute only one time?

```
data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;
```

34

p306d02

**Execution**

True

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

Partial orion.country

| Country | Country Name | Continent ID |
|---------|--------------|--------------|
| AU      | Australia    | 96           |
| CA      | Canada       | 91           |
| DE      | Germany      | 93           |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

36 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

Partial orion.country

| Country | Country Name | Continent ID |
|---------|--------------|--------------|
| AU      | Australia    | 96           |
| CA      | Canada       | 91           |
| DE      | Germany      | 93           |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

37 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |          |
|-----------------------|--------------|--------------|----------------------|----------|
| Country               | Country Name | Continent ID | KEY:                 | ContName |
| AU                    | Australia    | 96           |                      |          |
| CA                    | Canada       | 91           |                      |          |
| DE                    | Germany      | 93           |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

38 p306d02

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |          |
|-----------------------|--------------|--------------|----------------------|----------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | ContName |
| AU                    | Australia    | 96           |                      |          |
| CA                    | Canada       | 91           |                      |          |
| DE                    | Germany      | 93           |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |
|                       |              |              |                      |          |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

39 p306d02

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           |                      |                         |
| CA                    | Canada       | 91           |                      |                         |
| DE                    | Germany      | 93           |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |

**PDV**

| ContinentName | Country | CountryName | ContinentID | ► rc | ► _N_ |
|---------------|---------|-------------|-------------|------|-------|
|               |         |             | .           | .    | 1     |

40 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           |                      |                         |
| CA                    | Canada       | 91           |                      |                         |
| DE                    | Germany      | 93           |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |

**PDV**

| ContinentName | Country | CountryName | ContinentID | ► rc | ► _N_ |
|---------------|---------|-------------|-------------|------|-------|
|               |         |             | .           | .    | 1     |

41 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           | 91                   | North America           |
| CA                    | Canada       | 91           |                      |                         |
| DE                    | Germany      | 93           |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |

**PDV**

| ContinentName | Country | CountryName | ContinentID | ► rc | ► _N_ |
|---------------|---------|-------------|-------------|------|-------|
|               |         |             | .           | .    | 1     |

42 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           | 91                   | North America           |
| CA                    | Canada       | 91           | 93                   | Europe                  |
| DE                    | Germany      | 93           | 94                   | Africa                  |
|                       |              |              | 95                   | Asia                    |
|                       |              |              | 96                   | Australia/<br>Pacific   |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |
|                       |              |              |                      |                         |

**PDV**

| ContinentName | Country | CountryName | ContinentID | ► rc | ► _N_ |
|---------------|---------|-------------|-------------|------|-------|
|               |         |             | .           | .    | 1     |

43 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           | 91                   | North America           |
| CA                    | Canada       | 91           | 93                   | Europe                  |
| DE                    | Germany      | 93           | 94                   | Africa                  |
|                       |              |              | 95                   | Asia                    |
|                       |              |              | 96                   | Australia/<br>Pacific   |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

44 p306d02..

**Execution**



```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |
|-----------------------|--------------|--------------|----------------------|-------------------------|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |
| AU                    | Australia    | 96           | 91                   | North America           |
| CA                    | Canada       | 91           | 93                   | Europe                  |
| DE                    | Germany      | 93           | 94                   | Africa                  |
|                       |              |              | 95                   | Asia                    |
|                       |              |              | 96                   | Australia/<br>Pacific   |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               |         |             | .           | .  | 1   |

45 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |  |
|-----------------------|--------------|--------------|----------------------|-------------------------|--|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America           |  |
| CA                    | Canada       | 91           | 93                   | Europe                  |  |
| DE                    | Germany      | 93           | 94                   | Africa                  |  |
|                       |              |              | 95                   | Asia                    |  |
|                       |              |              | 96                   | Australia/<br>Pacific   |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               | AU      | Australia   | 96          | .  | 1   |

46 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

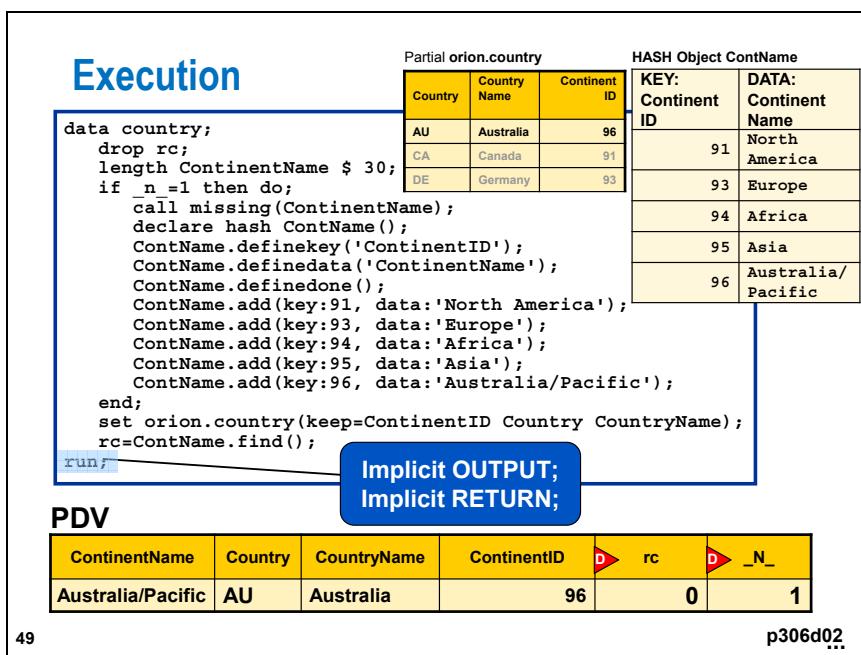
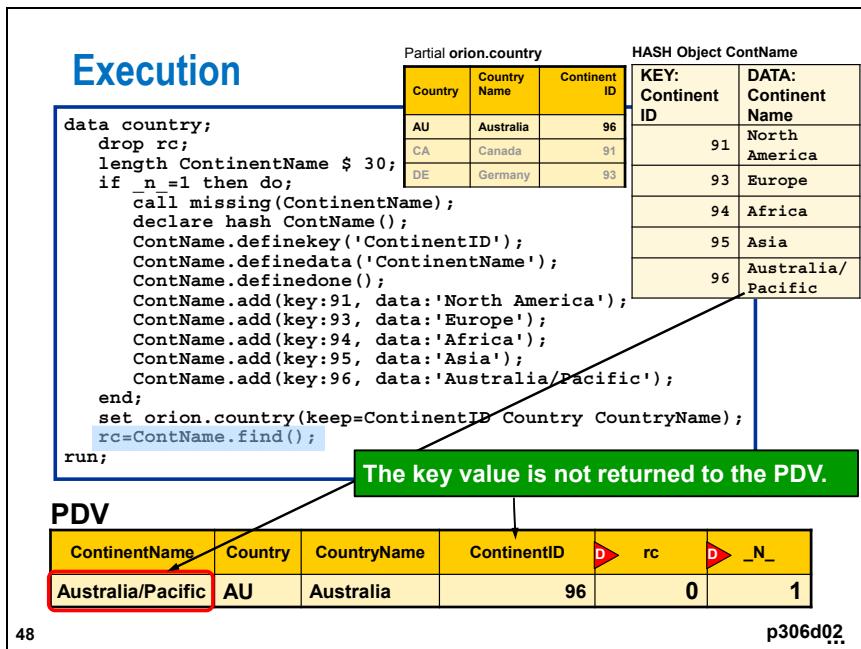
```

| Partial orion.country |              |              | HASH Object ContName |                         |  |
|-----------------------|--------------|--------------|----------------------|-------------------------|--|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America           |  |
| CA                    | Canada       | 91           | 93                   | Europe                  |  |
| DE                    | Germany      | 93           | 94                   | Africa                  |  |
|                       |              |              | 95                   | Asia                    |  |
|                       |              |              | 96                   | Australia/<br>Pacific   |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               | AU      | Australia   | 96          | 0  | 1   |

47 p306d02..



**Execution**

```

data country; Initialize PDV
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                      |  |
|-----------------------|--------------|--------------|----------------------|----------------------|--|
| Country               | Country Name | Continent ID | KEY: Continent ID    | DATA: Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America        |  |
| CA                    | Canada       | 91           | 93                   | Europe               |  |
| DE                    | Germany      | 93           | 94                   | Africa               |  |
|                       |              |              | 95                   | Asia                 |  |
|                       |              |              | 96                   | Australia/Pacific    |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               | AU      | Australia   | 96          | .  | 2   |

50 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

False

| Partial orion.country |              |              | HASH Object ContName |                      |  |
|-----------------------|--------------|--------------|----------------------|----------------------|--|
| Country               | Country Name | Continent ID | KEY: Continent ID    | DATA: Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America        |  |
| CA                    | Canada       | 91           | 93                   | Europe               |  |
| DE                    | Germany      | 93           | 94                   | Africa               |  |
|                       |              |              | 95                   | Asia                 |  |
|                       |              |              | 96                   | Australia/Pacific    |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
|               | AU      | Australia   | 96          | .  | 2   |

51 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |  |
|-----------------------|--------------|--------------|----------------------|-------------------------|--|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America           |  |
| CA                    | Canada       | 91           | 93                   | Europe                  |  |
| DE                    | Germany      | 93           | 94                   | Africa                  |  |
|                       |              |              | 95                   | Asia                    |  |
|                       |              |              | 96                   | Australia/<br>Pacific   |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
| CA            | Canada  |             | 91          | .  | 2   |

52 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

| Partial orion.country |              |              | HASH Object ContName |                         |  |
|-----------------------|--------------|--------------|----------------------|-------------------------|--|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America           |  |
| CA                    | Canada       | 91           | 93                   | Europe                  |  |
| DE                    | Germany      | 93           | 94                   | Africa                  |  |
|                       |              |              | 95                   | Asia                    |  |
|                       |              |              | 96                   | Australia/<br>Pacific   |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
| North America | CA      | Canada      | 91          | 0  | 2   |

53 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

Implicit OUTPUT;  
Implicit RETURN;

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
| North America | CA      | Canada      | 91          | 0  | 2   |

54 p306d02..

**Execution**

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
end;
set orion.country(keep=ContinentID Country CountryName);
rc=ContName.find();
run;

```

Continue until EOF

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc | _N_ |
|---------------|---------|-------------|-------------|----|-----|
| North America | CA      | Canada      | 91          | 0  | 2   |

55 p306d02..

**Setup for the Poll**

```
data country;
 drop rc;
 length ContinentName $ 30;
 if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName();
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 ContName.add(key:91, data:'North America');
 ContName.add(key:93, data:'Europe');
 ContName.add(key:94, data:'Africa');
 ContName.add(key:95, data:'Asia');
 ContName.add(key:96, data:'Australia/Pacific');
 end;
 set orion.country(keep=ContinentID Country CountryName);
 rc=ContName.find();
run;
```

| Partial orion.country |              |              | HASH Object ContName |                         |  |
|-----------------------|--------------|--------------|----------------------|-------------------------|--|
| Country               | Country Name | Continent ID | KEY:<br>Continent ID | DATA:<br>Continent Name |  |
| AU                    | Australia    | 96           | 91                   | North America           |  |
| CA                    | Canada       | 91           | 93                   | Europe                  |  |
| DE                    | Germany      | 93           | 94                   | Africa                  |  |
|                       |              |              | 95                   | Asia                    |  |
|                       |              |              | 96                   | Australia/<br>Pacific   |  |

**PDV**

| ContinentName | Country | CountryName | ContinentID | rc     | _N_ |
|---------------|---------|-------------|-------------|--------|-----|
| BZ            | Brazil  |             | 97          | 160038 | 8   |

56 p306d02

## 6.05 Multiple Choice Poll

What would be the value of **ContinentName** if the value of **ContinentID** were not found in the hash object (**rc ne 0**)?

- missing
- North America
- Australia/Pacific
- unknown

57

## Testing the Return Code

Use a subsetting IF statement to output the observation only if the FIND method found the key value in the hash object.

```
rc=ContName.find();
if rc=0;
```

|         |         |
|---------|---------|
| rc EQ 0 | success |
| rc NE 0 | failure |

59



## Exercises

---

### Level 1

#### 1. Using the ADD Method to Create a Hash Object with a Single Key

The following table shows the code that Orion Star uses for each type of order and a description of the order:

| Order Code | Sale Type     |
|------------|---------------|
| 1          | Retail Sale   |
| 2          | Catalog Sale  |
| 3          | Internet Sale |

The data set **orion.orders** contains the orders that were placed.

Partial **orion.orders**

| Obs | OrderID    | Type | EmployeeID | CustomerID | Order Date | Delivery Date |
|-----|------------|------|------------|------------|------------|---------------|
| 1   | 1230058123 | 1    | 121039     | 63         | 11JAN2007  | 11JAN2007     |
| 2   | 1230080101 | 2    | 99999999   | 5          | 15JAN2007  | 19JAN2007     |
| 3   | 1230106883 | 2    | 99999999   | 45         | 20JAN2007  | 22JAN2007     |
| 4   | 1230147441 | 1    | 120174     | 41         | 28JAN2007  | 28JAN2007     |
| 5   | 1230315085 | 1    | 120134     | 183        | 27FEB2007  | 27FEB2007     |

- a. Write a DATA step that creates a data set named **orders**.

- Use the ADD method to create a hash table containing the values of **OrderType** as the key values and the corresponding **SaleType** as the data values.
  - Use the FIND method to retrieve the sale type based on the variable **OrderType** in the data set **orion.orders**.
  - Keep only the variables **OrderID**, **OrderType**, and **SaleType**.
- b. Print the first five observations from the **orders** data set.

PROC PRINT Output

| orders Data Set |              |            |            |
|-----------------|--------------|------------|------------|
|                 | SaleType     | OrderID    | Order Type |
|                 | Retail Sale  | 1230058123 | 1          |
|                 | Catalog Sale | 1230080101 | 2          |
|                 | Catalog Sale | 1230106883 | 2          |
|                 | Retail Sale  | 1230147441 | 1          |
|                 | Retail Sale  | 1230315085 | 1          |

## Challenge

### 2. Using the ADD Method with a Composite Key

The following table shows the state code, state name, country code, and country names of the states and countries for the Orion Star employees:

| State | State Name   | Country | Country Name  |
|-------|--------------|---------|---------------|
| FL    | Florida      | US      | United States |
| PA    | Pennsylvania | US      | United States |
| CA    | California   | US      | United States |
|       |              | AU      | Australia     |

- a. Write a DATA step to create a data set named **emps**.
- Use the ADD method to add the composite key values for **State** and **Country** and the data values of **StateName** and **CountryName** from the table.
  - Read from the data set **orion.employeeaddresses**.
  - Use the FIND method to perform the table lookup.
- Hint: Use the online documentation to research arguments for the FIND method.

Hint: The resulting data set, **emps**, should have 424 observations. If you do not have 424 observations in the data set, determine why and fix the problem.

- b. Print the first five observations of the data set **emps**.

PROC PRINT Output

| emps Data Set |          |
|---------------|----------|
|               | Employee |
|               |          |

| StateName    | CountryName   | ID     | Country |
|--------------|---------------|--------|---------|
| Florida      | United States | 121044 | US      |
|              | Australia     | 120145 | AU      |
| Pennsylvania | United States | 120761 | US      |
| California   | United States | 120656 | US      |
| Pennsylvania | United States | 121107 | US      |

## 6.3 Loading a Hash Object from a SAS Data Set

---

### Objectives

- Load a hash object from a SAS data set.
- Use a hash object method to match records.

## Business Scenario

Orion Star needs to match continent codes to the code descriptions. The SAS data set **orion.country** has the **ContinentID** variable that contains the code, but no continent name.

Partial **orion.country**

| Country | CountryName   | Population | ContinentID |
|---------|---------------|------------|-------------|
| AU      | Australia     | 20000000   | 96          |
| CA      | Canada        | .          | 91          |
| DE      | Germany       | 80000000   | 93          |
| IL      | Israel        | 5000000    | 95          |
| TR      | Turkey        | 70000000   | 95          |
| US      | United States | 280000000  | 91          |
| ZA      | South Africa  | 43000000   | 94          |

64

## Business Scenario

The lookup data *is* stored in a SAS data set.

**orion.Continent**

| Continent ID | Continent Name    |
|--------------|-------------------|
| 91           | North America     |
| 93           | Europe            |
| 94           | Africa            |
| 95           | Asia              |
| 96           | Australia/Pacific |

65

...

## Loading a Hash Object from a SAS Data Set

Use the DECLARE statement DATASET argument to load a SAS data set into a hash object.

`orion.Continent`

| Continent ID | Continent Name    |
|--------------|-------------------|
| 91           | North America     |
| 93           | Europe            |
| 94           | Africa            |
| 95           | Asia              |
| 96           | Australia/Pacific |



Hash Object `ContName`

| Key:<br>ContinentID | Data:<br>ContinentName |
|---------------------|------------------------|
| 91                  | North America          |
| 93                  | Europe                 |
| 94                  | Africa                 |
| 95                  | Asia                   |
| 96                  | Australia/Pacific      |

66

## Defining a Hash Object

This example uses the DATASET argument.

```
DECLARE object object-reference(<arg_tag-1:value-1
<,...arg_tag-n: value-n>>);
```

```
declare hash
 ContName (dataset:'orion.continent');
```

67

## 6.06 Quiz

Use the DECLARE statement to create a hash object named **tree** and load it from the data set **work.naturepark**.

```
DECLARE object object-reference(<arg_tag-1:value-1
 <,...arg_tag-n: value-n>>);
```

---

68

## Loading a Hash Object from a SAS Data Set

Instead of using the ADD method to add data to the hash object, load the data directly with the DECLARE statement DATASET argument.

```
data country;
 drop rc;
 length ContinentName $ 30;
 if _n_=1 then do;
 call missing(ContinentName);
 declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 end;
 set orion.country(keep=ContinentID Country
 CountryName);
 rc=ContName.find();
run;
```

70

p306d03

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object  
ContName

| KEY:<br>Continent ID | DATA:<br>ContinentName |
|----------------------|------------------------|
| 91                   | North America          |
| 93                   | Europe                 |
| 94                   | Africa                 |
| 95                   | Asia                   |
| 96                   | Australia/Pacific      |

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName);
rc=ContName.find();
run;

```

PDV

| ContinentName | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ | ... |
|---------------|---------|-------------|-------------|-------|--------|-----|
|               |         |             | .           | .     | 1      | ... |

71

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object  
ContName

| KEY:<br>Continent ID | DATA:<br>ContinentName |
|----------------------|------------------------|
| 91                   | North America          |
| 93                   | Europe                 |
| 94                   | Africa                 |
| 95                   | Asia                   |
| 96                   | Australia/Pacific      |

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName);
rc=ContName.find();
run;

```

PDV

| ContinentName | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ | ... |
|---------------|---------|-------------|-------------|-------|--------|-----|
|               | AU      | Australia   | 96          | .     | 1      | ... |

72

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object ContName

| KEY: Continent ID | DATA: ContinentName |
|-------------------|---------------------|
| 91                | North America       |
| 93                | Europe              |
| 94                | Africa              |
| 95                | Asia                |
| 96                | Australia/Pacific   |

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
call missing(ContinentName);
declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
 ContinentName);
rc=ContName.find();
run;

```

PDV

| ContinentName | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ |
|---------------|---------|-------------|-------------|-------|--------|
|               | AU      | Australia   | 96          | 0     | 1      |

73

...

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object ContName

| KEY: Continent ID | DATA: ContinentName |
|-------------------|---------------------|
| 91                | North America       |
| 93                | Europe              |
| 94                | Africa              |
| 95                | Asia                |
| 96                | Australia/Pacific   |

```

data country;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
call missing(ContinentName);
declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
 ContinentName);
rc=ContName.find();
run;

```

PDV

| ContinentName     | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ |
|-------------------|---------|-------------|-------------|-------|--------|
| Australia/Pacific | AU      | Australia   | 96          | 0     | 1      |

74

...

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object  
ContName

| KEY:<br>Continent ID | DATA:<br>ContinentName |
|----------------------|------------------------|
| 91                   | North America          |
| 93                   | Europe                 |
| 94                   | Africa                 |
| 95                   | Asia                   |
| 96                   | Australia/Pacific      |

```

data country;
drop rc;
length ContinentName $ 17;
if _n_=1 then do;
call missing(ContinentName);
declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
CountryName);
rc=ContName.find(); if rc=0;
run;

```

**Implicit OUTPUT;  
Implicit RETURN;**

PDV

| ContinentName     | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ |
|-------------------|---------|-------------|-------------|-------|--------|
| Australia/Pacific | AU      | Australia   | 96          | 0     | 1      |

75 ...

## Execution

Partial orion.country

| Country | CountryName | Continent ID |
|---------|-------------|--------------|
| AU      | Australia   | 96           |
| CA      | Canada      | 91           |
| DE      | Germany     | 93           |
| IL      | Israel      | 95           |

Hash Object  
ContName

| KEY:<br>Continent ID | DATA:<br>ContinentName |
|----------------------|------------------------|
| 91                   | North America          |
| 93                   | Europe                 |
| 94                   | Africa                 |
| 95                   | Asia                   |
| 96                   | Australia/Pacific      |

```

data country;
drop rc;
length ContinentName $ 17;
if _n_=1 then do;
call missing(ContinentName);
declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
end;
set orion.country(keep=ContinentID Country
CountryName);
rc=ContName.find(); if rc=0;
run;

```

**Continue until EOF**

PDV

| ContinentName     | Country | CountryName | ContinentID | D▶ rc | D▶ _N_ |
|-------------------|---------|-------------|-------------|-------|--------|
| Australia/Pacific | AU      | Australia   | 96          | 0     | 1      |

76 ...

## Results

```
proc print data=country noobs;
 title 'Continent Descriptions';
run;
```

### PROC PRINT Output

| Continent Descriptions |         |               |             |
|------------------------|---------|---------------|-------------|
| ContinentName          | Country | CountryName   | ContinentID |
| Australia/Pacific      | AU      | Australia     | 96          |
| North America          | CA      | Canada        | 91          |
| Europe                 | DE      | Germany       | 93          |
| Asia                   | IL      | Israel        | 95          |
| Asia                   | TR      | Turkey        | 95          |
| North America          | US      | United States | 91          |
| Africa                 | ZA      | South Africa  | 94          |

77

p306d03

## 6.07 Multiple Choice Poll

The program **p306d03** created the variable **rc** and then dropped it. How can you avoid creating the variable so that you do not have to drop it?

- Use a WHERE statement or a WHERE= data set option.
- Use a KEEP= or DROP= data set option in **orion.country**.
- Test the result of the FIND method in the subsetting IF statement.
- Use a KEEP or DROP statement.

78

## Not Creating rc

```
data country;
 if 0 then set orion.continent;
 if _n_=1 then do;
 declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 end;
 set orion.country(keep=ContinentID Country
 CountryName);
 if ContName.find()=0;
run;
```

80

p306d04

## Defining PDV Variables

Instead of the LENGTH statement, you can use a non-executing SET statement.

```
data country;
 if 0 then set orion.continent;
 if _n_=1 then do;
 declare hash
 ContName(dataset:'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 end;
 set orion.country(keep=ContinentID Country
 CountryName);
 if ContName.find()=0;
run;
```

- The CALL MISSING statement is not needed because the SET statement adds **ContinentName** to the PDV and assigns an initial missing value.

81

p306d04

## Using DATA Set Options

SAS DATA set options can limit the amount of data loaded into a hash object.

```
data country2;
 if 0 then set orion.continent;
 if n_=1 then do;
 declare hash ContName(dataset:"orion.continent"
 (where=(ContinentName like 'A%')));
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 end;
 set orion.country(keep=ContinentID Country
 CountryName);
 if ContName.find()=0;
run;
```

82

p306d05

## Business Scenario

The Marketing Department wants to create a new data set that includes one observation per country in **sorted order** by **ContinentID**.

**work.countries**

| Continent ID | Continent Name    | Country | Country Name  | Population  |
|--------------|-------------------|---------|---------------|-------------|
| 91           | North America     | CA      | Canada        | .           |
| 91           | North America     | US      | United States | 280,000,000 |
| 93           | Europe            | DE      | Germany       | 80,000,000  |
| 94           | Africa            | ZA      | South Africa  | 43,000,000  |
| 95           | Asia              | IL      | Israel        | 5,000,000   |
| 95           | Asia              | TR      | Turkey        | 70,000,000  |
| 96           | Australia/Pacific | AU      | Australia     | 20,000,000  |

83

## Business Scenario

The **orion.country** data set contains four of the variables needed to create **work.countries**. It also contains duplicate values for **ContinentID**.

Partial **orion.country**

| Country | Country Name | Population | Continent ID |
|---------|--------------|------------|--------------|
| AU      | Australia    | 20,000,000 | 96           |
| CA      | Canada       | .          | 91           |
| DE      | Germany      | 80,000,000 | 93           |
| IL      | Israel       | 5,000,000  | 95           |
| TR      | Turkey       | 70,000,000 | 95           |

84

## Output Method

The OUTPUT method creates the **countries** data set and includes variables from **orion.country** and the hash object populated from **orion.continent**.

Partial **orion.country**

| Country | Country Name | Population | Continent ID | Continent ID | Continent Name | Country | Country Name  | Population  | Partial work.countries |
|---------|--------------|------------|--------------|--------------|----------------|---------|---------------|-------------|------------------------|
| AU      | Australia    | 20,000,000 | 96           | 96           | North America  | CA      | Canada        | .           |                        |
| CA      | Canada       | .          | 91           | 91           | North America  | US      | United States | 315,400,000 |                        |
| DE      | Germany      | 80,000,000 | 93           | 93           | Europe         | DE      | Germany       | 80,000,000  |                        |
| IL      | Israel       | 5,000,000  | 95           | 95           | Africa         | ZA      | Zambia        | 10,000,000  |                        |
| TR      | Turkey       | 70,000,000 | 95           | 95           | Asia           | IL      | Indonesia     | 250,000,000 | Asia                   |
|         |              |            | 96           | 96           |                |         |               |             | Australia/Pacific      |

85

## Output Method

A return code value of 0 indicates that the OUTPUT method was successful.

```
rc=object.OUTPUT(DATASET:'dataset-1 <(datasetoption)> '
 '<, ... <DATASET: 'dataset-n'>> ('datasetoption<(datasetoption)>');
```

0

Partial work.countries

| Continent ID | Continent Name | Country | Country Name  | Population  |
|--------------|----------------|---------|---------------|-------------|
| 91           | North America  | CA      | Canada        | .           |
| 91           | North America  | US      | United States | 280,000,000 |
| 93           | Europe         | DE      | Germany       | 80,000,000  |
| 94           | Africa         | ZA      | South Africa  | 43,000,000  |
| 95           | Asia           | IL      | Israel        | 5,000,000   |

86

## Output Method

```
data _null_;
 drop rc;
 length ContinentName $ 30;
 if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
 end;
 set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
 rc=ContName.find();
 rc=C.add();
 if eof then rc=C.output(dataset: 'work.countries');
run; p306d06
```

87

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

88

**DATA\_NULL\_**; prevents the creation of an output SAS data set.

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

89

Create hash object **ContName** for **ContinentName** lookup.

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');

Create hash object C to hold data to be output to work.countries.
The hash object is ordered in ascending order by the key value.);
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

90

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');

Define the key and data values
for the C hash object.);
 end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

91

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

92

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');

Read data from the orion.country data set and
look up ContinentID and retrieve ContinentName
from the ContName hash object. Name',
 , 'Population');

set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
rc=ContName.find();
rc=C.add();
if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

93

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 Add data to the C hash object.
 rc=C.add();
 if eof then rc=C.output(dataset: 'work.countries');
run; p306d06

```

94

...

## Output Method

```

data _null_;
drop rc;
length ContinentName $ 30;
if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(ordered: 'ascending');
 C.definekey('ContinentID');
 C.definedata('ContinentID','ContinentName',
 'Country', 'CountryName', 'Population');
 C.definedone();
end;
set orion.country(keep=ContinentID Country
 CountryName Population)
 end=end;
 rc=ContName.find();
 rc=C.add();
 if eof then rc=C.output(dataset: 'work.countries'); p306d06

```

95

Output the C hash object data to the **work.countries** data set.

## Output Method

The data set is ordered by **ContinentID**, but there are no observations representing Turkey and the United States.

```
proc print data=work.countries noobs;
 title1 'work.countries without
 Duplicate ContinentID Values';
run;
```

| work.countries without Duplicate ContinentID Values |                   |         |              |            |
|-----------------------------------------------------|-------------------|---------|--------------|------------|
| Continent<br>ID                                     | ContinentName     | Country | CountryName  | Population |
| 91                                                  | North America     | CA      | Canada       | .          |
| 93                                                  | Europe            | DE      | Germany      | 80,000,000 |
| 94                                                  | Africa            | ZA      | South Africa | 43,000,000 |
| 95                                                  | Asia              | IL      | Israel       | 5,000,000  |
| 96                                                  | Australia/Pacific | AU      | Australia    | 20,000,000 |

## MULTIDATA Argument

```
data _null_;
 drop rc;
 length ContinentName $ 30;
 if _n_=1 then do;
 call missing(ContinentName);
 declare hash ContName(dataset: 'orion.continent');
 ContName.definekey('ContinentID');
 ContName.definedata('ContinentName');
 ContName.definedone();
 declare hash C(multidata:'Y', ordered: 'ascending');
 C.definekey('ContinentName');
 C.definedata('CountryName', 'Population');
 C.defineoutput(dataset: 'work.countries');
 end;
 Use the MULTIDATA argument to enable multiple data items for each key.
 set orion.country(keep=ContinentID Country
 CountryName Population)
 end=eof;
 rc=ContName.find();
 rc=C.add();
 if eof then rc=C.output(dataset: 'work.countries');
run;
```

## MULTIDATA Argument

Multiple observations in **work.countries** were retrieved from the same **ContinentID** key value in hash object **C**.

```
proc print data=work.countries noobs;
 title 'work.countries with Duplicate
 ContinentID Values';
run;
```

| work.countries with Duplicate ContinentID Values |                   |               |               |             |            |
|--------------------------------------------------|-------------------|---------------|---------------|-------------|------------|
| Continent                                        | ID                | ContinentName | Country       | CountryName | Population |
| 91                                               | North America     | CA            | Canada        | .           |            |
| 91                                               | North America     | US            | United States | 280,000,000 |            |
| 93                                               | Europe            | DE            | Germany       | 80,000,000  |            |
| 94                                               | Africa            | ZA            | South Africa  | 43,000,000  |            |
| 95                                               | Asia              | IL            | Israel        | 5,000,000   |            |
| 95                                               | Asia              | TR            | Turkey        | 70,000,000  |            |
| 96                                               | Australia/Pacific | AU            | Australia     | 20,000,000  |            |

98

p306d07

## 6.08 Short Answer Poll

What is the advantage of testing the value of EOF before you execute the OUTPUT method?

```
set orion.country(keep=ContinentID Country
 CountryName
 Population) end=eof;
 rc=ContName.find();
 rc=C.add();
if eof then
 rc=C.output(dataset: 'work.countries');
```

99

### Advantages and Disadvantages of Hash Objects

| Advantages                                    | Disadvantages                         |
|-----------------------------------------------|---------------------------------------|
| Use of character and numeric keys             | Unique keys required prior to SAS 9.2 |
| Use of composite keys                         | DATA step only                        |
| Faster lookup than formats or merges or joins | Memory requirements                   |
| Ability to be loaded from a SAS data set      |                                       |
| Fine level of control (flexibility)           |                                       |
| Ability to do chained lookups                 |                                       |



### Exercises

---

#### Level 1

##### 3. Loading the Hash Object from a SAS Data Set

The data set **orion.customerType** contains the **CustomerTypeID** variable and the **CustomerType** variable, which is a description of the customer type.

##### **orion.customerType**

| orion.customerType |                |                                         |                 |                            |
|--------------------|----------------|-----------------------------------------|-----------------|----------------------------|
| Obs                | CustomerTypeID | CustomerType                            | CustomerGroupID | CustomerGroup              |
| 1                  | 1010           | Orion Club members inactive             | 10              | Orion Club members         |
| 2                  | 1020           | Orion Club members low activity         | 10              | Orion Club members         |
| 3                  | 1030           | Orion Club members medium activity      | 10              | Orion Club members         |
| 4                  | 1040           | Orion Club members high activity        | 10              | Orion Club members         |
| 5                  | 2010           | Orion Club Gold members low activity    | 20              | Orion Club Gold members    |
| 6                  | 2020           | Orion Club Gold members medium activity | 20              | Orion Club Gold members    |
| 7                  | 2030           | Orion Club Gold members high activity   | 20              | Orion Club Gold members    |
| 8                  | 3010           | Internet/Catalog Customers              | 30              | Internet/Catalog Customers |

The data set **orion.customer** contains the **CustomerID** variable and the **CustomerTypeID** variable.

##### Partial **orion.customer**

| Partial orion.customer |            |        |
|------------------------|------------|--------|
| Obs                    | CustomerID | TypeID |
| 1                      | 4          | 1020   |
| 2                      | 5          | 2020   |
| 3                      | 9          | 2020   |
| 4                      | 10         | 1040   |
| 5                      | 11         | 1040   |

- a. Write a DATA step to create a data set named **customers** that reads the variables **CustomerID** and **CustomerTypeID** from the data set **orion.customer**.
- Create a hash object and load it with the data from **orion.customertype**. The key should be the variable **CustomerTypeID**, and the data item should be the variable **CustomerType**.
  - Use the hash object to look up the **CustomerType** description.
- b. Print the first five observations of the **customers** data set.

PROC PRINT Output

| Partial customers Data Set              |            |        |
|-----------------------------------------|------------|--------|
| CustomerType                            | CustomerID | TypeID |
| Orion Club members low activity         | 4          | 1020   |
| Orion Club Gold members medium activity | 5          | 2020   |
| Orion Club Gold members medium activity | 9          | 2020   |
| Orion Club members high activity        | 10         | 1040   |
| Orion Club members high activity        | 11         | 1040   |

## Level 2

### 4. Loading Multiple Hash Objects from SAS Data Sets

The data set **orion.productlist** contains **ProductID** and **ProductName** for the products sold.

Partial **orion.productlist**

| Partial orion.productlist |              |                      |
|---------------------------|--------------|----------------------|
| Obs                       | ProductID    | ProductName          |
| 1                         | 210000000000 | Children             |
| 2                         | 210100000000 | Children Outdoors    |
| 3                         | 210100100000 | Outdoor things, Kids |
| 4                         | 210200000000 | Children Sports      |
| 5                         | 210200100000 | A-Team, Kids         |

The data set **orion.customerdim** contains **CustomerID**, **CustomerCountry**, and **CustomerName** for customers who made purchases.

Partial **orion.customerdim**

| Partial orion.customerdim |            |                     |                   |
|---------------------------|------------|---------------------|-------------------|
| Obs                       | CustomerID | Customer<br>Country | CustomerName      |
| 1                         | 4          | US                  | James Kvarniq     |
| 2                         | 5          | US                  | Sandrina Stephano |
| 3                         | 9          | DE                  | Cornelia Krahl    |
| 4                         | 10         | US                  | Karen Ballinger   |
| 5                         | 11         | DE                  | Elke Wallstab     |

The data set **orion.orderfact** contains **CustomerID** and information about the orders.

### Partial **orion.orderfact**

| Partial orion.orderfact |            |            |              |          |                   |
|-------------------------|------------|------------|--------------|----------|-------------------|
| Obs                     | CustomerID | Order Date | ProductID    | Quantity | TotalRetail Price |
| 1                       | 63         | 11JAN2007  | 220101300017 | 1        | \$16.50           |
| 2                       | 5          | 15JAN2007  | 230100500026 | 1        | \$247.50          |
| 3                       | 45         | 20JAN2007  | 240600100080 | 1        | \$28.30           |
| 4                       | 41         | 28JAN2007  | 240600100010 | 2        | \$32.00           |
| 5                       | 183        | 27FEB2007  | 240200200039 | 3        | \$63.60           |

- a. Create a data set named **billing** that reads **CustomerID**, **OrderDate**, **ProductID**, **Quantity**, and **TotalRetailPrice** from **orion.orderfact**.
  - Create a hash object from **orion.productlist** with the key **ProductID** and the data **ProductName**.
  - Create a hash object from **orion.customerdim** with the key **CustomerID** and the data **CustomerCountry** and **CustomerName**.
  - Use the two hash objects to look up **CustomerName**, **CountryName**, and **ProductName**.
- b. Sort the **billing** data set by ascending values of **CustomerID** and **ProductID**.
- c. Print the first five observations in **billing**.

### PROC PRINT Output

| Billing Information<br>Using a HASH Data Step Object |               |                   |                                     |
|------------------------------------------------------|---------------|-------------------|-------------------------------------|
| CustomerID                                           | CustomerName  | ProductID         | ProductName                         |
| 4                                                    | James Kvarniq | 220101400145      | Essence.baseball Cap                |
| 4                                                    | James Kvarniq | 230100100053      | Monster Men's Pants with Zipper     |
| 4                                                    | James Kvarniq | 240500100017      | A-team Sweat Round Neck, Small Logo |
| 4                                                    | James Kvarniq | 240500100029      | Men's Sweatshirt w/Hood Big Logo    |
| 4                                                    | James Kvarniq | 240500200083      | Force Technical Jacket w/Coolmax    |
| OrderDate                                            | Quantity      | TotalRetail Price |                                     |
| 16APR2008                                            | 1             | \$16.70           |                                     |
| 18DEC2008                                            | 2             | \$92.60           |                                     |
| 08APR2008                                            | 4             | \$214.00          |                                     |
| 08APR2008                                            | 1             | \$58.90           |                                     |
| 19AUG2008                                            | 3             | \$201.90          |                                     |

## Challenge

### 5. Loading the Hash Object from a SAS Data Set and Retrieving Multiple Values

The data set **orion.staff** contains the employee ID and the manager ID for that employee.

**Partial orion.staff**

| Partial orion.staff |            |            |  |           |                        |           |
|---------------------|------------|------------|--|-----------|------------------------|-----------|
| Obs                 | EmployeeID | Start Date |  | EndDate   | JobTitle               | Salary    |
|                     |            | Date       |  |           |                        |           |
| 1                   | 120101     | 01JUL2007  |  | 31DEC9999 | Director               | \$163,040 |
| 2                   | 120102     | 01JUN1993  |  | 31DEC9999 | Sales Manager          | \$108,255 |
| 3                   | 120103     | 01JAN1978  |  | 31DEC9999 | Sales Manager          | \$87,975  |
| 4                   | 120104     | 01JAN1985  |  | 31DEC9999 | Administration Manager | \$46,230  |
| 5                   | 120105     | 01MAY2003  |  | 31DEC9999 | Secretary I            | \$27,110  |

| Obs | Gender | Birth Date |  | EmpHire Date | EmpTerm |           |
|-----|--------|------------|--|--------------|---------|-----------|
|     |        | Date       |  |              | Date    | ManagerID |
| 1   | M      | 18AUG1980  |  | 01JUL2007    | .       | 120261    |
| 2   | M      | 11AUG1973  |  | 01JUN1993    | .       | 120101    |
| 3   | M      | 22JAN1953  |  | 01JAN1978    | .       | 120101    |
| 4   | F      | 11MAY1958  |  | 01JAN1985    | .       | 120101    |
| 5   | F      | 21DEC1978  |  | 01MAY2003    | .       | 120101    |

The data set **orion.employeeaddresses** contains the names of all employees.

**Partial orion.employeeaddresses**

| Partial orion.employeeaddresses |          |                      |  |            |  |        |
|---------------------------------|----------|----------------------|--|------------|--|--------|
| Obs                             | Employee |                      |  | Street     |  |        |
|                                 | ID       | EmployeeName         |  | StreetID   |  | Number |
| 1                               | 121044   | Abbott, Ray          |  | 9260116912 |  | 2267   |
| 2                               | 120145   | Aisbitt, Sandy       |  | 1600101803 |  | 30     |
| 3                               | 120761   | Akinfolarin, Tameaka |  | 9260121030 |  | 5      |
| 4                               | 120656   | Amos, Salley         |  | 9260123736 |  | 3524   |
| 5                               | 121107   | Anger, Rose          |  | 9260120989 |  | 744    |

| Obs | StreetName      | Postal       |       |       |         |
|-----|-----------------|--------------|-------|-------|---------|
|     |                 | City         | State | Code  | Country |
| 1   | Edwards Mill Rd | Miami-Dade   | FL    | 33135 | US      |
| 2   | Bingera Street  | Melbourne    |       | 2001  | AU      |
| 3   | Donnybrook Rd   | Philadelphia | PA    | 19145 | US      |
| 4   | Calico Ct       | San Diego    | CA    | 92116 | US      |
| 5   | Chapwith Rd     | Philadelphia | PA    | 19142 | US      |

The data set **orion.employeepayroll** has the employee ID and salary for each employee.

**Partial Listing of orion.employeepayroll**

| Partial orion.employeepayroll |            |          |        |           |           |          |        |            |
|-------------------------------|------------|----------|--------|-----------|-----------|----------|--------|------------|
| Obs                           | EmployeeID | Employee |        | Employee  | Employee  | Marital  |        |            |
|                               |            | Gender   | Salary | BirthDate | HireDate  | TermDate | Status | Dependents |
| 1                             | 120101     | M        | 163040 | 18AUG1980 | 01JUL2007 | .        | S      | 0          |
| 2                             | 120102     | M        | 108255 | 11AUG1973 | 01JUN1993 | .        | O      | 2          |
| 3                             | 120103     | M        | 87975  | 22JAN1953 | 01JAN1978 | .        | M      | 1          |
| 4                             | 120104     | F        | 46230  | 11MAY1958 | 01JAN1985 | .        | M      | 1          |
| 5                             | 120105     | F        | 27110  | 21DEC1978 | 01MAY2003 | .        | S      | 0          |

- a. Write a DATA step to create a data set named **manager** that reads the **EmployeeID** and **Salary** variables from **orion.employeepayroll**.
- Create hash objects from the data sets **orion.employeeaddresses** and **orion.staff**.
  - Use the hash object from **orion.staff** to return the **ManagerID** value for each **EmployeeID** in **orion.employeepayroll**.
  - Use the hash object from **orion.employeeaddresses** to retrieve the names for both employees and the manager for the employees.

Hint: Use the online documentation to research the arguments for the FIND method.

- b. Print the first five observations of the **manager** data set.

PROC PRINT Output

| manager Data Set  |                  |            |        |            |
|-------------------|------------------|------------|--------|------------|
| EmpName           | ManagerName      | EmployeeID | Salary | Manager ID |
| Lu, Patrick       | Highpoint, Harry | 120101     | 163040 | 120261     |
| Zhou, Tom         | Lu, Patrick      | 120102     | 108255 | 120101     |
| Dawes, Wilson     | Lu, Patrick      | 120103     | 87975  | 120101     |
| Billington, Karen | Lu, Patrick      | 120104     | 46230  | 120101     |
| Povey, Liz        | Lu, Patrick      | 120105     | 27110  | 120101     |

## 6.4 DATA Step Hiter Object

### Objectives

- Define a hiter object.
- Investigate the methods for manipulating a hiter object.
- Write a DATA step using a hiter object.

## Hash Object

The hash object stores and retrieves data based on lookup key values.

Hash Object

|  Key: | Data: |
|----------------------------------------------------------------------------------------|-------|
| 3                                                                                      | X     |
| 1                                                                                      | Y     |
| 2                                                                                      | Z     |

105

## Hash Iterator Object

Think of the hash iterator, or hiter, object as an ordered view of a hash object.

