SAS® Certification Review: Base Programming for SAS®9

Course Notes

SAS® Certification Review: Base Programming for SAS®9 Course Notes was developed by Michele Ensor. Additional contributions were made by Kent Reeve, Lori Rothenberg, Lorilyn Russell, Larry Stewart, and Su Chee Tay. Editing and production support was provided by the Curriculum Development and Support Department.

SAS® Certification Review: Base Programming for SAS®9 Course Notes

Book code E2408, course code LWCRB/CRB, prepared date 27Mar2013. LWCRB_004

ISBN 978-1-61290-698-0

Table of Contents

Course Description vi

Prerequisites vii

Chapter 1 Introduction 1-1

1.1 Specifics about the SAS Base Programming Exam 1-3

1.2 Specifics about This Review Course 1-10

1.3 SAS Fundamental Concepts 1-14

1.4 Answers to Questions 1-31

Chapter 2 Working with SAS Data Sets 2-1

2.1 Reading and Creating Data Sets 2-3

2.2 Selecting Observations 2-29

2.3 Sorting Observations with the SORT Procedure 2-42

2.4 Combining Data Sets 2-45

2.5 Answers to Questions 2-57

Chapter 3 Working with Raw Data and Microsoft Excel Files 3-1

3.1 Reading Raw Data Files: Part 1 3-3

3.2 Reading Raw Data Files: Part 2 3-16

3.3 Controlling When a Record Loads 3-32

3.4 Reading Microsoft Excel Files 3-38

3.5 Answers to Questions 3-42

Chapter 4 Creating Variables 4-1

4.1 Creating Variables with the Assignment Statement 4-3

4.2 Creating Variables Conditionally 4-8

4.3 Creating Accumulator Variables 4-19

4.4 Answers to Questions 4-25

Chapter 5 Manipulating Data 5-1

5.1 Using Functions to Manipulate Data 5-3

5.2 Converting Character and Numeric Data 5-34

5.3 Processing Data with DO Loops 5-41

5.4 Processing Data with Arrays 5-51

5.5 Answers to Questions 5-59

Chapter 6 Generating Reports 6-1

6.1 Creating Detail Reports with the PRINT Procedure 6-3

6.2 Creating Formats with the FORMAT Procedure 6-22

6.3 Creating Frequency Tables with the FREQ Procedure 6-30

6.4 Creating Summary Reports with the MEANS Procedure 6-35

6.5 Directing Reports to External Files with ODS 6-40

6.6 Answers to Questions 6-46

Chapter 7 Additional Information 7-1

7.1 More Specifics about the SAS Base Programming Exam 7-3

7.2 Additional Preparation Resources 7-6

7.3 Test-Taking Strategies 7-9

Appendix A Exercises and Solutions A-1

A.1 Exercises A-3

Chapter 1 .... A-3

Chapter 2 .... A-4

Chapter 3 .... A-7

Chapter 4 .... A-11

Chapter 5 .... A-14

Chapter 6 .... A-18

A.2 Solutions A-24

Chapter 1 .... A-24

Chapter 2 .... A-25

Chapter 3 .... A-26

Chapter 4 .... A-27

Chapter 5 .... A-29

Chapter 6 .... A-31

Appendix B Practice Exam B-1

B.1 SAS Fundamental Concepts B-3

B.2 Working with SAS Data Sets B-6

B.3 Working with Raw Data and Microsoft Excel Files B-12

B.4 Creating Variables B-20

B.5 Manipulating Data B-26

B.6 Generating Reports B-31

B.7 Scores B-39

B.8 Answers B-40

Course Description

This course provides a review of the majority of topics in the SAS® Base Programming Exam for SAS®9. It addresses the five exam content areas: Accessing Data, Creating Data Structures, Managing Data, Generating Reports, and Handling Errors.

To learn more…

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

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

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

Prerequisites

Before attending this course, you should

- be an experienced programmer with knowledge of the five exam content areas
- have taken SAS® Programming 1: Essentials and SAS® Programming 2: Data Manipulation Techniques or have equivalent experience.