Hash Object

|  Key | Hiter Object View |       |
|-----------------------------------------------------------------------------------------|-------------------|-------|
|                                                                                         | Key:              | Data: |
|                                                                                         | 1                 | Y     |
|                                                                                         | 2                 | Z     |
|                                                                                         | 3                 | X     |

- ✍ Declare the hash object before defining the hash iterator object.

106

## Business Scenario

The Marketing Department wants to identify which two customers ordered the most and least expensive items.

Partial orion.orderfact

| Customer ID | ProductID    | Quantity | Total Retail Price | CostPrice PerUnit | Discount |
|-------------|--------------|----------|--------------------|-------------------|----------|
| 63          | 220101300017 | 1        | \$16.50            | \$7.45            | .        |
| 5           | 230100500026 | 1        | \$247.50           | \$109.55          | .        |
| 45          | 240600100080 | 1        | \$28.30            | \$8.55            | .        |
| 41          | 240600100010 | 2        | \$32.00            | \$6.50            | .        |
| 183         | 240200200039 | 3        | \$63.60            | \$8.80            | .        |



107

## Hash Iterator Object Solution

The hash iterator object can retrieve hash object data in either ascending or descending key order to efficiently find these four customers.

Partial Hash Object

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID |
|----------------------------------|------------------------|
| 16.50                            | 63                     |
| 247.50                           | 5                      |
| 28.30                            | 45                     |
| 32.00                            | 41                     |
| .                                | .                      |
| .                                | .                      |
| 95.10                            | 10                     |
| 48.20                            | 10                     |
| 75.20                            | 89                     |
| 33.80                            | 5                      |

Partial Hiter View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID |
|----------------------------------|------------------------|
| 1937.20                          | 70100                  |
| 1796.00                          | 79                     |
| 1687.50                          | 16                     |
| 1561.80                          | 183                    |
| .                                | .                      |
| .                                | .                      |
| 3.20                             | 69                     |
| 3.00                             | 5                      |
| 2.70                             | 11171                  |
| 2.60                             | 79                     |

108

## Business Scenario

```

data top bottom;
 drop i;
 if N=1 then do;
 if 0 then set orion.orderfact(keep=CustomerID ProductID
 TotalRetailPrice);
 declare hash customer(dataset:'orion.orderfact',
 ordered:'descending');
 customer.definekey('TotalRetailPrice', 'CustomerID');
 customer.definedata('TotalRetailPrice', 'CustomerID',
 'ProductID');
 customer.definedone();
 declare hiter C('customer');
 end;
 C.first();
 do i=1 to 2;
 output top;
 C.next();
 end;
 C.last();
 do i=1 to 2;
 output bottom;
 C.prev();
 end;
 stop;
run;

```

109

p306d08

## Declaring a Hiter Object

The hash object loads the data in the order in which it occurs in the data set. The ORDERED argument specifies retrieving from the hash object by descending order of key values.

```

declare hash Customer(dataset:'orion.orderfact',
 ordered:'descending');
customer.definekey('TotalRetailPrice', 'CustomerID');
customer.definedata('TotalRetailPrice', 'CustomerID'
 'ProductID');
customer.definedone();
declare hiter C('customer');

```

110

p306d08

## Declaring a Hiter Object

The DEFINEKEY, DEFINEDATA, and DEFINEDONE methods execute.

```
declare hash Customer
 ordered:'descending');
customer.definekey('TotalRetailPrice', 'CustomerID');
customer.definedata('TotalRetailPrice', 'CustomerID'
 'ProductID');
customer.definedone();
declare hiter C('customer');
```

**The key value is not returned to the PDV.**

111

...

## Declaring a Hiter Object

The second DECLARE statement defines hash iterator object **C**.

```
declare hash Customer(dataset:'orion.orderfact',
 ordered:'descending');
customer.definekey('TotalRetailPrice', 'CustomerID');
customer.definedata('TotalRetailPrice', 'CustomerID'
 'ProductID');
customer.definedone();
declare hiter C('customer');
```

**DECLARE HITER iterator-name('hash-name');**

112

## Retrieving Hash Object Data with the Hash Iterator Object

Four hash iterator object methods return values based on the position of the rows in the hash object.

FIRST

NEXT

LAST

PREV

Hash Object

| Key: A | Data: B |
|--------|---------|
| 3      | X       |
| 1      | Y       |
| 2      | Z       |

Hiter Object View

| Key: A | Data: B |
|--------|---------|
| 3      | X       |
| 2      | Z       |
| 1      | Y       |

113

## Retrieving Hash Object Data with the Hash Iterator Object

The *FIRST* method returns the first data value in the underlying hash object.

```
declare hiter C('customer');
<more SAS code>
C.first();
```

Hash Object

| Key | Hiter Object View |         |
|-----|-------------------|---------|
|     | Key: A            | Data: B |
|     | 3                 | X       |
|     | 2                 | Z       |
|     | 1                 | Y       |

PDV

| A | B |
|---|---|
|   | X |

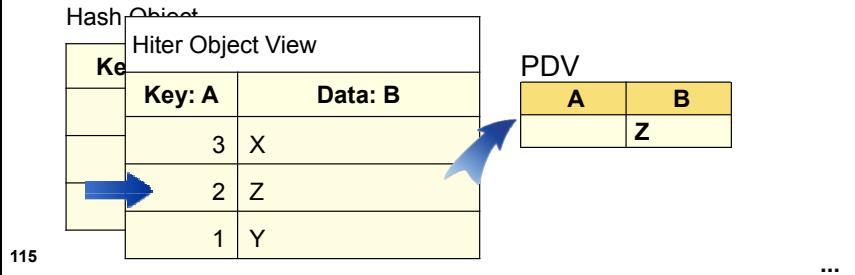
114

...

## Retrieving Hash Object Data with the Hash Iterator Object

Use the *NEXT* method to iterate through the hash object and return the data items in key order.

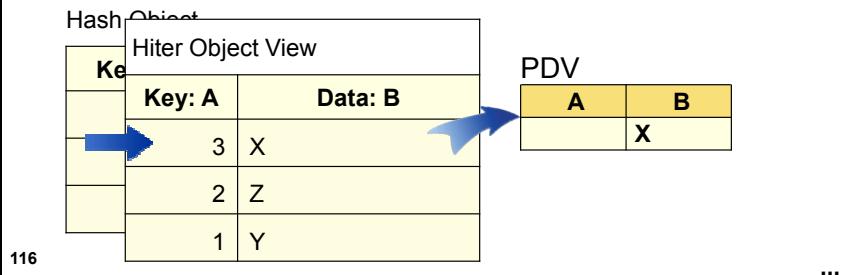
```
C.first();
<more SAS code>
C.next();
```



## Retrieving Hash Object Data with the Hash Iterator Object

Using the *NEXT* method without the *FIRST* method starts at the first item in the hash object.

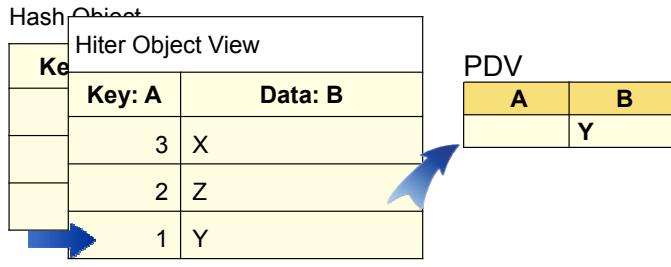
```
C.next();
```



## Retrieving Hash Object Data with the Hash Iterator Object

The *LAST* method returns the last data value in the underlying hash object.

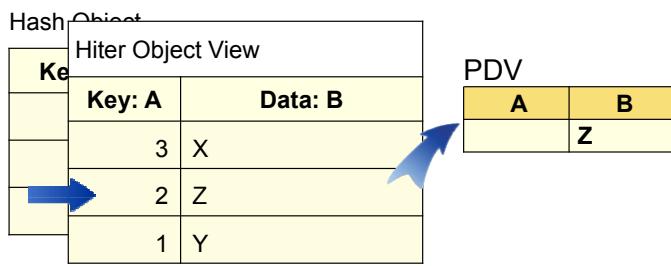
```
declare hiter C('customer');
<more SAS code>
C.last();
```



## Retrieving Hash Object Data with the Hash Iterator Object

Use the *PREV* method to return the data items in reverse key order.

```
C.prev();
```



## 6.09 Multiple Answer Poll

Which of the following are true regarding the hash iterator (hiter) object?

- a. You must define the hash object, which is referenced by the hiter object before the DECLARE statement for the hiter object.
- b. Ascending is the only valid method of moving through the hiter object.
- c. You can move through the hiter object using either the key or data items.
- d. The hiter object is an ordered view of the hash object.

119

## Declaring the Hash and Hiter Objects

```

data top bottom;
drop i;
if N=1 then do;
 if '0' then set orion.orderfact(keep=CustomerID ProductID
 TotalRetailPrice);
 declare hash customer(dataset:'orion.orderfact',
 ordered:'descending');
 customer.definekey('TotalRetailPrice', 'CustomerID');
 customer.definedata('TotalRetailPrice', 'CustomerID',
 'ProductID');
 customer.definedone();
 declare hiter C('customer');
end;
C. First();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;

```

121

p306d08

## Execution

Partial Hash Object **customer**

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 16.50                            | 63                     | 16.50                             | 63                      | 220101300017           |
| 247.50                           | 5                      | 247.50                            | 5                       | 230100500026           |
| 28.30                            | 45                     | 28.30                             | 45                      | 240600100080           |
| 32.00                            | 41                     | 32.00                             | 41                      | 240600100010           |
| .                                | .                      | .                                 | .                       | .                      |
| 95.10                            | 10                     | 95.10                             | 10                      | 240500200016           |
| 48.20                            | 10                     | 48.20                             | 10                      | 240500200122           |
| 75.20                            | 89                     | 75.20                             | 89                      | 240700200018           |
| 33.80                            | 5                      | 33.80                             | 5                       | 220101400130           |

```

data top bottom;
drop i;
if _N_=1 then do;
 if 0 then set orion.orderfact
 (keep=CustomerID ProductID
 TotalRetailPrice);
 declare hash customer
 (dataset:'orion.orderfact',
 ordered:'descending');
 customer.definekey('TotalRetailPrice',
 'CustomerID');
 customer.definedata('TotalRetailPrice',
 'CustomerID','ProductID');
 customer.definedone();
 declare hiter C('customer');
end;

```

Notice the unordered hash object.

PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D | i | D | _N_ |
|----------------|---------------|----------------------|---|---|---|-----|
| .              | .             | .                    | . | . | . | 1   |

122 ...

## Execution

Partial Hiter C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```

data top bottom;
drop i;
if _N_=1 then do;
 if 0 then set orion.orderfact
 (keep=CustomerID ProductID
 TotalRetailPrice);
 declare hash customer
 (dataset:'orion.orderfact',
 ordered:'descending');
 customer.definekey('TotalRetailPrice',
 'CustomerID');
 customer.definedata('TotalRetailPrice',
 'CustomerID','ProductID');
 customer.definedone();
 declare hiter C('customer');
end;

```

This is the hiter object's descending ordered view of the hash object.

PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D | i | D | _N_ |
|----------------|---------------|----------------------|---|---|---|-----|
| .              | .             | .                    | . | . | . | 1   |

123 ...

## Creating the Data Sets

```

data top bottom;
drop i;
if _N_=1 then do;
 if 0 then set orion.orderfact(keep=CustomerID ProductID
 TotalRetailPrice);
 declare hash customer(dataset:'orion.orderfact',
 ordered:'descending');
 customer.definekey('TotalRetailPrice', 'CustomerID');
 customer.definedata('TotalRetailPrice', 'CustomerID',
 'ProductID');
 customer.definedone();
 declare hiter C('customer');
end;
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;

```

124

p306d08

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500              |

```

C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;

```

Output current observation  
to work.top.

### PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | ► i | ► _N_ |
|----------------|---------------|----------------------|-----|-------|
| 70100          | 240200100173  | 1937.20              | 1   | 1     |

128

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 79             | 240200100076  | 1796.00              | 1      | 1        |

129

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 79             | 240200100076  | 1796.00              | 2      | 1        |

130

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

Output current observation  
to work.top.

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D <small>&gt;</small> | i | D <small>&gt;</small> | _N_ |
|----------------|---------------|----------------------|-----------------------|---|-----------------------|-----|
| 79             | 240200100076  | 1796.00              |                       | 2 |                       | 1   |

133

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D <small>&gt;</small> | i | D <small>&gt;</small> | _N_ |
|----------------|---------------|----------------------|-----------------------|---|-----------------------|-----|
| 16             | 230100700009  | 1687.50              |                       | 2 |                       | 1   |

134

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| <b>1687.50</b>                   | <b>16</b>              | <b>1687.50</b>                    | <b>16</b>               | <b>230100700009</b>    |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

PDV

| Customer<br>ID | Product<br>ID       | Total<br>RetailPrice | D | i        | D | _N_      |
|----------------|---------------------|----------------------|---|----------|---|----------|
| <b>16</b>      | <b>230100700009</b> | <b>1687.50</b>       |   | <b>3</b> |   | <b>1</b> |

135 ...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| <b>1687.50</b>                   | <b>16</b>              | <b>1687.50</b>                    | <b>16</b>               | <b>230100700009</b>    |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

Exit the DO loop.

PDV

| Customer<br>ID | Product<br>ID       | Total<br>RetailPrice | D | i        | D | _N_      |
|----------------|---------------------|----------------------|---|----------|---|----------|
| <b>16</b>      | <b>230100700009</b> | <b>1687.50</b>       |   | <b>3</b> |   | <b>1</b> |

136 ...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

Output current observation  
to work.bottom.

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 79             | 230100500045  | 2.60                 | 1      | 1        |

140

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
output top;
C.next();
end;
C.last();
do i=1 to 2;
output bottom;
C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 11171          | 240200100021  | 2.70                 | 1      | 1        |

141

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| <b>2.70</b>                      | <b>11171</b>           | <b>2.70</b>                       | <b>11171</b>            | <b>240200100021</b>    |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D | i | D | _N_ |
|----------------|---------------|----------------------|---|---|---|-----|
| 11171          | 240200100021  | 2.70                 |   | 2 |   | 1   |

142

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| <b>2.70</b>                      | <b>11171</b>           | <b>2.70</b>                       | <b>11171</b>            | <b>240200100021</b>    |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
```

Output current observation  
to work.bottom.

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D | i | D | _N_ |
|----------------|---------------|----------------------|---|---|---|-----|
| 11171          | 240200100021  | 2.70                 |   | 2 |   | 1   |

145

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 5              | 240100100433  | 3.00                 | 2      | 1        |

146

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```
C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;
```

## PDV

| Customer<br>ID | Product<br>ID | Total<br>RetailPrice | D<br>i | D<br>_N_ |
|----------------|---------------|----------------------|--------|----------|
| 5              | 240100100433  | 3.00                 | 3      | 1        |

147

...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.00                             | 5                      | 3.00                              | 5                       | 240100100433           |
| 2.70                             | 11171                  | 2.70                              | 11171                   | 240200100021           |
| 2.60                             | 79                     | 2.60                              | 79                      | 230100500045           |

```

C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;

```

Exit the DO loop.

PDV

| Customer ID | Product ID   | Total RetailPrice | D | i | D | _N_ |
|-------------|--------------|-------------------|---|---|---|-----|
| 5           | 240100100433 | 3.00              |   | 3 |   | 1   |

148 ...

## Execution

Partial Hiter Object C View

| KEY:<br>Total<br>Retail<br>Price | KEY:<br>Customer<br>ID | DATA:<br>Total<br>Retail<br>Price | DATA:<br>Customer<br>ID | DATA:<br>Product<br>ID |
|----------------------------------|------------------------|-----------------------------------|-------------------------|------------------------|
| 1937.20                          | 70100                  | 1937.20                           | 70100                   | 240200100173           |
| 1796.00                          | 79                     | 1796.00                           | 79                      | 240200100076           |
| 1687.50                          | 16                     | 1687.50                           | 16                      | 230100700009           |
| 1561.80                          | 183                    | 1561.80                           | 183                     | 240300300090           |
| .                                | .                      | .                                 | .                       | .                      |
| .                                | .                      | .                                 | .                       | .                      |
| 3.20                             | 69                     | 3.20                              | 69                      | 230100500004           |
| 3.0                              |                        |                                   |                         | 100433                 |
| 2.7                              |                        |                                   |                         | 100021                 |
| 2.6                              |                        |                                   |                         | 600045                 |

The STOP statement prevents the following note in the log:  
NOTE: DATA step stopped due to looping.

```

C.first();
do i=1 to 2;
 output top;
 C.next();
end;
C.last();
do i=1 to 2;
 output bottom;
 C.prev();
end;
stop;
run;

```

PDV

| Customer ID | Product ID   | Total RetailPrice | D | i | D | _N_ |
|-------------|--------------|-------------------|---|---|---|-----|
| 5           | 240100100433 | 3.00              |   | 3 |   | 1   |

149 ...

## PROC PRINT Output

```
proc print data=top;
 title 'Top 2 Big Spenders';
run;

proc print data=bottom;
 title 'Bottom 2 Frugal Spenders';
run;
```

| Top 2 Big Spenders       |            |              |                   |
|--------------------------|------------|--------------|-------------------|
| Obs                      | CustomerID | ProductID    | TotalRetail Price |
| 1                        | 70100      | 240200100173 | \$1,937.20        |
| 2                        | 79         | 240200100076 | \$1,796.00        |
| Bottom 2 Frugal Spenders |            |              |                   |
| Obs                      | CustomerID | ProductID    | TotalRetail Price |
| 1                        | 79         | 230100500045 | \$2.60            |
| 2                        | 11171      | 240200100021 | \$2.70            |

150

p306d08

## 6.10 Multiple Answer Poll

Which two of the following pairs of methods are used to retrieve data starting from the top and bottom of a hiter object?

- a. FIRST( ) and PREV( )
- b. FIRST( ) and NEXT( )
- c. LAST( ) and PREV( )
- d. LAST( ) and NEXT( )

151



## Exercises

---

### Level 1

#### 6. Using a Hiter Object

- a. Use the data set **orion.shoesales** to create two data sets named **expensive** and **leastexpensive**. The original data set contains the variables **ProductID**, **ProductName**, and **TotalRetailPrice**. The data set **expensive** should contain the five most expensive pairs of shoes and the data set **leastexpensive** should contain the five least expensive pairs of shoes.

Use the starter program **p306e06**.

#### Partial **orion.shoesales**

| Partial orion.shoesales Data Set |                                             |                   |
|----------------------------------|---------------------------------------------|-------------------|
| ProductID                        | ProductName                                 | TotalRetail Price |
| 220200200024                     | Pro Fit Gel Gt 2030 Women's Running Shoes   | \$178.50          |
| 220200100092                     | Big Guy Men's Air Terra Sebec Shoes         | \$83.00           |
| 240200100043                     | Bretagne Performance Tg Men's Golf Shoes L. | \$282.40          |
| 220100700024                     | Armadillo Road Dmx Women's Running Shoes    | \$99.70           |
| 220200300157                     | Hardcore Men's Street Shoes Large           | \$220.20          |

- b. Print the **expensive** and **leastexpensive** data sets.

#### **work.expensive**

| The Five Most Expensive Shoes |                                          |                   |
|-------------------------------|------------------------------------------|-------------------|
| ProductID                     | ProductName                              | TotalRetail Price |
| 240200100051                  | Bretagne Stabilites 2000 Goretex Shoes   | \$420.90          |
| 220200300129                  | Torino Men's Leather Adventure Shoes     | \$406.00          |
| 240200100227                  | Rubby Women's Golf Shoes w/Gore-Tex      | \$323.80          |
| 220100700024                  | Armadillo Road Dmx Women's Running Shoes | \$313.80          |
| 240200100225                  | Rubby Men's Golf Shoes w/Goretex         | \$306.20          |

#### **work.leastexpensive**

| The Five Least Expensive Shoes |                                       |                   |
|--------------------------------|---------------------------------------|-------------------|
| ProductID                      | ProductName                           | TotalRetail Price |
| 240100100433                   | Shoelace White 150 Cm                 | \$3.00            |
| 240100100434                   | Shoeshine Black                       | \$16.40           |
| 210200400020                   | Kids Baby Edge Max Shoes              | \$38.00           |
| 210200400070                   | Tony's Children's Deschutz (Bg) Shoes | \$41.60           |
| 220200100137                   | Big Guy Men's Multicourt Ii Shoes     | \$50.30           |

## Level 2

### 7. Using a Hiter Object

- a. Use the data set **orion.shoesales** to create a data set named **shoesales** that contains the five most expensive pairs of shoes and the five least expensive pairs of shoes. The original data set contains the variables **ProductID**, **ProductName**, and **TotalRetailPrice**. The data set should contain a new variable named **Rank** that has the values of *Top 1* to *Top 5* for the five most expensive and *Bottom 1* to *Bottom 5* for the five least expensive pairs of shoes.

**Partial orion.shoesales**

| Partial orion.shoesales Data Set |                                             |                   |
|----------------------------------|---------------------------------------------|-------------------|
| ProductID                        | ProductName                                 | TotalRetail Price |
| 220200200024                     | Pro Fit Gel Gt 2030 Women's Running Shoes   | \$178.50          |
| 220200100092                     | Big Guy Men's Air Terra Sebec Shoes         | \$83.00           |
| 240200100043                     | Bretagne Performance Tg Men's Golf Shoes L. | \$282.40          |
| 220100700024                     | Armadillo Road Dmx Women's Running Shoes    | \$99.70           |
| 220200300157                     | Hardcore Men's Street Shoes Large           | \$220.20          |

- b. Print the **shoesales** data set.

**work.shoesales**

| work.shoesales Data Set |              |                                          |                   |
|-------------------------|--------------|------------------------------------------|-------------------|
| Rank                    | ProductID    | ProductName                              | TotalRetail Price |
| Top 1                   | 240200100051 | Bretagne Stabilites 2000 Goretex Shoes   | \$420.90          |
| Top 2                   | 220200300129 | Torino Men's Leather Adventure Shoes     | \$406.00          |
| Top 3                   | 240200100227 | Rubby Women's Golf Shoes w/Gore-Tex      | \$323.80          |
| Top 4                   | 220100700024 | Armadillo Road Dmx Women's Running Shoes | \$313.80          |
| Top 5                   | 240200100225 | Rubby Men's Golf Shoes w/Goretex         | \$306.20          |
| Bottom 1                | 240100100433 | Shoelace White 150 Cm                    | \$3.00            |
| Bottom 2                | 240100100434 | Shoeshine Black                          | \$16.40           |
| Bottom 3                | 210200400020 | Kids Baby Edge Max Shoes                 | \$38.00           |
| Bottom 4                | 210200400070 | Tony's Children's Deschutz (Bg) Shoes    | \$41.60           |
| Bottom 5                | 220200100137 | Big Guy Men's Multicourt Ii Shoes        | \$50.30           |

**Challenge****8. Using a Hiter Object**

- a. Use a hiter object to create a data set named **different** that contains unique values of **CustomerID** and **OrderType** from the data set named **orion.orderfact**. There should be 100 observations in the data set **different**.
- b. Print the first five observations of the data set **different**.

**Partial work.different**

| No Duplicates |            |  |
|---------------|------------|--|
| CustomerID    | Order Type |  |
| 4             | 1          |  |
| 4             | 3          |  |
| 5             | 1          |  |
| 5             | 2          |  |
| 5             | 3          |  |

# 6.5 Solutions

---

## Solutions to Exercises

### 1. Using the ADD Method to Create a Hash Object with a Single Key

- a. Write a DATA step that creates a data set named **orders**.

- 1) Enter the following DATA step in the Editor window:

Partial p306s01

```
data orders;
 length SaleType $40;
 keep OrderID OrderType SaleType;
 if _N_=1 then do;
 declare hash Product();
 Product.definekey('OrderType');
 Product.definedata('SaleType');
 Product.definedone();
 Product.add(key:1, data:'Retail Sale');
 Product.add(key:2, data:'Catalog Sale');
 Product.add(key:3, data:'Internet Sale');
 call missing(SaleType);
 end;
 set orion.orders;
 rc=Product.find();
 if rc=0;
run;
```

- 2) Submit the DATA step.  
3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
128 data orders;
129 length SaleType $40;
130 keep OrderID OrderType SaleType;
131 if _N_=1 then do;
132 declare hash Product();
133 Product.definekey('OrderType');
134 Product.definedata('SaleType');
135 Product.definedone();
136 Product.add(key:1, data:'Retail Sale');
137 Product.add(key:2, data:'Catalog Sale');
138 Product.add(key:3, data:'Internet Sale');
139 call missing(SaleType);
140 end;
141 set orion.orders;
142 rc=Product.find();
143 if rc=0;
144 run;
```

**NOTE:** There were 490 observations read from the data set ORION.ORDERS.  
**NOTE:** The data set WORK.ORDERS has 490 observations and 3 variables.

```
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds
```

- b. Print the first five observations from the **orders** data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p306s01

```
proc print data=orders(obs=5) noobs;
 title 'orders Data Set';
run;
```

- 2) Submit the PROC PRINT step.  
 3) View the output to verify that it matches the expected output.  
 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
168
169 proc print data=orders(obs=5) noobs;
170 title 'orders Data Set';
171 run;
NOTE: There were 5 observations read from the data set WORK.ORDERS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

## 2. Using the ADD Method with a Composite Key

- a. Write a DATA step to create a data set named **emps**.

- 1) Enter the following DATA step in the Editor window.

Partial p306s02

```
data emps;
 length StateName $ 12 CountryName $30;
 keep EmployeeID Country CountryName StateName;
 if _N_=1 then do;
 declare hash C();
 C.definekey('State', 'Country');
 C.definedata('StateName', 'CountryName');
 C.definedone();
 C.add(key:'FL', key:'US', data:'Florida',
 data:'United States');
 C.add(key:'PA', key:'US', data:'Pennsylvania',
 data:'United States');
 C.add(key:'CA', key:'US', data:'California',
 data:'United States');
 C.add(key:' ', key:'AU', data:' ', data:'Australia');
 call missing(StateName, CountryName);
 end;
 set orion.employeeaddresses;
 rc=C.find(key:upcase(State), key:upcase(Country));
```

```
if rc=0;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
633486 data emps;
633487 length StateName $ 12 CountryName $30;
633488 keep EmployeeID Country CountryName StateName;
633489 if _N_=1 then do;
633490 declare hash C();
633491 C.definekey('State', 'Country');
633492 C.definedata('StateName', 'CountryName');
633493 C.definedone();
633494 C.add(key:'FL', key:'US', data:'Florida', data:'United States');
633495 C.add(key:'PA', key:'US', data:'Pennsylvania', data:'United States');
633496 C.add(key:'CA', key:'US', data:'California', data:'United States');
633497 C.add(key: ' ', key:'AU', data: ' ', data:'Australia');
633498 call missing(StateName, CountryName);
633499 end;
633500 set orion.employeeaddresses;
633501 rc=C.find(key:upcase(State), key:upcase(Country));
633502 if rc=0;
633503 run;

NOTE: There were 424 observations read from the data set ORION.EMPLOYEEADDRESSES.
NOTE: The data set WORK.EMPS has 424 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.73 seconds
 cpu time 0.03 seconds
```

- b. Print the first five observations of the data set **emps**.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p306s02

```
proc print data=emps(obs=5) noobs;
 title 'emps Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
633504 proc print data=emps(obs=5) noobs;
633505 title 'emps Data Set';
633506 run;

NOTE: There were 5 observations read from the data set WORK.EMPS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
```

### 3. Loading the Hash Object from a SAS Data Set

- a. Write a DATA step to create a data set named **customers** that reads the variables **CustomerID** and **CustomerTypeID** from the data set **orion.customer**.

- 1) Enter the following DATA step in the Editor window:

Partial p306s03

```
data customers;
 length CustomerType $40;
 keep CustomerID CustomerTypeID CustomerType;
 if _N_=1 then do;
 declare hash Customer(dataset:'orion.customertype');
 Customer.definekey('CustomerTypeID');
 Customer.definedata('CustomerType');
 Customer.definedone();
 call missing(CustomerType);
 end;
 set orion.Customer;
 if Customer.find()=0;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
1 data customers;
2 length CustomerType $40;
3 keep CustomerID CustomerTypeID CustomerType;
4 if _N_=1 then do;
5 declare hash Customer(dataset:'orion.customertype');
6 Customer.definekey('CustomerTypeID');
7 Customer.definedata('CustomerType');
8 Customer.definedone();
9 call missing(CustomerType);
10 end;
11 set orion.Customer;
12 if Customer.find()=0;
13 run;
NOTE: There were 8 observations read from the data set ORION.CUSTOMERTYPE.
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUSTOMERS has 77 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.40 seconds
 cpu time 0.14 seconds
```

- b. Print the first five observations of the **customers** data set.

- 1) Enter the following PROC PRINT step in the editor window:

Partial p306s03

```
proc print data=customers(obs=5) noobs;
 title 'customers Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

15 proc print data=customers(obs=5) noobs;
16 title 'customers Data Set';
17 run;
NOTE: There were 5 observations read from the data set WORK.CUSTOMERS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.06 seconds
 cpu time 0.06 seconds

```

#### 4. Loading Multiple Hash Objects from SAS Data Sets

- a. Create a data set named **billing** that reads **CustomerID**, **OrderDate**, **ProductID**, **Quantity**, and **TotalRetailPrice** from **orion.orderfact**.

- 1) Enter the following DATA step in the Editor window:

Partial p306s04

```

data billing;
 drop rc1 rc2;
 if _N_=1 then do;
 if 0 then set orion.productlist(keep=ProductID
 ProductName);
 if 0 then set orion.customerdim(keep=CustomerID
 CustomerCountry
 CustomerName);
 declare hash Prod(dataset:'orion.productlist');
 Prod.definekey('ProductID');
 Prod.definedata('ProductName');
 Prod.definedone();
 declare hash Customer(dataset:'orion.customerdim');
 customer.definekey('CustomerID');
 customer.definedata('CustomerCountry', 'CustomerName');
 customer.definedone();
end;
set orion.orderfact(keep=OrderDate Quantity ProductID
 TotalRetailPrice CustomerID);
rc1=Customer.find();
if rc1=0;
rc2=Prod.find();
if rc2=0;
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

354 data billing;
355 drop rc1 rc2;
356 if _N_=1 then do;
357 if 0 then set orion.productlist(keep=ProductID ProductName);
358 if 0 then set orion.customerdim(keep=CustomerID
359 CustomerCountry
360 CustomerName);
361 declare hash Prod(dataset:'orion.productlist');
362 Prod.definekey('ProductID');
363 Prod.definedata('ProductName');
364 Prod.definedone();
365 declare hash Customer(dataset:'orion.customerdim');
366 customer.definekey('CustomerID');
367 customer.definedata('CustomerCountry', 'CustomerName');
368 customer.definedone();
369 end;
370 set orion.orderfact(keep=OrderDate Quantity ProductID
371 TotalRetailPrice CustomerID);
372 rc1=Customer.find();
373 if rc1=0;
374 rc2=Prod.find();
375 if rc2=0;
376
377 run;

NOTE: There were 556 observations read from the data set ORION.PRODUCTLIST.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.BILLING has 617 observations and 8 variables.
NOTE: DATA statement used (Total process time):
 real time 1.01 seconds
 cpu time 0.03 seconds

```

- b. Sort the **billing** data set by ascending values of **CustomerID** and **ProductID**.

- 1) Enter the following PROC SORT step in the Editor window.

## Partial p306s04

```

proc sort data=billing;
 by CustomerID ProductID;
run;

```

- 2) Submit the PROC SORT step.
- 3) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

379 proc sort data=billing;
380 by CustomerID ProductID;
381 run;

NOTE: There were 617 observations read from the data set WORK.BILLING.
NOTE: The data set WORK.BILLING has 617 observations and 8 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.13 seconds
 cpu time 0.06 seconds

```

c. Print the first five observations in **billing**.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p306s05

```
proc print data=billing(obs=5) noobs;
 var CustomerID CustomerName ProductID ProductName
 OrderDate Quantity TotalRetailPrice;
 title1 'Billing Information';
 title2 'Using a HASH Data Step Object';
run;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
383 proc print data=billing(obs=5) noobs;
384 var CustomerID CustomerName ProductID ProductName
385 OrderDate Quantity TotalRetailPrice;
386 title1 'Billing Information';
387 title2 'Using a HASH Data Step Object';
388 run;
NOTE: There were 5 observations read from the data set WORK.BILLING.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

## 5. Loading the Hash Object from a SAS Data Set and Retrieving Multiple Values

a. Write a DATA step to create a data set named **manager** that reads the **EmployeeID** and **Salary** variables from **orion.employeepayroll**.

- 1) Enter the following DATA step in the Editor window:

Partial p306s05

```
data manager;
 length EmployeeName EmpName ManagerName $40;
 keep EmployeeID EmpName ManagerID ManagerName Salary;
 if _N_=1 then do;
 declare hash M(dataset:'orion.staff');
 M.definekey('EmployeeID');
 M.definedata('ManagerID');
 M.definedone();
 declare hash N(dataset:'orion.employeeaddresses');
 N.definekey('EmployeeID');
 N.definedata('EmployeeName');
 N.definedone();
 call missing(EmployeeName);
 end;
 set orion.employeepayroll(keep=EmployeeID Salary);
 rc1=M.find(key:EmployeeID);
 rc2=N.find(key:EmployeeID);
```

```

if rc2=0 then EmpName=EmployeeName;
else EmpName=' ';
rc3=N.find(key:ManagerID);
if rc3=0 then ManagerName=EmployeeName;
else ManagerName=' ';
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

#### Partial SAS Log

```

136 data manager;
137 length EmployeeName EmpName ManagerName $40;
138 keep EmployeeID EmpName ManagerID ManagerName Salary;
139 if _N_=1 then do;
140 declare hash M(dataset:'orion.staff');
141 M.definekey('EmployeeID');
142 M.definedata('ManagerID');
143 M.definedone();
144 declare hash N(dataset:'orion.employeeaddresses');
145 N.definekey('EmployeeID');
146 N.definedata('EmployeeName');
147 N.definedone();
148 call missing(EmployeeName);
149 end;
150 set orion.employeepayroll(keep=EmployeeID Salary);
151 rc1=M.find(key:EmployeeID);
152 rc2=N.find(key:EmployeeID);
153 if rc2=0 then EmpName=EmployeeName;
154 else EmpName=' ';
155 rc3=N.find(key:ManagerID);
156 if rc3=0 then ManagerName=EmployeeName;
157 else ManagerName=' ';
158 run;
NOTE: There were 424 observations read from the data set ORION.STAFF.
NOTE: There were 424 observations read from the data set ORION.EMPLOYEEADDRESSES.
NOTE: There were 424 observations read from the data set ORION.EMPLOYEEPAYROLL.
NOTE: The data set WORK.MANAGER has 424 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds

```

- b. Print the first five observations of the **manager** data set.

- 1) Enter the following PROC PRINT step in the Editor window:

#### Partial p306s05

```

proc print data=manager(obs=5) noobs;
 title "manager Data Set";
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.

- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

160 proc print data=manager(obs=5) noobs;
161 title "manager Data Set";
162 run;
NOTE: There were 5 observations read from the data set WORK.MANAGER.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

## 6. Using a Hiter Object

- a. Use the data set **orion.shoesales** to create two data sets named **expensive** and **leastexpensive**. The original data set contains the variables **ProductID**, **ProductName**, and **TotalRetailPrice**. The data set **expensive** should contain the five most expensive pairs of shoes and the data set **leastexpensive** should contain the five least expensive pairs of shoes.

Use the starter program **p306e06**.

- 1) Open the starter program.
- 2) Enter the following DATA step in the Editor window:

Partial **p306s06**

```

data expensive leastexpensive;
 drop i;
 if _N_=1 then do;
 if 0 then set orion.shoesales;
 declare hash Shoes(dataset:'orion.shoesales',
 ordered:'descending');
 shoes.definekey('TotalRetailPrice');
 shoes.definedata('TotalRetailPrice',
 'ProductID', 'ProductName');
 shoes.definedone();
 declare hiter S('Shoes');
 end;
 S.first();
 do i=1 to 5;
 output expensive;
 S.next();
 end;
 S.last();
 do i=1 to 5;
 output leastexpensive;
 S.prev();
 end;
 stop;
run;

```

- 3) Submit the DATA step.
- 4) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

1 data expensive leastexpensive;
2 drop i;
3 if _N_=1 then do;
4 if 0 then set orion.shoesales;
5 declare hash Shoes(dataset:'orion.shoesales',
6 ordered:'descending');
7 shoes.definekey('TotalRetailPrice');
8 shoes.definedata('TotalRetailPrice',
9 'ProductID', 'ProductName');
10 shoes.definedone();
11 declare hiter S('Shoes');
12 end;
13 S.first();
14 do i=1 to 5;
15 output expensive;
16 S.next();
17 end;
18 S.last();
19 do i=1 to 5;
20 output leastexpensive;
21 S.prev();
22 end;
23 stop;
24 run;
NOTE: There were 58 observations read from the data set ORION.SHOESALES.
NOTE: The data set WORK.EXPENSIVE has 5 observations and 3 variables.
NOTE: The data set WORK.LEASTEXPENSIVE has 5 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.14 seconds
 cpu time 0.06 seconds

```

- b. Print the **expensive** and **leastexpensive** data sets.

- 1) Enter the following PROC PRINT steps in the Editor window.

## Partial p306s06

```

proc print data=expensive noobs;
 title "The Five Most Expensive Shoes";
run;

proc print data=leastexpensive noobs;
 title "The Five Least Expensive Shoes";
run;

```

- 2) Submit the PROC PRINT steps.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the steps executed without errors.

## Partial SAS Log

```

27 proc print data=expensive noobs;
28 title "The Five Most Expensive Shoes";
29 run;
NOTE: There were 5 observations read from the data set WORK.EXPENSIVE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.07 seconds
 cpu time 0.00 seconds
30
31 proc print data=leastexpensive noobs;
32 title "The Five Least Expensive Shoes";
33 run;
NOTE: There were 5 observations read from the data set WORK.LEASTEXPENSIVE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

## 7. Using a Hiter Object

- a. Use the data set **orion.shoesales** to create a data set named **shoesales** that contains the five most expensive pairs of shoes and the five least expensive pairs of shoes. The original data set contains the variables **ProductID**, **ProductName**, and **TotalRetailPrice**. The data set should contain a new variable named **Rank** that has the values of *Top 1* to *Top 5* for the five most expensive and *Bottom 1* to *Bottom 5* for the five least expensive pairs of shoes.

- 1) Enter the following DATA step in the Editor window:

## Partial p306s07

```

data shoesales;
 drop i;
 length Rank $ 8;
 if _N_=1 then do;
 if 0 then set orion.shoesales;
 declare hash Shoes(dataset:'orion.shoesales',
 ordered:'descending');
 Shoes.definekey('TotalRetailPrice');
 Shoes.definedata('TotalRetailPrice',
 'ProductID', 'ProductName');
 Shoes.definedone();
 declare hiter S('Shoes');
 end;
 S.first();
 do i=1 to 5;
 Rank=catx(' ', 'Top', i);
 output;
 S.next();
 end;
 S.last();
 do i=1 to 5;
 Rank=catx(' ', 'Bottom', i);
 output;
 S.prev();
 end;

```

```
stop;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
36 data shoesales;
37 drop i;
38 length Rank $ 8;
39 if _N_=1 then do;
40 if 0 then set orion.shoesales;
41 declare hash Shoes(dataset:'orion.shoesales',
42 ordered:'descending');
43 Shoes.definekey('TotalRetailPrice');
44 Shoes.definedata('TotalRetailPrice',
45 'ProductID', 'ProductName');
46 Shoes.definedone();
47 declare hiter S('Shoes');
48 end;
49 S.first();
50 do i=1 to 5;
51 Rank=catx(' ', 'Top', i);
52 output;
53 S.next();
54 end;
55 S.last();
56 do i=1 to 5;
57 Rank=catx(' ', 'Bottom', i);
58 output;
59 S.prev();
60 end;
61 stop;
62 run;
NOTE: There were 58 observations read from the data set ORION.SHOESALES.
NOTE: The data set WORK.SHOESALES has 10 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.12 seconds
 cpu time 0.03 seconds
```

- b. Print the **shoesales** data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p306s07

```
proc print data=shoesales noobs;
 title "work.shoesales Data Set";
run;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

64 proc print data=shoesales noobs;
65 title "work.shoesales Data Set";
66 run;
NOTE: There were 10 observations read from the data set WORK.SHOESALES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 8. Using a Hiter Object

- a. Use a hiter object to create a data set named **different** that contains unique values of **CustomerID** and **OrderType** from the data set named **orion.orderfact**. There should be 100 observations in the data set **different**.

- 1) Enter the following DATA step in the Editor window:

Partial p306s08

```

data different;
drop rc;
if _N_=1 then do;
 if 0 then set orion.orderfact(keep=CustomerID
 OrderType);
 declare hash Orders(dataset: 'orion.orderfact',
 ordered: 'yes');
 declare hiter OF('Orders');
 orders.definekey('CustomerID', 'OrderType');
 orders.definedata('CustomerID', 'OrderType');
 orders.definedone();
end;
rc=OF.first();
do while (rc=0);
 output;
 rc=OF.next();
end;
stop;
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

69 data different;
70 drop rc;
71 if _N_=1 then do;
72 if 0 then set orion.orderfact(keep=CustomerID
 OrderType);
73 declare hash Orders(dataset: 'orion.orderfact',
 ordered: 'yes');
74 declare hiter OF('Orders');
75 orders.definekey('CustomerID', 'OrderType');
76 orders.definedata('CustomerID', 'OrderType');
77 orders.definedone();
78 end;
79 rc=OF.first();
80

```

```

82 do while (rc=0);
83 output;
84 rc=OF.next();
85 end;
86 stop;
87 run;

NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.DIFFERENT has 100 observations and 2 variables.
NOTE: DATA statement used (Total process time):
 real time 0.04 seconds
 cpu time 0.00 seconds

```

- b. Print the first five observations of the data set **different**.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p306s08

```

proc print data=different(obs=5) noobs;
 title "No Duplicates";
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

89 proc print data=different(obs=5) noobs;
90 title "No Duplicates";
91 run;

NOTE: There were 5 observations read from the data set WORK.DIFFERENT.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## Solutions to Student Activities (Polls/Quizzes)

### 6.02 Short Answer Poll – Correct Answer

Submit the program **p306a01** and examine the SAS log.  
You should receive an error. Why?

```
ERROR: Undeclared data symbol ContinentName for hash object at line 62 column 7.
ERROR: DATA STEP Component Object failure. Aborted during the EXECUTION phase.
```

The data component **ContinentName** was not defined  
as a data set variable. Use the LENGTH statement to  
create the variable **ContinentName**.

29

### 6.03 Short Answer Poll – Correct Answer

Submit the program **p306a02** and examine the SAS log.  
You should receive a note about **ContinentName**.  
Why do you get that note?

**NOTE: Variable ContinentName is uninitialized.**

A note is written to the log because the variable  
**ContinentName** does not appear on the left side  
of an equal sign (=). To suppress this note, use  
a CALL MISSING routine.

31

## 6.04 Short Answer Poll – Correct Answer

Why are the statements in the DO group programmed to execute only one time?

```
data country;
 drop rc;
 length ContinentName $ 30;
 if _n_=1 then do;
 call missing(Co
 declare hash Co
 ContName.define()
 ContName.define()
 ContName.define()
 ContName.add(k)
 ContName.add(k)
 ContName.add(k)
 ContName.add(k)
 ContName.add(k)
 end;
 set orion.country(keep=ContinentID Country CountryName);
 rc=ContName.find();
run;
```

The statements associated with the hash object are all executable. Hash object memory is not released and reused each time. You can potentially run out of memory if you have a lot of data to load into the hash object.

35

p306d02

## 6.05 Multiple Choice Poll – Correct Answer

What would be the value of **ContinentName** if the value of **ContinentID** were not found in the hash object (**rc ne 0**)?

- a. missing
- b. North America
- c. Australia/Pacific
- d. unknown

**ContinentName is initialized to missing at the top of the DATA step.**

58

## 6.06 Quiz – Correct Answer

Use the DECLARE statement to create a hash object named **tree** and load it from the data set **work.naturepark**.

```
DECLARE object object-reference(<arg_tag-1:value-1
<,...arg_tag-n: value-n>>);
```

```
declare hash tree(dataset: 'work.naturepark');
```

69

## 6.07 Multiple Choice Poll – Correct Answer

The program **p306d03** created the variable **rc** and then dropped it. How can you avoid creating the variable so that you do not have to drop it?

- a. Use a WHERE statement or a WHERE= data set option.
- b. Use a KEEP= or DROP= data set option in **orion.country**.
- c. Test the result of the FIND method in the subsetting IF statement.
- d. Use a KEEP or DROP statement.

79

## 6.08 Short Answer Poll – Correct Answer

What is the advantage of testing the value of EOF before you execute the OUTPUT method?

```
set orion.country(keep=ContinentID Country
 CountryName
 Population) end=eof;
 rc=ContName.find();
 rc=C.add();
if eof then
 rc=C.output(dataset: 'work.countries');
```

As a result of testing the value of EOF, the OUTPUT method is executed only once after the last observation is read.

100

## 6.09 Multiple Answer Poll – Correct Answers

Which of the following are true regarding the hash iterator (hiter) object?

- a. You must define the hash object, which is referenced by the hiter object before the DECLARE statement for the hiter object.
- b. Ascending is the only valid method of moving through the hiter object.
- c. You can move through the hiter object using either the key or data items.
- d. The hiter object is an ordered view of the hash object.

120

## 6.10 Multiple Answer Poll – Correct Answers

Which two of the following pairs of methods are used to retrieve data starting from the top and bottom of a hiter object?

- a. FIRST( ) and PREV( )
- b. FIRST( ) and NEXT( )
- c. LAST( ) and PREV( )
- d. LAST( ) and NEXT( )



# Chapter 7 Combining Data Horizontally

|                                                                     |             |
|---------------------------------------------------------------------|-------------|
| <b>7.1 DATA Step Merges and SQL Procedure Joins .....</b>           | <b>7-3</b>  |
| Demonstration: Using the DATA Step to Perform a Match-Merge .....   | 7-14        |
| Demonstration: Using a PROC SQL Join to Perform a Match-Merge ..... | 7-17        |
| Exercises .....                                                     | 7-20        |
| <b>7.2 Using an Index to Combine Data.....</b>                      | <b>7-25</b> |
| Demonstration: Using Multiple SET Statements with KEY= Options..... | 7-43        |
| Exercises .....                                                     | 7-48        |
| <b>7.3 Combining Summary and Detail Data.....</b>                   | <b>7-53</b> |
| Exercises .....                                                     | 7-66        |
| <b>7.4 Combining Data Conditionally .....</b>                       | <b>7-70</b> |
| Exercises .....                                                     | 7-85        |
| <b>7.5 Solutions .....</b>                                          | <b>7-89</b> |
| Solutions to Exercises .....                                        | 7-89        |
| Solutions to Student Activities (Polls/Quizzes) .....               | 7-125       |



# 7.1 DATA Step Merges and SQL Procedure Joins

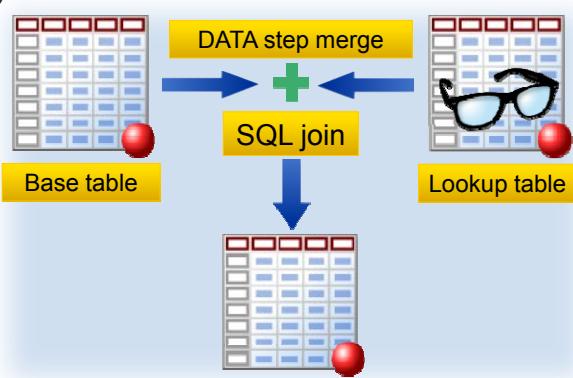
## Objectives

- Use the DATA step MERGE statement to combine multiple SAS data sets.
- Use the SQL procedure to join SAS data sets without a common variable.
- Describe the differences between the DATA step MERGE statement and PROC SQL.

3

## Combine Data Horizontally to Perform a Table Lookup

A DATA step merge or PROC SQL join is used to combine data sets horizontally, matching rows based on key columns or conditions.



4

## DATA Step Methods

The DATA step provides several methods to combine data horizontally.

### DATA Step



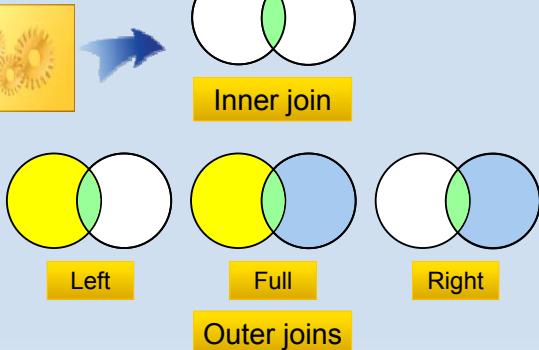
5

Information about the UPDATE and MODIFY statements is available on the extended learning page for this course.

## SQL Procedure Methods

There are also several types of SQL joins.

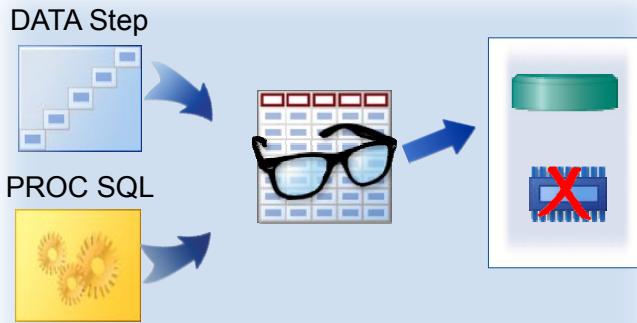
### PROC SQL



6

## Methods of Combining Data Horizontally

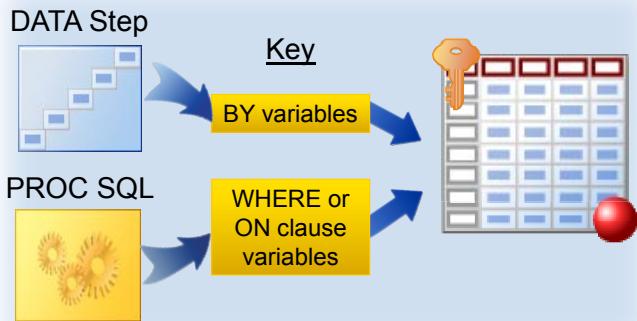
With these techniques, the lookup table resides on disk rather than in memory.



7

## Methods of Combining Data Horizontally

In DATA step merges, BY variables are the lookup keys.

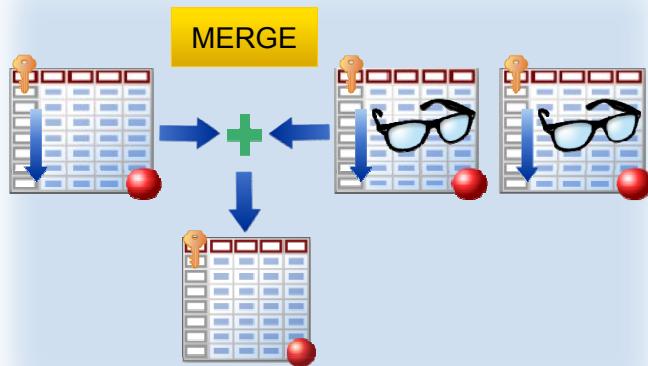


In SQL joins, WHERE or ON clause variables are the lookup keys.

8

## DATA Step Merge

In a DATA step merge, the data sets must be sorted or indexed by the key columns.



9

Indexed data sets can be used in a merge, but sorting is recommended.

## DATA Step Merge

Merge `orion.country` with `orion.continent` by `ContinentID`.

```
data countryinfo;
 merge orion.country
 orion.continent;
 by ContinentID;
run;
```

| Partial <code>orion.country</code> |     |              | Partial <code>orion.continent</code> |                | Partial results |              |                |
|------------------------------------|-----|--------------|--------------------------------------|----------------|-----------------|--------------|----------------|
| Country                            | ... | Continent ID | Continent ID                         | Continent Name | Country         | Continent ID | Continent Name |
| CA                                 | ... | 91           | 91                                   | North America  | CA              | 91           | North America  |
| US                                 | ... | 91           | 93                                   | Europe         | US              | 91           | North America  |
| DE                                 | ... | 93           | 94                                   | Africa         | DE              | 93           | Europe         |
| ZA                                 | ... | 94           | 95                                   | Asia           | ZA              | 94           | Africa         |

10

## Setup for the Poll

The following SAS program executes:

```
proc sort data=orion.country out=country;
 by ContinentID;
run;

data countryinfo;
 merge country orion.continent;
 by ContinentID;
run;
```

- ✍ The **orion.continent** data set is stored in ascending **ContinentID** order.

11

p307d01

## 7.01 Multiple Choice Poll

If the data set **country** has seven observations and the data set **orion.continent** has five observations, what stops the execution of the DATA step that contains a MERGE statement?

- a. the end of file for **country**
- b. the end of file for **orion.continent**
- c. the end of file for both data sets

12

## 7.02 Quiz

Data set **three** contains \_\_ observations where the value of **X** is 1.

DATA Step



| one |   | two |   | three |   |   |
|-----|---|-----|---|-------|---|---|
| X   | Y | X   | Z | X     | Y | Z |
| 1   | a | 1   | f |       |   |   |
| 1   | d | 1   | r |       |   |   |
| 4   | c | 1   | s |       |   |   |
| 5   | g | 3   | t |       |   |   |
|     |   | 4   | w |       |   |   |

```
data three;
 merge one two;
 by x;
run;
```

p307d02

14

## DATA Step Merge: A Sequential Process

Two observations represent the nonmatches.

DATA Step



| one |   | two |   | three |   |   |
|-----|---|-----|---|-------|---|---|
| X   | Y | X   | Z | X     | Y | Z |
| 1   | a | 1   | f | 1     | a | f |
| 1   | d | 1   | r | 1     | d | r |
| 4   | c | 1   | s | 1     | d | s |
| 5   | g | 3   | t | 3     |   | t |
|     |   | 4   | w | 4     | c | w |
|     |   |     |   | 5     | g |   |

```
data three;
 merge one two;
 by x;
run;
```

p307d02

16

## 7.03 Multiple Choice Poll

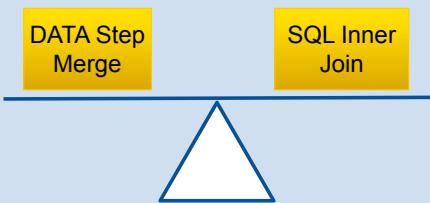
Which type of SQL join most closely mimics a DATA step merge?

- a. inner join
- b. full outer join
- c. left outer join
- d. right outer join

17

## DATA Step Match-Merges versus PROC SQL Joins

The DATA step merge and PROC SQL inner join can produce the same results. However, choosing one method over the other involves trade-offs.



19

## One-to-One Matches

The DATA step and PROC SQL sometimes produce the same results.

```
data dsthree;
 merge one two;
 by X;
run;
```

| one |   | two |   |
|-----|---|-----|---|
| X   | Y | X   | Z |
| 1   | a | 1   | f |
| 2   | b | 2   | g |

```
proc sql;
 create table sqlthree as
 select one.X, Y, Z
 from one, two
 where one.X=two.X;
quit;
```

One-to-one match

| dsthree |   |   | sqlthree |   |   |
|---------|---|---|----------|---|---|
| X       | Y | Z | X        | Y | Z |
| 1       | a | f | 1        | a | f |
| 2       | b | g | 2        | b | g |

20

P307d03

## One-to-Many Matches

The DATA step and PROC SQL sometimes produce the same results.

```
data dsthree;
 merge one two;
 by X;
run;
```

| one |   | two |   |
|-----|---|-----|---|
| X   | Y | X   | Z |
| 1   | a | 1   | f |
| 2   | b | 1   | r |
|     |   | 2   | g |

```
proc sql;
 create table sqlthree as
 select one.X, Y, Z
 from one, two
 where one.X=two.X;
quit;
```

One-to-many match

| dsthree |   |   | sqlthree |   |   |
|---------|---|---|----------|---|---|
| X       | Y | Z | X        | Y | Z |
| 1       | a | f | 1        | a | f |
| 1       | a | r | 1        | a | r |
| 2       | b | g | 2        | b | g |

21

P307d03

## Many-to-Many Matches

However, they do not *always* produce the same results.

```
data dsthree;
 merge one two;
 by X;
run;
```

| one |   | two |   |
|-----|---|-----|---|
| X   | Y | X   | Z |
| 1   | a |     | f |
| 1   | c |     | r |
| 2   | b |     | g |

```
proc sql;
 create table sqlthree as
 select one.X, Y, Z
 from one, two
 where one.X=two.X;
quit;
```

Many-to-many match

| dsthree |   |   |
|---------|---|---|
| X       | Y | Z |
| 1       | a | f |
| 1       | a | r |
| 1       | c | f |
| 1       | c | r |
| 2       | b | g |

sqlthree

| X | Y | Z |
|---|---|---|
| 1 | a | f |
| 1 | a | r |
| 1 | c | f |
| 1 | c | r |
| 2 | b | g |

22

P307d03

## Nonmatches

They do not produce the same results when there are nonmatches.

```
data dsthree;
 merge one two;
 by X;
run;
```

| one |   | two |   |
|-----|---|-----|---|
| X   | Y | X   | Z |
| 1   | a |     | f |
| 2   | b |     |   |
| 3   | c |     | t |

```
proc sql;
 create table sqlthree as
 select one.X, Y, Z
 from one, two
 where one.X=two.X;
quit;
```

Nonmatches

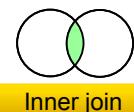
| dsthree |   |   |
|---------|---|---|
| X       | Y | Z |
| 1       | a | f |
| 2       | b |   |
| 3       | c | t |
| 4       |   | w |

sqlthree

| X | Y | Z |
|---|---|---|
| 1 | a | f |
| 3 | c | t |

23

P307d03



Inner join

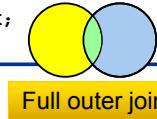
## Nonmatches

Use an SQL outer join to explicitly handle nonmatches.

```
data dsthree;
 merge one two;
 by X;
run;
```

| one |   | two |   |
|-----|---|-----|---|
| X   | Y | X   | Z |
| 1   | a |     | f |
| 2   | b |     | t |
| 3   | c |     | w |

```
proc sql;
 create table sqlthree as
 select coalesce(one.X,two.X) as X, Y, Z
 from one
 full outer join
 two
 on one.X=two.X;
quit;
```



Nonmatches

| dsthree |   |   | sqlthree |   |   |
|---------|---|---|----------|---|---|
| X       | Y | Z | X        | Y | Z |
| 1       | a | f | 1        | a | f |
| 2       | b |   | 2        | b |   |
| 3       | c | t | 3        | c | t |
| 4       |   | w | 4        |   | w |

24

P307d03

## 7.04 Quiz

Data set **three** contains \_\_\_ observations where the value of **X** is 1.

PROC SQL



Matches only

| one |   |
|-----|---|
| X   | Y |
| 1   | a |
| 1   | d |
| 4   | c |
| 5   | g |

| two |   |
|-----|---|
| X   | Z |
| 1   | f |
| 1   | r |
| 1   | s |
| 3   | t |
| 4   | w |

| three |   |   |
|-------|---|---|
| X     | Y | Z |
| 1     | a |   |
| 1     | d |   |
| 1     | c |   |
| 5     | g |   |
| 1     |   | f |
| 1     |   | r |
| 1     |   | s |
| 3     |   | t |
| 4     |   | w |

```
proc sql;
 create table three as
 select one.* , two.Z
 from one, two
 where one.X=two.X;
quit;
```

25

P307d03

## Using the DATA Step to Perform a Match-Merge

The Human Resources Department needs a SAS data set with the name of each employee's manager.

Desired output **work.names**

| Employee ID | Manager ID | Employee Name     | Manager Name |
|-------------|------------|-------------------|--------------|
| 120102      | 120101     | Zhou, Tom         | Lu, Patrick  |
| 120103      | 120101     | Dawes, Wilson     | Lu, Patrick  |
| 120104      | 120101     | Billington, Karen | Lu, Patrick  |
| 120105      | 120101     | Povey, Liz        | Lu, Patrick  |
| 120121      | 120102     | Elvish, Irenie    | Zhou, Tom    |

27

## Business Scenario

**Orion.staff** contains the IDs. **Orion.employeeaddresses** contains the names.

Partial **orion.staff**

| Employee ID | Start Date | Gender | Manager ID |
|-------------|------------|--------|------------|
| 120101      | 01JUL2007  | M      | 120261     |
| 120102      | 01JUN1993  | M      | 120101     |
| 120104      | 01JAN1985  | F      | 120101     |

Base table

Partial **orion.employeeaddresses**

| Employee ID | Employee Name        | State | Country |
|-------------|----------------------|-------|---------|
| 120144      | Abbott, Ray          | FL    | US      |
| 120145      | Aisbitt, Sandy       |       | AU      |
| 120761      | Akinfolarin, Tameaka | PA    | US      |

Lookup table

28

**Business Scenario**

This scenario requires combining the data twice.

**Partial orion.staff**

| Employee ID | Start Date | Gender | Manager ID |
|-------------|------------|--------|------------|
| 120101      | 01JUL2007  | M      | 120261     |
| 120102      | 01JUN1993  | M      | 120101     |

Patrick Lu is both an employee and a manager.

**Partial orion.employeeaddresses**



| Employee ID | Employee Name | State | Country |
|-------------|---------------|-------|---------|
| 120101      | Lu, Patrick   |       | AU      |
| 120102      | Zhou, Tom     |       | AU      |

**Partial work.names**



| Employee ID | Manager ID | Employee Name | Manager Name     |
|-------------|------------|---------------|------------------|
| 120101      | 120261     | Lu, Patrick   | Highpoint, Harry |
| 120102      | 120101     | Zhou, Tom     | Lu, Patrick      |

29



## Using the DATA Step to Perform a Match-Merge

### p307d04

This demonstration illustrates merging SAS data sets twice to perform a table lookup.

1. Open the program **p307d04**.
2. Submit the first PROC SORT, DATA **temp1**, and PROC PRINT steps.

#### Partial p307d04

```

proc sort data=orion.employeeaddresses(keep=EmployeeID EmployeeName)
 out=addresses;
 by EmployeeID;
run;

data temp1;
 keep EmployeeName EmployeeID ManagerID;
 merge orion.staff(in=S keep=EmployeeID ManagerID)
 addresses(in=A);
 by EmployeeID;
 if S and A; /* Matches only */
run;

proc print data=temp1(obs=5) noobs;
run;

```

3. Examine the SAS log to verify that the steps ran without errors and created the correct number of observations and variables.

## Partial SAS Log

```

1 proc sort data=orion.employeeaddresses(keep=EmployeeID
2 EmployeeName)
3 out=addresses;
4 by EmployeeID;
5 run;
NOTE: There were 424 observations read from the data set ORION.EMPLOYEEADDRESSES.
NOTE: The data set WORK.ADDRESSES has 424 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.12 seconds
 cpu time 0.03 seconds
6
7 data temp1;
8 keep EmployeeName EmployeeID ManagerID;
9 merge orion.staff(in=S keep=EmployeeID ManagerID)
10 addresses(in=A);
11 by EmployeeID;
12 if S and A; /* Matches only */
13 run;
NOTE: There were 424 observations read from the data set ORION.STAFF.
NOTE: There were 424 observations read from the data set WORK.ADDRESSES.
NOTE: The data set WORK.TEMP1 has 424 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.15 seconds
 cpu time 0.06 seconds
15 proc print data=temp1(obs=5) noobs;
16 run;
NOTE: There were 5 observations read from the data set WORK.TEMP1.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.21 seconds
 cpu time 0.04 seconds

```

## PROC PRINT Output

| EmployeeID | ManagerID | EmployeeName      |
|------------|-----------|-------------------|
| 120101     | 120261    | Lu, Patrick       |
| 120102     | 120101    | Zhou, Tom         |
| 120103     | 120101    | Dawes, Wilson     |
| 120104     | 120101    | Billington, Karen |
| 120105     | 120101    | Povey, Liz        |

4. Submit the last three steps (PROC SORT, a DATA step, and PROC PRINT).

## Partial p307d04

```

proc sort data=temp1;
 by ManagerID;
run;

data names;
 merge temp1(in=T)
 addresses(in=A rename=(EmployeeID=ManagerID
 EmployeeName=ManagerName));
 by ManagerID;
 if A and T; /* Matches only */
run;

```

```
proc print data=names(obs=5) noobs;
 title "Names Data Set";
run;
```

5. Examine the SAS log to verify that the steps ran without errors and created the correct number of observations and variables.

#### Partial SAS Log

```
14 proc sort data=temp1;
15 by ManagerID;
16 run;
NOTE: There were 424 observations read from the data set WORK.TEMP1.
NOTE: The data set WORK.TEMP1 has 424 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.00 seconds
17
18 data names;
19 merge temp1(in=T)
20 addresses(in=A rename=(EmployeeID=ManagerID
21 EmployeeName=ManagerName));
22 by ManagerID;
23 if A and T; /*Matches only*/
24 run;
NOTE: There were 424 observations read from the data set WORK.TEMP1.
NOTE: There were 424 observations read from the data set WORK.ADDRESSES.
NOTE: The data set WORK.NAMES has 423 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.00 seconds
25
26 proc print data=names(obs=5) noobs;
27 title "Names Data Set";
28 run;
NOTE: There were 5 observations read from the data set WORK.NAMES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.17 seconds
 cpu time 0.03 seconds
```

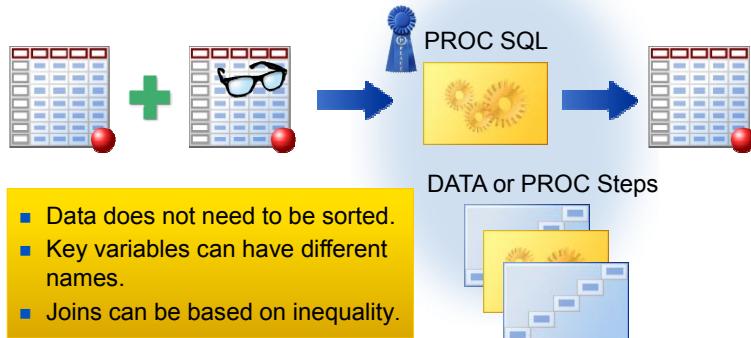
6. View the output to verify that it matches the expected output.

#### PROC PRINT Output

| Names Data Set |           |                   |             |
|----------------|-----------|-------------------|-------------|
| EmployeeID     | ManagerID | EmployeeName      | ManagerName |
| 120102         | 120101    | Zhou, Tom         | Lu, Patrick |
| 120103         | 120101    | Dawes, Wilson     | Lu, Patrick |
| 120104         | 120101    | Billington, Karen | Lu, Patrick |
| 120105         | 120101    | Povey, Liz        | Lu, Patrick |
| 120121         | 120102    | Elvish, Irenie    | Zhou, Tom   |

## Using PROC SQL to Perform a Table Lookup

PROC SQL can often do in one step what it takes multiple DATA and PROC steps to accomplish.



31



## Using a PROC SQL Join to Perform a Match-Merge

### p307d05

This demonstration illustrates performing a match-merge using the PROC SQL inner join.

1. Open the program **p307d05**.
2. Submit the PROC SQL step.

#### Partial p307d05

```
proc sql;
 create table namessql as
 select e.EmployeeID,
 e.EmployeeName,
 ManagerID,
 m.EmployeeName as ManagerName
 from orion.staff,
 orion.employeeaddresses as e,
 orion.employeeaddresses as m
 where e.EmployeeID=staff.EmployeeID
 and m.EmployeeID=staff.ManagerID
 order by ManagerID,
 EmployeeID;
quit;
```

**orion.employeeaddresses** is listed twice in the FROM list. This is called a *reflexive join*.

3. Examine the SAS log to verify that the steps ran without errors and created the correct number of observations and variables.

## Partial SAS Log

```

33 proc sql;
34 create table namessql as
35 select e.EmployeeID,
36 e.EmployeeName,
37 ManagerID,
38 m.EmployeeName as ManagerName
39 from orion.staff,
40 orion.employeeaddresses as e,
41 orion.employeeaddresses as m
42 where e.EmployeeID=staff.EmployeeID
43 and m.EmployeeID=staff.ManagerID
44 order by ManagerID,
45 EmployeeID;
NOTE: Table WORK.NAMESSQL created, with 423 rows and 4 columns.
46 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.23 seconds
 cpu time 0.06 seconds

```

4. Submit the PROC PRINT step.

```

proc print data=namessql(obs=5) noobs;
 title "Employee and Manager Names";
run;

```

5. Examine the SAS log to verify that the step ran without errors and printed the correct number of observations.

## Partial SAS Log

```

47 proc print data=namessql(obs=5) noobs;
48 title "Employee and Manager Names";
49 run;
NOTE: There were 5 observations read from the data set WORK.NAMESQL.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
50
51 title;

```

6. View the output to verify that it matches the expected output.

## PROC PRINT Output

| Employee and Manager Names |                   |           |             |
|----------------------------|-------------------|-----------|-------------|
| Employee                   |                   | ManagerID | ManagerName |
| ID                         | EmployeeName      |           |             |
| 120102                     | Zhou, Tom         | 120101    | Lu, Patrick |
| 120103                     | Dawes, Wilson     | 120101    | Lu, Patrick |
| 120104                     | Billington, Karen | 120101    | Lu, Patrick |
| 120105                     | Povey, Liz        | 120101    | Lu, Patrick |
| 120121                     | Elvish, Irenie    | 120102    | Zhou, Tom   |

## Comparing the DATA Step Merge and SQL Inner Join

| DATA Step Match Merge                                                                                       | SQL Inner Join                                                                                 |
|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| There is no limit to the number of data sets or the size of the data sets other than disk space and memory. | The maximum number of tables that can be joined at one time is 256.                            |
| Data is processed sequentially so that observations with duplicate BY values are joined one-to-one.         | Data is processed using a Cartesian product for duplicate BY values.                           |
| Multiple data sets can be created.                                                                          | Only one data set can be created with one CREATE TABLE statement.                              |
| Complex business logic can be incorporated using IF-THEN or SELECT/WHEN logic.                              | CASE logic can be used for business logic. However, it is not as flexible as DATA step syntax. |
| The data sets being merged must be sorted or indexed on the BY variables.                                   | The data sets being joined do not need to be sorted or indexed.                                |
| An exact match on the BY-variables' values must be found.                                                   | Inequality joins can be performed.                                                             |
| Like-named BY variables must be available in all data sets.                                                 | Matches can be performed on columns with nonmatching names.                                    |

### Default Action with a DATA Step Merge

Variable name, type, and length attributes are established by the first data set.

Example:

```
data c;
 merge a b;
 by x;
run;
```

| a                                                                                                                                                                                                            | b      | c |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|---|-----|------|---|------|---|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|-----|------|---|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|-----|------|---|------|---|------|
| <table border="1"> <thead> <tr> <th>x</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>N 8</td> <td>\$ 4</td> </tr> <tr> <td>1</td> <td>ABCD</td> </tr> <tr> <td>1</td> <td>ZZZZ</td> </tr> </tbody> </table> | x      | y | N 8 | \$ 4 | 1 | ABCD | 1 | ZZZZ | <table border="1"> <thead> <tr> <th>x</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>N 8</td> <td>\$ 6</td> </tr> <tr> <td>1</td> <td>EFGHIJ</td> </tr> </tbody> </table> | x | y | N 8 | \$ 6 | 1 | EFGHIJ | <table border="1"> <thead> <tr> <th>x</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>N 8</td> <td>\$ 4</td> </tr> <tr> <td>1</td> <td>EFGH</td> </tr> <tr> <td>1</td> <td>ZZZZ</td> </tr> </tbody> </table> | x | y | N 8 | \$ 4 | 1 | EFGH | 1 | ZZZZ |
| x                                                                                                                                                                                                            | y      |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| N 8                                                                                                                                                                                                          | \$ 4   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| 1                                                                                                                                                                                                            | ABCD   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| 1                                                                                                                                                                                                            | ZZZZ   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| x                                                                                                                                                                                                            | y      |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| N 8                                                                                                                                                                                                          | \$ 6   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| 1                                                                                                                                                                                                            | EFGHIJ |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| x                                                                                                                                                                                                            | y      |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| N 8                                                                                                                                                                                                          | \$ 4   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| 1                                                                                                                                                                                                            | EFGH   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |
| 1                                                                                                                                                                                                            | ZZZZ   |   |     |      |   |      |   |      |                                                                                                                                                                            |   |   |     |      |   |        |                                                                                                                                                                                                              |   |   |     |      |   |      |   |      |

## Default Action with a DATA Step Merge

In a one-to-one or one-to-many merge, variable values might come from the last data set.

Example:

```
data c;
 merge a b;
 by X;
run;
```

| <b>a</b> |           | <b>b</b> |           |
|----------|-----------|----------|-----------|
| X<br>N 8 | y<br>\$ 4 | X<br>N 8 | Y<br>\$ 6 |
| 1        | ABCD      | 1        | EFGHIJ    |
| 1        | ZZZZ      |          |           |

**c**

| X<br>N 8 | y<br>\$ 4 |
|----------|-----------|
| 1        | EFGH      |
| 1        | ZZZZ      |

**WARNING:** Multiple lengths were specified for the variable y by input data set(s).  
This can cause truncation of data.

34

p307d06



## Exercises

---

### Level 1

#### 1. Merging or Joining Three Data Sets

The data set **orion.orderfact** has details about purchases that were made.

Partial **orion.orderfact**

| Partial orion.orderfact |              |            |                   |                   |            |
|-------------------------|--------------|------------|-------------------|-------------------|------------|
| CustomerID              | EmployeeID   | StreetID   | OrderDate         | Delivery Date     | OrderID    |
| 63                      | 121039       | 9260125492 | 11JAN2007         | 11JAN2007         | 1230058123 |
| 5                       | 99999999     | 9260114570 | 15JAN2007         | 19JAN2007         | 1230080101 |
| 45                      | 99999999     | 9260104847 | 20JAN2007         | 22JAN2007         | 1230106883 |
| 41                      | 120174       | 1600101527 | 28JAN2007         | 28JAN2007         | 1230147441 |
| 183                     | 120134       | 1600100760 | 27FEB2007         | 27FEB2007         | 1230315085 |
| Order Type              | ProductID    | Quantity   | TotalRetail Price | CostPrice PerUnit | Discount   |
| 1                       | 220101300017 | 1          | \$16.50           | \$7.45            | .          |
| 2                       | 230100500026 | 1          | \$247.50          | \$109.55          | .          |
| 2                       | 240600100080 | 1          | \$28.30           | \$8.55            | .          |
| 1                       | 240600100010 | 2          | \$32.00           | \$6.50            | .          |
| 1                       | 240200200039 | 3          | \$63.60           | \$8.80            | .          |

The data set **orion.customerdim** has customers' names.

#### Partial **orion.customerdim**

| Partial orion.customerdim |                   |
|---------------------------|-------------------|
| CustomerID                | CustomerName      |
| 4                         | James Kvarniq     |
| 5                         | Sandrina Stephano |
| 9                         | Cornelia Krahel   |
| 10                        | Karen Ballinger   |
| 11                        | Elke Wallstab     |

The data set **orion.productdim** has the product names and supplier names.

#### Partial **orion.productdim**

| Partial orion.productdim |                                     |                         |
|--------------------------|-------------------------------------|-------------------------|
| ProductID                | ProductName                         | SupplierName            |
| 210200100009             | Kids Sweat Round Neck, Large Logo   | A Team Sports           |
| 210200100017             | Sweatshirt Children's O-Neck        | A Team Sports           |
| 210200200022             | Sunfit Slow Swimming Trunks         | Nautlius SportsWear Inc |
| 210200200023             | Sunfit Stockton Swimming Trunks Jr. | Nautlius SportsWear Inc |
| 210200300006             | Fleece Cuff Pant Kid'S              | Eclipse Inc             |

- a. Modify the program **p307e01**. Combine the three data sets to create a data set named **purchases** that contains the customer name, product name, and supplier name for the customers in the **orion.orderfact** data set.

Order the data by **ProductID**.

- b. Print the first five observations of the **purchases** data set.

#### PROC PRINT Output

| Partial purchases Data Set |                                     |                         |
|----------------------------|-------------------------------------|-------------------------|
| CustomerName               | ProductName                         | SupplierName            |
| Kyndal Hooks               | Kids Sweat Round Neck, Large Logo   | A Team Sports           |
| Annmarie Leveille          | Sweatshirt Children's O-Neck        | A Team Sports           |
| Najma Hicks                | Sunfit Slow Swimming Trunks         | Nautlius SportsWear Inc |
| Yan Kozlowski              | Sunfit Stockton Swimming Trunks Jr. | Nautlius SportsWear Inc |
| Kyndal Hooks               | Fleece Cuff Pant Kid'S              | Eclipse Inc             |

## Level 2

### 2. Merging or Joining Data to Create Multiple Data Sets

The data set **orion.orderfact** has details about purchases that were made.

**Partial orion.orderfact**

| Partial orion.orderfact |              |                   |           |                   |            |
|-------------------------|--------------|-------------------|-----------|-------------------|------------|
| CustomerID              | EmployeeID   | StreetID          | OrderDate | Delivery Date     | OrderID    |
| 63                      | 121039       | 9260125492        | 11JAN2007 | 11JAN2007         | 1230058123 |
| 5                       | 99999999     | 9260114570        | 15JAN2007 | 19JAN2007         | 1230080101 |
| 45                      | 99999999     | 9260104847        | 20JAN2007 | 22JAN2007         | 1230106883 |
| 41                      | 120174       | 1600101527        | 28JAN2007 | 28JAN2007         | 1230147441 |
| 183                     | 120134       | 1600100760        | 27FEB2007 | 27FEB2007         | 1230315085 |
| Order Type              |              | TotalRetail Price |           | CostPrice PerUnit | Discount   |
| 1                       | 220101300017 | 1                 | \$16.50   | \$7.45            | .          |
| 2                       | 230100500026 | 1                 | \$247.50  | \$109.55          | .          |
| 2                       | 240600100080 | 1                 | \$28.30   | \$8.55            | .          |
| 1                       | 240600100010 | 2                 | \$32.00   | \$6.50            | .          |
| 1                       | 240200200039 | 3                 | \$63.60   | \$8.80            | .          |

The data set **orion.customerdim** has the customers' names.

**Partial orion.customerdim**

| Partial orion.customerdim |                   |
|---------------------------|-------------------|
| CustomerID                | CustomerName      |
| 4                         | James Kvarnig     |
| 5                         | Sandrina Stephano |
| 9                         | Cornelia Krahl    |
| 10                        | Karen Ballinger   |
| 11                        | Elke Wallstab     |

The data set **orion.productdim** has the product names and supplier names.

**Partial orion.productdim**

| Partial orion.productdim |                                     |                         |
|--------------------------|-------------------------------------|-------------------------|
| ProductID                | ProductName                         | SupplierName            |
| 210200100009             | Kids Sweat Round Neck, Large Logo   | A Team Sports           |
| 210200100017             | Sweatshirt Children's O-Neck        | A Team Sports           |
| 210200200022             | Sunfit Slow Swimming Trunks         | Nautlius SportsWear Inc |
| 210200200023             | Sunfit Stockton Swimming Trunks Jr. | Nautlius SportsWear Inc |
| 210200300006             | Fleece Cuff Pant Kid'S              | Eclipse Inc             |

- Modify **p307e02** by adding a DATA step to merge the three data sets and create the following three data sets:
  - a data set named **nopurchases** that contains the customers who did not make any purchases. It should only include the variables CustomerID, and CustomerName.
  - a data set named **purchases** that contains the customers that exist in the **orion.orderfact** data set. It should only include the variables CustomerID, CustomerName, ProductID, OrderID, Quantity, and TotalRetailPrice

- a data set named **noproducts** that contains the product names and suppliers for products that were not purchased. It should only include the variables ProductID, ProductName, and SupplierName.

Review the log to verify the correct number of observations as shown below.

#### Partial SAS Log

```

NOTE: There were 617 observations read from the data set WORK.ORDERFACT.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.TEMP has 617 observations and 13 variables.
NOTE: The data set WORK.NOPURCHASES has 2 observations and 2 variables.
NOTE: DATA statement used (Total process time):
 real time 0.21 seconds
 cpu time 0.06 seconds

<more SAS log>

NOTE: There were 617 observations read from the data set WORK.TEMP.
NOTE: There were 481 observations read from the data set ORION.PRODUCTDIM.
NOTE: The data set WORK.PURCHASES has 617 observations and 6 variables.
NOTE: The data set WORK.NOPRODUCTS has 0 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.15 seconds
 cpu time 0.06 seconds

```

- b. Print the first five observations of each data set.

#### work.nopurchases

| Partial nopurchases Data Set |                 |
|------------------------------|-----------------|
| CustomerID                   | CustomerName    |
| 33                           | Rolf Robak      |
| 42                           | Thomas Leitmann |

#### Partial work.purchases

| Partial purchases Data Set |            |          |             |                   |
|----------------------------|------------|----------|-------------|-------------------|
| CustomerID                 | OrderID    | Quantity | TotalRetail | CustomerName      |
| 90                         | 1243960910 | 2        | \$69.40     | Kyndal Hooks      |
| 49                         | 1234198497 | 1        | \$39.00     | Annmarie Leveille |
| 79                         | 1235926178 | 2        | \$36.00     | Najma Hicks       |
| 52                         | 1240886449 | 1        | \$19.80     | Yan Kozlowski     |
| 90                         | 1242149082 | 1        | \$14.30     | Kyndal Hooks      |

## Challenge

### 3. Merging or Joining Multiple Data Sets

The data set **orion.organizationdim** has the levels of managers for each employee.

 Not all **ManagerLevel** variables have values nor do all employees have six levels of managers.

**Partial orion.organizationdim**

| Partial orion.organizationdim |         |         |         |         |         |         |
|-------------------------------|---------|---------|---------|---------|---------|---------|
| EmployeeID                    | Manager | Manager | Manager | Manager | Manager | Manager |
|                               | Level1  | Level2  | Level3  | Level4  | Level5  | Level6  |
| 120101                        | 120261  | 120259  | .       | .       | .       | .       |
| 120102                        | 120101  | 120261  | 120259  | .       | .       | .       |
| 120103                        | 120101  | 120261  | 120259  | .       | .       | .       |
| 120104                        | 120101  | 120261  | 120259  | .       | .       | .       |
| 120105                        | 120101  | 120261  | 120259  | .       | .       | .       |

The data set **orion.employeeaddresses** contains the employee IDs and the employee names for all employees.

**Partial orion.employeeaddresses**

| Partial orion.employeeaddresses |                      |
|---------------------------------|----------------------|
| Employee                        | Employee             |
| ID                              | Name                 |
| 121044                          | Abbott, Ray          |
| 120145                          | Aisbitt, Sandy       |
| 120761                          | Akinfolarin, Tameaka |
| 120656                          | Amos, Salley         |
| 121107                          | Anger, Rose          |

- Create a data set named **managernames** that contains the **EmployeeID** variable, the six **ManagerID** variables, and the six manager names.
- Print observations 420 through 424 of the **managernames** data set.

**Partial work.managernames**

| Partial managernames Data |                  |         |                  |         |                 |         |          |                   |
|---------------------------|------------------|---------|------------------|---------|-----------------|---------|----------|-------------------|
| Obs                       | EmployeeID       | Manager | Manager          | Manager | Manager         | Manager | Manager  | Manager1Name      |
|                           |                  | Level1  | Level2           | Level3  | Level4          | Level5  | Level6   |                   |
| 420                       | 121144           | 121142  | 121141           | 120261  | 120259          | .       | .        | Steiber, Reginald |
| 421                       | 121145           | 121142  | 121141           | 120261  | 120259          | .       | .        | Steiber, Reginald |
| 422                       | 121146           | 121141  | 120261           | 120259  | .               | .       | .        | Bleu, Henri Le    |
| 423                       | 121147           | 121142  | 121141           | 120261  | 120259          | .       | .        | Steiber, Reginald |
| 424                       | 121148           | 121141  | 120261           | 120259  | .               | .       | .        | Bleu, Henri Le    |
| Obs                       | Manager2Name     |         | Manager3Name     |         | Manager4Name    |         | Manager5 | Manager6          |
|                           |                  |         |                  |         |                 |         | Name     | Name              |
| 420                       | Bleu, Henri Le   |         | Highpoint, Harry |         | Miller, Anthony |         |          |                   |
| 421                       | Bleu, Henri Le   |         | Highpoint, Harry |         | Miller, Anthony |         |          |                   |
| 422                       | Highpoint, Harry |         | Miller, Anthony  |         |                 |         |          |                   |
| 423                       | Bleu, Henri Le   |         | Highpoint, Harry |         | Miller, Anthony |         |          |                   |
| 424                       | Highpoint, Harry |         | Miller, Anthony  |         |                 |         |          |                   |

## 7.2 Using an Index to Combine Data

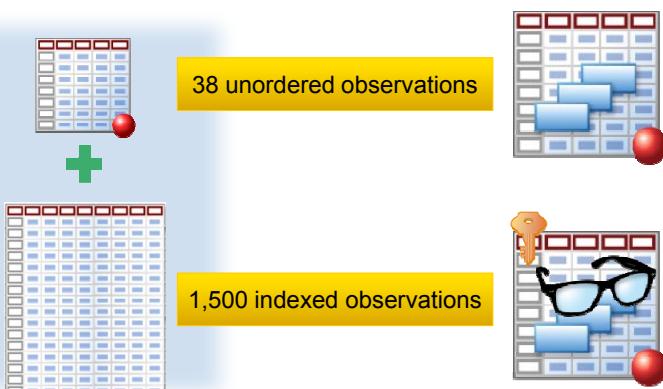
### Objectives

- Use the SET statement with the KEY= option to combine two SAS data sets.
- Use \_IORC\_ to control output.

38

### Business Scenario

The Marketing Department wants to combine the data from two data sets. One of the data sets is much larger than the other.



39

## 7.05 Multiple Choice Poll

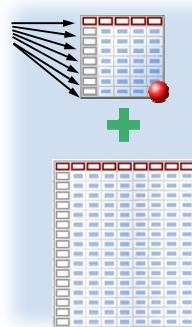
If you use a DATA step merge to combine the data sets, how many observations are read from the larger data set, **orion.customerdimmore**?