Chapter 1 Introduction

1.1 Specifics about the SAS Base Programming Exam 1-3

1.2 Specifics about This Review Course 1-10

1.3 SAS Fundamental Concepts..1-14

1.4 Answers to Questions..1-31

1.1 Specifics about the SAS Base Programming Exam

SAS Certification

SAS offers professional certifications that validate a candidate's knowledge within the following areas:

- SAS Foundation
- SAS Advanced Analytics
- SAS Information Management
- SAS Data Management
- SAS Business Intelligence

6

SAS Foundation Credentials and Exams

SAS offers the following credentials and exams in the SAS Foundation area:

Credentials	Certification Exams
SAS Certified Base Programmer for SAS®9	➢ SAS Base Programming for SAS®9
SAS Certified Advanced Programmer for SAS®9	➢ SAS Base Programming for SAS®9 ➢ SAS Advanced Programming for SAS®9
SAS Certified Clinical Trials Programmer Using SAS®9	➢ Clinical Trials Programming Using SAS®9 OR ➢ SAS Base Programming for SAS®9 ➢ Clinical Trials Programming Using SAS®9 – Accelerated Version

7

SAS Base Programming Exam for SAS®9

The intended candidate for the SAS Base Programming Exam is someone with current SAS programming ***experience*** in the following five content areas:

1. Accessing Data
2. Creating Data Structures
3. Managing Data
4. Generating Reports
5. Handling Errors

In addition, the candidate must be familiar with the enhancements and new functionality available in SAS®9.

8

1. Accessing Data

- Use FORMATTED and LIST input to read raw data files.
- Use INFILE statement options to control processing when you read raw data files.
- Use various components of an INPUT statement to process raw data files, including column and line pointer controls and trailing @ controls.
- Combine SAS data sets.
- Access Microsoft Excel workbooks.

10

2. Creating Data Structures

- Create temporary and permanent SAS data sets.
- Create and manipulate SAS date values.
- Export data to standard and comma-delimited raw data files.
- Control which observations and variables in a SAS data set are processed and output.

11

3. Managing Data

- Investigate SAS libraries using utility procedures.
- Sort observations in a SAS data set.
- Use assignment statements in the DATA step.
- Modify variable attributes using options and statements in the DATA step.
- Accumulate subtotals and totals using DATA step statements.
- Use SAS functions to manipulate character data, numeric data, and SAS date values.
- Use SAS functions to convert character data to numeric and vice versa.
- Process data using DO loops and SAS arrays.
- Validate and clean data.

12

4. Generating Reports

- Use the PRINT procedure to generate list reports.
- Use Base SAS procedures to generate summary reports and frequency tables.
- Enhance reports through the use of labels, SAS formats, user-defined formats, titles, footnotes, and SAS reporting options.
- Use ODS statements to direct reports to external files.

13

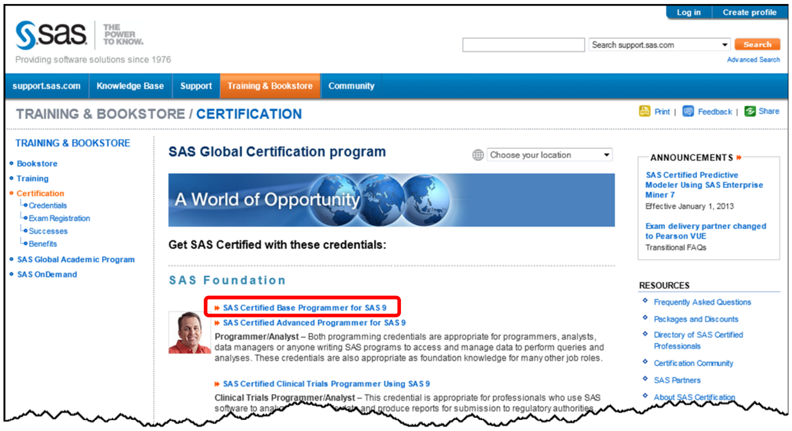
5. Handling Errors

- Identify and resolve programming logic errors.
- Recognize and correct syntax errors.
- Examine and resolve data errors.

14

Exam Details

For up-to-date information about the SAS Base Programming exam, go to http://support.sas.com/certify.



16

Exam Details

Go to the following link to complete the exam details:
http://support.sas.com/certify/creds/bp.html#t2

- Exam is administered by SAS and _____________.
- _____ multiple-choice and short-answer questions
- Must achieve score of ______ correct to pass
- _______ to complete exam
- Exam taken on a computer
- Closed book
- Score received after completing the exam

Exam details are subject to change at any time.

17

Multiple-Choice Questions

All multiple-choice questions contain a question and four possible answers.

For example:

Which city is the headquarters for SAS? ← **question**

A. Boston
B. Cary
C. Detroit
D. San Francisco

four possible answers

18

Multiple-Choice Questions

There is only one correct answer.