- a. 38
- b. 1,500

40

## Business Scenario

**Orion.catalog** is not indexed or sorted by **CustomerID**.  
The entire file must be read sequentially.



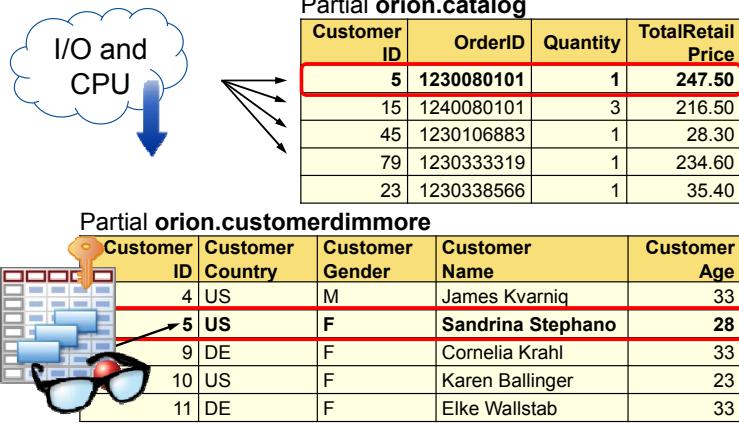
Partial **orion.catalog**

| Customer ID | OrderID    | Quantity | TotalRetail Price |
|-------------|------------|----------|-------------------|
| 5           | 1230080101 | 1        | 247.50            |
| 15          | 1240080101 | 3        | 216.50            |
| 45          | 1230106883 | 1        | 28.30             |
| 79          | 1230333319 | 1        | 234.60            |
| 23          | 1230338566 | 1        | 35.40             |

42

## Business Scenario

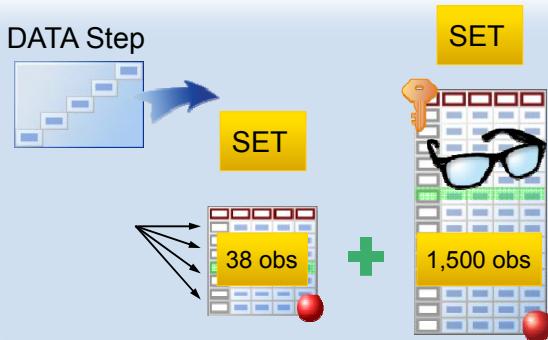
The larger file is indexed by **CustomerID**. Using the index to access observations directly can save I/O and CPU time.



43

## Multiple SET Statements

Using multiple SET statements enables SAS to read the smaller data set sequentially and the larger data set directly using its index.

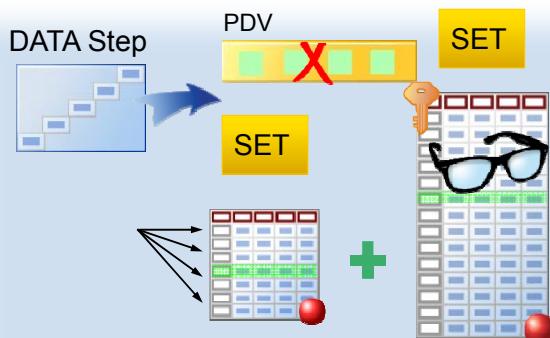


44

- The DATA step stops processing at the first end-of-file marker that is encountered.

## Multiple SET Statements

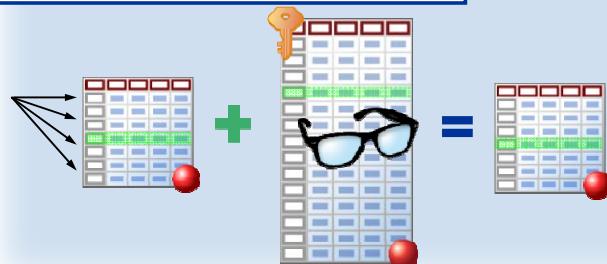
Existing SAS variables read in a SET statement are not reinitialized across DATA step iterations.



## Multiple SET Statements

```
data catalogcustomers;
 set orion.catalog;
 set orion.customerdimmore;
run;
```

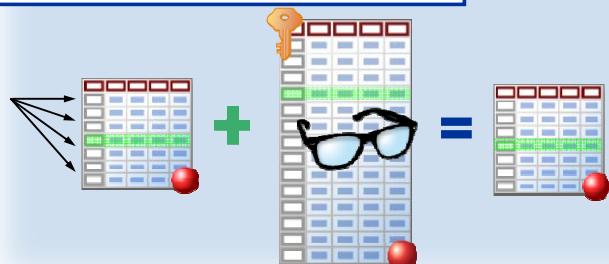
```
DATA data-set-name;
 SET SAS-data-set;
 SET SAS-data-set;
RUN;
```



## Multiple SET Statements

By itself, this syntax does not perform a table lookup.

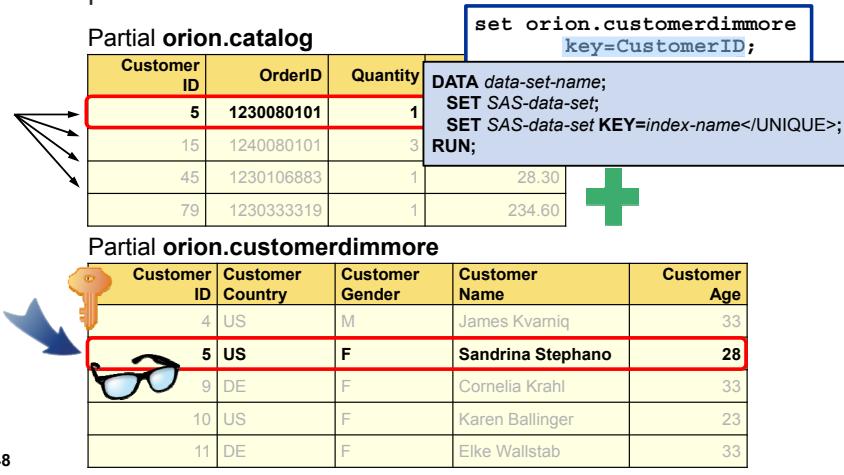
```
data catalog.customers;
 set orion.catalog;
 set orion.customerdimmore;
run;
```



47

## Multiple SET Statements and the KEY= Option

The KEY= option changes the reading of data from sequential to direct access.

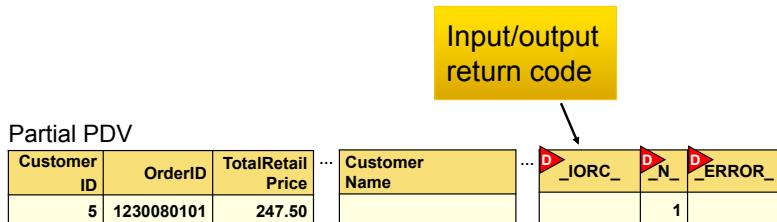


48

## Using the \_IORC\_ Automatic Variable

When the KEY= option is used, SAS creates the temporary variable `_IORC_` to hold the input/output return code. It indicates whether the indexed file has an entry for the current key value.

```
set orion.customerdimmore key=CustomerID;
```

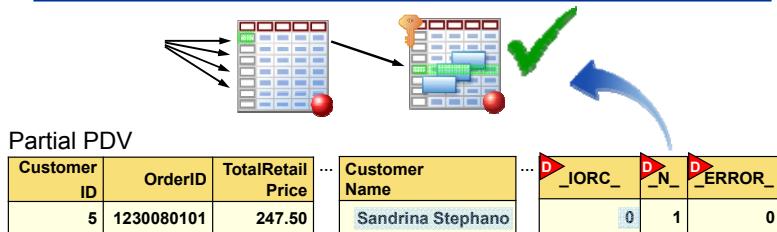


49

## Using the \_IORC\_ Automatic Variable

If the value of `_IORC_` is 0, SAS found an observation matching the current value in the KEY= index.

```
set orion.customerdimmore key=CustomerID;
```

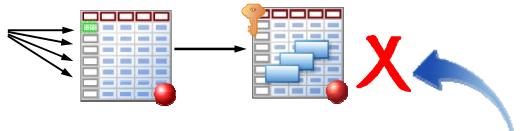


50

## Using the \_IORC\_ Automatic Variable

A nonzero `_IORC_` value indicates that SAS did not find an observation matching the current value in the KEY= index. A nonzero `_ERROR_` value indicates that a data error has occurred. Ulrich Heyde is not customer 66.

```
set orion.customerdimmore key=CustomerID;
```



Partial PDV

| Customer ID | OrderID    | TotalRetail Price | Customer Name | _IORC_  | _N_ | _ERROR_ |
|-------------|------------|-------------------|---------------|---------|-----|---------|
| 66          | 1220106883 | 128.15            | Ulrich Heyde  | 1230015 | 7   | 1       |

51

## 7.06 Quiz

Open and submit the program p307a01. Check the log.

If the **CustomerID** value 15 was not found in the index, from where is the **CustomerName** value coming?

Partial SAS Log

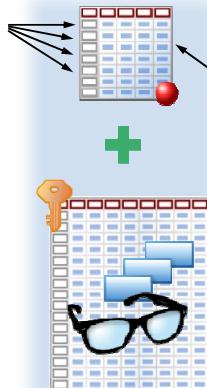
```
81 data catalogcustomers (keep=CustomerID OrderID Quantity TotalRetailPrice
82 CustomerCountry CustomerGender CustomerName
83 CustomerAge);
84 errors(keep=CustomerID);
85 set orion.catalog(keep=CustomerID OrderID
86 Quantity TotalRetailPrice);
87 set orion.customerdimmore key=CustomerID;
88 if _IORC_=0 then output catalogcustomers;
89 else output errors;
90 run;

CustomerID=15 OrderID=1240080101 Quantity=3 TotalRetailPrice=$216.50
CustomerCountry=US CustomerGender=F CustomerName=Sandrina Stephano
CustomerFirstName=Sandrina CustomerLastName=Stephano CustomerBirthDate=09JUL1983
CustomerAgeGroup=15-30 years CustomerType=Orion Club Gold members medium activity
CustomerGroup=Orion Club Gold members CustomerAge=28 _ERROR_=1 _IORC_=1230015 N_=2
```

52

## Using the \_IORC\_ Automatic Variable

Use `_IORC_` to control output.



```
data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
 set orion.catalog (keep=CustomerID OrderID
 Quantity TotalRetailPrice);
 set orion.customerdimmore key=CustomerID;
 if _IORC_=0 then output catalogcustomers;
 else do;
 ERROR=0;
 output errors;
 end;
run;
```

54

p307d07

## Execution

`orion.catalog (obs=2)`

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```
data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
 set orion.catalog (keep=CustomerID OrderID
 Quantity TotalRetailPrice);
 set orion.customerdimmore key=CustomerID;
 if _IORC_=0 then output catalogcustomers;
 else do;
 ERROR=0;
 output errors;
 end;
run;
```

Initialize the PDV.

Simplified Index on  
`orion.customerdimmore`

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

Partial PDV

| Customer ID | OrderID | TotalRetail Price | ... | Customer Name | ... | D_IORC_ | D_N_ | D_ERROR_ |
|-------------|---------|-------------------|-----|---------------|-----|---------|------|----------|
| .           | .       | .                 | .   | .             | ... | 0       | 1    | 0        |

55

p307d07

...

## Execution

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

Partial PDV

| Customer ID | OrderID    | TotalRetail Price | Customer Name | D_IORC_ | D_N_ | D_ERROR_ |
|-------------|------------|-------------------|---------------|---------|------|----------|
| 5           | 1230080101 | 247.50            |               | 0       | 1    | 0        |

56 p307d07 ...

Simplified Index on **orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |
| .           | .                  |

## Execution

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

Check the index for the current key value.

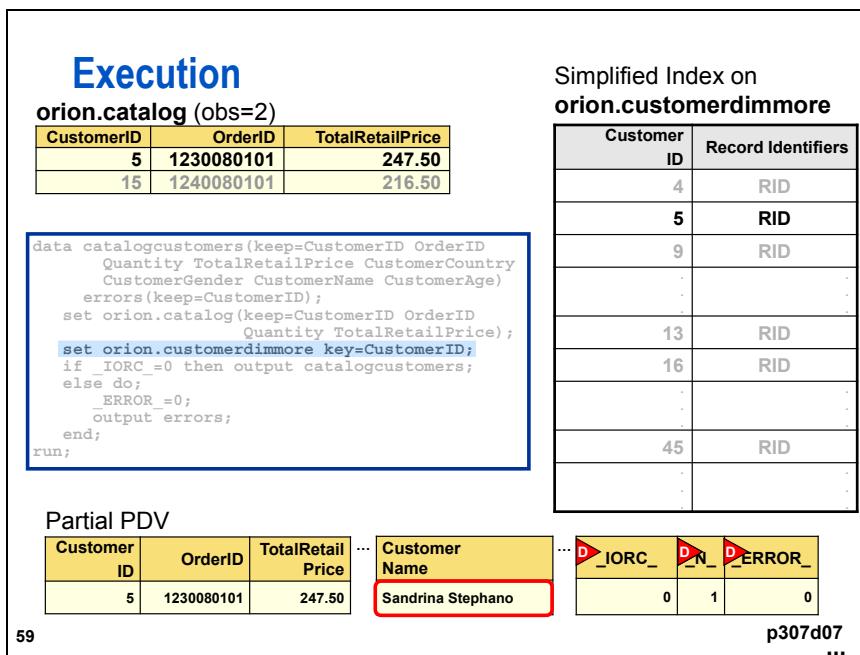
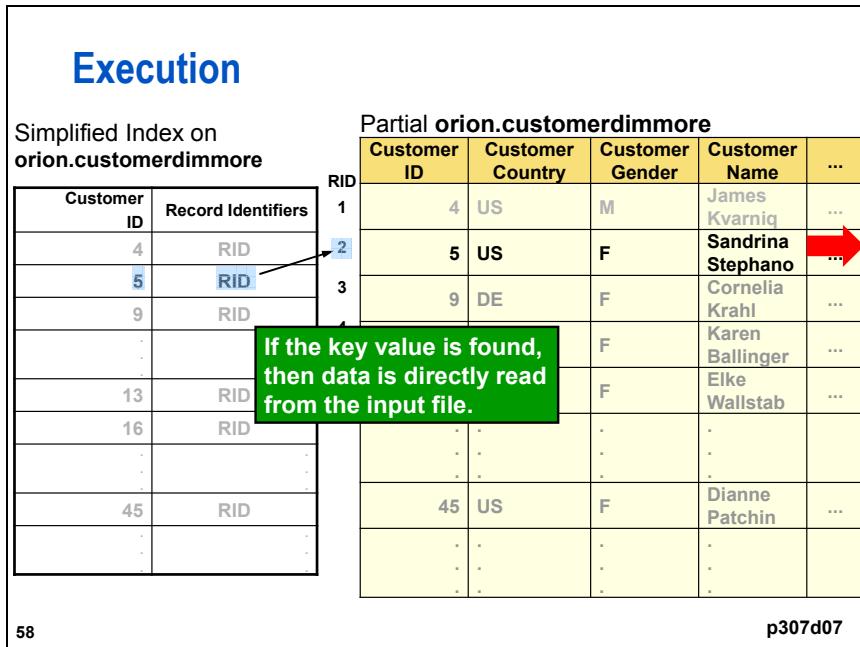
Partial PDV

| Customer ID | OrderID    | TotalRetail Price | Customer Name | D_IORC_ | D_N_ | D_ERROR_ |
|-------------|------------|-------------------|---------------|---------|------|----------|
| 5           | 1230080101 | 247.50            |               | 0       | 1    | 0        |

57 p307d07 ...

Simplified Index on **orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |
| .           | .                  |



**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

**True, output to work.catalogcustomers.**

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | _IORC_ | _N_ | _ERROR_ |
|-------------|------------|-------------------|-------------------|--------|-----|---------|
| 5           | 1230080101 | 247.50            | Sandrina Stephano | 0      | 1   | 0       |

60 p307d07 ...

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
 errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

**Implicit return**

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | _IORC_ | _N_ | _ERROR_ |
|-------------|------------|-------------------|-------------------|--------|-----|---------|
| 5           | 1230080101 | 247.50            | Sandrina Stephano | 0      | 1   | 0       |

61 p307d07 ...

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

Reinitialize PDV.

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | D_IORC_ | D_N_ | D_ERROR_ |
|-------------|------------|-------------------|-------------------|---------|------|----------|
| 5           | 1230080101 | 247.50            | Sandrina Stephano | 0       | 2    | 0        |

62 p307d07 ...

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

Read the second observation from orion.catalog.

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | D_IORC_ | D_N_ | D_ERROR_ |
|-------------|------------|-------------------|-------------------|---------|------|----------|
| 15          | 1240080101 | 216.50            | Sandrina Stephano | 0       | 2    | 0        |

63 p307d07 ...

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| ...         | ...                |
| 13          | RID                |
| 16          | RID                |
| ...         | ...                |
| 45          | RID                |
| ...         | ...                |

**CustomerID value 15 is not found in the index.**

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price |
|-------------|------------|-------------------|
| 15          | 1240080101 | 216.50            |

| Customer Name     | D_IORC_ | D_N_ | D_ERROR_ |
|-------------------|---------|------|----------|
| Sandrina Stephano | 1230015 | 2    | 1        |

64 p307d07 ...

## 7.07 Quiz

Edit the program **p307a01**.

- Replace the ELSE statement with the following ELSE DO group:

```
else do;
 ERROR=0;
 output errors;
end;
```

- Resubmit the program and look at the log.  
Why are there no messages now?

65

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Suppress the data error message. Output to work.errors.**

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | _IORC_  | _N_ | _ERROR_ |
|-------------|------------|-------------------|-------------------|---------|-----|---------|
| 15          | 1240080101 | 216.50            | Sandrina Stephano | 1230015 | 2   | 0       |

67 p307d07 ...

**Execution**

**orion.catalog (obs=2)**

| CustomerID | OrderID    | TotalRetailPrice |
|------------|------------|------------------|
| 5          | 1230080101 | 247.50           |
| 15         | 1240080101 | 216.50           |

```

data catalogcustomers(keep=CustomerID OrderID
 Quantity TotalRetailPrice CustomerCountry
 CustomerGender CustomerName CustomerAge)
errors(keep=CustomerID);
set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
set orion.customerdimmore key=CustomerID;
if _IORC_=0 then output catalogcustomers;
else do;
 ERROR=0;
 output errors;
end;
run;

```

**Simplified Index on orion.customerdimmore**

| Customer ID | Record Identifiers |
|-------------|--------------------|
| 4           | RID                |
| 5           | RID                |
| 9           | RID                |
| .           | .                  |
| 13          | RID                |
| 16          | RID                |
| .           | .                  |
| 45          | RID                |
| .           | .                  |

**Continue until EOF in orion.catalog.**

**Partial PDV**

| Customer ID | OrderID    | TotalRetail Price | Customer Name     | _IORC_  | _N_ | _ERROR_ |
|-------------|------------|-------------------|-------------------|---------|-----|---------|
| 15          | 1240080101 | 216.50            | Sandrina Stephano | 1230015 | 2   | 0       |

68 p307d07 ...

## Reference Information

### Monitoring I/O Error Conditions

Use the automatic variable **\_IORC\_** with the %SYSRC autocall macro to test for specific I/O error conditions that are created when you use the KEY= option in the SET statement.

General form for using %SYSRC with **\_IORC\_**:

**IF \_IORC\_=%SYSRC(*mnemonic*) THEN...**

| Mnemonic | Meaning                                             |
|----------|-----------------------------------------------------|
| _DSENOM  | No matching observation was found.                  |
| _SOK     | The observation was located. _SOK has a value of 0. |

The %SYSRC macro is in the AUTOCALL library. You must have the MACRO system option in effect to use this macro. Consult SAS Online Documentation for more information. Follow the path shown below:

**Knowledge Base** ⇒ **Documentation** ⇒ **Product Index A-Z** ⇒ **Base SAS** ⇒ **SAS 9.4 Macro Language: Reference** ⇒ **Macro Language Dictionary** ⇒ **AutoCall Macros**

## Using the IORCMMSG Function

The IORCMMSG function returns the formatted error message that is associated with the current value of the automatic variable `_IORC_`.

General form of the IORCMMSG function:

*character-variable*=IORCMMSG();

- ✍ *Character-variable* specifies a character variable with a length of 200, unless the length was previously assigned.

Example:

### p307d07a

```
data catalogcustomers(keep=CustomerID OrderID TotalRetailPrice
 CustomerCountry CustomerGender CustomerName
 CustomerAge)
 errors(keep=CustomerID);
 set orion.catalog(keep=CustomerID OrderID
 Quantity TotalRetailPrice);
 set orion.customerdimmore key=CustomerID;
 if _IORC_=0 then output catalogcustomers;
 else do;
 output errors;
 Message=iorcmmsg();
 ERROR=0;
 putlog _N_ ' The problem is ' Message;
 end;
run;
```

## Multiple SET Statements with Duplicate Key Values

What happens if there are duplicate values for the key variable in the base or lookup data sets (or both)?

The results depend on whether the duplicates are contiguous.



69

## Multiple SET Statements with Duplicate Key Values

In this example, data set **one** has contiguous duplicates of the **CustomerID** value. Each has a match in data set **two**.

```
data three;
 set one;
 set two key=CustomerID;
run;
```

| one        |   | two        |   |
|------------|---|------------|---|
| CustomerID | X | CustomerID | Y |
| A          | 1 | A          | 1 |
| A          | 2 | A          | 2 |
| A          | 3 | A          | 3 |

One-to-one matches

70

p307d08

## Multiple SET Statements with Duplicate Key Values

Data set **one** has contiguous duplicates. Some do not have a separately occurring match in data set **two**.

```
data three;
 set one;
 set two key=CustomerID;
run;
```

**one**

| CustomerID | X |
|------------|---|
| A          | 1 |
| A          | 2 |
| A          | 3 |

**two**

| CustomerID | Y |
|------------|---|
| A          | 1 |
| A          | 2 |

```
customerid=A x=3 y=2 _ERROR_=1 _IORC_=1230015 _N_=3
NOTE: There were 3 observations read from the data set WORK.ONE.
NOTE: The data set WORK.THREE has 3 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

71

p307d08

## Multiple SET Statements with Duplicate Key Values

The UNIQUE option forces SAS to ignore duplicates.

```
data three;
 set one;
 set two key=CustomerID/unique;
run;
```

**one**

| CustomerID | X |
|------------|---|
| A          | 1 |
| A          | 2 |
| A          | 3 |

**two**

| CustomerID | Y |
|------------|---|
| A          | 1 |
| A          | 2 |
| B          | 3 |

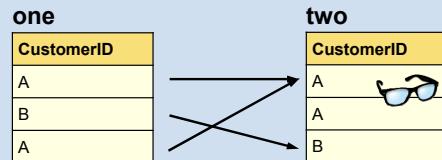
72

p307d08

## Multiple SET Statements with Duplicate Key Values

The non-contiguous duplicates in data set **one** match only the first occurrence in data set **two**.

```
data three;
 set one;
 set two key=CustomerID;
run;
```



73

p307d08

## Using Two SET Statements with the KEY= Option

| Advantages                                                                     | Disadvantages                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Only the necessary observations are read.                                      | An index on one data set is required.                                                                                                                                                                                              |
| An existing index is used.                                                     | Creating and maintaining an index uses resources.                                                                                                                                                                                  |
| <code>_IORC_</code> can be used to control how nonmatching data is handled.    | When the indexed data set is not sorted by the key variables, there can be considerable increase in I/O. This increase in I/O is because of the random access of the data set and the additional I/O required to access the index. |
| The availability of DATA step syntax provides the full power of the DATA step. |                                                                                                                                                                                                                                    |

## Comparing SET/SET KEY=, Merging, and SQL

| Match-Merge                                                                                         | SQL Inner Join                                                                                      | SET/SET KEY=                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| There is no limit to the number of data sets or the size of the data sets other than disk space.    | The maximum number of tables that can be joined at one time is 256.                                 | There is no limit to the number of data sets or the size of the data sets other than disk space.                                                                                                                                                                                                  |
| Data is processed sequentially so that observations with duplicate BY values are joined one-to-one. | Data is processed using a Cartesian product for duplicate BY values.                                | The data set listed in the first SET statement is read sequentially. The data sets in the second and subsequent SET statements are processed via the index. Observations in the first data set with duplicates are treated differently depending on whether the duplicates are contiguous or not. |
| Multiple data sets can be created.                                                                  | Only one data set can be created with one CREATE TABLE statement.                                   | Multiple data sets can be created.                                                                                                                                                                                                                                                                |
| Complex business logic can be incorporated using IF-THEN or SELECT/WHEN logic.                      | The CASE clause can be used for business logic. However, it is not as flexible as DATA step syntax. | Complex business logic can be incorporated using IF-THEN or SELECT/WHEN logic.                                                                                                                                                                                                                    |
| The data sets being merged must be sorted or indexed on the BY variable (or variables).             | The data sets being joined do not have to be sorted or indexed.                                     | The data sets in all but the first SET statement must have the index named on the KEY= option.                                                                                                                                                                                                    |
| An exact match on the value (or values) of the BY variable (or variables) must be found.            | Inequality joins can be performed.                                                                  | An exact match on the key value is required.                                                                                                                                                                                                                                                      |
| Like-named BY variables must be available in all data sets.                                         | Common variables are not required to be in all data sets.                                           | The indexed variable (or variables) must be in all data sets.                                                                                                                                                                                                                                     |



## Using Multiple SET Statements with KEY= Options

**p307d09**

This demonstration illustrates using multiple SET statements with the KEY= option.

You can use multiple SET statements with the KEY= option to combine several data sets. For example, create a data set that contains the customers who ordered products from the Internet and from the catalog.

1. Open the program **p307d09**.

2. Submit the first PROC SQL step, which creates the **CustomerID** index on the **orion.customerdim** data set.



The **CustomerID** index on the **orion.customerdim** data set is created for this example. It is deleted at the end of the program.

#### Partial p307d09

```
proc sql;
 create index CustomerID
 on orion.customerdim(CustomerID);
quit;
```

3. Examine the SAS log to verify that the step ran without errors and created the **CustomerID** index.

#### Partial SAS Log

```
1 proc sql;
2 create index CustomerID
3 on orion.customerdim(CustomerID);
NOTE: Simple index CustomerID has been defined.
4 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.49 seconds
 cpu time 0.14 seconds
```

4. Examine the DATA step that creates **work.cataloginternet** and **work.others**. The following points are of special interest:

- The data set **orion.catalog** is read sequentially.
- The data set **orion.customerdim** is read using the index on **CustomerID**. The **IORC\_DIM** variable is created to track whether the key variable value is found in the **orion.customerdim** index.
- The data set **orion.internet** is read using the index on **CustomerID**. The **IORC\_INT** variable is created to track whether the key variable value is found in the **orion.internet** index. The **OrderID**, **TotalRetailPrice**, and **Quantity** variables are renamed so that both the values from **orion.catalog** and **orion.internet** are in the **work.cataloginternet** data set. Without the renaming, the values from **orion.internet** would overwrite the like-named values from **orion.catalog**.

#### Partial p307d09

```
data cataloginternet catalognoncust catalogcust;
 keep CustomerID OrderID Quantity TotalRetailPrice CustomerName
 IntOrderID IntTotPrice IntQuant IORC_DIM IORC_INT;
 set orion.catalog(keep=CustomerID OrderID Quantity
 TotalRetailPrice in=InCat);
 set orion.customerdim key=CustomerID;
 IORC_DIM=_IORC_;
 set orion.internet(rename=(OrderID=IntOrderID
 TotalRetailPrice=IntTotPrice
 Quantity=IntQuant))
 key=CustomerID;
 IORC_INT=_IORC_;
 if IORC_DIM=0 and IORC_INT=0 then output cataloginternet;
 else if IORC_DIM ne 0 and IORC_INT ne 0 then do;
 ERROR_=0;
```

```

 output catalognoncust;
 end;
 else do;
 ERROR=0;
 output catalogcust;
 end;
run;

```

5. Submit the DATA step.
6. Examine the SAS log to determine that the **work.cataloginternet**, **work.catalognoncust**, and **work.catalogcust** data sets were created appropriately.

#### Partial SAS Log

```

8 data cataloginternet catalognoncust catalogcust;
9 keep CustomerID OrderID Quantity TotalRetailPrice CustomerName IntOrderID IntTotPrice
9! IntQuant
10 IORC_DIM IORC_INT;
11 set orion.catalog(keep=CustomerID OrderID Quantity TotalRetailPrice in=InCat);
12 set orion.customerdim key=CustomerID;
13 IORC_DIM=_IORC_;
14 set orion.internet(rename=(OrderID=IntOrderID TotalRetailPrice=IntTotPrice
14! Quantity=IntQuant))
15 key=CustomerID;
16 IORC_INT=_IORC_;
17 if IORC_DIM=0 and IORC_INT=0 then output cataloginternet;
18 else if IORC_DIM ne 0 and IORC_INT ne 0 then do;
19 _ERROR_=0;
20 output catalognoncust;
21 end;
22 else do;
23 _ERROR_=0;
24 output catalogcust;
25 end;
26 run;

NOTE: There were 38 observations read from the data set ORION.CATALOG.
NOTE: The data set WORK.CATALOGINTERNET has 6 observations and 10 variables.
NOTE: The data set WORK.CATALOGNONCUST has 2 observations and 10 variables.
NOTE: The data set WORK.CATALOGCUST has 30 observations and 10 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds

```



There are no data errors in the SAS log because the ELSE DO groups reset **\_ERROR\_** to 0.

7. Print the first five observations from the three new data sets. Use the variable labels that are stored in the data sets.

#### Partial p307d09

```

proc print data=cataloginternet(obs=5) noobs label;
 title 'Customers Who Ordered from Both Catalog and Internet';
run;
proc print data=catalognoncust(obs=5) noobs label;
 title 'Catalog Only Orders with no link to Customer Database';

```

```

run;
proc print data=catalogcust(obs=5) noobs label;
 title 'Catalog Only Orders from existing Customers';
run;

```

8. Examine the SAS log to verify that no errors were encountered.

#### Partial SAS Log

```

166
167 proc print data=cataloginternet(obs=5) noobs label;
168 title 'Customers Who Ordered from Both Catalog and Internet';
169 run;

NOTE: There were 5 observations read from the data set WORK.CATALOGINTERNET.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.28 seconds
 cpu time 0.10 seconds

170 proc print data=catalognoncust(obs=5) noobs label;
171 title 'Catalog Only Orders with no link to Customer Database';
172 run;

NOTE: There were 2 observations read from the data set WORK.CATALOGNONCUST.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

173 proc print data=catalogcust(obs=5) noobs label;
174 title 'Catalog Only Orders from existing Customers';
175 run;

NOTE: There were 5 observations read from the data set WORK.CATALOGCUST.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

#### PROC PRINT Output

| Customers Who Ordered from Both Catalog and Internet |            |                  |                              |                   |
|------------------------------------------------------|------------|------------------|------------------------------|-------------------|
| Customer ID                                          | Order ID   | Quantity Ordered | Total Price for This Product | Customer Name     |
| 5                                                    | 1230080101 | 1                | \$247.50                     | Sandrina Stephano |
| 79                                                   | 1230333319 | 1                | \$234.60                     | Najma Hicks       |
| 24                                                   | 1234958242 | 1                | \$11.90                      | Robyn Klem        |
| 41                                                   | 1238161695 | 2                | \$17.00                      | Wendell Summersby |
| 53                                                   | 1239972644 | 1                | \$102.90                     | Dericka Pockran   |
| Total Retail                                         |            |                  |                              |                   |
| IORC_DIM                                             | Order ID   | Quantity Ordered | Price for This Product       | IORC_INT          |
| 0                                                    | 1242140006 | 1                | \$31.40                      | 0                 |
| 0                                                    | 1235926178 | 2                | \$36.00                      | 0                 |
| 0                                                    | 1241054779 | 2                | \$195.60                     | 0                 |
| 0                                                    | 1232723799 | 1                | \$21.80                      | 0                 |
| 0                                                    | 1241086052 | 3                | \$37.80                      | 0                 |

| Catalog Only Orders with no link to Customer Database |            |                  |                                     |                   |  |
|-------------------------------------------------------|------------|------------------|-------------------------------------|-------------------|--|
| Customer ID                                           | Order ID   | Quantity Ordered | Total Retail Price for This Product |                   |  |
|                                                       |            |                  | Customer Name                       |                   |  |
| 15                                                    | 1240080101 | 3                | \$216.50                            | Sandrina Stephano |  |
| 66                                                    | 1220106883 | 2                | \$128.15                            | Ulrich Heyde      |  |
| IORC_DIM                                              | Order ID   | Quantity Ordered | Total Retail Price for This Product |                   |  |
|                                                       |            |                  | Iorc_INT                            |                   |  |
| 1230015                                               | 1242140006 | 1                | \$31.40                             | 1230015           |  |
| 1230015                                               | 1235926178 | 2                | \$36.00                             | 1230015           |  |
| Catalog Only Orders from existing Customers           |            |                  |                                     |                   |  |
| Customer ID                                           | Order ID   | Quantity Ordered | Total Retail Price for This Product |                   |  |
|                                                       |            |                  | Customer Name                       |                   |  |
| 45                                                    | 1230106883 | 1                | \$28.30                             | Dianne Patchin    |  |
| 23                                                    | 1230338566 | 1                | \$35.40                             | Tulio Devereaux   |  |
| 16                                                    | 1230450371 | 2                | \$128.40                            | Ulrich Heyde      |  |
| 2788                                                  | 1230478006 | 2                | \$60.60                             | Serdar Yucel      |  |
| 12386                                                 | 1230503155 | 1                | \$31.80                             | Avinoam Zweig     |  |
| IORC_DIM                                              | Order ID   | Quantity Ordered | Total Retail Price for This Product |                   |  |
|                                                       |            |                  | Iorc_INT                            |                   |  |
| 0                                                     | 1242140006 | 1                | \$31.40                             | 1230015           |  |
| 0                                                     | 1235926178 | 2                | \$36.00                             | 1230015           |  |
| 0                                                     | 1235926178 | 2                | \$36.00                             | 1230015           |  |
| 0                                                     | 1235926178 | 2                | \$36.00                             | 1230015           |  |
| 0                                                     | 1235926178 | 2                | \$36.00                             | 1230015           |  |

9. Submit the final PROC SQL step to drop the index **CustomerID** from **orion.customerdim**.

```
proc sql;
 drop index CustomerID
 from orion.customerdim;
quit;
```

10. Examine the SAS log to verify that the index was dropped.

Partial SAS Log

```
166 proc sql;
167 drop index CustomerID
168 from orion.customerdim;
NOTE: Index CustomerID has been dropped.
169 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.03 seconds
```

|          |              |
|----------|--------------|
| cpu time | 0.00 seconds |
|----------|--------------|



## Exercises

---

### Level 1

#### 4. Combining Data Sets Using an Index to Create One Data Set

The SAS data set **orion.salesstaff** contains information about the employees who work in Sales.

Partial **orion.salesstaff**

| orion.salesstaff SAS Data Set |                |          |        |           |              |
|-------------------------------|----------------|----------|--------|-----------|--------------|
| EmployeeID                    | JobTitle       | Salary   | Gender | BirthDate | EmpHire Date |
| 120121                        | Sales Rep. II  | \$26,600 | F      | 02AUG1948 | 01JAN1978    |
| 120134                        | Sales Rep. II  | \$28,015 | M      | 06JUN1953 | 01JAN1978    |
| 120151                        | Sales Rep. II  | \$26,520 | F      | 21NOV1948 | 01JAN1978    |
| 120154                        | Sales Rep. III | \$30,490 | F      | 20JUL1948 | 01JAN1978    |
| 120166                        | Sales Rep. IV  | \$30,660 | M      | 14JUN1948 | 01JAN1978    |

| EmpTerm Date | ManagerID | SSN         | EmployeeName           |
|--------------|-----------|-------------|------------------------|
| .            | 120102    | 42-8321-982 | Elvish, Irenie         |
| 30JUN2010    | 120102    | 905-76-7767 | Shannan, Sian          |
| .            | 120103    | 798-16-4924 | Phaiyakounh, Julianna  |
| .            | 120102    | 534-14-1428 | Hayawardhana, Caterina |
| 31AUG2010    | 120102    | 878-79-9390 | Nowd, Fadi             |

The SAS data set **orion.organizationdim** contains information about all employees. There are no indexes on any variables.

Partial **orion.organizationdim**

| orion.organizationdim SAS Data Set |         |                 |                  |                  |                  |  |
|------------------------------------|---------|-----------------|------------------|------------------|------------------|--|
| EmployeeID                         | Country | Company         | Department       | Section          | OrgGroup         |  |
| 120101                             | AU      | Orion Australia | Sales Management | Sales Management | Sales Management |  |
| 120102                             | AU      | Orion Australia | Sales Management | Sales Management | Sales Management |  |
| 120103                             | AU      | Orion Australia | Sales Management | Sales Management | Sales Management |  |
| 120104                             | AU      | Orion Australia | Administration   | Administration   | Administration   |  |
| 120105                             | AU      | Orion Australia | Administration   | Administration   | Administration   |  |

| JobTitle               | EmployeeName      | Gender | Salary    | BirthDate | HireDate  |
|------------------------|-------------------|--------|-----------|-----------|-----------|
| Director               | Patrick Lu        | M      | \$163,040 | 18AUG1980 | 01JUL2007 |
| Sales Manager          | Tom Zhou          | M      | \$108,255 | 11AUG1973 | 01JUN1993 |
| Sales Manager          | Wilson Dawes      | M      | \$87,975  | 22JAN1953 | 01JAN1978 |
| Administration Manager | Kareen Billington | F      | \$46,230  | 11MAY1958 | 01JAN1985 |

| Secretary I       |                | Liz Povey      |                | F              | \$27,110       | 21DEC1978      | 01MAY2003      |
|-------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Employee TermDate | Manager Levels | Manager Level1 | Manager Level2 | Manager Level3 | Manager Level4 | Manager Level5 | Manager Level6 |
| .                 | 2              | 120261         | 120259         | .              | .              | .              | .              |
| .                 | 3              | 120101         | 120261         | 120259         | .              | .              | .              |
| .                 | 3              | 120101         | 120261         | 120259         | .              | .              | .              |
| .                 | 3              | 120101         | 120261         | 120259         | .              | .              | .              |
| .                 | 3              | 120101         | 120261         | 120259         | .              | .              | .              |

- a. Modify **p307e04** to create a SAS data set named **salesemps**. Use the index on **EmployeeID** to combine the two data sets, **orion.salesstaff** and **orion.organizationdim**. Read only the variables **EmployeeID**, **Department**, **Section**, and **OrgGroup** from **orion.organizationdim**.
- b. Check the SAS log to ensure that you do not have any data errors.
- c. Print the first five observations of the **salesemps** SAS data set.

PROC PRINT Output

| Sales Employee Data<br>(First 5 Observations) |                        |          |        |            |              |                          |           |
|-----------------------------------------------|------------------------|----------|--------|------------|--------------|--------------------------|-----------|
| EmployeeID                                    | JobTitle               | Salary   | Gender | BirthDate  | EmpHire Date | EmpTerm Date             | ManagerID |
| 120121                                        | Sales Rep. II          | \$26,600 | F      | 02AUG1948  | 01JAN1978    | .                        | 120102    |
| 120134                                        | Sales Rep. II          | \$28,015 | M      | 06JUN1953  | 01JAN1978    | 30JUN2010                | 120102    |
| 120151                                        | Sales Rep. II          | \$26,520 | F      | 21NOV1948  | 01JAN1978    | .                        | 120103    |
| 120154                                        | Sales Rep. III         | \$30,490 | F      | 20JUL1948  | 01JAN1978    | .                        | 120102    |
| 120166                                        | Sales Rep. IV          | \$30,660 | M      | 14JUN1948  | 01JAN1978    | 31AUG2010                | 120102    |
| SSN                                           | EmployeeName           |          |        | Department | Section      | OrgGroup                 |           |
| 42-8321-982                                   | Elvish, Irenie         |          |        | Sales      | Sales        | Assorted Sports Articles |           |
| 905-76-7767                                   | Shannan, Sian          |          |        | Sales      | Sales        | Golf                     |           |
| 798-16-4924                                   | Phaiyakounh, Julianna  |          |        | Sales      | Sales        | Outdoors                 |           |
| 534-14-1428                                   | Hayawardhana, Caterina |          |        | Sales      | Sales        | Racket Sports            |           |
| 878-79-9390                                   | Nowd, Fadi             |          |        | Sales      | Sales        | Running - Jogging        |           |

## Level 2

### 5. Combining Data Sets Using an Index to Monitor Data Integrity

The data set **orion.shoevendors** contains information about the shoe products and their suppliers. There are vendors in **orion.shoevendors** whose products are not in the Orion Star price list, which should contain all prices.

Partial **orion.shoevendors**

| orion.shoevendors Data Set |                  |                      |                 |                                        |                         |  |
|----------------------------|------------------|----------------------|-----------------|----------------------------------------|-------------------------|--|
| Product Line               | Product Category | Product Group        | ProductID       | ProductName                            | SupplierID              |  |
| 21                         | 2102             | 2102004              | 210200400002    | Deschutes Boys Outdoors Training Shoes | 1303                    |  |
| 21                         | 2102             | 2102004              | 210200400005    | Kid Air Terra Grande Running Shoes     | 1303                    |  |
| 21                         | 2102             | 2102004              | 210200400007    | Kid Equivalent Street Shoes            | 1303                    |  |
| 21                         | 2102             | 2102004              | 210200400009    | Kid Impeccably Strong(Bg) Basket Shoes | 1303                    |  |
| 21                         | 2102             | 2102004              | 210200400012    | Kid Trainer Lite V(Bp) Street Shoes    | 1303                    |  |
| Supplier                   |                  |                      |                 | Line                                   | MfgSuggestedRetailPrice |  |
| Name                       | Country          | GroupName            | CategoryName    | Name                                   | RetailPrice             |  |
| Eclipse Inc                | US               | Eclipse, Kid's Shoes | Children Sports | Children                               | \$69.00                 |  |
| Eclipse Inc                | US               | Eclipse, Kid's Shoes | Children Sports | Children                               | \$73.00                 |  |
| Eclipse Inc                | US               | Eclipse, Kid's Shoes | Children Sports | Children                               | \$79.00                 |  |
| Eclipse Inc                | US               | Eclipse, Kid's Shoes | Children Sports | Children                               | \$114.00                |  |
| Eclipse Inc                | US               | Eclipse, Kid's Shoes | Children Sports | Children                               | \$54.00                 |  |

The data set **orion.shoeprices** contains pricing information for all shoes.

Partial **orion.shoeprices**

| shoeprices Data Set |              |                  |                  |  |
|---------------------|--------------|------------------|------------------|--|
| Obs                 | ProductID    | TotalRetailPrice | CostPricePerUnit |  |
| 1                   | 210200400002 | \$41.80          | \$21.00          |  |
| 2                   | 210200400005 | \$97.80          | \$24.55          |  |
| 3                   | 210200400007 | \$108.90         | \$18.25          |  |
| 4                   | 210200400009 | \$56.50          | \$28.35          |  |
| 5                   | 210200400012 | \$33.40          | \$16.80          |  |

- a. Modify **p307e05** to create a SAS data set named **shoes** and a SAS data set named **errors**. Use an index on **ProductID** to combine the two data sets, **orion.shoevendors** and **orion.shoeprices**.

- Read only the variables **ProductID**, **ProductName**, **SupplierName**, and **MfgSuggestedRetailPrice** from **orion.shoevendors**.
- Read only the variables **ProductID**, **TotalRetailPrice**, and **CostPricePerUnit** from **orion.shoeprices**.
- The **shoes** data set should have the price information for the shoe products.



The data set **shoes** should have 357 observations.

- The **errors** data set should contain data that is in **orion.shoevendors**, but that is not in the **orion.shoeprices** data.
- The **errors** data set should contain only the variables **ProductID**, **ProductName**, and **SupplierName**.



The **errors** data set can be used to determine why these vendors do not have observations in **pricelist**. The data set **errors** should have four observations.

- b. Print the first five observations of the **shoes** SAS data set.

## PROC PRINT Output

| The shoes Data Set          |                                        |                      |
|-----------------------------|----------------------------------------|----------------------|
| First 5 Observations        |                                        |                      |
| ProductID                   | ProductName                            | Supplier             |
| 210200400002                | Deschutes Boys Outdoors Training Shoes | Eclipse Inc          |
| 210200400005                | Kid Air Terra Grande Running Shoes     | Eclipse Inc          |
| 210200400007                | Kid Equivalent Street Shoes            | Eclipse Inc          |
| 210200400009                | Kid Impeccably Strong(Bg) Basket Shoes | Eclipse Inc          |
| 210200400012                | Kid Trainer Lite V(Bp) Street Shoes    | Eclipse Inc          |
| MfgSuggested<br>RetailPrice | TotalRetail<br>Price                   | CostPrice<br>PerUnit |
| \$69.00                     | \$41.80                                | \$21.00              |
| \$73.00                     | \$97.80                                | \$24.55              |
| \$79.00                     | \$108.90                               | \$18.25              |
| \$114.00                    | \$56.50                                | \$28.35              |
| \$54.00                     | \$33.40                                | \$16.80              |

- c. Print the **errors** SAS data set.

## PROC PRINT Output

| The errors Data Set |                                             |             |
|---------------------|---------------------------------------------|-------------|
| ProductID           | ProductName                                 | Supplier    |
|                     |                                             | Name        |
| 210200400027        | Toddle Children's Air Mantra (3) (Bg) Shoes | Eclipse Inc |
| 210200400047        | Toddler Fit Shoes                           | Eclipse Inc |
| 210201000174        | Freestyle Children's Leather Street Shoes   | 3Top Sports |
| 220200100123        | Big Guy Men's Deschutz Slide Shoes          | Eclipse Inc |

- d. Submit the PROC Datasets step to delete the ProductID index.

**Challenge****6. Combining Data Sets Using an Index and Using the Macro Facility to Monitor Errors**

The data set **orion.firstinternetorder** contains the first order that a customer placed via the Internet.

**Partial orion.firstinternetorder**

| orion.firstinternetorder SAS Data Set |            |            |           |               |            |
|---------------------------------------|------------|------------|-----------|---------------|------------|
| CustomerID                            | EmployeeID | StreetID   | OrderDate | Delivery Date | OrderID    |
| 4                                     | 99999999   | 9260106519 | 02MAR2008 | 03MAR2008     | 1232410925 |
| 5                                     | 99999999   | 9260114570 | 02MAY2011 | 07MAY2011     | 1242140006 |
| 9                                     | 99999999   | 3940106659 | 15APR2008 | 20APR2008     | 1232698281 |
| 11                                    | 99999999   | 3940108592 | 29OCT2007 | 03NOV2007     | 1231653765 |
| 19                                    | 99999999   | 3940106547 | 26DEC2007 | 30DEC2007     | 1231976710 |

| ProductID    | Quantity | TotalRetail<br>Price | CostPrice<br>PerUnit | Discount |
|--------------|----------|----------------------|----------------------|----------|
| 240800200030 | 1        | \$47.70              | \$18.80              | .        |
| 240100100159 | 1        | \$31.40              | \$13.90              | .        |
| 230100600035 | 1        | \$29.40              | \$14.15              | .        |
| 230100200047 | 1        | \$72.70              | \$35.20              | .        |
| 240300100020 | 4        | \$56.40              | \$6.05               | .        |

The data set **orion.internet** contains multiple orders that a customer placed via the Internet. There is an index on the **OrderID** variable.

#### Partial **orion.internet**

| orion.internet SAS Data Set |            |            |           |               |            |
|-----------------------------|------------|------------|-----------|---------------|------------|
| CustomerID                  | EmployeeID | StreetID   | OrderDate | Delivery Date | OrderID    |
| 70046                       | 99999999   | 2600100017 | 02APR2007 | 03APR2007     | 1230500669 |
| 36                          | 99999999   | 9260128237 | 18APR2007 | 20APR2007     | 1230591675 |
| 171                         | 99999999   | 1600101555 | 01MAY2007 | 04MAY2007     | 1230657844 |
| 11171                       | 99999999   | 2600100032 | 07MAY2007 | 09MAY2007     | 1230690733 |
| 17023                       | 99999999   | 2600100021 | 20JUN2007 | 25JUN2007     | 1230931366 |

| ProductID    | Quantity | TotalRetail<br>Price | CostPrice<br>PerUnit | Discount |
|--------------|----------|----------------------|----------------------|----------|
| 240200100131 | 2        | \$148.60             | \$41.35              | .        |
| 240500100039 | 1        | \$34.50              | \$15.40              | .        |
| 240100100646 | 1        | \$109.90             | \$46.80              | .        |
| 240200100043 | 2        | \$282.40             | \$69.40              | .        |
| 240200200007 | 2        | \$166.80             | \$8.35               | .        |

- a. Create a data set named **processedorders** that contains the variables from **orion.firstinternetorder** and a variable named **Comment**. Use the index on the variable **OrderID** to retrieve the matching observation from **orion.internet**.
  - The new variable **Comment** has the value *Order has been processed* if the order ID is in both **orion.firstinternetorder** and **orion.internet**. The value is *Order has not been processed* if the order ID is not in both data sets.
  - Use the %SYSRC AUTOCALL macro that is described in the reference information in this chapter. In addition, refer to SAS documentation by following the path shown below:

**Knowledge Base**  $\Rightarrow$  **Documentation**  $\Rightarrow$  **Product Index A-Z**  $\Rightarrow$  **Base SAS**  $\Rightarrow$  **SAS 9.4 Macro Language: Reference**  $\Rightarrow$  **Macro Language Dictionary**  $\Rightarrow$  **AutoCall Macros**
- b. Print observations 6 through 10 of **processedorders**.

## PROC PRINT Output

| Internet Orders<br>Observations 6-10 |              |            |                   |                   |               |  |
|--------------------------------------|--------------|------------|-------------------|-------------------|---------------|--|
| Comment                              | CustomerID   | EmployeeID | StreetID          | OrderDate         | Delivery Date |  |
| Order has been processed.            | 20           | 99999999   | 9260118934        | 18MAY2010         | 24MAY2010     |  |
| Order has been processed.            | 24           | 99999999   | 9260115784        | 02JAN2011         | 05JAN2011     |  |
| Order has not been processed         | 25           | 99999999   | 9260114570        | 15JAN2011         | 19JAN2011     |  |
| Order has been processed.            | 27           | 99999999   | 9260105670        | 28JAN2011         | 02FEB2011     |  |
| Order has been processed.            | 31           | 99999999   | 9260128428        | 25APR2011         | 29APR2011     |  |
|                                      |              |            | TotalRetail Price | CostPrice PerUnit | Discount      |  |
| OrderID                              | ProductID    | Quantity   |                   |                   |               |  |
| 1239226632                           | 220200100190 | 3          | \$190.50          | \$29.95           | .             |  |
| 1241054779                           | 240800200021 | 2          | \$195.60          | \$42.45           | .             |  |
| 1230080101                           | 230100500026 | 1          | \$247.50          | \$109.55          | .             |  |
| 1241286432                           | 240800200009 | 2          | \$174.40          | \$34.90           | .             |  |
| 1242076538                           | 220200200022 | 1          | \$57.30           | \$33.90           | .             |  |

## 7.3 Combining Summary and Detail Data

---

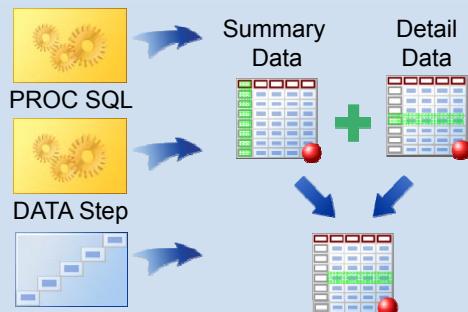
### Objectives

- Combine the summarized output SAS data set from PROC SUMMARY with a detail SAS data set.
- Use the SQL procedure to combine summary and detail data.
- Use the SQL procedure to calculate the summary statistic and combine it with every observation in the data set.
- Use the DATA step to calculate the summary statistic and combine it with every observation in the data set.

## Introduction

This section explores how to combine summary and detail data using PROC SUMMARY, PROC SQL, and the DATA step.

### PROC SUMMARY



79

## Business Scenario

Human Resources needs to analyze the salaries of Orion Star employees for an annual salary survey. The data set **orion.totalsalaries** has one observation for every value of **ManagerID**.

Partial **orion.totalsalaries**

| Manager ID | Numemps | DeptSal     |
|------------|---------|-------------|
| 120101     | 4       | \$269,570   |
| 120102     | 48      | \$1,344,595 |
| 120103     | 30      | \$793,835   |
| 120104     | 15      | \$425,215   |
| 120259     | 6       | \$941,155   |
| 120260     | 3       | \$216,065   |
| 120261     | 6       | \$595,935   |
| 120262     | 10      | \$545,255   |



80

## Business Scenario

Use **orion.totalsalaries** to calculate percentages of the total company payroll for each manager. This requires a few steps.

Partial **orion.totalsalaries**

| Manager ID | Numemps | DeptSal     | GrandTot     | Percent |
|------------|---------|-------------|--------------|---------|
| 120101     | 4       | \$269,570   | \$15,695,800 | 1.72%   |
| 120102     | 48      | \$1,344,595 | \$15,695,800 | 8.57%   |
| 120103     | 30      | \$793,835   | \$15,695,800 | 5.06%   |
| 120104     | 15      | \$425,215   | \$15,695,800 | 2.71%   |
| 120259     | 6       | \$941,155   | \$15,695,800 | 6.00%   |
| 120260     | 3       | \$216,065   | \$15,695,800 | 1.38%   |
| 120261     | 6       | \$595,935   | \$15,695,800 | 3.80%   |
| 120262     | 10      | \$545,255   | \$15,695,800 | 3.47%   |

81

## Business Scenario

Calculate the grand total of **DeptSal** and store it in a summary data set.

Partial **orion.totalsalaries**

| Manager ID | Numemps | DeptSal     |
|------------|---------|-------------|
| 120101     | 4       | \$269,570   |
| 120102     | 48      | \$1,344,595 |
| 120103     | 30      | \$793,835   |
| 120104     | 15      | \$425,215   |
| 120259     | 6       | \$941,155   |
| 120260     | 3       | \$216,065   |
| 120261     | 6       | \$595,935   |
| 120262     | 10      | \$545,255   |

Detail Data Set



Summary Data Set



$$= \$15,695,800$$

82

## Business Scenario

Combine the summary data with each row of the detailed data. Divide each detail amount by the grand total to calculate the percent of payroll by manager.

Partial `orion.totalsalaries`

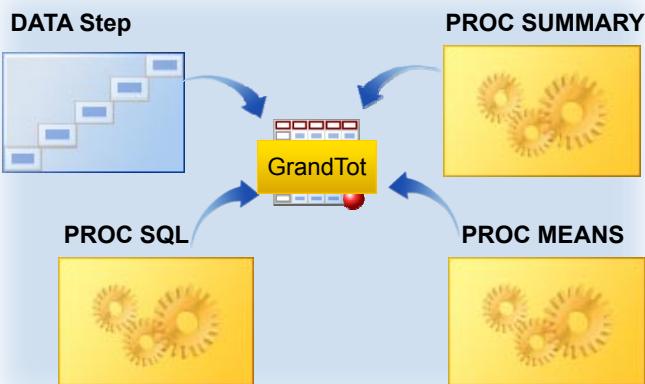
| Manager ID | Numemps | DeptSal     |
|------------|---------|-------------|
| 120101     | 4       | \$269,570   |
| 120102     | 48      | \$1,344,595 |
| 120103     | 30      | \$793,835   |
| 120104     | 15      | \$425,215   |
| 120259     | 6       | \$941,155   |
| 120260     | 3       | \$216,065   |
| 120261     | 6       | \$595,935   |
| 120262     | 10      | \$545,255   |

| GrandTot     | Percent |
|--------------|---------|
| \$15,695,800 | 1.72%   |
| \$15,695,800 | 8.57%   |
| \$15,695,800 | 5.06%   |
| \$15,695,800 | 2.71%   |
| \$15,695,800 | 6.00%   |
| \$15,695,800 | 1.38%   |
| \$15,695,800 | 3.80%   |
| \$15,695,800 | 3.47%   |

83

## Creating a Summary Data Set

There are several ways to create the summary data set.



84

## Creating a Summary Data Set

PROC SUMMARY generates descriptive statistics.  
The OUTPUT statement with the OUT= option creates  
a SAS data set with the summary statistics.

```
proc summary data=orion.totalsalaries;
 var DeptSal;
 output out=summary sum=GrandTot;
run;
```

```
PROC SUMMARY <option(s)><statistic-keyword(s)>;
 VAR variable(s);
 OUTPUT <OUT=SAS-data-set>
 <output-statistic-specification(s)>;
RUN;
```

work.summary

GrandTot

85

p307d10

The output data set has variables that contain the requested statistics, plus these variables:

|               |                                                    |
|---------------|----------------------------------------------------|
| <u>_TYPE_</u> | Information about the level of the class variables |
|---------------|----------------------------------------------------|

|               |                                                        |
|---------------|--------------------------------------------------------|
| <u>_FREQ_</u> | Number of observations that an output level represents |
|---------------|--------------------------------------------------------|

## Combining the Summary and Detail Data

Use two SET statements in the DATA step to combine  
the summary and detailed data.

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

SET summary  
data

GrandTot  
is retained.

SET detail  
data

Calculate  
percentage

p307d10

86

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

Initialize the PDV.

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| .        | .          | .        | .       | .       | 1   |

87

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

True. Read first observation from work.summary.

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | .          | .        | .       | .       | 1   |

88

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

Read first observation from orion.totalsalaries.

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120101     | 4        | 269570  | .       | 1   |

89

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

Calculate Percent.

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120101     | 4        | 269570  | 0.0172  | 1   |

90

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

Implicit OUTPUT;  
Implicit RETURN;

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120101     | 4        | 269570  | 0.0172  | 1   |

91

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

Re-initialize the PDV.

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120101     | 4        | 269570  | .       | 2   |

92

p307d10

## Execution

```

data percent;
if _N_=1 then
 set summary(keep=GrandTot);
set orion.totalsalaries;
Percent=DeptSal/GrandTot;
format Percent percent8.2;
run;

```

False

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

work.summary

| GrandTot |
|----------|
| 15695800 |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120101     | 4        | 269570  | .       | 2   |

93

p307d10

## Execution

```

data percent;
if _N_=1 then
 set summary(keep=GrandTot);
set orion.totalsalaries;
Percent=DeptSal/GrandTot;
format Percent percent8.2;
run;

```

Read second observation  
from orion.totalsalaries.  
Calculate Percent.

work.summary

| GrandTot |
|----------|
| 15695800 |

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

PDV

| GrandTot | Manager ID | Num Emps | DeptSal | Percent | _N_ |
|----------|------------|----------|---------|---------|-----|
| 15695800 | 120102     | 48       | 1344595 | 0.857   | 2   |

94

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID    | NumEmps   | DeptSal        |
|---------------|-----------|----------------|
| 120101        | 4         | 269570         |
| <b>120102</b> | <b>48</b> | <b>1344595</b> |
| 120103        | 30        | 793835         |
| 120104        | 15        | 425215         |
| 120259        | 6         | 941155         |
| 120260        | 3         | 216065         |
| :             | :         | :              |

Implicit OUTPUT;  
Implicit RETURN;

work.summary

| GrandTot        |
|-----------------|
| <b>15695800</b> |

PDV

| GrandTot        | Manager ID    | Num Emps  | DeptSal        | Percent      | _N_      |
|-----------------|---------------|-----------|----------------|--------------|----------|
| <b>15695800</b> | <b>120102</b> | <b>48</b> | <b>1344595</b> | <b>0.857</b> | <b>2</b> |

p307d10

## Execution

```
data percent;
 if _N_=1 then
 set summary(keep=GrandTot);
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;
```

Partial orion.totalsalaries

| Manager ID | NumEmps | DeptSal |
|------------|---------|---------|
| 120101     | 4       | 269570  |
| 120102     | 48      | 1344595 |
| 120103     | 30      | 793835  |
| 120104     | 15      | 425215  |
| 120259     | 6       | 941155  |
| 120260     | 3       | 216065  |
| :          | :       | :       |

Continue until EOF  
in orion.totalsalaries.

work.summary

| GrandTot        |
|-----------------|
| <b>15695800</b> |

PDV

| GrandTot        | Manager ID    | Num Emps  | DeptSal        | Percent      | _N_       |
|-----------------|---------------|-----------|----------------|--------------|-----------|
| <b>15695800</b> | <b>121145</b> | <b>45</b> | <b>1216055</b> | <b>0.077</b> | <b>53</b> |

p307d10

96

## Creating a Summary Data Set

Partial PROC PRINT Output

| Percentage of Total Salaries<br>for Each Manager |           |         |             |         |
|--------------------------------------------------|-----------|---------|-------------|---------|
| GrandTot                                         | ManagerID | Numemps | DeptSal     | Percent |
| \$15,695,800                                     | 120101    | 4       | \$269,570   | 1.72%   |
| \$15,695,800                                     | 120102    | 48      | \$1,344,595 | 8.57%   |
| \$15,695,800                                     | 120103    | 30      | \$793,835   | 5.06%   |
| \$15,695,800                                     | 120104    | 15      | \$425,215   | 2.71%   |
| \$15,695,800                                     | 120259    | 6       | \$941,155   | 6.00%   |

97

p307d10

## 7.08 Quiz

Open and submit the program **p307a02**.

1. How many observations are in the resulting data set?
2. Why?
3. Why did the **p307d10** program generate 53 observations?

98

## Combining Summary and Detail Data Using the SQL Procedure

A PROC SQL join can combine the detail data set with the summary data set. Without a WHERE clause, a Cartesian product is used to match the summary data with every detailed observation.

```
proc sql;
 create table percentsql as
 select ManagerID,
 DeptSal,
 GrandTot 'Grand Total',
 DeptSal/GrandTot as Percent
 format=percent8.2
 from orion.totalsalaries, summary;
quit;
```

**NOTE:** The execution of this query involves performing one or more Cartesian product joins that cannot be optimized.

102

p307d11

## Combining Summary and Detail Data Using the SQL Procedure

Calculate the summary and merge it with the detail data in a single PROC SQL step.

```
proc sql;
 create table percentsql as
 select ManagerID,
 DeptSal,
 sum(DeptSal) as GrandTot 'Grand Total',
 DeptSal / calculated GrandTot
 as Percent format=percent8.2
 from orion.totalsalaries;
quit;
```

**NOTE:** The query requires remerging summary statistics back with the original data.

103

p307d12

In addition to using the SQL procedure for calculating the percentages in one step, you can also use the REPORT procedure and the TABULATE procedure. These procedures can also calculate percentages in one step.

p307d13

```

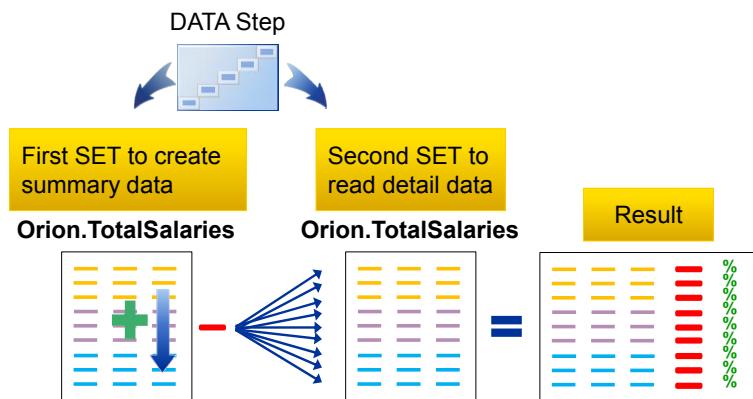
proc report data=orion.totalsalaries
 out=report_pct(drop=_break_)
 nowd;
 column ManagerID DeptSal DeptSal=PctSal;
 define ManagerID / display 'Manager ID';
 define DeptSal / sum 'Department Salaries';
 define PctSal / pctsum format=percent8.2
 'Percent of Total Salaries';
run;

proc tabulate data=orion.totalsalaries
 out=tab_pct(drop=_type_ _page_ _table_);
 class ManagerID;
 var DeptSal;
 table ManagerID='Manager ID',
 DeptSal='Department Salaries' * (sum*f=dollar14.2 pctsum);
run;

```

## Combining the Summary and Detail Data Using SET Statements

Use two SET statements to calculate the **GrandTot** variable and calculate the percentages.



104

## Combining the Summary and Detail Data Using SET Statements

```

data percent(drop=i);
 if _N_=1 then do i=1 to TotObs;
 set orion.totalsalaries(keep=DeptSal) nobs=TotObs;
 GrandTot+DeptSal;
 end;
 set orion.totalsalaries;
 Percent=DeptSal/GrandTot;
 format Percent percent8.2;
run;

```

Create summary data.  
Read detail data and calculate Percent.

### Partial SAS Log

```

NOTE: There were 53 observations read from the data set ORION.TOTALSALARIES.
NOTE: There were 53 observations read from the data set ORION.TOTALSALARIES.
NOTE: The data set WORK.PERCENT has 53 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.07 seconds
 cpu time 0.03 seconds

```

105

p307d14



## Exercises

---

### Level 1

#### 7. Combining Summary Data Containing an Average with Detail Data

The data set **orion.customerdim** contains the age for each customer.

##### Partial **orion.customerdim**

| Partial orion.customerdim Data Set |                     |                    |                                         |                     |
|------------------------------------|---------------------|--------------------|-----------------------------------------|---------------------|
| Customer ID                        | Customer Country    | Customer Gender    | Customer Name                           | Customer First Name |
| 4                                  | US                  | M                  | James Kvarniq                           | James               |
| 5                                  | US                  | F                  | Sandrina Stephano                       | Sandrina            |
| 9                                  | DE                  | F                  | Cornelia Krahl                          | Cornelia            |
| 10                                 | US                  | F                  | Karen Ballinger                         | Karen               |
| 11                                 | DE                  | F                  | Elke Wallstab                           | Elke                |
| Customer Last Name                 | Customer Birth Date | Customer Age Group | Customer Type Name                      |                     |
| Kvarniq                            | 27JUN1978           | 31-45 years        | Orion Club members low activity         |                     |
| Stephano                           | 09JUL1983           | 15-30 years        | Orion Club Gold members medium activity |                     |
| Krahl                              | 27FEB1978           | 31-45 years        | Orion Club Gold members medium activity |                     |
| Ballinger                          | 18OCT1988           | 15-30 years        | Orion Club members high activity        |                     |
| Wallstab                           | 16AUG1978           | 31-45 years        | Orion Club members high activity        |                     |

| Customer                |     |
|-------------------------|-----|
| Customer Group Name     | Age |
| Orion Club members      | 33  |
| Orion Club Gold members | 28  |
| Orion Club Gold members | 33  |
| Orion Club members      | 23  |
| Orion Club members      | 33  |

- a. Use PROC SUMMARY to store the average age of all customers in a temporary file named **work.average**.
- b. Create a SAS data set named **agedif**. It combines **work.average** with **orion.customerdim** in order to determine the difference between each customer's age and the average for all customers. (You can use any method presented in this section.)
- c. Print the first five observations of the **agedif** SAS data set.

#### PROC PRINT Output

| The agedif Data Set |            |          |     |            |
|---------------------|------------|----------|-----|------------|
| AvgAge              | CustomerID | Customer | Age | Difference |
| 41.9740             | 4          | 33       |     | -8.9740    |
| 41.9740             | 5          | 28       |     | -13.9740   |
| 41.9740             | 9          | 33       |     | -8.9740    |
| 41.9740             | 10         | 23       |     | -18.9740   |
| 41.9740             | 11         | 33       |     | -8.9740    |

## Level 2

### 8. Combining Summary Data with Multiple Statistics with Detail Data

The data set **orion.employeedonations** contains the contributions of employees to various charities. The contributions are reported for each of four quarters.

#### Partial **orion.employeedonations**

| orion.employeedonations SAS Data Set |      |      |      |      |                                                      |
|--------------------------------------|------|------|------|------|------------------------------------------------------|
| EmployeeID                           | Qtr1 | Qtr2 | Qtr3 | Qtr4 | Recipients                                           |
| 120265                               | .    | .    | .    | 25   | Mitleid International 90%, Save the Baby Animals 10% |
| 120267                               | 15   | 15   | 15   | 15   | Disaster Assist, Inc. 80%, Cancer Cures, Inc. 20%    |
| 120269                               | 20   | 20   | 20   | 20   | Cancer Cures, Inc. 10%, Cuidadores Ltd. 90%          |
| 120270                               | 20   | 10   | 5    | .    | AquaMissions International 10%, Child Survivors 90%  |
| 120271                               | 20   | 20   | 20   | 20   | Cuidadores Ltd. 80%, Mitleid International 20%       |
| <b>PaidBy</b>                        |      |      |      |      |                                                      |
| Cash or Check                        |      |      |      |      |                                                      |
| Payroll Deduction                    |      |      |      |      |                                                      |
| Payroll Deduction                    |      |      |      |      |                                                      |
| Cash or Check                        |      |      |      |      |                                                      |
| Payroll Deduction                    |      |      |      |      |                                                      |

- Select any method to create a SAS data set named **compare** by performing the following tasks:
  - Calculate the total contribution for each employee.
  - Determine the average of the total contribution for all of the employees.
  - Calculate the difference between the average and each individual employee's total contribution.
- Print the first five observations of the **compare** SAS data set to match the output below.

Partial PROC PRINT Output

| The compare Data Set |      |      |      |      |                |              |            |
|----------------------|------|------|------|------|----------------|--------------|------------|
| EmployeeID           | Qtr1 | Qtr2 | Qtr3 | Qtr4 | Total Donation | Avg Donation | Difference |
| 120265               | .    | .    | .    | 25   | 25             | 47.2581      | -22.2581   |
| 120267               | 15   | 15   | 15   | 15   | 60             | 47.2581      | 12.7419    |
| 120269               | 20   | 20   | 20   | 20   | 80             | 47.2581      | 32.7419    |
| 120270               | 20   | 10   | 5    | .    | 35             | 47.2581      | -12.2581   |
| 120271               | 20   | 20   | 20   | 20   | 80             | 47.2581      | 32.7419    |

## Challenge

### 9. Combining Summary Data Containing a Weighted Average and Detail Data

The SAS data set **orion.orderfact** contains the variables **CostPricePerUnit** and **Quantity**.

Partial Listing of **orion.orderfact**

| orion.orderfact SAS Data Set |              |            |                   |                   |            |
|------------------------------|--------------|------------|-------------------|-------------------|------------|
| CustomerID                   | EmployeeID   | StreetID   | OrderDate         | Delivery Date     | OrderID    |
| 63                           | 121039       | 9260125492 | 11JAN2007         | 11JAN2007         | 1230058123 |
| 5                            | 99999999     | 9260114570 | 15JAN2007         | 19JAN2007         | 1230080101 |
| 45                           | 99999999     | 9260104847 | 20JAN2007         | 22JAN2007         | 1230106883 |
| 41                           | 120174       | 1600101527 | 28JAN2007         | 28JAN2007         | 1230147441 |
| 183                          | 120134       | 1600100760 | 27FEB2007         | 27FEB2007         | 1230315085 |
| Order Type                   | ProductID    | Quantity   | TotalRetail Price | CostPrice PerUnit | Discount   |
| 1                            | 220101300017 | 1          | \$16.50           | \$7.45            | .          |
| 2                            | 230100500026 | 1          | \$247.50          | \$109.55          | .          |
| 2                            | 240600100080 | 1          | \$28.30           | \$8.55            | .          |
| 1                            | 240600100010 | 2          | \$32.00           | \$6.50            | .          |
| 1                            | 240200200039 | 3          | \$63.60           | \$8.80            | .          |

The data set **orion.productdim** contains the variables **ProductID** and **ProductName**.

Partial Listing of **orion.productdim**

| orion.productdim SAS Data Set |                         |                 |                        |                                     |
|-------------------------------|-------------------------|-----------------|------------------------|-------------------------------------|
| ProductID                     | ProductLine             | Category        | ProductGroup           | ProductName                         |
| 210200100009                  | Children                | Children Sports | A-Team, Kids           | Kids Sweat Round Neck,Large Logo    |
| 210200100017                  | Children                | Children Sports | A-Team, Kids           | Sweatshirt Children's O-Neck        |
| 210200200022                  | Children                | Children Sports | Bathing Suits, Kids    | Sunfit Slow Swimming Trunks         |
| 210200200023                  | Children                | Children Sports | Bathing Suits, Kids    | Sunfit Stockton Swimming Trunks Jr. |
| 210200300006                  | Children                | Children Sports | Eclipse, Kid's Clothes | Fleece Cuff Pant Kid'S              |
| <br>Supplier                  |                         |                 |                        |                                     |
| Country                       | SupplierName            | SupplierID      |                        |                                     |
| US                            | A Team Sports           | 3298            |                        |                                     |
| US                            | A Team Sports           | 3298            |                        |                                     |
| US                            | Nautlius SportsWear Inc | 6153            |                        |                                     |
| US                            | Nautlius SportsWear Inc | 6153            |                        |                                     |
| US                            | Eclipse Inc             | 1303            |                        |                                     |

- Select any method to create a SAS data set named **products** by performing the following tasks:
  - Calculate the total **CostPricePerUnit** weighted by **Quantity**.
  - Combine the weighted total with the **orion.orderfact** data. Create a new variable named **Percent** that is based on the actual total cost (**CostPricePerUnit \*Quantity**) and the weighted total.
- Print the first five observations of the **products** SAS data set.

## PROC PRINT Output

| The products Data Set |          |                      |                                     |         |
|-----------------------|----------|----------------------|-------------------------------------|---------|
| CustomerID            | Quantity | CostPrice<br>PerUnit | ProductName                         | Percent |
| 90                    | 2        | \$15.50              | Kids Sweat Round Neck,Large Logo    | 0.068%  |
| 49                    | 1        | \$17.35              | Sweatshirt Children's O-Neck        | 0.038%  |
| 79                    | 2        | \$7.05               | Sunfit Slow Swimming Trunks         | 0.031%  |
| 52                    | 1        | \$8.25               | Sunfit Stockton Swimming Trunks Jr. | 0.018%  |
| 90                    | 1        | \$7.70               | Fleece Cuff Pant Kid'S              | 0.017%  |

## 7.4 Combining Data Conditionally

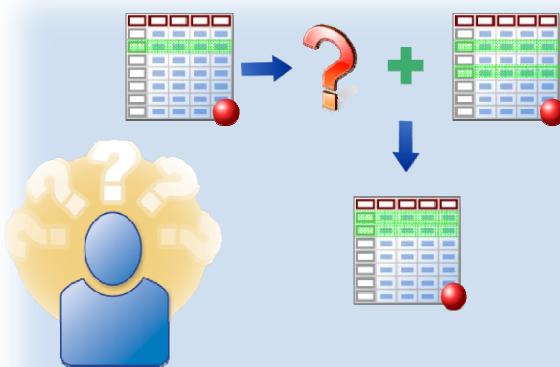
### Objectives

- Combine data conditionally with the SQL procedure.
- Combine data conditionally using multiple SET statements.

109

### Introduction

Sometimes, data must be combined based on a condition other than equality.



110

## Business Scenario

Orion Star's Sales Department needs to determine **TotalRetailPrice** in euros for September orders. The conversion rate from dollars to euros changes each week.



111

## Business Scenario

The **orion.orderfact** data set stores the total retail price in dollars. The **orion.rates** data set has the conversion rates for each week in September. The date ranges are exclusive and exhaustive.

**OrderDate** should be between **Sdate** and **Edate**.

```
where OrderDate between Sdate and Edate;
```

Partial orion.orderfact

| CustomerID | OrderDate | TotalRetailPrice |
|------------|-----------|------------------|
| 928        | 04SEP2011 | \$86.30          |
| 27         | 05SEP2011 | \$78.40          |
| ...        | ...       |                  |
| 12         | 18SEP2011 | \$87.20          |

Partial orion.rates

| Sdate     | Edate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

112

## 7.09 Poll

Can a DATA step merge be used for this task?

- Yes
- No

113

## Conditionally Combining Data

The SQL procedure can be used to combine data based on a condition other than equality.

```
proc sql nonumber;
 create table euros as
 select CustomerID, OrderDate, ProductID,
 TotalRetailPrice, SDate, EDate,
 AvgRate format=8.6,
 TotalRetailPrice*AvgRate as EuroPrice
 format=Euro10.2
 from orion.orderfact, orion.rates
 where OrderDate between SDate and EDate;

title 'euros SQL Data Set';
 select * from euros(obs=5);
quit;
```

115

p307d15

## Conditionally Combining Data

SQL Inner Join Results

Partial work.euros

| Customer ID | Order Date | Product ID   | Total Retail Price | Sdate     | Edate     | Avg Rate | Euro Price |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|
| 928         | 04SEP2011  | 230100600030 | \$86.30            | 01SEP2011 | 07SEP2011 | 0.731167 | €63.10     |
| 27          | 05SEP2011  | 240500200082 | \$78.40            | 01SEP2011 | 07SEP2011 | 0.731167 | €57.32     |
| 31          | 06SEP2011  | 220200100137 | \$50.30            | 01SEP2011 | 07SEP2011 | 0.731167 | €36.78     |
| 45          | 06SEP2011  | 230100600015 | \$78.20            | 01SEP2011 | 07SEP2011 | 0.731167 | €57.18     |
| 5           | 09SEP2011  | 210200500016 | \$52.50            | 08SEP2011 | 14SEP2011 | 0.721840 | €37.90     |
| ...         | ...        | ...          | ...                | ...       | ...       | ...      | ...        |

116

## Conditionally Combining Data

The DATA step can be used in this scenario also, but it requires two SET statements. The first reads only September orders. The second reads the conversion data.

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Orion.orderfact must be sorted by OrderDate.  
Orion.rates must be sorted by Sdate.

117

p307d16



To use multiple SET statements in this fashion, both data sets must be sorted in order (ascending or descending) by the variables tested in the DO WHILE statement.

## Conditionally Combining Data

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2007           | 07SEP2011 | 0.73117 |  |
| 27                      | ... | 05SEP2011  | ... | 78.40              | 08SEP2007           | 14SEP2011 | 0.72184 |  |
| 31                      | ... | 06SEP2011  | ... | 50.30              | 15SEP2007           | 21SEP2011 | 0.71589 |  |

| PDV | Customer ID | Order Date | Product ID | Total Retail Price | SDate | EDate | Avg Rate | Euro Price | D | _N_ |
|-----|-------------|------------|------------|--------------------|-------|-------|----------|------------|---|-----|
|     | .           | .          | .          | .                  | .     | .     | .        | .          | . | .   |

118

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Initialize the PDV.

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2011           | 07SEP2011 | 0.73117 |  |
| 27                      | ... | 05SEP2011  | ... | 78.40              | 08SEP2011           | 14SEP2011 | 0.72184 |  |
| 31                      | ... | 06SEP2011  | ... | 50.30              | 15SEP2011           | 21SEP2011 | 0.71589 |  |

| PDV | Customer ID | Order Date | Product ID | Total Retail Price | SDate | EDate | Avg Rate | Euro Price | D | _N_ |
|-----|-------------|------------|------------|--------------------|-------|-------|----------|------------|---|-----|
|     | .           | .          | .          | .                  | .     | .     | .        | .          | . | 1   |

119

...

## Execution

```

data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;

```

Read the first observation from orion.orderfact.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

120

...

## 7.10 Poll

When is a DO WHILE condition checked?

- a. at the start of the loop
- b. at the end of the loop

121

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

The DO WHILE condition is true. The DO loop executes.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate | EDate | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-------|-------|----------|------------|---|-----|
| 928         | 04SEP2011  | 230100600030 | 86.30              | .     | .     | .        | .          | D | 1   |

123

It is true that 04SEP2011 is *not* between . and .

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Read the first observation from orion.rates.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 928         | 04SEP2011  | 230100600030 | 86.30              | 01SEP2011 | 07SEP2011 | 0.73117  | .          | D | 1   |

124

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

PDV

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

Customer ID
Order Date
Product ID
Total Retail Price
SDate
EDate
Avg Rate
Euro Price
D
N

|     |           |              |       |           |           |         |   |   |
|-----|-----------|--------------|-------|-----------|-----------|---------|---|---|
| 928 | 04SEP2011 | 230100600030 | 86.30 | 01SEP2011 | 07SEP2011 | 0.73117 | . | 1 |
|-----|-----------|--------------|-------|-----------|-----------|---------|---|---|

125 ...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

The DO WHILE condition is false.  
The DO loop does not execute.

PDV

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

Customer ID
Order Date
Product ID
Total Retail Price
SDate
EDate
Avg Rate
Euro Price
D
N

|     |           |              |       |           |           |         |   |   |
|-----|-----------|--------------|-------|-----------|-----------|---------|---|---|
| 928 | 04SEP2011 | 230100600030 | 86.30 | 01SEP2011 | 07SEP2011 | 0.73117 | . | 1 |
|-----|-----------|--------------|-------|-----------|-----------|---------|---|---|

126 04SEP2011 is between 01SEP2011 and 07SEP2011. ...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

**Calculate EuroPrice.**

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

127

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

**Implicit OUTPUT;  
Implicit RETURN;**

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

128

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
end;
EuroPrice=TotalRetailPrice*AvgRate;
format EuroPrice Euro10.2;
run;
```

Reinitialize the PDV.

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2011           | 07SEP2011 | 0.73117 |  |
| 27                      | ... | 05SEP2011  | ... | 78.40              | 08SEP2011           | 14SEP2011 | 0.72184 |  |
| 31                      | ... | 06SEP2011  | ... | 50.30              | 15SEP2011           | 21SEP2011 | 0.71589 |  |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 928         | 04SEP2011  | 230100600030 | 86.30              | 01SEP2011 | 07SEP2011 | 0.73117  | .          |   | 2   |

129

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
end;
EuroPrice=TotalRetailPrice*AvgRate;
format EuroPrice Euro10.2;
run;
```

Read the next observation from orion.orderfact.

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2011           | 07SEP2011 | 0.73117 |  |
| 27                      | ... | 05SEP2011  | ... | 78.40              | 08SEP2011           | 14SEP2011 | 0.72184 |  |
| 31                      | ... | 06SEP2011  | ... | 50.30              | 15SEP2011           | 21SEP2011 | 0.71589 |  |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 27          | 05SEP2011  | 240500200082 | 78.40              | 01SEP2011 | 07SEP2011 | 0.73117  | .          |   | 2   |

130

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

The DO WHILE condition is false.  
The DO loop does not execute.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D<br>_N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|----------|
| 27          | 05SEP2011  | 240500200082 | 78.40              | 01SEP2011 | 07SEP2011 | 0.73117  | .          | 2        |

131      05SEP2011 is between 01SEP2011 and 07SEP2011.      ...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Calculate EuroPrice.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| 27          | ... | 05SEP2011  | ... | 78.40              |
| 31          | ... | 06SEP2011  | ... | 50.30              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D<br>_N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|----------|
| 27          | 05SEP2011  | 240500200082 | 78.40              | 01SEP2011 | 07SEP2011 | 0.73117  | 57.32      | 2        |

132      ...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2011           | 07SEP2011 | 0.73117 |  |
| 27                      | ... | 05SEP2011  | ... | 78.40              | 08SEP2011           | 14SEP2011 | 0.72184 |  |
| 31                      | ... | 06SEP2011  | ... | 50.30              | 15SEP2011           | 21SEP2011 | 0.71589 |  |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 27          | 05SEP2011  | 240500200082 | 78.40              | 01SEP2011 | 07SEP2011 | 0.73117  | 57.32      |   | 2   |

133

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Continue until OrderDate  
is 09SEP2011.

PDV

| Partial orion.orderfact |     |            |     |                    | Partial orion.rates |           |         |  |
|-------------------------|-----|------------|-----|--------------------|---------------------|-----------|---------|--|
| Customer ID             | ... | Order Date | ... | Total Retail Price | Sdate               | EDate     | AvgRate |  |
| 928                     | ... | 04SEP2011  | ... | 86.30              | 01SEP2011           | 07SEP2011 | 0.73117 |  |
| ...                     | ... | ...        | ... | ...                | 08SEP2011           | 14SEP2011 | 0.72184 |  |
| 5                       | ... | 09SEP2011  | ... | 52.50              | 15SEP2011           | 21SEP2011 | 0.71589 |  |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 5           | 09SEP2011  | 210200500016 | 52.50              | 01SEP2011 | 07SEP2011 | 0.73117  | .          |   | 5   |

134

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

The DO WHILE condition is true.  
The DO loop executes.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| ...         | ... | ...        | ... | ...                |
| 5           | ... | 09SEP2011  | ... | 52.50              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

135

It is true that 09SEP2011 is *not* between 01SEP2011 and 07SEP2011. ...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

Read the next observation  
from orion.rates.

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| ...         | ... | ...        | ... | ...                |
| 5           | ... | 09SEP2011  | ... | 52.50              |

PDV

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

136

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

PDV

| Partial orion.orderfact |     |            |     |
|-------------------------|-----|------------|-----|
| Customer ID             | ... | Order Date | ... |
| 928                     | ... | 04SEP2011  | ... |
| ...                     | ... | ...        | ... |
| 5                       | ... | 09SEP2011  | ... |

| Partial orion.rates |           |         |
|---------------------|-----------|---------|
| Sdate               | EDate     | AvgRate |
| 01SEP2011           | 07SEP2011 | 0.73117 |
| 08SEP2011           | 14SEP2011 | 0.72184 |
| 15SEP2011           | 21SEP2011 | 0.71589 |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D<br>_N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|----------|
| 5           | 09SEP2011  | 210200500016 | 52.50              | 08SEP2011 | 14SEP2011 | 0.72184  | .          | 5        |

137

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

The DO WHILE condition is false.  
The DO loop does not execute.

PDV

| Partial orion.orderfact |     |            |     |
|-------------------------|-----|------------|-----|
| Customer ID             | ... | Order Date | ... |
| 928                     | ... | 04SEP2011  | ... |
| ...                     | ... | ...        | ... |
| 5                       | ... | 09SEP2011  | ... |

| Partial orion.rates |           |         |
|---------------------|-----------|---------|
| Sdate               | EDate     | AvgRate |
| 01SEP2011           | 07SEP2011 | 0.73117 |
| 08SEP2011           | 14SEP2011 | 0.72184 |
| 15SEP2011           | 21SEP2011 | 0.71589 |

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D<br>_N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|----------|
| 5           | 09SEP2011  | 210200500016 | 52.50              | 08SEP2011 | 14SEP2011 | 0.72184  | .          | 5        |

138

09SEP2011 is between 08SEP2011 and 14SEP2011.

...

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

**Calculate EuroPrice.**

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| 928         | ... | 04SEP2011  | ... | 86.30              |
| ...         | ... | ...        | ... | ...                |
| 5           | ... | 09SEP2011  | ... | 52.50              |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 5           | 09SEP2011  | 210200500016 | 52.50              | 08SEP2011 | 14SEP2011 | 0.72184  | 37.90      | D | 5   |

139

...

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| 01SEP2011 | 07SEP2011 | 0.73117 |
| 08SEP2011 | 14SEP2011 | 0.72184 |
| 15SEP2011 | 21SEP2011 | 0.71589 |

## Execution