For example:

Which city is the headquarters for SAS?

A. Boston
B. Cary ← **one correct answer**
C. Detroit
D. San Francisco

19

Multiple-Choice Questions

Given the following data set:

State	City
MA	Boston
NC	Cary
MI	Detroit
CA	San Francisco

Support information might appear prior to the question.

What is the value of **`City`** for the second observation?

A. Boston

B. Cary ← one correct answer

C. Detroit

D. San Francisco

20

Short-Answer Questions

All short-answer questions contain a question with explicit instructions for entering the answer. Support information can appear before the question.

For example:

question

What is the maximum length for a SAS variable name?

Enter your numeric answer. ← explicit instructions

32 ← correct answer

21

1.2 Specifics about This Review Course

SAS Certification Review: Base Programming for SAS®9

This course is a review of the majority of the topics in the SAS Base Programming Exam for SAS®9.

The five content areas of the exam are presented in six chapters.

- Accessing Data
- Creating Data Structures
- Managing Data
- Generating Reports
- Handling Errors

Chapter	Section
1	SAS Fundamental Concepts
2	Working with SAS Data Sets
3	Working with Raw Data and Excel Files
4	Creating Variables
5	Manipulating Data
6	Generating Reports

23

SAS Certification Review: Base Programming for SAS®9

This course is a review of the majority of the topics in the SAS Base Programming Exam for SAS®9.

re·view
1: to view or see again
2: to examine or study again

This course assumes prior knowledge of the topics.

24

SAS Certification Review: Base Programming for SAS®9

This course is a review of ~~the~~ majority ~~of the~~ topics in the SAS Base Programming Exam for SAS®9.

ma·jor·i·ty
1 : a number or percentage equaling more than half of a total

The certification exam can include topics beyond those topics discussed in this course.

25

SAS Certification Review: Base Programming for SAS®9

This review course will

- refresh your mind on some details that you might have forgotten
- include topics with which you might have limited experience
- familiarize you with SAS terminology
- determine in what areas you need to do additional studying
- help you determine whether you are ready to take the exam
- provide practice for answering SAS questions.

26

Questions in This Course

Different question format types are used throughout the lecture portion of this course.

- multiple choice with one correct answer (A-D)
- multiple choice with more than one correct answer (1-6)
- yes or no
- discussion
- polls

Remember: The certification exam does not use all of these question format types.

27

Questions in This Course

All questions in this course are identified with a Q in the top right corner of the slide.

Selecting Observations

The following program is submitted:

```
data work.portion;
   set sashelp.retail(firstobs=5 obs=10);
run;
```

How many observations are in the output data set?

A. 5
B. 6
C. 10
D. 14

72

28

Questions in This Course

All answers to the questions are identified with an **A** in the top right corner of the slide.

Selecting Observations

The following program is submitted:

```
data work.portion;
   set sashelp.retail(firstobs=5 obs=10);
run;
```

How many observations are in the output data set?

A. 5
B. 6
C. 10
D. 14

73

Answer slides are not in your course notes.
A summary of the correct answers is located in the last section of each chapter.

29

Exercises in This Course (Appendix A)

Different exercise format types are used throughout this course.

- True or False
- Answer the Questions
- Match the Values
- Fill in the Blanks
- Fill in the Table
- Complete the Program
- Add a Statement
- Find the Mistakes

Some exercises are completed ***before*** a topic is reviewed, whereas other exercises are completed ***after*** a topic is reviewed.

Remember: The certification exam does not use all of these exercise format types.

30

Practice Exam in This Course (Appendix B)

The practice exam contains 50 questions in six sections.

Chapter	Section	Questions	Time Limit
1	SAS Fundamental Concepts	5	9 minutes
2	Working with SAS Data Sets	9	15 minutes
3	Working with Raw Data and Excel Files	9	15 minutes
4	Creating Variables	9	15 minutes
5	Manipulating Data	9	15 minutes
6	Generating Reports	9	15 minutes

Remember: Refer to the certification website for the actual number of questions and time limit.

31

1.3 SAS Fundamental Concepts

Refer to Exercise 1 for Chapter 1 in Appendix A.