```
data euros;
 set orion.orderfact(where=
 (OrderDate between '01SEP2011'd and '30SEP2011'd)
 keep=CustomerID OrderDate ProductID TotalRetailPrice);
 do while (not (SDate <= OrderDate <= EDate));
 set orion.rates;
 end;
 EuroPrice=TotalRetailPrice*AvgRate;
 format EuroPrice Euro10.2;
run;
```

**Continue until EOF in orion.orderfact.**

Partial orion.orderfact

| Customer ID | ... | Order Date | ... | Total Retail Price |
|-------------|-----|------------|-----|--------------------|
| ...         | ... | ...        | ... | ...                |
| ...         | ... | ...        | ... | ...                |
| 11          | ... | 28SEP2011  | ... | 78.20              |

PDV

| Customer ID | Order Date | Product ID   | Total Retail Price | SDate     | EDate     | Avg Rate | Euro Price | D | _N_ |
|-------------|------------|--------------|--------------------|-----------|-----------|----------|------------|---|-----|
| 11          | 28SEP2011  | 220200100002 | 78.20              | 22SEP2011 | 30SEP2011 | 0.70725  | 55.31      | D | 10  |

140

Partial orion.rates

| Sdate     | EDate     | AvgRate |
|-----------|-----------|---------|
| ...       | ...       | ...     |
| ...       | ...       | ...     |
| 22SEP2011 | 30SEP2011 | 0.70725 |

## 7.11 Multiple Choice Poll

Does SAS encounter the end-of-file marker (EOF) for **orion.rates**?

- a. SAS encounters the EOF, but it does not stop the DATA step because there are two SET statements.
- b. SAS encounters the EOF for **orion.rates**, so there are only four observations in the resulting data set **euros**.
- c. The DO WHILE statement prevents the data set **orion.rates** from being read a fifth time, so the EOF is never encountered.

141

## Comparison of Techniques

| Two SET Statements                            | SQL Procedure                 |
|-----------------------------------------------|-------------------------------|
| Availability of all DATA step functionality   | Easy to code                  |
| Fast, sequential processing of both data sets | No sorting or indexing needed |



Perform benchmark tests with your data to determine which is most efficient.



## Exercises

---

### Level 1

#### 10. Combining Two Data Sets Conditionally Using the SQL Procedure

The data set **orion.agesmod** contains information about age groups.

##### **orion.agesmod**

| orion.agesmod SAS Data Set |           |          |
|----------------------------|-----------|----------|
| Description                | First Age | Last Age |
| 15-29 years                | 15        | 30       |
| 30-44 years                | 30        | 45       |
| 45-59 years                | 45        | 60       |
| 60-75 years                | 60        | 75       |

The data set **orion.customer** contains information about customers.

#### Partial **orion.customer**

| orion.customer Data Set |                       |        |                 |                   |                           |                          |
|-------------------------|-----------------------|--------|-----------------|-------------------|---------------------------|--------------------------|
| CustomerID              | Country               | Gender | Personal        |                   | Customer<br>First<br>Name | Customer<br>Last<br>Name |
|                         |                       |        | ID              | CustomerName      |                           |                          |
| 4                       | US                    | M      |                 | James Kvarniq     | James                     | Kvarniq                  |
| 5                       | US                    | F      |                 | Sandrina Stephano | Sandrina                  | Stephano                 |
| 9                       | DE                    | F      |                 | Cornelia Krahel   | Cornelia                  | Krahel                   |
| 10                      | US                    | F      |                 | Karen Ballinger   | Karen                     | Ballinger                |
| 11                      | DE                    | F      |                 | Elke Wallstab     | Elke                      | Wallstab                 |
| BirthDate               |                       |        | CustomerAddress | StreetID          | Street<br>Number          | Customer<br>TypeID       |
| 27JUN1978               | 4382 Gralyn Rd        |        | 9260106519      | 4382              | 1020                      |                          |
| 09JUL1983               | 6468 Cog Hill Ct      |        | 9260114570      | 6468              | 2020                      |                          |
| 27FEB1978               | Kallstaderstr. 9      |        | 3940106659      | 9                 | 2020                      |                          |
| 18OCT1988               | 425 Bryant Estates Dr |        | 9260129395      | 425               | 1040                      |                          |
| 16AUG1978               | Carl-Zeiss-Str. 15    |        | 3940108592      | 15                | 1040                      |                          |

- a. Use the SQL procedure to create a data set named **agegroups**, which contains the customer ID, name, age, and age group as of January 1, 2012. (The variable **Description** is in the **orion.agesmod** SAS data set.) Order the data by **CustomerID**.

Hint: The following expression calculates the customer's age:

```
Int(yrdif(BirthDate,'01Jan2012'd, 'AGE'))
```

 The AGE argument is new in SAS 9.3.

The customer's age should be between **FirstAge** and **LastAge** to retrieve the appropriate age group description.

- b. Print the first five observations of the **agegroups** data set.

#### PROC PRINT Output

| agegroups Data Set |                   |     |             |
|--------------------|-------------------|-----|-------------|
| CustomerID         | CustomerName      | Age | Description |
| 4                  | James Kvarniq     | 33  | 30-44 years |
| 5                  | Sandrina Stephano | 28  | 15-29 years |
| 9                  | Cornelia Krahel   | 33  | 30-44 years |
| 10                 | Karen Ballinger   | 23  | 15-29 years |
| 11                 | Elke Wallstab     | 33  | 30-44 years |

## Level 2

### 11. Combining Two Data Sets Conditionally Using the DATA Step DO Loop

The data set **orion.agesmod** contains information about age groups.

**orion.agesmod**

| orion.agesmod SAS Data Set |     |                  |
|----------------------------|-----|------------------|
| Description                | Age | First<br>LastAge |
| 15-29 years                | 15  | 30               |
| 30-44 years                | 30  | 45               |
| 45-59 years                | 45  | 60               |
| 60-75 years                | 60  | 75               |

The data set **orion.customer** contains information about customers.

Partial **orion.customer**

| orion.customer SAS Data Set |                       |        |                 |                   |                 |                   |
|-----------------------------|-----------------------|--------|-----------------|-------------------|-----------------|-------------------|
| CustomerID                  | Country               | Gender | Personal        | Customer          |                 |                   |
|                             |                       |        | ID              | CustomerName      | First Name      | Customer LastName |
| 4                           | US                    | M      |                 | James Kvarniq     | James           | Kvarniq           |
| 5                           | US                    | F      |                 | Sandrina Stephano | Sandrina        | Stephano          |
| 9                           | DE                    | F      |                 | Cornelia Krahel   | Cornelia        | Krahel            |
| 10                          | US                    | F      |                 | Karen Ballinger   | Karen           | Ballinger         |
| 11                          | DE                    | F      |                 | Elke Wallstab     | Elke            | Wallstab          |
| BirthDate                   |                       |        | CustomerAddress | Street ID         | Customer TypeID |                   |
| 27JUN1978                   | 4382 Gralyn Rd        |        |                 | 9260106519        | 4382            | 1020              |
| 09JUL1983                   | 6468 Cog Hill Ct      |        |                 | 9260114570        | 6468            | 2020              |
| 27FEB1978                   | Kallstадterstr. 9     |        |                 | 3940106659        | 9               | 2020              |
| 18OCT1988                   | 425 Bryant Estates Dr |        |                 | 9260129395        | 425             | 1040              |
| 16AUG1978                   | Carl-Zeiss-Str. 15    |        |                 | 3940108592        | 15              | 1040              |

- a. Modify p307e11 to use a DATA step to combine the **agesmod** and **customer** data sets based on a customer's age as of January 1, 2012. Create a data set named **agegroups**, which contains the customer ID, name, birthdate, age, and age description.

Hint: Use the following calculation to determine someone's age as of January 1, 2012:

```
int(yrdif(BirthDate, '01Jan2012'd, 'AGE'))
```

- b. From the **agegroups** data set, print only the observations that fall into the highest age category.

Partial PROC PRINT Output (First 7 of 14 observations)

| Highest Age Category |                    |           |     |             |
|----------------------|--------------------|-----------|-----|-------------|
| CustomerID           | CustomerName       | BirthDate | Age | Description |
| 195                  | Cosi Rimmington    | 11NOV1948 | 63  | 60-75 years |
| 92                   | Lendon Celii       | 14SEP1948 | 63  | 60-75 years |
| 61                   | Carsten Maestrini  | 08JUL1948 | 63  | 60-75 years |
| 183                  | Duncan Robertshawe | 25MAR1948 | 63  | 60-75 years |
| 2788                 | Serdar Yucel       | 02JAN1948 | 63  | 60-75 years |
| 70108                | Patrick Leach      | 14APR1943 | 68  | 60-75 years |
| 33                   | Rolf Robak         | 24FEB1943 | 68  | 60-75 years |

## Challenge

### 12. Combining Two Data Sets Conditionally Using the DATA Step Hash Object

The data set **orion.agesmod** contains information about age groups.

**orion.agesmod**

| orion.agesmod SAS Data Set |     |         |
|----------------------------|-----|---------|
| Description                | Age | LastAge |
| 15-29 years                | 15  | 30      |
| 30-44 years                | 30  | 45      |
| 45-59 years                | 45  | 60      |
| 60-75 years                | 60  | 75      |

The data set **orion.customer** contains information about customers.

Partial **orion.customer**

| orion.customer SAS Data Set |                       |        |            |                   |                    |                      |
|-----------------------------|-----------------------|--------|------------|-------------------|--------------------|----------------------|
| CustomerID                  | Country               | Gender | Personal   | CustomerName      | Customer           | Customer<br>LastName |
|                             |                       |        | ID         |                   | First<br>Name      |                      |
| 4                           | US                    | M      |            | James Kvarniq     | James              | Kvarniq              |
| 5                           | US                    | F      |            | Sandrina Stephano | Sandrina           | Stephano             |
| 9                           | DE                    | F      |            | Cornelia Krahel   | Cornelia           | Krahel               |
| 10                          | US                    | F      |            | Karen Ballinger   | Karen              | Ballinger            |
| 11                          | DE                    | F      |            | Elke Wallstab     | Elke               | Wallstab             |
| BirthDate                   | CustomerAddress       |        | StreetID   | Street<br>Number  | Customer<br>TypeID |                      |
| 27JUN1978                   | 4382 Gralyn Rd        |        | 9260106519 | 4382              | 1020               |                      |
| 09JUL1983                   | 6468 Cog Hill Ct      |        | 9260114570 | 6468              | 2020               |                      |
| 27FEB1978                   | Kallstadterstr. 9     |        | 3940106659 | 9                 | 2020               |                      |
| 18OCT1988                   | 425 Bryant Estates Dr |        | 9260129395 | 425               | 1040               |                      |
| 16AUG1978                   | Carl-Zeiss-Str. 15    |        | 3940108592 | 15                | 1040               |                      |

- a. Use the DATA step and a hash object to create a data set named **agegroups** that contains the customer ID, name, age, and age group as of January 1, 2012. (The variable **Description** is in the **orion.agesmod** SAS data set.)
- b. Print the first five observations of the **agegroups** data set.

PROC PRINT Output

| agegroups |             |            |                   |     |
|-----------|-------------|------------|-------------------|-----|
|           | Description | CustomerID | CustomerName      | Age |
|           | 30-44 years | 4          | James Kvarniq     | 33  |
|           | 15-29 years | 5          | Sandrina Stephano | 28  |
|           | 30-44 years | 9          | Cornelia Krahel   | 33  |
|           | 15-29 years | 10         | Karen Ballinger   | 23  |
|           | 30-44 years | 11         | Elke Wallstab     | 33  |

# 7.5 Solutions

---

## Solutions to Exercises

### 1. Merging or Joining Three Data Sets

- a. Modify the program **p307e01**. Combine the three data sets to create a data set named **purchases** that contains the customer name, product name, and supplier name for the customers in the **orion.orderfact** data set.

- 1) Order the data by **ProductID**.
- 2) Enter one of the following solutions in the Editor window:

**p307s01**

Merge Solution

```
proc sort data=orion.orderfact(keep=CustomerID ProductID)
 out=orderfact;
 by CustomerID;
run;
data temp;
 merge orderfact(in=O)
 orion.customerdim(keep=CustomerID CustomerName in=C);
 by CustomerID;
 if O and C;
run;
proc sort data=temp;
 by ProductID;
run;
data purchases;
 keep CustomerName ProductName SupplierName;
 merge temp(in=T)
 orion.productdim(keep=ProductID ProductName
 SupplierName in=P);
 by ProductID;
 if P and T;
run;
```

## PROC SQL Solution

```

proc sql;
 create table purchases as
 select CustomerName,
 ProductName,
 SupplierName
 from orion.orderfact as o,
 orion.productdim as p,
 orion.customerdim as c
 where o.CustomerID=c.CustomerID
 and o.ProductID=p.ProductID
 order by o.ProductID;
 title 'Partial purchases Data Set';
 reset flow=15 25;
 select *
 from purchases(obs=5);
title;
quit;

```

- 3) Submit the SAS code.
- 4) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log – DATA Step Merge Solution

```

591 proc sort data=orion.orderfact(keep=CustomerID ProductID)
592 out=orderfact;
593 by CustomerID;
594 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.ORDERFACT has 617 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.04 seconds
 cpu time 0.00 seconds
595
596 data temp;
597 merge orderfact(in=0)
598 orion.customerdim(keep=CustomerID CustomerName in=C);
599 by CustomerID;
600 if 0 and C;
601 run;
NOTE: There were 617 observations read from the data set WORK.ORDERFACT.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.TEMP has 617 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.01 seconds
602
603 proc sort data=temp;
604 by ProductID;
605 run;
NOTE: There were 617 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.TEMP has 617 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

```

606
607 data purchases;
608 keep CustomerName ProductName SupplierName;
609 merge temp(in=T)
610 orion.productdim(keep=ProductID ProductName SupplierName in=P);
611 by ProductID;
612 if P and T;
613 run;
NOTE: There were 617 observations read from the data set WORK.TEMP.
NOTE: There were 481 observations read from the data set ORION.PRODUCTDIM.
NOTE: The data set WORK.PURCHASES has 617 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds

```

#### Partial SAS Log – PROC SQL Solution

```

614 proc sql;
615 create table purchases as
616 select CustomerName,
617 ProductName,
618 SupplierName
619 from orion.orderfact as o,
620 orion.productdim as p,
621 orion.customerdim as c
622 where o.CustomerID=c.CustomerID
623 and o.ProductID=p.ProductID
624 order by o.ProductID;
NOTE: The query as specified involves ordering by an item that doesn't appear in its
SELECT
 clause.
NOTE: Table WORK.PURCHASES created, with 617 rows and 3 columns.

```

- b. Print the first five observations of the **purchases** data set.

- 1) Enter one of the following solutions in the Editor window:

#### Partial p307s01 – DATA Step Merge Solution

```

proc print data=purchases(obs=5) noobs;
 title 'Partial Purchases Data Set';
run;

```

#### PROC SQL Solution

```

proc sql;
title 'Partial purchases Data Set';
 reset flow=15 25;
 select *
 from purchases(obs=5);
title;
quit;

```

- 2) View the output to verify that it matches the expected output.
- 3) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log – DATA Step Merge Solution

```

84 proc print data=purchases(obs=5) noobs;
85 title 'Partial purchases Data Set';
86 run;
NOTE: There were 5 observations read from the data set WORK.PURCHASES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

## Partial SAS Log – PROC SQL Solution

```

157 title 'Partial purchases Data Set';
158 reset flow=15 25;
159 select *
160 from purchases(obs=5);
161 title;
162 quit;

```

**2. Merging or Joining Data to Create Multiple Data Sets**

- a. Modify **p307e02** to merge the three data sets and create the following three data sets:
- a data set named **nopurchases** that contains the customers who did not make any purchases. It should only include the variables CustomerID, and CustomerName.
  - a data set named **purchases** that contains the customers that exist in the **orion.orderfact** data set. It should only include the variables Customerid, CustomerName, ProductID, OrderID, Quantity, and TotalRetailPrice
  - a data set named **noproducts** that contains the product names and suppliers for products that were not purchased. It should only include the variables ProductID, ProductName, and SupplierName.

- 1) Enter one of the following solutions in the Editor window:

Partial **p307s02** – DATA Step Merge Solution

```

proc sort data=orion.orderfact out=orderfact;
 by CustomerID;
run;
data temp
 nopurchases(keep=CustomerID CustomerName);
 merge orderfact(in=O)
 orion.customerdim(keep=CustomerID CustomerName in=C);
 by CustomerID;
 if O and C then output temp;
 else if C and not O then output nopurchases;
run;
proc sort data=temp;
 by ProductID;
run;

```

```

data purchases(keep=Customerid CustomerName ProductID
 OrderID Quantity TotalRetailPrice)
 noproducts(keep=ProductID ProductName
 SupplierName);
 merge temp(in=T)
 orion.productdim(keep=ProductID ProductName
 SupplierName in=P);
 by ProductID;
 if P and T then output purchases;
 else if P and not T then output noproducts;
run;

```

## PROC SQL Solution

```

proc sql;
 create table nopurchases as
 select CustomerID,
 CustomerName
 from orion.customerdim
 where customerdim.CustomerID not in
 (select CustomerID from orion.orderfact);
 create table noproducts as
 select ProductID, ProductName, SupplierName
 from orion.productdim
 where productdim.ProductID not in
 (select ProductID from orion.orderfact);
 create table purchases as
 select c.Customerid,
 CustomerName,
 o.ProductID,
 OrderID,
 Quantity,
 TotalRetailPrice
 from orion.orderfact as o,
 orion.productdim as p,
 orion.customerdim as c
 where o.CustomerID=c.CustomerID
 and o.ProductID=p.ProductID
 order by o.ProductID;
quit;

```

- 2) Submit the SAS code and check the SAS log for warnings or error messages.

## Partial SAS Log – DATA Step Merge Solution

```

246 proc sort data=orion.orderfact out=orderfact;
247 by CustomerID;
248 run;

```

```

NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.ORDERFACT has 617 observations and 12 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

```

249
250 data temp
251 nopurchases(keep=CustomerID CustomerName);
252 merge orderfact(in=0)
253 orion.customerdim(keep=CustomerID CustomerName in=C);
254 by CustomerID;
255 if 0 and C then output temp;
256 else if C and not 0 then output nopurchases;
257 run;

NOTE: There were 617 observations read from the data set WORK.ORDERFACT.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.TEMP has 617 observations and 13 variables.
NOTE: The data set WORK.NOPURCHASES has 2 observations and 2 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

258
259 proc sort data=temp;
260 by ProductID;
261 run;

NOTE: There were 617 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.TEMP has 617 observations and 13 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

262
263 data purchases(keep=Customerid CustomerName ProductID OrderID Quantity
TotalRetailPrice)
264 noproducts(keep=ProductID ProductName
 SupplierName);
265 merge temp(in=T)
266 orion.productdim(keep=ProductID ProductName
 SupplierName in=P);
267 by ProductID;
268 if P and T then output purchases;
269 else if P and not T then output noproducts;
270 run;

NOTE: There were 617 observations read from the data set WORK.TEMP.
NOTE: There were 481 observations read from the data set ORION.PRODUCTDIM.
NOTE: The data set WORK.PURCHASES has 617 observations and 6 variables.
NOTE: The data set WORK.NOPRODUCTS has 0 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## Partial SAS Log – PROC SQL Solution

```

328 proc sql;
329 create table nopurchases as
330 select CustomerID,
331 CustomerName

```

```

332 from orion.customerdim
333 where customerdim.CustomerID not in
334 (select CustomerID from orion.orderfact);
NOTE: Table WORK.NOPURCHASES created, with 2 rows and 2 columns.

335
336 create table nopproducts as
337 select ProductID, ProductName, SupplierName
338 from orion.productdim
339 where productdim.ProductID not in
340 (select ProductID from orion.orderfact);
NOTE: Table WORK.NOPRODUCTS created, with 0 rows and 3 columns.

341
342 create table purchases as
343 select c.Customerid, CustomerName, o.ProductID, OrderID, Quantity,
343! TotalRetailPrice
344 from orion.orderfact as o,
345 orion.productdim as p,
346 orion.customerdim as c
347 where o.CustomerID=c.CustomerID
348 and o.ProductID=p.ProductID
349 order by o.ProductID;
NOTE: Table WORK.PURCHASES created, with 617 rows and 6 columns.

```

- b. Print the first five observations of each data set.

- 1) Enter one of the following solutions in the Editor window:

Partial **p307s02** – PROC PRINT Solution

```

proc print data=nopproducts(obs=5) noobs;
 title 'Partial nopproducts Data Set';
run;
proc print data=purchases(obs=5) noobs;
 title 'Partial purchases Data Set';
run;
proc print data=nopurchases(obs=5) noobs;
 title 'Partial nopurchases Data Set';
run;

```

PROC SQL Solution

```

proc sql;
title 'Partial nopproducts Data Set';
 select *
 From nopproducts(obs=5);

title 'Partial purchases Data Set';
 select *
 From purchases(obs=5);

title 'Partial nopurchases Data Set';
 select *
 From nopurchases(obs=5);
quit;

```

```
quit;
```

- 2) View the output to verify that it matches the expected output.
- 3) Examine the SAS log to verify that the step executed without errors.

#### Partial SAS Log – DATA Step Merge Solution

```

117 proc print data=nopurchases(obs=5) noobs;
118 title 'Partial nopurchases Data Set';
119 run;
NOTE: There were 2 observations read from the data set WORK.NOPURCHASES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
120
121
122 proc print data=purchases(obs=5) noobs;
123 title 'Partial purchases Data Set';
124 run;
NOTE: There were 5 observations read from the data set WORK.PURCHASES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds
125
126 proc print data=noproducts(obs=5) noobs;
127 title 'Partial noproducts Data Set';
128 run;
NOTE: No observations in data set WORK.NOPRODUCTS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

#### Partial SAS Log – PROC SQL Solution

```

130 title 'Partial noproducts Data Set';
131 select *
132 From noproducts(obs=5);
NOTE: No rows were selected.
133
134 title 'Partial purchases Data Set';
135 select *
136 From purchases(obs=5);
137
138 title 'Partial nopurchases Data Set';
139 select *
140 From nopurchases(obs=5);
141 quit;

```

### 3. Merging or Joining Multiple Data Sets

- a. Create a data set named **managernames** that contains the **EmployeeID** variable, the six **ManagerID** variables, and the six manager names.

- 1) Enter one of the following solutions in the Editor window:

#### Partial p307s03 – PROC SQL Solution

```
proc sql;
 create table ManagerNames as
```

```

select e.EmployeeID,
 ManagerLevel1,
 ManagerLevel2,
 ManagerLevel3,
 ManagerLevel4,
 ManagerLevel5,
 ManagerLevel6,
 m1.EmployeeName as Manager1Name,
 m2.EmployeeName as Manager2Name,
 m3.EmployeeName as Manager3Name,
 m4.EmployeeName as Manager4Name,
 m5.EmployeeName as Manager5Name,
 m6.EmployeeName as Manager6Name
 from orion.organizationdim as e
 left join orion.employeeaddresses as m1
 on e.ManagerLevel1=m1.EmployeeID
 left join orion.employeeaddresses as m2
 on e.ManagerLevel2=m2.EmployeeID
 left join orion.employeeaddresses as m3
 on e.ManagerLevel3=m3.EmployeeID
 left join orion.employeeaddresses as m4
 on e.ManagerLevel4=m4.EmployeeID
 left join orion.employeeaddresses as m5
 on e.ManagerLevel5=m5.EmployeeID
 left join orion.employeeaddresses as m6
 on e.ManagerLevel6=m6.EmployeeID
 order by e.EmployeeID;
quit;

```

#### DATA Step Merge Solution

```

proc sort data=orion.employeeaddresses
 out=empaddresses(rename=(EmployeeID=ManagerLevel1
 EmployeeName=Manager1Name));
 by EmployeeID;
run;
proc sort data=orion.organizationdim
 out=man1(keep=EmployeeID ManagerLevel1-ManagerLevel6);
 by ManagerLevel1;
run;
data manager1;
 merge man1(in=M)
 empaddresses(keep=ManagerLevel1 Manager1Name);
 by ManagerLevel1;
 if M;
run;
proc sort data=manager1 out=man2;
 by ManagerLevel2;
run;
data manager2;
 merge man2(in=M)
 empaddresses(rename=(ManagerLevel1=ManagerLevel2
 Manager1Name=Manger2Name));
 by ManagerLevel2;
run;

```

```
keep=ManagerLevel1 Manager1Name);
by ManagerLevel2;
if M;
run;
proc sort data=manager2 out=man3;
by ManagerLevel3;
run;
data manager3;
merge man3(in=M)
empaddresses(rename=(ManagerLevel1=ManagerLevel3
Manager1Name=Manger3Name)
keep=ManagerLevel1 Manager1Name);
by ManagerLevel3;
if M;
run;
proc sort data=manager3 out=man4;
by ManagerLevel4;
run;
data manager4;
merge man4(in=M)
empaddresses(rename=(ManagerLevel1=ManagerLevel4
Manager1Name=Manger4Name)
keep=ManagerLevel1 Manager1Name);
by ManagerLevel4;
if M;
run;
proc sort data=manager4 out=man5;
by ManagerLevel5;
run;
data manager5;
merge man5(in=M)
empaddresses(rename=(ManagerLevel1=ManagerLevel5
Manager1Name=Manger5Name)
keep=ManagerLevel1 Manager1Name);
by ManagerLevel5;
if M;
run;
proc sort data=manager5 out=man6;
by ManagerLevel6;
run;
data managernames;
merge man6(in=M)
empaddresses(rename=(ManagerLevel1=ManagerLevel6
Manager1Name=Manger6Name)
keep=ManagerLevel1 Manager1Name);
by ManagerLevel6;
if M;
run;
proc sort data=managernames;
by EmployeeID;
run;
```

- 2) Submit the SAS code.
- 3) Check the SAS log to verify that the code executed without errors.

Partial SAS Log – PROC SQL Solution

```

proc sql;
224 create table ManagerNames as
225 select e.EmployeeID,
226 ManagerLevel1,
227 ManagerLevel2,
228 ManagerLevel3,
229 ManagerLevel4,
230 ManagerLevel5,
231 ManagerLevel6,
232 m1.EmployeeName as Manager1Name,
233 m2.EmployeeName as Manager2Name,
234 m3.EmployeeName as Manager3Name,
235 m4.EmployeeName as Manager4Name,
236 m5.EmployeeName as Manager5Name,
237 m6.EmployeeName as Manager6Name
238 from orion.organizationdim as e
239 left join orion.employeeaddresses as m1
240 on e.ManagerLevel1=m1.EmployeeID
241 left join orion.employeeaddresses as m2
242 on e.ManagerLevel2=m2.EmployeeID
243 left join orion.employeeaddresses as m3
244 on e.ManagerLevel3=m3.EmployeeID
245 left join orion.employeeaddresses as m4
246 on e.ManagerLevel4=m4.EmployeeID
247 left join orion.employeeaddresses as m5
248 on e.ManagerLevel5=m5.EmployeeID
249 left join orion.employeeaddresses as m6
250 on e.ManagerLevel6=m6.EmployeeID
251 order by e.EmployeeID;
NOTE: Table WORK.MANAGERNAMES created, with 424 rows and 13 columns.

252 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 2.24 seconds
 cpu time 0.34 seconds

253
254 /* DATA step Solution */
255
256 proc sort data=orion.employeeaddresses
257 out=empaddresses(rename=(EmployeeID=ManagerLevel1
258 EmployeeName=Manager1Name));
259 by EmployeeID;
260 run;

NOTE: There were 424 observations read from the data set ORION.EMPLOYEEADDRESSES.
NOTE: The data set WORK.EMPADDRESSES has 424 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.12 seconds
 cpu time 0.03 seconds

```

```
261
262 proc sort data=orion.organizationdim
263 out=man1(keep=EmployeeID ManagerLevel1-ManagerLevel6);
264 by ManagerLevel1;
265 run;

NOTE: There were 424 observations read from the data set ORION.ORGANIZATIONDIM.
NOTE: The data set WORK.MAN1 has 424 observations and 7 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
266
267 data manager1;
268 merge man1(in=M)
269 empaddresses(keep=ManagerLevel1 Manager1Name);
270 by ManagerLevel1;
271 if M;
272 run;

NOTE: There were 424 observations read from the data set WORK.MAN1.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGER1 has 424 observations and 8 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.01 seconds

273
274 proc sort data=manager1 out=man2;
275 by ManagerLevel2;
276 run;

NOTE: There were 424 observations read from the data set WORK.MANAGER1.
NOTE: The data set WORK.MAN2 has 424 observations and 8 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.00 seconds

277
278 data manager2;
279 merge man2(in=M)
280 empaddresses(rename=(ManagerLevel1=ManagerLevel2
281 Manager1Name=Manger2Name)
282 keep=ManagerLevel1 Manager1Name);
283 by ManagerLevel2;
284 if M;
285 run;

NOTE: There were 424 observations read from the data set WORK.MAN2.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGER2 has 424 observations and 9 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

286
287 proc sort data=manager2 out=man3;
288 by ManagerLevel3;
```

```

289 run;

NOTE: There were 424 observations read from the data set WORK.MANAGER2.
NOTE: The data set WORK.MAN3 has 424 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

290
291 data manager3;
292 merge man3(in=M)
293 empaddresses(rename=(ManagerLevel1=ManagerLevel3
294 Manager1Name=Manger3Name)
295 keep=ManagerLevel1 Manager1Name);
296 by ManagerLevel3;
297 if M;
298 run;

NOTE: There were 424 observations read from the data set WORK.MAN3.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGER3 has 424 observations and 10 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

299
300 proc sort data=manager3 out=man4;
301 by ManagerLevel4;
302 run;

NOTE: There were 424 observations read from the data set WORK.MANAGER3.
NOTE: The data set WORK.MAN4 has 424 observations and 10 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

303
304 data manager4;
305 merge man4(in=M)
306 empaddresses(rename=(ManagerLevel1=ManagerLevel4
307 Manager1Name=Manger4Name)
308 keep=ManagerLevel1 Manager1Name);
309 by ManagerLevel4;
310 if M;
311 run;

NOTE: There were 424 observations read from the data set WORK.MAN4.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGER4 has 424 observations and 11 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

312
313 proc sort data=manager4 out=man5;
314 by ManagerLevel5;
315 run;

```

```

NOTE: There were 424 observations read from the data set WORK.MANAGER4.
NOTE: The data set WORK.MAN5 has 424 observations and 11 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

316
317 data manager5;
318 merge man5(in=M)
319 empaddresses(rename=(ManagerLevel1=ManagerLevel5
320 Manager1Name=Manger5Name)
321 keep=ManagerLevel1 Manager1Name);
322 by ManagerLevel5;
323 if M;
324 run;

NOTE: There were 424 observations read from the data set WORK.MAN5.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGER5 has 424 observations and 12 variables.
NOTE: DATA statement used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

325
326 proc sort data=manager5 out=man6;
327 by ManagerLevel6;
328 run;

NOTE: There were 424 observations read from the data set WORK.MANAGER5.
NOTE: The data set WORK.MAN6 has 424 observations and 12 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

329
330 data managernames;
331 merge man6(in=M)
332 empaddresses(rename=(ManagerLevel1=ManagerLevel6
333 Manager1Name=Manger6Name)
334 keep=ManagerLevel1 Manager1Name);
335 by ManagerLevel6;
336 if M;
337 run;

NOTE: There were 424 observations read from the data set WORK.MAN6.
NOTE: There were 424 observations read from the data set WORK.EMPADDRESSES.
NOTE: The data set WORK.MANAGERNAMES has 424 observations and 13 variables.
NOTE: DATA statement used (Total process time):
 real time 0.04 seconds
 cpu time 0.01 seconds

```

- b. Print observations 420 through 424 of the **managernames** data set.

- 1) Enter the following PROC SORT and PRINT steps in the Editor window:

## Partial p307s03

```

proc sort data=managernames;
 by EmployeeID;
run;
proc print data=managernames(firstobs=420 obs=424);
 title 'Partial ManagerNames Data';
 title2 'FirstObs=420';
run;
title;

```

- 2) View the output to verify that it matches the expected output.
- 3) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

344 proc print data=managernames(firstobs=420 obs=424);
345 title 'Partial ManagerNames Data';
346 title2 'FirstObs=420';
347 run;

NOTE: There were 5 observations read from the data set WORK.MANAGERNAMES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 4. Combining Data Sets Using an Index to Create One Data Set

- a. Modify p307e04 to create a SAS data set named **salesemps**. Use the index on **EmployeeID** to combine the two data sets, **orion.salesstaff** and **orion.organizationdim**. Read only the variables **EmployeeID**, **Department**, **Section**, and **OrgGroup** from **orion.organizationdim**.

- 1) Write the DATA step to combine the **orion.salesstaff** and **orion.organizationdim** data sets. Use the **EmployeeID** index.

- a) Enter the following DATA step in the Editor window:

## Partial p307s04

```

data salesemps;
 set orion.salesstaff;
 set orion.organizationdim(keep=EmployeeID Department
 Section OrgGroup)key=EmployeeID;
 if _IORC_=0;
run;

```

- b) Submit the DATA step.

- b. Check the SAS log to ensure that you do not have any data errors.

## Partial SAS Log

```

174 data salesemps;
175 set orion.salesstaff;
176 set orion.organizationdim(keep=EmployeeID Department Section OrgGroup)
177 key=EmployeeID;
178 if _IORC_=0;
179 run;

NOTE: There were 163 observations read from the data set ORION.SALESSTAFF.

```

```
NOTE: The data set WORK.SALESEMP has 163 observations and 13 variables.
NOTE: DATA statement used (Total process time):
 real time 0.12 seconds
 cpu time 0.03 seconds
```

- c. Print the first five observations of the **salesemps** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s04

```
proc print data=salesemps(obs=5) noobs;
 title 'Sales Employee Data';
 title2 '(First 5 Observations)';
run;
title;
```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

```
215 proc print data=salesemps(obs=5) noobs;
216 title 'Sales Employee Data';
217 title2 '(First 5 Observations)';
218 run;
```

```
NOTE: There were 5 observations read from the data set WORK.SALESEMP.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

## 5. Combining Data Sets Using an Index to Monitor Data Integrity

- a. Modify **p307e05** to create a SAS data set named **shoes** and a SAS data set named **errors**. Use an index on **ProductID** to combine the two data sets **orion.shoevendors** and **orion.shoeprices**.

- 1) Enter the following DATA step in the Editor window:

Partial p307s05

```
data shoes errors(keep=ProductID ProductName SupplierName);
 set orion.shoevendors(keep=ProductID ProductName
 SupplierName MfgSuggestedRetailPrice);
 set orion.shoeprices(keep=ProductID TotalRetailPrice
 CostPriceperUnit)
 key=ProductID;
 if _IORC_=0 then output shoes;
 else do;
 ERROR=0;
 output errors;
 end;
run;
```

- 2) Submit the DATA step.

- 3) Check the SAS log to ensure that you do not have any data errors. Verify that **work.shoes** and **work.errors** were both created with the correct number of observations and variables.

Partial SAS Log

```

16 data shoes errors(keep=ProductID ProductName SupplierName);
17 set orion.shoevendors(keep=ProductID ProductName SupplierName
18 MfgSuggestedRetailPrice);
19 set orion.shoeprices(keep=ProductID TotalRetailPrice CostPriceperUnit)
20 key=ProductID;
21 if _IORC_=0 then output shoes;
22 else do;
23 _ERROR_=0;
24 output errors;
25 end;
26 run;
NOTE: There were 361 observations read from the data set ORION.SHOEVENDORS.
NOTE: The data set WORK.SHES has 357 observations and 6 variables.
NOTE: The data set WORK.ERRORS has 4 observations and 3 variables.
NOTE: DATA statement used (Total process time):
 real time 0.10 seconds
 cpu time 0.03 seconds

```

- b. Print the first five observations of the **shoes** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p308s05

```

proc print data=shoes(obs=5) noobs;
 title 'The Shoes Data Set';
 title2 'First 5 Observations';
run;

```

- 2) Submit the PROC PRINT step.  
3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

35 proc print data=shoes(obs=5);
36 title 'The Shoes Data Set';
37 title2 'First 5 Observations';
38 run;
NOTE: There were 5 observations read from the data set WORK.SHES.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

- 4) View the output to verify that it matches the expected output.  
c. Print the **errors** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s05

```

proc print data=errors noobs;
 title 'The errors Data Set';
run;

```

- 2) Submit the PROC PRINT step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

38 proc print data=errors;
39 title 'The errors Data Set';
40 run;
NOTE: There were 4 observations read from the data set WORK.ERRORS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

- 4) View the output to verify that it matches the expected output.

- d. Submit the PROC Datasets step to delete the ProductID index.

Partial SAS Log

```

41 proc datasets lib=orion nolist;
42 modify shoeprices;
43 index delete ProductID;
NOTE: All indexes defined on ORION.SHOEPRICES.DATA have been deleted.
44 quit;

NOTE: MODIFY was successful for ORION.SHOEPRICES.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 6. Combining Data Sets Using an Index and Using the Macro Facility to Monitor Errors

- a. Create a data set named **processedorders** that contains the variables from **orion.firstinternetorder** and a variable named **Comment**. Use the index on the variable **OrderID** to retrieve the matching observation from **orion.internet**.

- 1) Enter the following DATA step in the Editor window:

Partial p307s06

```

data processedorders;
length Comment $30;
set orion.firstinternetorder;
set orion.internet key=OrderID;
select (_iorc_);
when (%sysrc(_sok)) do;
 Comment='Order has been processed.';
 output;
end;
when (%sysrc(_dsenom)) do;
 ERROR=0;
 Comment='Order has not been processed.';
 output;
end;
otherwise;
end;
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

43 data processedorders;
44 length Comment $30;
45 set orion.firstinternetorder;
46 set orion.internet key=OrderID;
47 select (_iorc_);
48 when (%sysrc(_sok)) do;
49 Comment='Order has been processed.';
50 output;
51 end;
52 when (%sysrc(_dsenom)) do;
53 _ERROR_=0;
54 Comment='Order has not been processed.';
55 output;
56 end;
57 otherwise;
58 end;
59 run;
NOTE: There were 38 observations read from the data set ORION.FIRSTINTERNETORDER.
NOTE: The data set WORK.PROCESSEDORDERS has 38 observations and 12 variables.
NOTE: DATA statement used (Total process time):
 real time 0.23 seconds
 cpu time 0.07 seconds

```

- b. Print observations six through 10 of **processedorders**.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s06

```

proc print data=processedorders(firstobs=6 obs=10) noobs;
 title 'Internet Orders';
 title2 'Observations 6-10';
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

67 proc print data=processedorders(firstobs=6 obs=10) noobs;
68 title 'Internet Orders';
68 title2 'Observations 6-10';
69 run;
NOTE: There were 5 observations read from the data set WORK.PROCESSEDORDERS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

## 7. Combining Summary Data Containing an Average with Detail Data

- a. Use PROC SUMMARY to store the average age of all customers in a temporary file **work.average**.

- 1) Enter the following PROC SUMMARY step in the Editor window:

Partial p307s07

```
proc summary data=orion.customerdim;
 var CustomerAge;
 output out=average mean=AvgAge;
run;
```

- 2) Submit the PROC SUMMARY step.

- 3) Examine the data values portion of **work.average**. The value for **AvgAge** should be **41.9740**.

Partial p307s07

| This is the average Data Set |        |        |         |
|------------------------------|--------|--------|---------|
| Obs                          | _TYPE_ | _FREQ_ | AvgAge  |
| 1                            | 0      | 77     | 41.9740 |

- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
1 proc summary data=orion.customerdim;
2 var CustomerAge;
3 output out=average mean=AvgAge;
4 run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.AVERAGE has 1 observations and 3 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
 real time 0.40 seconds
 cpu time 0.17 seconds

5 proc print data=work.average;
6 title 'This is the Average data set';
7 run;
NOTE: There were 1 observations read from the data set WORK.AVERAGE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.04 seconds
 cpu time 0.03 seconds
```

- b. Create a SAS data set named **agedif**. It combines **work.average** with the **orion.customerdim** in order to determine the difference between each customer's age and the average for all customers. (You can use any method presented in this section.)

### PROC SUMMARY and the DATA Step

- 1) Enter the following DATA step in the Editor window:

Partial p307s07

```
data agedif;
 if _N_=1 then set average(keep=AvgAge);
 set orion.customerdim(keep=CustomerID CustomerAge);
 AgeDifference=CustomerAge-AvgAge;
run;
```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

12 data agedif;
13 if _N_=1 then set average(keep=AvgAge);
14 set orion.customerdim(keep=CustomerID CustomerAge);
15 AgeDifference=CustomerAge-AvgAge;
16 run;
NOTE: There were 1 observations read from the data set WORK.AVERAGE.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.AGEDIF has 77 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 25.02 seconds
 cpu time 0.12 seconds

```

### PROC SUMMARY and PROC SQL

- 4) Enter the following PROC SQL step in the Editor window:

Partial p307s07

```

proc sql;
 create table agedif as
 select AvgAge,
 CustomerID,
 CustomerAge,
 CustomerAge-AvgAge as AgeDifference
 from orion.customerdim,
 average;
quit;

```

- 5) Submit the PROC SQL step.
- 6) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

17 proc sql;
18 create table agedif as
19 select AvgAge,
20 CustomerID,
21 CustomerAge,
22 CustomerAge-AvgAge as AgeDifference
23 from orion.customerdim,
24 average;
NOTE: The execution of this query involves performing one or more Cartesian product
 joins that can not be optimized.
NOTE: Table WORK.AGEDIF created, with 77 rows and 4 columns.
25 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

### PROC SQL Only

- 7) Enter the following PROC SQL step in the Editor window:

## Partial p307s07

```

proc sql;
 create table agedif as
 select mean(CustomerAge) as AvgAge,
 CustomerID,
 CustomerAge,
 CustomerAge-calculated AvgAge as AgeDifference
 from orion.customerdim;
quit;

```

- 8) Submit the PROC SQL step.
- 9) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

26 proc sql;
27 create table agedif as
28 select mean(CustomerAge) as AvgAge,
29 CustomerID,
30 CustomerAge,
31 CustomerAge-calculated AvgAge as AgeDifference
32 from orion.customerdim;
NOTE: The query requires remerging summary statistics back with the original data.
NOTE: Table WORK.AGEDIF created, with 77 rows and 4 columns.
33 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

**DATA Step Only**

- 10) Enter the following DATA step in the Editor window:

## Partial p307s07

```

data agedif;
 drop i TotAge;
 if _N_=1 then do i=1 to TotObs;
 set orion.customerdim(keep=CustomerAge) nobs=TotObs;
 TotAge+CustomerAge;
 end;
 set orion.customerdim(keep=CustomerID CustomerAge);
 AvgAge=TotAge/TotObs;
 AgeDifference=CustomerAge-AvgAge;
run;

```

- 11) Submit the DATA step.
- 12) Examine the SAS log to verify that the step executed without errors.

## Partial SAS Log

```

34 data agedif;
35 drop i TotAge;
36 if _N_=1 then do i=1 to TotObs;
37 set orion.customerdim(keep=CustomerAge) nobs=TotObs;
38 TotAge+CustomerAge;

```

```

39 end;
40 set orion.customerdim(keep=CustomerID CustomerAge);
41 AvgAge=TotAge / TotObs;
42 AgeDifference=CustomerAge-AvgAge;
43 run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: There were 77 observations read from the data set ORION.CUSTOMERDIM.
NOTE: The data set WORK.AGEDIF has 77 observations and 4 variables.
NOTE: DATA statement used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds

```

- c. Print the first five observations of the **agedif** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s07

```

proc print data=agedif(obs=5) noobs;
 var AvgAge CustomerID CustomerAge AgeDifference;
 title 'The agedif Data Set';
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

```

45 proc print data=agedif(obs=5) noobs;
45 var AvgAge CustomerID CustomerAge AgeDifference;
46 title 'The agedif Data Set';
47 run;
NOTE: There were 5 observations read from the data set WORK.AGEDIF.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 33.41 seconds
 cpu time 0.04 seconds
48
49 title;

```

## 8. Combining Summary Data with Multiple Statistics with Detail Data

- a. Select any method to create a SAS data set named **compare** by performing the following tasks:

### PROC SUMMARY and the DATA Step

- 1) Enter the following DATA and PROC SUMMARY steps in the Editor window:

Partial p307s08

```

data donations;
 set orion.employeedonations;
 TotalDonation=sum(of Qtr1-Qtr4);
run;
proc summary data=donations;
 var TotalDonation;
 output out=totals mean=AvgDonation;
run;
data compare;

```

```

if _N_=1 then set totals;
set donations;
Difference=TotalDonation-AvgDonation;
run;

```

- 2) Submit the three steps.
- 3) Examine the SAS log to verify that the steps executed without errors.

#### Partial SAS Log

```

131 data donations;
132 set orion.employeedonations;
133 TotalDonation=sum(of Qtr1-Qtr4);
134 run;
NOTE: There were 124 observations read from the data set ORION.EMPLOYEEDONATIONS.
NOTE: The data set WORK.DONATIONS has 124 observations and 8 variables.
NOTE: DATA statement used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
135
136 proc summary data=donations;
137 var TotalDonation;
138 output out=totals mean=AvgDonation;
139 run;
NOTE: There were 124 observations read from the data set WORK.DONATIONS.
NOTE: The data set WORK.TOTALS has 1 observations and 3 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
140
141 data compare;
142 if _N_=1 then set totals;
143 set donations;
144 Difference=TotalDonation-AvgDonation;
145 run;
NOTE: There were 1 observations read from the data set WORK.TOTALS.
NOTE: There were 124 observations read from the data set WORK.DONATIONS.
NOTE: The data set WORK.COMPARE has 124 observations and 12 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

#### PROC SUMMARY and PROC SQL

- 4) Enter the following PROC SQL and PROC SUMMARY steps in the Editor window:

#### Partial p307s08

```

proc sql;
 create table donations as
 select EmployeeID,
 Qtr1,
 Qtr2,
 Qtr3,
 Qtr4,
 Recipients,
 PaidBy,

```

```

 sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonation
 from orion.employeedonations;
proc summary data=donations;
 var TotalDonation;
 output out=totals mean=AvgDonation;
run;
proc sql;
 create table compare as
 select AvgDonation,
 donations.*,
 TotalDonation-AvgDonation as Difference
 from totals,
 donations;
quit;

```

- 5) Submit the three steps.
- 6) Examine the SAS log to verify that the steps executed without errors.

#### Partial SAS Log

```

146 proc sql;
147 create table donations as
148 select EmployeeID,
149 Qtr1,
150 Qtr2,
151 Qtr3,
152 Qtr4,
153 Recipients,
154 PaidBy,
155 sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonation
156 from orion.employeedonations;
NOTE: Table WORK.DONATIONS created, with 124 rows and 8 columns.
157
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
158 proc summary data=donations;
159 var TotalDonation;
160 output out=totals mean=AvgDonation;
161 run;
NOTE: There were 124 observations read from the data set WORK.DONATIONS.
NOTE: The data set WORK.TOTALS has 1 observations and 3 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
162
163 proc sql;
164 create table compare as
165 select AvgDonation,
166 donations.*,
167 TotalDonation-AvgDonation as Difference
168 from totals,
169 donations;
NOTE: The execution of this query involves performing one or more Cartesian product
 Joins that can not be optimized.
NOTE: Table WORK.COMPARE created, with 124 rows and 10 columns.
170 quit;

```

```
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

### PROC SQL Only

- 7) Enter the following PROC SQL step in the Editor window:

Partial p307s08

```
proc sql;
 create table compare as
 select mean(sum(Qtr1, Qtr2, Qtr3, Qtr4)) as AvgDonation,
 EmployeeDonations.*,
 sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonation,
 calculated TotalDonation-calculated AvgDonation as
 Difference
 from orion.employeedonations;
quit;
```

- 8) Submit the PROC SQL step.  
 9) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
171 proc sql;
172 create table compare as
173 select mean(sum(Qtr1, Qtr2, Qtr3, Qtr4)) as AvgDonation,
174 EmployeeDonations.*,
175 sum(Qtr1, Qtr2, Qtr3, Qtr4) as TotalDonation,
176 calculated TotalDonation-calculated AvgDonation as Difference
177 from orion.employeedonations;
NOTE: The query requires remerging summary statistics back with the original data.
NOTE: Table WORK.COMPARE created, with 124 rows and 10 columns.
178 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
```

### DATA Step Only

- 10) Enter the following DATA step in the Editor window:

Partial p307s08

```
data compare;
 drop i;
 if _N_=1 then do i=1 to TotObs;
 set orion.employeedonations(keep=Qtr1-Qtr4) nobs=TotObs;
 Total+sum(of Qtr1-Qtr4);
 end;
 set orion.employeedonations;
 TotalDonation=sum(of Qtr1-Qtr4);
 AvgDonation=Total/TotObs;
 Difference=TotalDonation-AvgDonation;
run;
```

- 11) Submit the DATA step.
- 12) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

179 data compare;
180 drop i;
181 if _N_=1 then do i=1 to TotObs;
182 set orion.employeedonations(keep=Qtr1-Qtr4) nobs=TotObs;
183 Total+sum(of Qtr1-Qtr4);
184 end;
185 set orion.employeedonations;
186 TotalDonation=sum(of Qtr1-Qtr4);
187 AvgDonation=Total/TotObs;
188 Difference=TotalDonation-AvgDonation;
189 run;
NOTE: There were 124 observations read from the data set ORION.EMPLOYEEDONATIONS.
NOTE: There were 124 observations read from the data set ORION.EMPLOYEEDONATIONS.
NOTE: The data set WORK.COMPARE has 124 observations and 11 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

- b. Print the first five observations of the **compare** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s08

```

proc print data=compare(obs=5) noobs;
 var EmployeeID Qtr1 Qtr2 Qtr3 Qtr4
 TotalDonation AvgDonation Difference;
 title 'The compare Data Set';
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

190 proc print data=compare(obs=5) noobs;
191 var AvgDonation EmployeeID Qtr1 Qtr2 Qtr3 Qtr4
192 Recipients PaidBy TotalDonation Difference;
193 title 'The compare Data Set';
194 run;
NOTE: There were 5 observations read from the data set WORK.COMPARE.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.00 seconds
 cpu time 0.00 seconds
195
196 title;

```

## 9. Combining Summary Data Containing a Weighted Average and Detail Data

- a. Select any method to create a SAS data set named **products** by performing the following tasks:

## PROC SUMMARY and the DATA Step

- 1) Enter the following PROC SUMMARY, PROC SORT, and DATA steps in the Editor window:

Partial p307s09

```
proc summary data=orion.orderfact;
 var CostPricePerUnit;
 weight Quantity;
 output out=totals sum=TotalCost;
run;
proc sort data=orion.orderfact out=orderfact;
 by ProductID;
run;
data products(keep=CustomerID CostPricePerUnit Quantity
 Percent ProductName);
 if _N_=1 then set totals(keep=TotalCost);
 merge orderfact(keep=CustomerID ProductID CostPricePerUnit
 Quantity in=O)
 orion.productdim(keep=ProductID ProductName in=P);
 by ProductID;
 if O and P;
 Percent=(CostPricePerUnit*Quantity)/TotalCost;
 format Percent percent9.3;
run;
```

- 2) Submit the three steps.  
 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
227 proc summary data=orion.orderfact;
228 var CostPricePerUnit;
229 weight Quantity;
230 output out=totals sum=TotalCost;
231 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.TOTALS has 1 observations and 3 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
 real time 0.03 seconds
 cpu time 0.01 seconds
232
233 proc sort data=orion.orderfact out=orderfact;
234 by ProductID;
235 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.ORDERFACT has 617 observations and 12 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.06 seconds
 cpu time 0.03 seconds
236
237 data products(keep=CustomerID CostPricePerUnit Quantity Percent ProductName);
238 if _N_=1 then set totals(keep=TotalCost);
239 merge orderfact(keep=CustomerID ProductID CostPricePerUnit Quantity in=O)
 orion.productdim(keep=ProductID ProductName in=P);
240 by ProductID;
```

```

242 if 0 and P;
243 Percent=(CostPricePerUnit*Quantity)/TotalCost;
244 format Percent percent9.3;
245 run;
NOTE: There were 1 observations read from the data set WORK.TOTALS.
NOTE: There were 617 observations read from the data set WORK.ORDERFACT.
NOTE: There were 481 observations read from the data set ORION.PRODUCTDIM.
NOTE: The data set WORK.PRODUCTS has 617 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.04 seconds
 cpu time 0.03 seconds

```

## PROC SUMMARY and PROC SQL

- 4) Enter the following PROC SUMMARY and SQL steps in the Editor window:

Partial p307s09

```

proc summary data=orion.orderfact;
 var CostPricePerUnit;
 weight Quantity;
 output out=totals sum=TotalCost;
run;
proc sql;
 create table products as
 select CustomerID,
 CostPricePerUnit,
 Quantity,
 ProductName,
 (Quantity*CostPricePerUnit)/TotalCost as Percent
 format=percent9.3
 from totals,
 orion.orderfact,
 orion.productdim
 where orderfact.ProductID=productdim.ProductID;
quit;

```

- 5) Submit the two steps.  
6) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

216 proc summary data=orion.orderfact;
217 var CostPricePerUnit;
218 weight Quantity;
219 output out=totals sum=TotalCost;
220 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.TOTALS has 1 observations and 3 variables.
NOTE: PROCEDURE SUMMARY used (Total process time):
 real time 0.79 seconds
 cpu time 0.01 seconds
221
222 proc sql;
223 create table products as

```

```

224 select CustomerID,
225 CostPricePerUnit,
226 Quantity,
227 ProductName,
228 (Quantity*CostPricePerUnit)/TotalCost as Percent format=percent9.3
229 from totals,
230 orion.orderfact,
231 orion.productdim
232 where orderfact.ProductID=productdim.ProductID;
NOTE: The execution of this query involves performing one or more Cartesian product
 Joins that can not be optimized.
NOTE: Table WORK.PRODUCTS created, with 617 rows and 5 columns.
233 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

```

**PROC SQL Only**

- 7) Enter the following PROC SQL step in the Editor window:

Partial p307s09

```

proc sql;
 create table products as
 select CustomerID,
 CostPricePerUnit,
 Quantity,
 ProductName,
 (Quantity*CostPricePerUnit) /
 sum(Quantity*CostPricePerUnit)
 as Percent format=percent9.3
 from orion.orderfact,
 orion.productdim
 where orderfact.ProductID=productdim.ProductID;
quit;

```

- 8) Submit the PROC SQL step.  
 9) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

246 proc sql;
247 create table products as
248 select CustomerID,
249 CostPricePerUnit,
250 Quantity,
251 ProductName,
252 (Quantity*CostPricePerUnit)/sum(Quantity*CostPricePerUnit)
253 as Percent format=percent9.3
254 from orion.orderfact,
255 orion.productdim
256 where orderfact.ProductID=productdim.ProductID;
NOTE: The query requires remerging summary statistics back with the original data.
NOTE: Table WORK.PRODUCTS created, with 617 rows and 5 columns.
257 quit;

```

|                                                |
|------------------------------------------------|
| NOTE: PROCEDURE SQL used (Total process time): |
| real time           0.96 seconds               |
| cpu time           0.01 seconds                |

### DATA Step Only

- 10) Enter the following DATA step in the Editor window:

Partial p307s09

```
proc sort data=orion.orderfact out=orderfact;
 by ProductID;
run;
data products(keep=CustomerID CostPricePerUnit Quantity
 Percent ProductName);
 if _N_=1 then do i=1 to TotObs;
 set orion.orderfact nobs=TotObs;
 TotalCost+(Quantity*CostPricePerUnit);
 end;
 merge orderfact(keep=CustomerID ProductID CostPricePerUnit
 Quantity in=0)
 orion.productdim(keep=ProductID ProductName in=P);
 by ProductID;
 if O and P;
 Percent=(CostPricePerUnit*Quantity)/TotalCost;
 format Percent percent9.3;
run;
```

- 11) Submit the DATA step.  
 12) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
258 proc sort data=orion.orderfact out=orderfact;
259 by ProductID;
260 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.ORDERFACT has 617 observations and 12 variables.
NOTE: PROCEDURE SORT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
261
262 data products(keep=CustomerID CostPricePerUnit Quantity Percent ProductName);
263 if _N_=1 then do i=1 to TotObs;
264 set orion.orderfact nobs=TotObs;
265 TotalCost+(Quantity*CostPricePerUnit);
266 end;
267 merge orderfact(keep=CustomerID ProductID CostPricePerUnit Quantity in=0)
 orion.productdim(keep=ProductID ProductName in=P);
268 by ProductID;
269 if O and P;
270 Percent=(CostPricePerUnit*Quantity)/TotalCost;
271 format Percent percent9.3;
272 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: There were 617 observations read from the data set WORK.ORDERFACT.
```

```

NOTE: There were 481 observations read from the data set ORION.PRODUCTDIM.
NOTE: The data set WORK.PRODUCTS has 617 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.03 seconds
 cpu time 0.03 seconds

```

- b. Print the first five observations of the **products** SAS data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p307s09

```

proc print data=products(obs=5) noobs;
 var CustomerID Quantity CostPricePerUnit ProductName Percent;
 title 'The products Data Set';
run;
title;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

274 proc print data=products(obs=5) noobs;
275 var CustomerID Quantity CostPricePerUnit ProductName Percent;
276 title 'The products Data Set';
277 run;
NOTE: There were 5 observations read from the data set WORK.PRODUCTS.
NOTE: PROCEDURE PRINT used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds
278
279 title;

```

## 10. Combining Two Data Sets Conditionally Using the SQL Procedure

- a. Use the SQL procedure to create a data set named **agegroups**, which contains the customer ID, name, age, and age group as of January 1, 2012. (The variable **Description** is in the **orion.agesmod** SAS data set.) Order the data by **CustomerID**.

- 1) Enter the following PROC SQL step in the Editor window:

Partial p307s10

```

proc sql;
 create table agegroups as
 select CustomerID,
 CustomerName,
 int(yrdif(BirthDate, '01Jan2012'd, 'AGE')) as Age,
 Description
 from orion.customer,
 orion.agesmod
 where calculated Age between FirstAge and LastAge
 order by CustomerID;

```

```
quit;
```

- 2) Submit the PROC SQL step.
- 3) Examine the SAS log to verify that the step executed without errors.

#### Partial SAS Log

```
23 proc sql;
24 create table agegroups as
25 select CustomerID,
26 CustomerName,
27 int(yrdif(BirthDate, '01Jan2012'd, 'AGE')) as Age,
28 Description
29 from orion.customer,
30 orion.agesmod
31 where calculated Age between FirstAge and LastAge
32 order by CustomerID;
NOTE: The execution of this query involves performing one or more Cartesian product
 joins that can not be optimized.
NOTE: Table WORK.AGEGROUPS created, with 77 rows and 4 columns.
33 quit;
NOTE: PROCEDURE SQL used (Total process time):
 real time 2:33.53
 cpu time 0.17 seconds
```

- b. Print the first five observations of the **agegroups** data set.
    - 1) Enter the following PROC PRINT step in the Editor window:
- Partial p307s10**
- ```
proc print data=agegroups(obs=5) noobs;
  title 'agegroups Data Set';
run;
```
- 2) Submit the PROC PRINT step.
 - 3) View the output to verify that it matches the expected output.
 - 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```
35 proc print data=agegroups(obs=5) noobs;
36   title 'agegroups Data Set';
37 run;
NOTE: There were 5 observations read from the data set WORK.AGEGROUPS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          33.19 seconds
      cpu time           0.17 seconds
```

11. Combining Two Data Sets Conditionally Using the DATA Step DO Loop

- a. Modify **p307e11** to use a DATA step to combine the **agesmod** and **customer** data sets based on a customer's age as of January 1, 2012. Create a data set named **agegroups**, which contains the customer ID, name, birthdate, age, and age description.

Hint: Use the following calculation to determine someone's age as of January 1, 2012:

```
int(yrdif(BirthDate, '01Jan2012'd, 'AGE'))
```

- 1) Enter the following DATA step in the Editor window:

Partial **p307s11**

```
data agegroups;
  keep CustomerID CustomerName BirthDate Age Description;
  set customer;
  Age=int(yrdif(BirthDate, '01Jan2012'd, 'AGE'));
  do while (not (FirstAge le Age lt LastAge));
    set orion.agesmod;
  end;
run;
```

- 2) Submit the SAS steps.
- 3) Examine the SAS log to verify that the steps executed without errors.

Partial SAS Log

```
18  proc sort data=orion.customer(keep=CustomerID BirthDate
19                                CustomerName)out=customer;
20      by descending BirthDate;
21  run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUSTOMER has 77 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.06 seconds
      cpu time          0.03 seconds
22
23  data agegroups;
24      keep CustomerID CustomerName BirthDate Age Description;
25      set customer;
26      Age=int(yrdif(BirthDate, '01Jan2012'd, 'AGE'));
27      do while (not (FirstAge le Age lt LastAge));
28        set orion.agesmod;
29      end;
30  run;
NOTE: There were 77 observations read from the data set WORK.CUSTOMER.
NOTE: There were 4 observations read from the data set ORION.AGESMOD.
NOTE: The data set WORK.AGEGROUPS has 77 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time          0.12 seconds
      cpu time          0.04 seconds
```

- b. From the **agegroups** data set, print only the observations that fall into the highest age category.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial **p307s11**

```
proc print data=agegroups noobs;
  where description='60-75 years';
  title 'Highest Age Category';
run;
```

- 2) Submit the PROC PRINT step.

- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

35  proc print data=agegroups noobs;
36  where description='60-75 years';
37  title 'Highest Age Category';
38  run;
NOTE: There were 14 observations read from the data set WORK.AGEGROUPS.
      WHERE description='60-75 years';
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds

```

12. Combining Two Data Sets Conditionally Using the DATA Step Hash Object

- a. Use the DATA step and a hash object to create a data set named **agegroups** that contains the customer ID, name, age, and age group as of January 1, 2012. (The variable **Description** is in the **orion.agesmod** SAS data set.)

- 1) Enter the following DATA step in the Editor window:

Partial p307s12

```

data agegroups;
  keep CustomerID CustomerName Age Description;
  if _N_=1 then do;
    if 0 then set orion.agesmod;
    declare hash AG(dataset: 'orion.agesmod',
                    ordered: 'ascending');
    AG.definekey('FirstAge');
    AG.definedata('FirstAge', 'LastAge', 'Description');
    AG.definedone();
    declare hiter A('AG');
  end;
  set orion.customer(keep=CustomerID BirthDate
                     CustomerName);
  Age=int(yrdif(Birth_Date, '01Jan2012'd, 'AGE'));
  A.first();
  do until (rc ne 0);
    if FirstAge <= Age < LastAge then do;
      output;
      leave;
    end;
    else if FirstAge > Age then leave;
    rc=A.next();
  end;
run;

```

- 2) Submit the DATA step.
- 3) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

37  data agegroups;
38  keep CustomerID CustomerName Age Description;

```

```

39      if _N_=1 then do;
40          if 0 then set orion.agesmod;
41          declare hash AG(dataset: 'orion.agesmod', ordered: 'ascending');
42          AG.defineKey('FirstAge');
43          AG.defineData('FirstAge', 'LastAge', 'Description');
44          AG.defineDone();
45          declare hiter A('AG');
46      end;
47      set orion.customer(keep=CustomerID BirthDate CustomerName);
48      Age=int(yrdif(BirthDate, '01Jan2012'd, 'AGE'));
49      A.first();
50      do until (rc ne 0);
51          if FirstAge <= Age < LastAge then do;
52              output;
53              leave;
54          end;
55          else if FirstAge > Age then leave;
56          rc=A.next();
57      end;
58  run;
NOTE: There were 4 observations read from the data set ORION.AGESMOD.
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.AGEGRUPGS has 77 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.32 seconds
      cpu time          0.09 seconds

```

- b. Print the first five observations of the **agegroups** data set.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial **p307s12**

```

proc print data=agegroups(obs=5) noobs;
    title 'agegroups';
run;

```

- 2) Submit the PROC PRINT step.
- 3) View the output to verify that it matches the expected output.
- 4) Examine the SAS log to verify that the step executed without errors.

Partial SAS Log

```

35  proc print data=agegroups(obs=5) noobs;
36      title 'agegroups';
37  run;
NOTE: There were 5 observations read from the data set WORK.AGEGRUPGS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          33.19 seconds
      cpu time          0.17 seconds

```

Solutions to Student Activities (Polls/Quizzes)

7.01 Multiple Choice Poll – Correct Answer

If the data set **country** has seven observations and the data set **orion.continent** has five observations, what stops the execution of the DATA step that contains a MERGE statement?

- a. the end of file for **country**
- b. the end of file for **orion.continent**
- c.** the end of file for both data sets

The DATA step sequentially reads and attempts to match all observation from all data sets that are listed in the MERGE statement.

13

7.02 Quiz – Correct Answer

Data set **three** contains 3 observations where the value of **X** is 1.

DATA Step



one		two		three		
X	Y	X	Z	X	Y	Z
1	a	1	f	1	a	f
1	d	1	r	1	d	r
4	c	1	s	1	d	s
5	g	3	t	3		t
		4	w	4	c	w
				5	g	

```
data three;
  merge one two;
  by x;
run;
```

NOTE: MERGE statement has more than one data set with repeats of BY values.

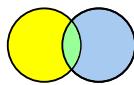
15

p307d02

7.03 Multiple Choice Poll – Correct Answer

Which type of SQL join most closely mimics a DATA step merge?

- a. inner join
- b. full outer join**
- c. left outer join
- d. right outer join



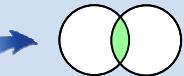
Like the DATA step, both matches and nonmatches are returned by a full outer join.

18

7.04 Quiz – Correct Answer

Data set **three** contains **6** observations where the value of **X** is **1**.

PROC SQL



Matches only

one	two	three
X	Z	X
1 a	1 f	1 a f
1 d	1 r	1 a r
4 c	1 s	1 a s
5 g	3 t	1 d F
	4 w	1 d r

one	two	three
X	Z	X
1 a	1 f	1 a f
1 d	1 r	1 a r
4 c	1 s	1 a s
5 g	3 t	1 d F
	4 w	1 d r

```
proc sql;
  create table three as
    select one.* , two.Z
    from one, two
    where one.X=two.X;
quit;
```

P307d03

26

7.05 Multiple Choice Poll – Correct Answer

If you use a DATA step merge to combine the data sets, how many observations are read from the larger data set, orion.customerdimmore?

- a. 38
- b. 1,500**

The MERGE statement reads all observations from all of the input data sets sequentially. If there is a great disparity in size between the input data sets, merging might not be the best technique for combining them.

41

7.06 Quiz – Correct Answer

Along with other variable values, the CustomerName value was retained from the previous successful match.

Partial orion.catalog

Customer ID	OrderID	Quantity	TotalRetail Price
5	1230080101	1	247.50
15	1240080101	3	216.50
5	1230080101	1	247.50

Partial orion.customerdimmore

Customer ID	Customer Country	Customer Name
4	US	James Kvarnig
5	US	Sandrina Stephano
9	DE	Cornelia Krahl

Partial SAS Log

```
CustomerID=15 OrderID=1240080101 Quantity=3 TotalRetailPrice=$216.50
CustomerCountry=US CustomerGender=F CustomerName=Sandrina Stephano
CustomerFirstName=Sandrina CustomerLastName=Stephano
CustomerBirthDate=09JUL1983 CustomerAgeGroup=15-30 years CustomerType=Orion
Club Gold members medium activity CustomerGroup=Orion Club Gold members
CustomerAge=28 _ERROR_=1 _IORC_=1230015 N_=2
```

53

7.07 Quiz – Correct Answer

Edit the program **p307a01**.

1. Replace the ELSE statement with the following
ELSE DO group:

```
else do;  
    _ERROR_=0;  
    output errors;  
end;
```

2. Resubmit the program and look at the log.
Why are there no messages now?

**The contents of the PDV are printed in the log only
when _ERROR_=1.**

66

7.08 Quiz – Correct Answer

Open and submit the program **p307a02**.

1. How many observations are in the resulting data set?
only one observation

99

continued...

7.08 Quiz – Correct Answer

Open and submit the program **p307a02**.

1. How many observations are in the resulting data set?

only one observation

2. Why?

When the statement

```
set summary(keep=GrandTot);
```

executes a second time, SAS encounters the end of the file in the work.summary data set because that data set has only one observation in it.

100

continued...

7.08 Quiz – Correct Answer

Open and submit the program **p307a02**.

3. Why did the **p307d10** program generate 53 observations?

The statement

```
set summary(keep=GrandTot);
```

does not execute a second time because execution of the SET statement is controlled by the condition IF _N_=1.

101

7.09 Poll – Correct Answer

Can a DATA step merge be used for this task?

- Yes
- No

The DATA step merge requires matches to be based on equality.

114

7.10 Poll – Correct Answer

When is a DO WHILE condition checked?

- a. at the start of the loop
- b. at the end of the loop

DO WHILE conditions are checked at the top of the loop. If the condition is false, the loop does not execute.

122

7.11 Multiple Choice Poll – Correct Answer

Does SAS encounter the end-of-file marker (EOF) for **orion.rates**?

- a. SAS encounters the EOF, but it does not stop the DATA step because there are two SET statements.
- b. SAS encounters the EOF for **orion.rates**, so there are only four observations in the resulting data set **euros**.
- c. The DO WHILE statement prevents the data set **orion.rates** from being read a fifth time, so the EOF is never encountered.

Chapter 8 User-Defined Functions and Formats

8.1 User-Defined Functions	8-3
Demonstration: Creating and Using a User-Defined Function	8-11
Demonstration: Using a User-Defined Function in an SQL Step	8-14
Exercises	8-16
8.2 User-Defined Formats	8-18
Demonstration: Creating Formats from Functions.....	8-22
Demonstration: Creating Functions from Formats.....	8-27
Demonstration: User-Defined Informats	8-38
Exercises	8-41
8.3 Solutions	8-43
Solutions to Exercises	8-43
Solutions to Student Activities (Polls/Quizzes)	8-51

8.1 User-Defined Functions

Objectives

- Introduce the FCMP procedure.
- Examine the syntax for the FCMP procedure.
- Create a function using the FCMP procedure.
- Apply the user-defined function.

3

Business Scenario

Many Orion Star data sets have employee names that are structured as *Lastname, Firstname*. To simplify production code, create a user-defined function to rearrange the name.

The diagram illustrates a transformation process. On the left, a table titled "EmployeeName" contains five rows of names: Lu, Patrick; Zhou, Tom; Dawes, Wilson; Billington, Karen; and Povey, Liz. A blue arrow points from this table to another table on the right, titled "Rearranged_Name". This second table also has five rows, where each name is rearranged to follow the pattern "Firstname Lastname": Patrick Lu; Tom Zhou; Wilson Dawes; Karen Billington; and Liz Povey.

EmployeeName	Rearranged_Name
Lu, Patrick	Patrick Lu
Zhou, Tom	Tom Zhou
Dawes, Wilson	Wilson Dawes
Billington, Karen	Karen Billington
Povey, Liz	Liz Povey

4

8.01 Quiz

Fill in the blanks:

A SAS function is a routine that accepts _____, performs a data manipulation, and returns a _____.

5

8.02 Multiple Answer Poll

Which functions can be used to extract first name and last name values and combine them as shown below?

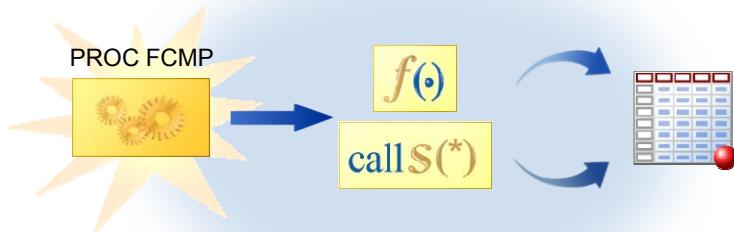
- a. SUBSTR
- b. LEFT
- c. CATX
- d. SCAN
- e. TRIM



7

PROC FCMP

Use the Function Compiler procedure to build user-defined functions and call routines with DATA step syntax. Store the user-defined functions in a SAS data set.

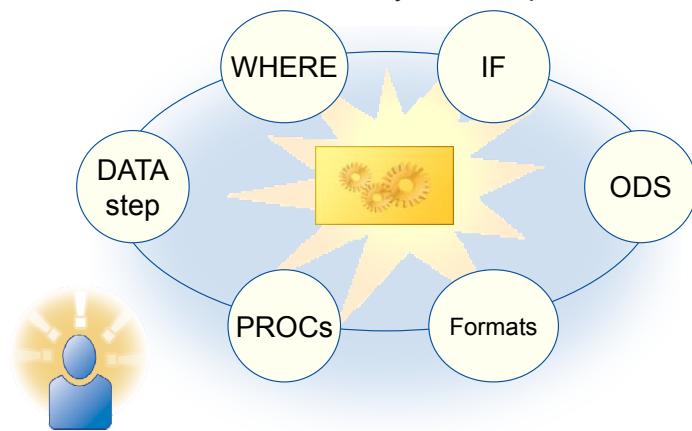


9

- A *SAS function* is a routine that accepts arguments, performs a computation or other operation, and returns either a character or numeric value.
- A *CALL routine* alters variable values or performs other system functions. CALL routines are similar to functions, but do not return a value, so you cannot use them in assignment statements or expressions.

Why Use PROC FCMP?

As with functions that are supplied by SAS, user-defined functions can be used in many different places.



10

Advantages of PROC FCMP

Here are some advantages of writing your own functions:

- ✓ can build a library of reusable routines
- ✓ simplify complex programs
- ✓ routines independent from their use
- ✓ read, write, and maintain complex code easily

11

PROC FCMP

```

proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
    return(catx(' ', scan(name, 2, ','), scan(name, 1, ',')));
  endsub;
quit;
PROC FCMP OUTLIB=libref.data-set.package;
  FUNCTION function-name(argument-1 <$>,...,
                        argument-n <$>) <$> <length>;
  programming-statements;
  RETURN(expression);
  ENDSUB;
QUIT;

```

12

p308d01

PROC FCMP Statement

The OUTLIB= option in the PROC FCMP statement specifies the three-level name of an output package to which the compiled functions are written.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
  return(catx(' ', scan(name, 2, ' '), scan(name, 1, ' ')));
  endsub;
  PROC FCMP OUTLIB=libref.data-set.package;
quit;
```

- ✍ The **dev** package is a collection of routines that have unique names and are stored in the **orion.functions** data set.

13

p308d01

The OUTLIB= or OUTCAT= argument is required.

A *package* is any collection of related routines that are specified by the user. It is a way of grouping related subroutines and functions within a data set.

FUNCTION Statement

A function definition begins with the FUNCTION statement and ends with an ENDSUB statement. Define the function attributes and specify the arguments in the FUNCTION statement.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
  ...
  endsub;
quit; FUNCTION function-name(argument-1 <$>,...,
                                argument-n <$>) <$> <length>;
  ...
ENDSUB;
```

- ✍ Multiple functions can be defined per FCMP step. Do not use the same name as a function supplied by SAS.

14

p308d01

FUNCTION Statement

Character arguments must be followed by a dollar sign. In this example, the value returned by the ReverseName function is a character value with a maximum length of 40 characters.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
  ...
  endsub;
quit; FUNCTION function-name(argument-1 <$>,...,
          argument-n <$>) <$> <length>;
          programming-statements;
          RETURN(expression);
ENDSUB;
```

15

p308d01

RETURN Statement

A RETURN statement is required for each function definition. It defines the expression that generates the value that the function returns.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
  return(catx('',scan(name,2,','),scan(name,1,',')));
  endsub;
quit; RETURN(expression);
```

16

p308d01

RETURN Statement

A RETURN statement can be embedded within conditional expressions if needed.

```
proc fcmp outlib=orion.functions.dev;
function ReverseName(name $) $ 40;
if gender='F' then
  return(catx(' ','Ms.',scan(name,2,','),scan(name,1,',')));
else if gender='M' then
  return(catx(' ','Mr.',scan(name,2,','),scan(name,1,',')));
else
  return(catx(' ',scan(name,2,','),scan(name,1,',')));
endsub;
quit;
```

RETURN(expression);

17

p308d01

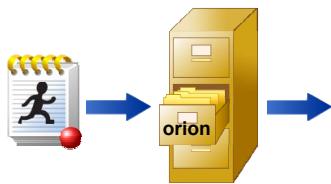


Not all DATA step statements are available in PROC FCMP. Some DATA step statements are available but behave differently in PROC FCMP. Consult the SAS documentation for more specifics about PROC FCMP and DATA step differences.

Accessing the Newly Defined Function

The CMPLIB= SAS system option specifies one or more data sets that SAS searches for user-defined function entries. The default is **work.functions**.

```
options cmplib=orion.functions;
CMPLIB=libref.data-set | (libref.data-set-1 ... libref.data-set-n)
```



Partial PROC PRINT Output

EmployeeName	FMLName
Abbott, Ray	Ray Abbott
Aisbitt, Sandy	Sandy Aisbitt
Akinfolarin, Tameaka	Tameaka Akinfolarin
Amos, Salley	Salley Amos
Anger, Rose	Rose Anger
Anstey, David	David Anstey

18

8.03 Poll

The CMPLIB option is required to create user-defined functions in PROC FCMP.

- True
- False

19

Accessing a User-Defined Function

Use the function in the DATA step to create **FMLName**, which stores the rearranged name value.

```
options cmplib=orion.functions;
data work.emplist;
  set orion.employeeaddresses;
  FMLName=ReverseName (EmployeeName) ;
  keep EmployeeID EmployeeName FMLName;
run;
```

21

p308d01

Accessing a User-Defined Function

The ReverseName function uses the **EmployeeName** value as the argument to return the rearranged **FMLName**.

```

options cmplib=orion.functions;
data work.empList;
  set orion.employeeaddresses;
  FMLName=ReverseName(EmployeeName);
  keep EmployeeID EmployeeName FMLName;
run;
proc fcmp outlib=orion.functions.HR;
  function ReverseName(name $) $40;
    return(catx(' ',scan(name,2,'.'),scan(name,1,'.')));
  endsub;
quit;

```

PDV

EmployeeID	EmployeeName	...	Country	FMLName
N 8	\$ 40	D..	D \$ 2	\$ 40
121044	Abbott, Ray	...	US	Ray Abbott

22

p308d01

8.04 Quiz

Where did the type and length attributes for the new variable come from?

PDV

EmployeeID	EmployeeName	...	Country	FMLName
N 8	\$ 40		\$ 2	\$ 40

23



Creating and Using a User-Defined Function

p308d01

This demonstration illustrates how to create and use a user-defined function that reverses the last name and first name in a character string.

Creating a User-Defined Function

1. Open the program file **p308d01**.
2. Highlight and submit the PROC FCMP step.

Partial **p308d01**

```
/* Create the ReverseName function */
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
    return(catx(' ',scan(name,2,','),scan(name,1,',')));
  endsub;
quit;
```

3. Examine the SAS log.

Partial SAS Log

```
1 /* Create the ReverseName function */
2 proc fcmp outlib=orion.functions.dev;
3   function ReverseName(name $) $ 40;
4     return(catx(' ',scan(name,2,','),scan(name,1,',')));
5   endsub;
6 quit;

NOTE: Function reversename saved to orion.functions.dev.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.44 seconds
      cpu time           0.03 seconds
      cpu time           0.17 seconds
```

Accessing a User-Defined Function

1. Highlight and submit the OPTIONS statement, the DATA step, and the PROC PRINT step.

Partial Listing of **p308d01**

```
/* Use the ReverseName function */
options cmplib=orion.functions;

data work.emplist;
  set orion.employeeaddresses;
  FMLName=reversename(EmployeeName);
  keep EmployeeID EmployeeName FMLName;
run;

title 'Reverse Names Function in a Data Step';
proc print data=work.emplist;
run;
```

2. Examine the SAS log.

Partial SAS Log

```

8 /* Use the ReverseName function */
9 options cmplib=orion.functions;
10
11 data work.emplist;
12   set orion.employeeaddresses;
13   FMLName=reversename(EmployeeName);
14   keep EmployeeID EmployeeName FMLName;
15 run;

NOTE: There were 424 observations read from the data set ORION.EMPLOYEEADDRESSES.
NOTE: The data set WORK.EMPLIST has 424 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time          0.36 seconds
      cpu time           0.03 seconds

16 title 'Reverse Names Function in a Data Step';
17 proc print data=work.emplist;
18 run;

NOTE: There were 424 observations read from the data set WORK.EMPLIST.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.01 seconds

```

3. Examine the PRINT procedure output.

Partial PROC PRINT Output

Reverse Names Function in a Data Step				
	Obs	ID	EmployeeName	FMLName
	1	121044	Abbott, Ray	Ray Abbott
	2	120145	Aisbitt, Sandy	Sandy Aisbitt
	3	120761	Akinfolarin, Tameaka	Tameaka Akinfolarin
	4	120656	Amos, Salley	Salley Amos
	5	121107	Anger, Rose	Rose Anger

Accessing a User-Defined Function

After the function is defined, it can be used as needed.

```
proc sql;
  create table work.empinfo as
    select s.employeeid,
           ReverseName(s.EmployeeName) as FMLName,
           s.jobtitle,
           s.gender,
           e.birthdate,
           e.employeeshiredate,
      from orion.salesstaff as s,
           orion.employeepayroll as e
      where s.employeeid=e.employeeid;
quit;
```

26

p308d02



Using a User-Defined Function in an SQL Step

p308d02

This demonstration illustrates how to use a user-defined function in an SQL step.

Using a User-Defined Function in PROC SQL

1. Open the program file **p308d02**.
2. Highlight and submit the first PROC SQL step that joins **orion.salestaff** and **orion.employeepayroll** to create **work.empinfo**. The **ReverseName** function is used in the SELECT statement to calculate a new column named **FMLName**.

Partial **p308d02**

```
proc sql;
  create table work.empinfo as
    select s.employeeid,
           ReverseName(s.EmployeeName) as FMLName,
           s.jobtitle,
           s.gender,
           e.birthdate,
           e.employeeshiredate
      from orion.salesstaff as s,
           orion.employeepayroll as e
      where s.employeeid=e.employeeid;
quit;
```

3. Examine the SAS log.

Partial SAS Log

```

1   proc sql;
2   create table work.empinfo as
3   select s.employeeid,
4         ReverseName(s.EmployeeName) as FMLName,
5         s.jobtitle,
6         s.gender,
7         e.birthdate,
8         e.employeeshiredate
9   from orion.salesstaff as s,
10      orion.employeepayroll as e
11 where s.employeeid=e.employeeid;
NOTE: Table WORK.EMPINFO created, with 163 rows and 6 columns.
12 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.31 seconds
      cpu time           0.06 seconds

```

4. Highlight and submit the second PROC SQL step.

Partial Listing of p308d02

```

title 'Reverse Names Function in an SQL Step';
proc sql;
  select * from work.empinfo;
quit;

```

5. Examine the SAS log.

Partial SAS Log

```

15 title 'Reverse Names Function in an SQL Step';
16 proc sql;
17   select * from work.empinfo;
18 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

```

6. Examine the SQL procedure output.

Partial PROC SQL Output

Reverse Names Function in an SQL Step						
Employee ID	FMLName	Employee Job Title	Employee Gender	Employee		
				Birth Date	Hire Date	
120121	Irenie Elvish	Sales Rep. II	F	02AUG1948	01JAN1978	
120134	Sian Shannan	Sales Rep. II	M	06JUN1953	01JAN1978	
120151	Julianne Phaiyakounh	Sales Rep. II	F	21NOV1948	01JAN1978	
120154	Caterina Hayawardhana	Sales Rep. III	F	20JUL1948	01JAN1978	
120166	Fadi Nowd	Sales Rep. IV	M	14JUN1948	01JAN1978	
120172	Edwin Comber	Sales Rep. III	M	01APR1948	01JAN1978	
120174	Doungkamol Simms	Sales Rep. I	F	10JAN1948	01JAN1978	
121018	Julienne Magolan	Sales Rep. II	F	03JAN1948	01JAN1978	



Exercises

Level 1

1. Using the FCMP Procedure to Store a Formula in a Function

- Open the program file **p308e01** and submit it.

```
data refunds;
  set orion.orderfact(keep=EmployeeID Quantity
                      TotalRetailPrice);
  if Quantity > 2 then RefundAmt=Quantity*TotalRetailPrice*0.10;
  else RefundAmt=Quantity * TotalRetailPrice * 0.05;
run;
proc print data=refunds (obs=5) noobs;
  title 'Partial refunds Data Set';
run;
```

- Add a PROC FCMP step before the DATA step to encapsulate the IF/THEN-ELSE logic into a function named **REFUND**. Store the function in **work.functions.Marketin**g.
- Modify the DATA step to use the REFUND function to create a variable named **RefundAmt**. Set the appropriate system option so that the new function can be used. The DATA step should *not* contain any IF/THEN-ELSE logic.
- Print the first five observations of the SAS data set **refunds**.

PROC PRINT Output

Partial refunds Data Set

EmployeeID	Quantity	TotalRetail Price	Refund Amt
121039	1	\$16.50	0.825
99999999	1	\$247.50	12.375
99999999	1	\$28.30	1.415
120174	2	\$32.00	3.200
120134	3	\$63.60	19.080

Level 2

2. Using the FCMP Procedure to Create a User-Defined Function

The Orion Star Marketing Department needs a report that concatenates **CustomerID** with a comment. The comment text should be based on the last order that the customer placed.

- Use PROC FCMP to encapsulate the conditional logic below into a function named **MKT**. Store the function in **work.functions.Marketin**g.

Concatenate the customer ID along with comment text as defined below, separated with a hyphen.

If **OrderType**=1, then the comment text is *Mail In-Store Coupon*.

If **OrderType**=2, then the comment text is *Send New Catalog*.

If **OrderType**=3, then the comment text is *Send Email*.

For example, if the last order placed by customer 123456 was an Internet order (**OrderType**=3), the function should return ‘123456 – Send Email’.

- The function should accept two arguments: ID and Type.
 - The returned string should be 40 characters long.
- b.** Use PROC SORT and the DATA step to process **orion.orderfact** and create a temporary data set named **work.LastOrder**. It should be based on the date of the last order placed by each customer (**OrderDate**).

- c.** In the same DATA step, use the MKT function to create a new variable named **MarketingComment**. Set the appropriate system option so that the new function can be used. The DATA step should *not* contain any IF/THEN-ELSE logic.
- d.** Write a PROC PRINT step to generate the report shown below.

Partial PROC PRINT Output

Obs	CustomerID	OrderDate	Marketing Comment	
			Order	Type
1	4	18DEC2008	1	4 - Mail In-Store Coupon
2	5	26DEC2011	1	5 - Mail In-Store Coupon
3	9	25DEC2009	3	9 - Send Email
4	10	13DEC2011	1	10 - Mail In-Store Coupon
5	11	28SEP2011	3	11 - Send Email

8.2 User-Defined Formats

Objectives

- Create a format that uses a user-defined function.
- Create a function that uses a user-defined format.
- Use formats to group data.
- Define the INVALUE statement.

31

User-Defined Formats: Review

Use PROC FORMAT to create user-defined formats and informats. User-defined formats have the following characteristics:

- ✓ are stored as catalog entries, documented with the FMTLIB option
- ✓ can be used to translate and group data for analysis
- ✓ can be used to perform table lookups
- ✓ can be created and managed from the contents of a data set using the CNTLIN= and CNTLOUT= options
- ✓ can be permanently stored and shared using the LIBRARY= option and the FMTSEARCH system option

32

8.05 Quiz

Write an OPTIONS statement that adds the **orion.formats** catalog to the path that SAS searches when it looks for format definitions.

33

Business Scenario

Generate reports with rearranged employee names.