SAS Steps

The DATA step and the PROC (procedure) step are the two types of steps in a SAS program.

```
data work.enroll;
   infile 'enrollment.dat';
   input @1 first_name $8. @9 last $9.
         @20 state2 $2. @23 _age_ 2.
         @26 enrolldate date9.;
run;
```

DATA Step

```
libname project 'C:\workshop\winsas\lwcrb';

proc sort data=work.enroll
          out=project.enroll;
   by last;
run;
```

PROC Step

```
proc print data=project.enroll;
   var last state2 _age_ enrolldate;
   format enrolldate mmddyy10.;
run;
```

PROC Step

37

DATA Step

In general, the DATA step manipulates data.

- The input for a DATA step can be of several types, such as raw data or a SAS data set.
- The output from a DATA step can be of several types, such as a SAS data set or a report.

input raw data

output data set

```
data work.enroll;
   infile 'enrollment.dat';
   input @1 first_name $8. @9 last $9.
         @20 state2 $2. @23 _age_ 2.
         @26 enrolldate date9.;
run;
```

38

DATA Step

Yes or No: Do all SAS programs contain a DATA step?

39

PROC Step

In general, the PROC step analyzes data, produces output, or manages SAS files.

- The input for a PROC step is usually a SAS data set.
- The output from a PROC step can be of several types, such as a report or an updated SAS data set.

input data set

output data set

```
proc sort data=work.enroll
          out=project.enroll;
   by last;
run;

proc print data=project.enroll;
   var last state2 _age_ enrolldate;
   format enrolldate mmddyy10.;
run;
```

input data set

output report

41

SAS Statements

A SAS statement is a series of items that might include keywords, SAS names, special characters, and operators.

The two types of SAS statements are as follows:

- those that are used in DATA and PROC steps
- those that are global in scope and can be used anywhere in a SAS program

All SAS statements end with a semicolon.

42

SAS Statements

Q

How many statements are in the PROC SORT step?

```
proc sort data=work.enroll
          out=project.enroll;
   by last;
run;

proc print data=project.enroll;
   var last state2 _age_ enrolldate;
   format enrolldate mmddyy10.;
run;
```

43

Global Statements

Global statements

- are used anywhere in a SAS program
- stay in effect until changed or canceled, or until you end your SAS session.

```
libname project 'C:\workshop\winsas\lwcrb';

proc sort data=work.enroll
          out=project.enroll;
   by last;
run;
```

global statement

45

Global Statements

What are some additional examples of global statements?

1. DATA
2. TITLE
3. LABEL
4. FORMAT
5. OPTIONS
6. FOOTNOTE

46

SAS Data Sets

A SAS data set has these characteristics:

- is a SAS file stored in a SAS library that SAS creates and processes
- contains data values that are organized as a table of observations (rows) and variables (columns)
- contains descriptor information such as the data types and lengths of the variables

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

48

SAS Libraries

A SAS library is a collection of one or more SAS files, including SAS data sets, that are referenced and stored as a unit.

- In a directory-based operating environment, a SAS library is a group of SAS files that are stored in the same directory.
- In z/OS (OS/390), a SAS library is a group of SAS files that are stored in an operating environment file.

49

SAS Libraries

A logical name (*libref*) can be assigned to a SAS library using the LIBNAME statement.

```
libname project 'C:\workshop\winsas\lwcrb';
```

libref

physical location of SAS library

The *libref*

- can be up to 8 characters long
- must begin with a letter (A-Z) or an underscore (_)
- can contain only letters, digits (0-9), or underscores.

50

SAS Libraries

Which of the following sentences is ***true*** concerning the LIBNAME statement?

A. The LIBNAME statement must go in a DATA step.

B. The LIBNAME statement must end in a semicolon.

C. The LIBNAME statement must be the first statement in a program.

D. The LIBNAME statement must be followed by the RUN statement.

51

Two-Level SAS Data Set Names

A SAS data set can be referenced using a two-level SAS data set name.

```
proc sort data=work.enroll out=project.enroll;
```

libref ***data-set*** ***libref*** ***data-set***

- *libref* is the logical name that is associated with the physical location of the SAS library.
- *data-set* is the data set name, which can be up to 32 characters long, must begin with a letter or an underscore, and can contain letters, digits, and underscores.

53

One-Level SAS Data Set Names

A data set referenced with a one-level name is automatically assigned to the **Work** library by default.

For example, the following two statements are equivalent:

```
proc sort data=work.enroll out=project.enroll;
```

```
proc sort data=enroll out=project.enroll;
```

54

Temporary and Permanent SAS Data Sets

How many of the following data sets are permanent data sets?

```
work.enroll
```

```
temp.enroll
```

```
project.enroll
```

```
enroll
```

55

Temporary and Permanent SAS Data Sets

A *temporary* SAS data set is one that exists only for the current SAS session or job.

- The **Work** library is a temporary data library.
- Data sets held in the **Work** library are deleted at the end of the SAS session.

A *permanent* SAS data set is one that resides on the external storage medium of your computer and is not deleted when the SAS session terminates.

- Any data library referenced with a LIBNAME statement is considered a permanent data library by default.

57

Variables

Data values are organized into columns called *variables*.

Variables have attributes, such as name and type, that enable you to identify them and that define how they can be used.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

58

Variable Names

By default, a SAS variable name

- can be up to 32 characters long
- must begin with a letter (A-Z) or an underscore (_)
- can contain only letters, digits (0-9), or underscores.

first_name	last	state2	age	enrolldate

59

Variable Names

Which of the following variable names is valid?

A. street#
B. zip_code
C. 2address
D. last name

60

Variable Types

The two types of SAS variables are listed below:

- character
- numeric

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

three character variables

two numeric variables

62

Variable Types: Character

Character variables are stored with a length of 1 to 32,767 bytes with 1 character equaling 1 byte.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

8 bytes | 9 bytes | 2 bytes

Character variables can contain letters (A-Z), numeric digits (0-9), and other special characters (_, #, %, &,…).

63

Variable Types: Numeric

Numeric variables are stored as floating-point numbers with a default byte size of 8.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

8 bytes 8 bytes

To be stored as a floating point number, the numeric value can contain numeric digits (0-9), plus or minus sign, decimal point, and E for scientific notation.

64

Variable Types

How should a date be stored in SAS?

A. character
B. numeric

65

SAS Dates

A SAS date value is a value that represents the number of days between January 1, 1960, and a specified date.

- Dates before January 1, 1960, are negative numbers.
- Dates after January 1, 1960, are positive numbers.

To reference a SAS date value in a SAS program, use a SAS date constant.

- A SAS date constant is a date (DDMMMYYYY) in quotation marks followed by the letter D.
- Example:

```
'12NOV1986'd
```

67

SAS Dates

Q

What is the numeric SAS date value for December 25, 1959?

A. -6
B. -7
C. 6
D. 8

68

Missing Data

Missing data is a value that indicates that no data value is stored for the variable in the current observation.

- A missing numeric value is displayed as a single period (.).
- A missing character value is displayed as a blank space.

VIEWTABLE: Project.Enroll

	first_name	last	state2	_age_	enrolldate
1	Danny	Brown	CO	.	15684
2	William	Johnson		22	17318
3	Samantha	McCormick	CA	47	16674
4	Tina	Stewart	TX	53	14287

70

CONTENTS Procedure

The CONTENTS procedure shows the descriptor portion of a SAS data set.

```
proc contents data=project.enroll;
run;
```

```
    Alphabetic List of Variables and Attributes

          #    Variable       Type    Len

          4    _age_          Num       8
          5    enrolldate     Num       8
          1    first_name     Char      8
          2    last           Char      9
          3    state2         Char      2
```

partial output

The VARNUM option can be used to print the variable list in the order of the variables' positions in the data set.

71

CONTENTS Procedure

Q

Which step displays the directory of the **Project** library and suppresses printing the contents of individual data sets?

A.
```
proc contents data=project;
run;
```

B.
```
proc contents data=project.all;
run;
```

C.
```
proc contents data=project nocontents;
run;
```

D.
```
proc contents data=project._all_ nods;
run;
```

72

PRINT Procedure

The PRINT procedure can show the data portion of a SAS data set.

```
proc print data=project.enroll;
run;
```

```
       first_
Obs    name        last          state2    _age_    enrolldate

 1     Danny       Brown           CO        .         15684
 2     William     Johnson                  22         17318
 3     Samantha    McCormick       CA       47         16674
 4     Tina        Stewart         TX       53         14287
```

74

CONTENTS and PRINT Procedures

Which program has the correct syntax for the CONTENTS and PRINT procedures?

A.
```
proc contents project.enroll;
run;
proc print project.enroll;
run;
```

B.
```
proc contents data=project.enroll
run;
proc print data=project.enroll
run;
```

C.
```
proc contents data=project.enroll;
run;
proc print data=project.enroll;
run;
```

75

Comments

There are two ways to add comments in a SAS program.

```
   *The SORT procedure is creating a data set;

proc sort data=work.enroll
          out=project.enroll;
   by last;
run;

   /* The PRINT procedure is creating a report. */

proc print data=project.enroll;
   var last state2 _age_ enrolldate;
   format enrolldate mmddyy10.;
run;
```

During processing, SAS ignores text in comments.