Reverse Name Function Applied using Format Statement						
Obs	EmployeeID	EmployeeName	City	State	PostalCode	Country
1	121044	Ray Abbott	Miami-Dade	FL	33135	US
2	120145	Sandy Aisbitt	Melbourne		2001	AU
3	120761	Tameaka Akinfolarin	Philadelphia	PA	19145	US
4	120656	Salley Amos	San Diego	CA	92116	US
5	121107	Rose Anger	Philadelphia	PA	19142	US

35

Business Scenario

Use PROC FCMP to create the ReverseName function and store it in a shared library. Then create a format using the function.



36

Creating Formats from Functions

Step 1: Define the ReverseName function. Store it in a permanent library.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $ 40;
    return(catx('', scan(name, 2, ','), scan(name, 1, ',')));
  endsub;
quit;
```

37

p308d04

Creating Formats from Functions

Step 2: Define the new format, and reference the function.

```
proc format library=orion fmtlib;
  value $FmtRevName (default=40)
    ' ' = 'Missing Name'
    other= [ReverseName()];
run;
```

VALUE <\$>format name other=[function name()];

- To use an existing function, enclose the function name in square brackets.

38

p308d04

Creating Formats from Functions

Step 3: Use the new format.

```
options fmtsearch=(orion);
title1 'Reverse Name Function';
title2 'Applied using Format Statement';

proc print data=orion.employeeaddresses(obs=5) noobs;
  var EmployeeID EmployeeName City State
      PostalCode Country;
  format employeeName $FmtRevName.;
run;
```

- Remember to specify the FMTSEARCH= path for user-defined formats.

39

p308d04

Creating Formats from Functions

PROC PRINT Output

Reverse Name Format Applied using Format Statement					
Employee ID	EmployeeName	City	State	Postal Code	Country
121044	Ray Abbott	Miami-Dade	FL	33135	US
120145	Sandy Aisbitt	Melbourne		2001	AU
120761	Tameaka Akinfolarin	Philadelphia	PA	19145	US
120656	Salley Amos	San Diego	CA	92116	US
121107	Rose Anger	Philadelphia	PA	19142	US

40

p308d04



Creating Formats from Functions

p308d04

This demonstration illustrates how to use user-defined functions and formats to generate custom results.

Using a Function to Generate a Format

1. Open the program file **p308d04**.
2. If the ReverseName function is not yet defined, uncomment and execute the PROC FCMP step.

Partial p308d04

```
/* proc fcmp outlib=orion.functions.dev;
   function ReverseName(name $) $ 40;
   return(catx(' ',scan(name,2,','),scan(name,1,',')));
   endsub;
quit; */
```

3. Examine the SAS log to verify that the function was successfully created.

SAS Log

```
78  proc fcmp outlib=orion.functions.dev;
79    function ReverseName(name $) $ 40;
80      return(catx(' ',scan(name,2,','),scan(name,1,',')));
81    endsub;
82  quit;

NOTE: Function ReverseName saved to orion.functions.dev.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.72 seconds
      cpu time           0.06 seconds
```

4. Submit the OPTIONS statement and PROC FORMAT step to create the \$FmtRevName format from the ReverseName user-defined function.

Partial p308d04

```
options cmplib=orion.functions;
proc format library=orion fmtlib;
  value $FmtRevName (default=40)
    '='='Missing Name'
    other=[ReverseName()];
run;
```



The FMTLIB option prints information about informats or formats in the catalog that is specified in the LIBRARY= option or the default catalog **work.formats**, if no catalog is specified.

5. Examine the SAS log to verify that the formats were created.

SAS Log

```
82  options cmplib=orion.functions;
83  proc format library=orion fmtlib;
84    value $FmtRevName (default=40)
85      '='='Missing Name'
86      other=[ReverseName()];
NOTE: Format $FMTREVNAME has been written to ORION.FORMATS.
87  run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time          0.05 seconds
      cpu time           0.01 seconds
```

6. Verify that your output matches the expected output.

PROC FORMAT Output

FORMAT NAME: \$FMTREVNAME LENGTH: 13 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 40 FUZZ: 0			
START	END	LABEL (VER. 9.4)	28APR2014:12:02:01
OTHER	**OTHER**	Missing Name [REVERSENAME()]	

7. Submit the TITLE statements and the PROC PRINT step, which uses the \$FmtRevName format.

Partial p308d04

```
options fmtsearch=(orion);

title1 'Reverse Name Format';
title2 'Applied using Format Statement';

proc print data=orion.employeeaddresses(obs=5) noobs;
  var employeeid employeename city state postalcode country;
  format employeename $FmtRevName.;
```

```
run;
```

8. Examine the SAS log and verify that the steps ran without errors.

SAS Log

```
84  proc print data=orion.employeeaddresses(obs=5) noobs;
85  var employeeid employeeName city state postalcode country;
86  format employeeName $FmtRevName.;
87  run;
```

NOTE: There were 5 observations read from the data set ORION.EMPLOYEEADDRESSES.

NOTE: PROCEDURE PRINT used (Total process time):

real time	0.28 seconds
cpu time	0.23 seconds

9. Verify that your output matches the expected output.

Output

Reverse Name Format
Applied using Format Statement

Obs	Employee ID	EmployeeName	City	State	Postal Code	Country
1	121044	Ray Abbott	Miami-Dade	FL	33135	US
2	120145	Sandy Aisbitt	Melbourne		2001	AU
3	120761	Tameaka Akinfolarin	Philadelphia	PA	19145	US
4	120656	Salley Amos	San Diego	CA	92116	US
5	121107	Rose Anger	Philadelphia	PA	19142	US

Business Scenario

Create a user-defined function that returns a United States state code or Australian province code. The data set **orion.employeeaddresses** contains the postal codes for employees in the US and Australia.



Employees with State/Province Codes				
Country	PostalCode	State/Province	EmployeeID	EmployeeName
AU	2001	NSW	120145	Aisbitt, Sandy
AU	2165	NSW	120185	Bahlman, Sharon
AU	2119	NSW	120109	Baker, Gabriele
AU	1225	NSW	120188	Baran, Shanmuganathan
AU	2114	NSW	120144	Barbis, Viney
AU	8003	VIC	120168	Barcoe, Selina

Business Scenario

Use PROC FORMAT to design a format for Australian postal codes. Then use the format in a PROC FCMP function to display US and Australian states and provinces.

PROC FORMAT



PROC FCMP



43

Creating Functions from Formats

Step 1: Define the necessary format.

```
proc format;
  value $postabb
    '1000'-'1999', '2000'-'2599',
    '2619'-'2898', '2921'-'2999'='NSW'
    '0200'-'0299', '2600'-'2618',
    '2900'-'2920'          ='ACT'
    '3000'-'3999', '8000'-'8999'='VIC'
    '4000'-'4999', '9000'-'9999'='QLD'
    '5000'-'5799', '5800'-'5999'='SA'
    '6000'-'6797', '6800'-'6999'='WA'
    '7000'-'7799', '7800'-'7999'='TAS'
    '0800'-'0899', '0900'-'0999'='NT';
run;
```

44

p308d05

Creating Functions from Formats

Step 2: Define the necessary function.

```
proc fcmp outlib=orion.functions.char;
  function StateProv(Country $,Code $) $ 4;
    if upcase(country) = 'AU'
      then strp=put(code,$postabb.);
    else if upcase(country)='US'
      then strp=zipstate(code);
    else strp=' ';
    return(strp);
  endsub;
quit;
```

45

p308d05

Creating Functions from Formats

Step 3: Use the function.

```
options cmplib=orion.functions;

title 'Employees with State/Province Codes';

proc sql;
  select Country,
    PostalCode,
    StateProv(Country,PostalCode) 'State/Province',
    EmployeeID,
    EmployeeName
  from orion.employeeaddresses
  order by Country, EmployeeName;
quit;
```

46

p308d05

Creating Functions from Formats

Partial PROC SQL Output

Employees with State/Province Codes			Employee	
Country	PostalCode	State/Province	ID	EmployeeName
AU	2001	NSW	120145	Aisbitt, Sandy
AU	2165	NSW	120185	Bahlman, Sharon
AU	2119	NSW	120109	Baker, Gabriele
AU	1225	NSW	120188	Baran, Shanmuganathan
AU	2114	NSW	120144	Barbis, Viney
AU	8003	VIC	120168	Barcoe, Selina
AU	1115	NSW	120182	Barreto, Geok-Seng
AU	1670	NSW	120104	Billington, Karen
AU	3150	VIC	120183	Blanton, Brig

47

p308d05



Creating Functions from Formats

Using a Format to Generate a Function

1. Open the program file **p308d05**.
2. Submit the PROC FORMAT code that creates the \$POSTABB user-defined format.

Partial **p308d05**

```
proc format;
  value $postabb
    '1000'-'1999', '2000'-'2599',
    '2619'-'2898', '2921'-'2999'='NSW'
    '0200'-'0299', '2600'-'2618',
    '2900'-'2920'          ='ACT'
    '3000'-'3999', '8000'-'8999'='VIC'
    '4000'-'4999', '9000'-'9999'='QLD'
    '5000'-'5799', '5800'-'5999'='SA'
    '6000'-'6797', '6800'-'6999'='WA'
    '7000'-'7799', '7800'-'7999'='TAS'
    '0800'-'0899', '0900'-'0999'='NT';
run;
```

3. Examine the SAS log to verify that the format was created.

SAS Log

```

96  proc format;
97    value $postabb
98      '1000'-'1999', '2000'-'2599',
99      '2619'-'2898', '2921'-'2999'='NSW'
100     '0200'-'0299', '2600'-'2618',
101     '2900'-'2920'           ='ACT'
102     '3000'-'3999', '8000'-'8999'='VIC'
103     '4000'-'4999', '9000'-'9999'='QLD'
104     '5000'-'5799', '5800'-'5999'='SA'
105     '6000'-'6797', '6800'-'6999'='WA'
106     '7000'-'7799', '7800'-'7999'='TAS'
107     '0800'-'0899', '0900'-'0999'='NT';
NOTE: Format $POSTABB has been output.
110 run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time          0.05 seconds
      cpu time           0.00 seconds

```

4. Submit the PROC FCMP step that creates the StateProv function.

Partial p308d05

```

proc fcmp outlib=orion.functions.char;
  function StateProv(Country $,Code $) $ 4;
    if upcase(country) ='AU' then stpr=put(code,$postabb.);
    else if upcase(country)='US' then stpr=zipstate(code);
    else stpr=' ';
    return(stpr);
  endsub;
quit;

```

5. Examine the SAS log to verify that the function was created.

SAS Log

```

111 proc fcmp outlib=orion.functions.char;
112   function StateProv(Country $,Code $) $ 4;
113     if upcase(country) ='AU' then stpr=put(code,$postabb.);
114     else if upcase(country)='US' then stpr=zipstate(code);
115     else stpr=' ';
116     return(stpr);
117   endsub;
118 quit;

NOTE: Function StateProv saved to orion.functions.char.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.39 seconds
      cpu time           0.04 seconds

```

6. Submit the OPTIONS statement and the PROC SQL step that references the StateProv function.

Partial p308d05

```
options cmplib=orion.functions;

title 'Employees with State/Province Codes';

proc sql;
  select Country,
         PostalCode,
         StateProv(Country,PostalCode) 'State/Province',
         EmployeeID,
         EmployeeName
    from orion.employeeaddresses
   order by Country, EmployeeName;
quit;
```

7. Examine the SAS log and verify that the steps ran without errors.

SAS Log

```
119 options cmplib=orion.functions;
120
121 title 'Employees with State/Province Codes';
122
123 proc sql;
124   select Country,
125         PostalCode,
126         StateProv(Country,PostalCode) 'State/Province',
127         EmployeeID,
128         EmployeeName
129   from orion.employeeaddresses
130   order by Country, EmployeeName;
131 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.51 seconds
      cpu time           0.20 seconds
```

8. Verify that your output matches the expected output:

Partial Output

Employees with State/Province Codes				
Country	PostalCode	State/Province	Employee	
			ID	EmployeeName
AU	2001	NSW	120145	Aisbitt, Sandy
AU	2165	NSW	120185	Bahlman, Sharon
AU	2119	NSW	120109	Baker, Gabriele
AU	1225	NSW	120188	Baran, Shanmuganathan
AU	2114	NSW	120144	Barbis, Viney
AU	8003	VIC	120168	Barcoe, Selina
AU	1115	NSW	120182	Barreto, Geok-Seng
AU	1670	NSW	120104	Billington, Karen
AU	3150	VIC	120183	Blanton, Brig

Business Scenario

Only employees hired before 1994 are eligible for certain benefits. Create a report that displays the hire date for those who are eligible or a comment if the employee is not eligible. The data set **orion.employeepayroll** contains the hire dates for all employees.

EmployeeID	EmployeeHireDate
120101	** Not Eligible **
120102	June 1, 1993
120103	January 1, 1978
120104	January 1, 1985
120105	** Not Eligible **



50

Alternative Date Formats

The SAS format WORDDATE20. can be used to display the dates for those who are eligible. Combine this with a user-defined format to display a comment for those who are not eligible.

```
proc format;
  value benefit
    low-'31dec1993'd=[worddate20.]
    '01jan1994'd-high='** Not Eligible **';
run;
```

- ✍ To use an existing format, enclose the format name in square brackets.

51

p308d06

Alternative Date Formats

```
proc print data=orion.employeepayroll(obs=5) noobs;
  var employeeid employeeshiredate;
  format employeeshiredate benefit.;
  title 'Nested formats';
run;
```

Partial PROC PRINT Output

Nested formats	
EmployeeID	EmployeeHireDate
120101	** Not Eligible **
120102	June 1, 1993
120103	January 1, 1978
120104	January 1, 1985
120105	** Not Eligible **

52

p308d06

8.06 Poll

You can nest only formats supplied by SAS.

- True
- False

53

Business Scenario

Create a report that displays the highest-paid and lowest-paid employee in each salary group.



Highest and Lowest Salary by Group

Employee ID	Salary	Salary Group	Employee Hire Date
121084	\$22,710	Under \$30,000	01JAN1995
120155	\$29,990	Under \$30,000	01APR2010
120738	\$30,025	\$30,000 to \$35,000	01JAN1978
120727	\$34,925	\$30,000 to \$35,000	01JUN1990
120728	\$35,070	\$35,000 to \$50,000	01JAN1986
120691	\$49,240	\$35,000 to \$50,000	01JAN1978
120813	\$50,865	Over \$50,000	01JAN1997
120259	\$433,800	Over \$50,000	01SEP1993

55

Business Scenario

The data set **orion.employeepayroll** has the salaries for all employees.

Partial **orion.employeepayroll**

EmployeeID	Salary
120101	\$28,090
120102	\$207,885
120103	\$85,495
120104	\$43,650



56

Grouping Data by Formatted Value

Step 1: Create the format for the groups.

```
proc format;
  value salgrp
    low-<30000 = 'Under $30,000'
    30000-<35000 = '$30,000 to $35,000'
    35000-<50000 = '$35,000 to $50,000'
    50000-high = 'Over $50,000';
run;
```

57

p308d07

Grouping Data by Formatted Value

Step 2: Sort the data and use the GROUPFORMAT option in the DATA step BY statement to select the subset based on the formatted values.

```
proc sort data=orion.employeepayroll
          out=sorted;
  by salary;
run;
data highlowsal;
  set sorted;
  by groupformat salary;
  format salary salgrp.;
  Salgroup=put(salary,salgrp.);
  if first.salary or last.salary;
run;
```

BY GROUPFORMAT variable-name <NOTSORTED>;

58

p308d07



The NOTSORTED option specifies that observations with the same BY value are grouped together but are not necessarily sorted in alphabetical or numeric order.

Grouping Data by Formatted Value

Step 3: Use the subset to create the report.

```
title 'Lowest and Highest Salary';
title2 'by Salary Group';
proc print data=highlowsal label;
  id SalGroup;
  by SalGroup notsorted;
  var Salary EmployeeID;
  format salary dollar9.;
  label salgroup='Salary Group';
run;
```

59

p308d07

Grouping Data by Formatted Value

PROC PRINT Output

Lowest and Highest Salary by Salary Group		
Salary Group	Salary	Employee ID
Under \$30,000	\$22,710	121084
	\$29,990	120155
\$30,000 to \$35,000	\$30,025	120738
	\$34,925	120727
\$35,000 to \$50,000	\$35,070	120728
	\$49,240	120691
Over \$50,000	\$50,865	120813
	\$433,800	120259

60

p308d07

8.07 Multiple Choice Poll

Which of the following is true of the GROUPFORMAT option?

- a. available in both the DATA step and PROC steps
- b. must be used with the NOTSORTED option
- c. enables the DATA step to process data in formatted groups
- d. can be used in a WHERE or BY statement

61

Business Scenario

The Marketing Department needs to create a SAS data set from the **newcustomers.dat** file. The file contains demographic information for new customers. Some of the data was incorrectly entered and should be cleansed as the data is read.

newcustomers.dat

	1	1	2	2	3	3	4	4	5	5	6
1	--5	0	--5	0	--5	0	--5	0	--5	0	--5
4	m	James	Kvarniq		1		47.7		Good		
5	F	Sandrina	Stephano		-2		27.75		Excellent		
9	F	Cornelia	Krah1		1		29.4		Fair		
10	f	Karen	Ballinger		2		27.75		Excellent		
11	F	Elke	Wallstab		1		72.7		Good		

63

Business Scenario

Create informats to ensure the following:

- Values for **Gender** are coded either as *Male* or *Female*.
- Negative quantities are converted to missing values.
- Customer ratings are converted to numbers for calculating rating statistics (*Excellent*=4, and so on).

newcustomers.dat

	1	1	2	2	3	3	4	4	5	5	6
1	---	5	---	0	---	5	---	0	---	5	---
4	m	James	Kvarnig		1	47.7	Good				
5	F	Sandrina	Stephano		-2	27.75	Excellent				
9	F	Cornelia	Krahil		1	29.4	Fair				
10	f	Karen	Ballinger		2	27.75	Excellent				
11	F	Elke	Wallstab		1	72.7	Good				

64

User-Defined Informats

Step 1: Create the informats.

```
proc format library=orion;
  invaluel $eval
    'Excellent'=4
    'Good'=3
    'Fair'=2
    'Poor'=1;
  invaluel quant
    1-high=_same_
    other=_error_;
  invaluel $Gender (upcase)
    'M'='Male'
    'F'='Female';
run;

PROC FORMAT LIBRARY=libref.CATALOG;
  INVALUE numinfmt 'value1' = informat-value-1
    'value2' = informat-value-2
    'valuen' = informat-value-n;
RUN;
```

65

p308d08

Special Keywords

SAME indicates that a value in the domain is to be mapped into the same value in the range.

ERROR indicates that a value or set of values should be excluded from the domain.

```
proc format library=orion;
  invalue $eval
    'Excellent'=4
    'Good'=3
    'Fair'=2
    'Poor'=1;
  invalue quant
    1-high=_same_
    other=_error_;
  invalue $Gender (upcase)
    'M'='Male'
    'F'='Female';
run;
```

66

p308d08

UPCASE Option

The **UPCASE** option in the INVALUE statement automatically uppercases all input values before they are compared to the informat domain.

```
proc format library=orion;
  invalue $eval
    'Excellent'=4
    'Good'=3
    'Fair'=2
    'Poor'=1;
  invalue quant
    1-high=_same_
    other=_error_;
  invalue $Gender (upcase)
    'M'='Male'
    'F'='Female';
run;
```

67

p308d08

User-Defined Informats

Step 2: Apply the informats.

```
options fmtsearch=(orion);

data newcustomers;
  infile "&path\newcustomers.dat" firstobs=2;
  input @1 CustomerID
        @11 CustomerGender $Gender.
        @17 CustomerName $20.
        @37 Quantity quant2.
        @44 TotalRetailPrice
        @53 Rating $eval.;
run;
```

68

p308d08

User-Defined Informats

```
proc print data=newcustomers(obs=5) noobs;
  title 'User Defined Informats';
run;
```

Partial PROC PRINT Output

User Defined Informats					
Customer ID	Customer Gender	CustomerName	Quantity	Total Retail Price	Rating
4	Male	James Kvarniq	1	47.70	3
5	Female	Sandrina Stephano	.	27.75	4
9	Female	Cornelia Krahl	1	29.40	2
10	Female	Karen Ballinger	2	27.75	4
11	Female	Elke Wallstab	1	72.70	3

69

p308d08



User-Defined Informats

p308d08

Creating User-Defined Informats

1. Open the program file **p308d08**.

2. Submit the PROC FORMAT step, which creates three user-defined informats, and the DATA step, which reads the **newcustomers.dat** file.

Partial p308d08

```

proc format library=orion;
  invalide $eval
    'Excellent'=4
    'Good'=3
    'Fair'=2
    'Poor'=1;
  invalide quant
    1-high=_same_
    other=_error_;
  invalide $Gender (upcase)
    'M'='Male'
    'F'='Female';
run;

options fmtsearch=(orion);

data newcustomers;
  infile "&path\newcustomers.dat" firstobs=2;
  input @1 CustomerID
    @11 CustomerGender $Gender.
    @17 CustomerName $20.
    @37 Quantity quant2.
    @44 TotalRetailPrice
    @53 Rating $eval.;
run;

proc print data=newcustomers(obs=5) noobs;
  title 'User Defined Informats';
run;

```

3. Examine the SAS log to verify that the PROC FORMAT step ran without errors and that the DATA step ran with four reported, detailed data errors.

Partial SAS Log

```

96  proc format library=orion;
97    invalide $eval
98      'Excellent'=4
99      'Good'=3
100     'Fair'=2
101     'Poor'=1;
NOTE: Informat $EVAL has been output.
102    invalide quant
103      1-high=_same_
104      other=_error_;
NOTE: Informat QUANT has been output.
105    invalide $Gender (upcase)
106      'M'='Male'

```

```

107      'F'='Female';
NOTE: Informat $GENDER has been output.
108  run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

109
110 options fmtsearch=(orion);
111
112 data newcustomers;
113   infile "&path\newcustomers.dat" firsttobs=2;
114   input @1 CustomerID
115     @11 CustomerGender $Gender.
116     @17 CustomerName $20.
117     @37 Quantity quant2.
118     @44 TotalRetailPrice
119     @53 rating $eval.;
120 run;

NOTE: The infile "S:\workshop\newcustomers.dat" is:
      Filename=S:\workshop\newcustomers.dat,
      RECFM=V,LRECL=32767,File Size (bytes)=5252,
      Last Modified=28Jul2014:16:26:38,
      Create Time=28Jul2014:15:44:47

NOTE: Invalid data for Quantity in line 3 37-38.
RULE:    -----1-----2-----3-----4-----5-----6-----7-----8---
3       5      F      Sandrina Stephano   -2      27.75   Excellent    65
CustomerID=5 CustomerGender=Female CustomerName=Sandrina Stephano Quantity=.
TotalRetailPrice=27.75 Rating=4 _ERROR_=1 _N_=2
NOTE: Invalid data for Quantity in line 13 37-38.
13      20     m      Michael Dineley   -2      27.75   Poor        65
CustomerID=20 CustomerGender=Male CustomerName=Michael Dineley Quantity=.
TotalRetailPrice=27.75
Rating=1 _ERROR_=1 _N_=12
NOTE: Invalid data for Quantity in line 72 37-38.
72      70100   f      Wilma Yeargan   -2      27.75   Excellent    65
CustomerID=70100 CustomerGender=Female CustomerName=Wilma Yeargan Quantity=.
TotalRetailPrice=27.75 Rating=4 _ERROR_=1 _N_=71
NOTE: Invalid data for Quantity in line 77 37-38.
77      70210   m      Alex Santinello -2      27.75   Fair         65
CustomerID=70210 CustomerGender=Male CustomerName=Alex Santinello Quantity=.
TotalRetailPrice=27.75 Rating=2 _ERROR_=1 _N_=76

NOTE: 77 records were read from the infile
      "S:\workshop\newcustomers.dat".
      The minimum record length was 65.
      The maximum record length was 65.
NOTE: The data set WORK.NEWCUSTOMERS has 77 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds

```

4. Submit the PROC PRINT step, which prints the **work.newcustomers** data set.

Partial p308d08

```
proc print data=newcustomers(obs=5) noobs;
   title 'Using the INVALUE Statement';
run;
```

5. Examine the SAS log to verify that the step ran without errors.

SAS Log

```
121 proc print data=newcustomers(obs=5) noobs;
122   title 'Using the INVALUE Statement';
123 run;
NOTE: There were 5 observations read from the data set WORK.NEWCUSTOMERS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
```

6. Verify that your output matches the expected output.

PROC PRINT Output

User Defined Informats					
Customer ID	Customer Gender	CustomerName	Quantity	Total Price	Retail Rating
4	Male	James Kvarniq	1	47.70	3
5	Female	Sandrina Stephano	.	27.75	4
9	Female	Cornelia Krahl	1	29.40	2
10	Female	Karen Ballinger	2	27.75	4
11	Female	Elke Wallstab	1	72.70	3

**Exercises****Level 1****3. Creating a Format Based on a Function**

- a. Currently, **CustomerAgeGroup** is hardcoded in the **orion.customerdim** data set. Create a function named **CustAge** that automatically groups the customers based on their age. Store the function in **work.functions.DateType**.

- 1) Calculate customer ages in years.

Hint: Use the following formula to calculate age:

```
int((today() - date)/365.25)
```

- 2) Create the following four groups based on age:

- Under 30
- 30 to 44 years

- 45 to 60 years
 - More than 60
- b. Create a format named CUSTGROUP based on the function. Set the appropriate system option so that the new function can be used.
- c. Open p308e03 and substitute the format for XXXXX in the PROC PRINT step.

```
proc print data = orion.customerdim(obs=5) noobs label;
  var CustomerID CustomerName CustomerBirthDate;
  title 'Customer Age Group Information';
  format CustomerBirthDate XXXXX;
  label CustomerID='Customer ID' CustomerName='Customer Name'
        CustomerBirthDate='Age Group';
run;
```

- d. Verify that your output matches the expected output.

PROC PRINT Output

Customer Age Group Information		
Customer ID	Customer Name	Age Group
4	James Kvarniq	30 to 44
5	Sandrina Stephano	30 to 44
9	Cornelia Krahl	30 to 44
10	Karen Ballinger	Under 30
11	Elke Wallstab	30 to 44



Results might differ depending on when the program is executed.

Level 2

4. Reading Raw Data with User-Defined Informats

The raw data file **sales1.dat** has employee information for the Australian and United States sales staff.

Layout for **sales1.dat**

Field Description	Starting Column	Length of Field	Data Type
Employee ID	1	6	Numeric
First Name	8	12	Character
Last Name	21	18	Character
Gender	40	1	Character
Job Title	43	20	Character
Salary	64	8	Numeric \$100,000

Field Description	Starting Column	Length of Field	Data Type
→ Country	73	2	Character 'AU' or 'US'
Birth Date	76	10	Numeric mm/dd/yyyy
→ Hire Date	87	10	Numeric mm/dd/yyyy

- a. Modify **p308e04.sas** to create and use the following informats to aid in reading the **sales1.dat** file:

- 1) a character informat named \$Gender to read the gender codes as follows:

Raw Value	Variable Value
f or F	Female
m or M	Male

- 2) a character informat named \$Country to read the country codes as follows:

Raw Value	Variable Value
au, Au, aU, AU	Australia
us, Us, uS, US	United States

- b. Make sure that the two user-defined informats are permanently stored in the **Orion** library.
- c. Modify the DATA step that creates **sales_staff** to read the fields indicated by arrows in the layout table. Use the new user-defined informats where applicable. Make sure that the appropriate option is set to tell SAS to search the **Orion** library for user-defined informats.
- d. Print the first 5 observations of **sales_staff** and add an appropriate title.

Partial PROC PRINT Output

Australian and US Sales Staff					
Employee_ID	Gender	Job_Title	Salary	Country	Hire_Date
120102	Male	Sales Manager	108255	Australia	12205
120103	Male	Sales Manager	87975	Australia	6575
120121	Female	Sales Rep. II	26600	Australia	6575
120122	Female	Sales Rep. II	27475	Australia	8217
120123	Female	Sales Rep. I	26190	Australia	18901

8.3 Solutions

Solutions to Exercises

1. Using the FCMP Procedure to Store a Formula in a Function

- a. Open the program file **p308e01** and submit it.

- 1) Open the program file **p308e01**.

- 2) Submit the DATA and PROC PRINT steps.
- 3) Examine the SAS log to verify that the steps ran without errors.

Partial SAS Log

```

307  data refunds;
308    set orion.orderfact(keep=EmployeeID Quantity
309                      TotalRetailPrice);
310    if Quantity > 2 then RefundAmt=Quantity * TotalRetailPrice * 0.10;
311    else RefundAmt=Quantity * TotalRetailPrice * 0.05;
312  run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.TEST has 617 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time          0.00 seconds
314
315  proc print data=test(obs=5) noobs;
316    title 'Partial refunds Data Set';
317  run;
NOTE: There were 5 observations read from the data set WORK.TEST.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds

```

- 4) Verify that the output matches the expected output.
- b. Use PROC FCMP to encapsulate the IF/THEN logic into a function named REFUND. Store the function in **work.functions.Marketing**.

- 1) Enter the following PROC FCMP step in the Editor window:

Partial p308s01

```

proc fcmp outlib=work.functions.Marketing;
  function REFUND(Quantity, Price);
    if Quantity > 2 then return(Quantity * Price * 0.10);
    else return(Quantity * Price * 0.05);
  endsub;
quit;

```

- 2) Submit the PROC FCMP step.
- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```

288  proc fcmp outlib=work.functions.Marketing;
289    function REFUND (Quantity, Price);
290      if Quantity > 2 then RefundAmt=Quantity * TotalRetailPrice * 0.10;
291      else RefundAmt=Quantity * TotalRetailPrice * 0.05;
292  run;
NOTE: Function REFUND saved to work.functions.Marketing.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.06 seconds
      cpu time          0.03 seconds

```

- c. Write a DATA step to read **orion.orderfact**. Create a data set named **refunds** that uses the REFUND function to create a variable named **RefundAmt**. Set the appropriate system option so that the new function can be used. The DATA step should *not* contain any IF/THEN-ELSE logic.

- 1) Enter the following OPTIONS statement and DATA step in the Editor window:

Partial p308s01

```
options cmplib=work.functions;

data refunds;
  set orion.orderfact(keep=EmployeeID Quantity
                      TotalRetailPrice);
  RefundAmt=REFUND(Quantity, TotalRetailPrice);
run;
```

- 2) Submit the OPTIONS statement and DATA step.
- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
337 options cmplib=work.functions;
338
339 data refunds;
340   set orion.orderfact(keep=EmployeeID Quantity
341                     TotalRetailPrice);
342   RefundAmt=REFUND(Quantity, TotalRetailPrice);
343 run;
NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.REFUNDS has 617 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 second
```

- d. Print the first five observations of the SAS data set **refunds**.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p308s01

```
proc print data=refunds (obs=5) noobs;
  title 'Partial refunds Data Set';
run;
```

- 2) Submit the PROC PRINT step.
- 3) Verify that the output matches the expected output.
- 4) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
345 proc print data=refunds (obs=5) noobs;
346   title 'Partial refunds Data Set';
347 run;
NOTE: There were 5 observations read from the data set WORK.REFUNDS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

2. Using the FCMP Procedure to Create a User-Defined Function

The Orion Star Marketing Department needs a report that concatenates **CustomerID** with a comment. The comment text should be based on the last order that the customer placed.

- a. Use PROC FCMP to encapsulate the conditional logic below into a function named MKT. Store the function in **work.functions.Marketing**.

Concatenate the customer ID along with comment text as defined below, separated with a hyphen.

If **OrderType**=1, then the comment text is *Mail In-Store Coupon*.

If **OrderType**=2, then the comment text is *Send New Catalog*.

If **OrderType**=3, then the comment text is *Send Email*.

For example, if the last order placed by customer 123456 was an Internet order (**OrderType**=3), the function should return ‘123456 – Send Email’.

- 1) Enter the following PROC FCMP step in the Editor window:

Partial p308s02

```
proc fcmp outlib=orion.functions.Marketing;
  function MKT(ID, Type) $ 40;
    if Type=1
      then return(catx(' - ',put(ID,12.),'Mail In-Store Coupon'));
    else if Type=2
      then return(catx(' - ',put(ID,12.),'Send New Catalog'));
    else if Type=3
      then return(catx(' - ',put(ID,12.),'Send Email'));
    endsub;
quit;
```

- 2) Submit the PROC FCMP step.

- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
63 proc fcmp outlib=orion.functions.Marketing;
64   function MKT(ID, Type) $ 40;
65     if Type=1 then return(catx(' - ',put(ID,12.),'Mail In-Store Coupon'));
66     else if Type=2 then return(catx(' - ',put(ID,12.),'Send New Catalog'));
67     else if Type=3 then return(catx(' - ',put(ID,12.),'Send Email'));
68   endsub;
69 quit;
NOTE: Function MKT saved to orion.functions.Marketing.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.42 seconds
      cpu time           0.04 seconds
```

- b. Use PROC SORT and the DATA step to process **orion.orderfact** and create a temporary data set named **work.LastOrder**. It should be based on the date of the last order placed by each customer (**OrderDate**).

- 1) Enter the following PROC SORT step in the Editor window:

Partial p308s02

```
proc sort data=orion.orderfact out=work.LastOrder;
  by CustomerID OrderDate;
run;
```

- 2) Submit the PROC SORT step.
- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
72 proc sort data=orion.orderfact out=work.LastOrder;
73   by CustomerID OrderDate;
74 run;

NOTE: There were 617 observations read from the data set ORION.ORDERFACT.
NOTE: The data set WORK.LASTORDER has 617 observations and 12 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds
```

- c. In the same DATA step, use the MKT function to create a new variable named **MarketingComment**. Set the appropriate system option so that the new function can be used. The DATA step should *not* contain any IF/THEN-ELSE logic.

- 1) Enter the following options statement and DATA step in the Editor window:

Partial p308s02

```
options cmplib=orion.functions;

data work.LastOrder;
  set work.LastOrder;
  by CustomerID OrderDate;
  if last.CustomerID;
  MarketingComment=MKT(CustomerID,OrderType);
run;
```

- 2) Submit the options statement and DATA step.
- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
676 options cmplib=orion.functions;
677
678 Data work.LastOrder;
679   set work.LastOrder;
680   by CustomerID OrderDate;
681   if last.CustomerID;
682   MarketingComment=MKT(CustomerID,OrderType);
683 run;

NOTE: There were 617 observations read from the data set WORK.LASTORDER.
NOTE: The data set WORK.LASTORDER has 75 observations and 13 variables.
NOTE: DATA statement used (Total process time):
      real time          0.34 seconds
      cpu time          0.04 seconds
```

- d. Write a PROC PRINT step to generate the report.

- 1) Enter the following PROC PRINT step in the Editor window:

Partial p308s02

```
title 'Marketing Comment';
proc print data=work.LastOrder(obs=5) noobs;
  var CustomerID OrderDate OrderType MarketingComment;
run;
```

- 2) Submit the PROC PRINT step.
 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
84  title 'Marketing Comment';
85  proc print data=work.LastOrder;
86  var CustomerID OrderDate OrderType MarketingComment;
87  run;

NOTE: There were 75 observations read from the data set WORK.LASTORDER.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.04 seconds
      cpu time          0.04 seconds
```

3. Creating a Format Based on a Function

- e. Currently, **CustomerAgeGroup** is hardcoded in the **orion.customerdim** data set. Create a function named **CustAge** that automatically groups the customers based on their birthdates and on the current date. Store the function in **work.functions.DateType**.

- 1) Enter the following PROC FCMP step in the Editor window:

Partial p308s03

```
proc fcmp outlib=orion.functions.DateType;
  function CustAge(Date) $ 15;
    age=int((today()-date)/365.25);
    if age < 30 then return('Under 30');
    else if age < 45 then return('30 to 44 years');
    else if age < 60 then return('45 to 60 years');
    else return('Over 60');
  endsub;
quit;
```

- 2) Submit the PROC FCMP step.
 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
299  proc fcmp outlib=orion.functions.DateType;
300    function CustAge(Date) $ 15;
301      age=int((today()-date)/365.25);
302      if age < 30 then return('Under 30');
303      else if age < 45 then return('30 to 44 years');
304      else if age < 60 then return('45 to 60 years');
305      else return('Over 60');
306    endsub;
```

```
307 quit;

NOTE: Function CustAge saved to orion.functions.DateType.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.62 seconds
      cpu time          0.06 seconds
```

- b. Create a format named CUSTGROUP based on the function. Set the appropriate system option so that the new function can be used

- 1) Enter the following OPTIONS statement and PROC FORMAT step in the Editor window:

Partial p308s03

```
options cmplib=orion.functions;

proc format;
  value custgroup other=[custage()];
run;
```

- 2) Submit the OPTIONS statement and the PROC FORMAT step.

- 3) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```
170 options cmplib=orion.functions;
171
172 proc format;
173   value custgroup other=[custage()];
NOTE: Format CUSTGROUP has been output.
174 run;
NOTE: PROCEDURE FORMAT used (Total process time):
      real time          0.00 seconds
      cpu time          0.00 seconds
```

- c. Open p308e03 and substitute the format for XXXXX in the PROC PRINT step.

- 1) Open the program file p308e03.
- 2) Replace the XXXX in the FORMAT statement with the CUSTGROUP format.

Partial p308s03

```
proc print data = orion.customerdim(obs=5) noobs label;
  var CustomerID CustomerName CustomerBirthDate;
  title 'Customer Age Group Information';
  format CustomerBirthDate custgroup.;
  label CustomerID='Customer ID' CustomerName='Customer Name'
        CustomerBirthDate='Age Group';
run;
```

- 3) Submit the PROC PRINT step.
- 4) Verify that the output matches the expected output.
- 5) Examine the SAS log to verify that the step ran without errors.

Partial SAS Log

```

175  proc print data = orion.customerdim(obs=5) noobs label;
176      var CustomerID CustomerName CustomerBirthDate;
177      title 'Customer Age Group Information';
178      format CustomerBirthDate custgroup.;
179      label CustomerID='Customer ID' CustomerName='Customer Name'
180          CustomerBirthDate='Age Group';
181      run;
NOTE: There were 5 observations read from the data set ORION.CUSTOMERDIM.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.14 seconds
      cpu time          0.07 seconds

```

4. Reading Raw Data with User-Defined Informats

The raw data file **sales1.dat** has employee information for the Australian and United States sales staff. The record layout is shown in the table in the exercise.

- a. Modify **p308e04.sas** to create and use the following informats to aid in reading the **sales1.dat** file:

- 1) a character informat named \$Gender to read the gender codes as follows:

Raw Value	Variable Value
f or F	Female
m or M	Male

- 2) a character informat named \$Country to read the country codes as follows:

Raw Value	Variable Value
au, Au, aU, AU	Australia
us, Us, uS, US	United States

- b. Make sure that the two user-defined informats are permanently stored in the **Orion** library.

```

proc format library=orion;
  invalide $gender (upcase)
    'M'='Male'
    'F'='Female';
  invalide $country (upcase)
    'AU'='Australia'
    'US'='United States';
run;

```

- c. Modify the DATA step that creates **sales_staff** to read the fields indicated by arrows in the layout table. Use the new user-defined informats where applicable. Make sure that the appropriate option is set to tell SAS to search the **Orion** library for user-defined informats.

```

Options fmtsearch=(orion);

data sales_staff;
  infile "&path\sales1.dat";
  input @1 Employee_ID 6.
        @40 Gender :$gender.
        @43 Job_Title $20.

```

```

@64 Salary Dollar8.
@73 Country :$country.
@87 Hire_Date mmddyy10.;

run;

```

- d. Print the first 5 observations from **sales_staff** and add an appropriate title.

```

title 'Australian and US Sales Staff';
proc print data=sales_staff(obs=5) noobs;
run;
title;

```

Solutions to Student Activities (Polls/Quizzes)

8.01 Quiz – Correct Answer

Fill in the blanks:

A SAS function is a routine that accepts arguments performs a data manipulation, and returns a value.

All functions have common syntax. Arguments are enclosed in parentheses and separated by a comma.

function-name(argument-1,argument-2,...,argument-n)

8.02 Multiple Answer Poll – Correct Answers

Which functions can be used to extract first name and last name values and combine them as shown below?

- a. SUBSTR
- b. LEFT
- c. CATX
- d. SCAN
- e. TRIM



Some examples include the following:

```
trim(left(scan(EmployeeName,-1,'')))||' '|left(scan(EmployeeName,1,''))  
complib(scan(EmployeeName,2,'')||scan(EmployeeName,1,''))  
catx(' ',scan(EmployeeName,2,''),scan(EmployeeName,1,''))
```

8

8.03 Poll – Correct Answer

The CMPLIB option is required to create user-defined functions in PROC FCMP.

- True
- False

The CMPLIB option is not needed to *create* user-defined functions and subroutines. It is needed to *use* them.

20

8.04 Quiz – Correct Answer

Where did the type and length attributes for the new variable come from?

PDV

EmployeeID	EmployeeName	...	Country	FMLName
N 8	\$ 40		\$ 2	\$ 40

The type and length attributes came from the function definition.

```
proc fcmp outlib=orion.functions.dev;
  function ReverseName(name $) $40;
    return(catx(' ',scan(name,2,' '),scan(name,1,' ')));
  endsub;
quit;
```

24

8.05 Quiz – Correct Answer

Write an OPTIONS statement that adds the **orion.formats** catalog to the path that SAS searches when it looks for format definitions.

```
options fmtsearch=(ORION);
```

The default search path is as follows:

1. formats supplied by SAS
2. workformats
3. libraryformats
4. <permanent library>.formats
5. <permanent library>.<other format catalog>

34

8.06 Poll – Correct Answer

You can nest only formats supplied by SAS.

- True
- False

You can nest formats that are supplied by SAS *and* user-defined formats.

54

8.07 Multiple Choice Poll – Correct Answer

Which of the following is true of the GROUPFORMAT option?

- a. available in both the DATA step and PROC steps
- b. must be used with the NOTSORTED option
- c. enables the DATA step to process data in formatted groups
- d. can be used in a WHERE or BY statement

The GROUPFORMAT option can be used only in the DATA step and only in a BY statement. It does not require the NOTSORTED option.

62

Chapter 9 Learning More

9.1 Introduction.....	9-3
-----------------------	-----

9.1 Introduction

Objectives

- Identify the areas of support that SAS offers.
- Identify the next steps after the completion of this course.

2

Customer Support

SAS provides a variety of resources to help customers.



<http://support.sas.com/resourcekit/>

3

Education

SAS Education provides comprehensive training, including

- more than 200 course offerings
- world-class instructors
- multiple delivery methods
- worldwide training centers.

<http://support.sas.com/training/>



4

SAS Global Certification Program

SAS Education also provides

- globally recognized certifications
- preparation materials
- practice exams.



<http://support.sas.com/certify/>

5

Networking

Social media channels and user group organizations enable you to

- interact with other SAS users and SAS staff
- learn new programming tips and tricks
- get exclusive discounts.



For training-specific information:

<http://support.sas.com/training/socialmedia>

6

SAS Books

SAS Books offers a complete selection of publications, including



- e-Books
- CD-ROM
- hard-copy books
- books written by outside authors.

<http://support.sas.com/bookstore/>

1-800-727-3228

7

Beyond This Course

To expand your SAS skills, remember to activate the **extended learning page** for this course.



Individual learning software is available.

8

Next Steps

To learn more about:



Enroll in this course:

- SAS® Macro Language 1: Essentials
- SAS® Macro Language 2: Advanced Techniques

- SAS/GRAPH®: Essentials
- Producing Maps with SAS/GRAPH®



9

continued...

Next Steps

To learn more about:



Enroll in this course:

- SAS® Report Writing 1: Essentials
- ODS Graphics: Essentials

- SAS® SQL 1: Essentials
- SAS® SQL 2: Processing Data Efficiently in Real-World Scenarios