77

SAS Log

The SAS log is a record of your submitted SAS program.

```
125  libname project 'C:\workshop\winsas\lwcrb\data';

NOTE: Libref PROJECT was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\workshop\winsas\lwcrb\data

126  proc sort data=work.enroll
127            out=project.enroll;
128     by last;
129  run;

NOTE: There were 4 observations read from the data set WORK.ENROLL.
NOTE: The data set PROJECT.ENROLL has 4 observations and 5 variables.
```

- Original program statements are identified by line numbers.
- SAS messages can include the words NOTE, INFO, WARNING, or ERROR.

78

SAS Log

What are the issues with the following program based on the SAS log?

```
154  proc content data=project.enroll;
ERROR: Procedure CONTENT not found.
155  run;

NOTE: The SAS System stopped processing this step because of errors.

156  proc print project.enroll;
                --------------
                22            200
ERROR 22-322: Syntax error, expecting one of the following: ;, DATA,
              DOUBLE, HEADING, LABEL, N, NOOBS, OBS, ROUND, ROWS,
              SPLIT, STYLE, UNIFORM, WIDTH.
ERROR 200-322: The symbol is not recognized and will be ignored.
157  run;

NOTE: The SAS System stopped processing this step because of errors.
```

79

1.4 Answers to Questions

Question Slide Number	Answer
39	No
43	three statements
46	2., 5., and 6.
51	B.
55	two permanent data sets (**temp.enroll** and **project.enroll**)
60	B.
65	B.
68	B.
72	D.
75	C.
79	• CONTENTS misspelled • DATA= missing

Chapter 2 Working with SAS Data Sets

2.1 Reading and Creating Data Sets 2-3

2.2 Selecting Observations...2-29

2.3 Sorting Observations with the SORT Procedure...2-42

2.4 Combining Data Sets...2-45

2.5 Answers to Questions..2-57

2.1 Reading and Creating Data Sets

Reading and Creating Data Sets

```
data work.newprice;
   set golf.supplies;
   <additional programming statements>
run;
```

1. What is the name of the data set being read?

2. What is the name of the data set being created?

3

DATA Statement

The DATA statement begins a DATA step and provides names for any output SAS data sets that are created.

```
data work.newprice;      output data set
   set golf.supplies;
   <additional programming statements>
run;
```

- The DATA statement can create temporary or permanent data sets.

5

SET Statement

The SET statement reads an observation from one or more SAS data sets for further processing in the DATA step.

```
data work.newprice;
   set golf.supplies;
   <additional programming statements>
run;
```

input data set

- By default, the SET statement reads all variables and all observations from the input data sets.
- The SET statement can read temporary or permanent data sets.

6

Additional Programming Statements

Additional programming statements can be added to perform further processing in the DATA step.

For example, an assignment statement can be added to create a new variable based on an expression.

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

creates the variable saleprice based on the price variable from the golf.supplies data set

7

Additional Programming Statements

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

input data set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

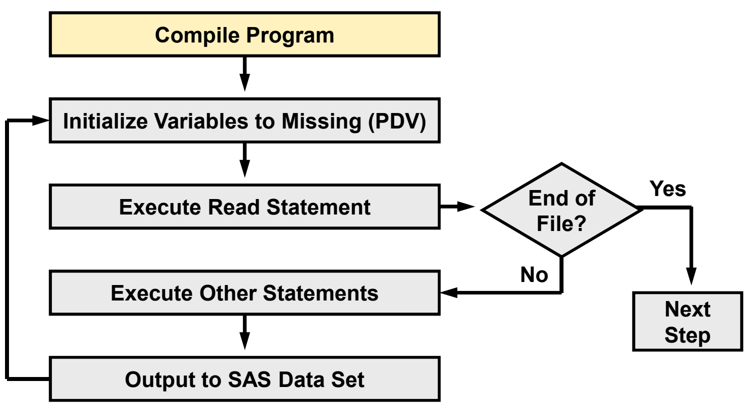
How many variables are in the output data set **work.newprice**?

A. 3
B. 4
C. 5
D. 6

8

DATA Step Execution

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



10

Compilation Phase

During the compilation phase, SAS does the following:

- checks the syntax of the SAS statements
- translates the statements into machine code
- identifies the name, type, and length of each variable

The following three items are potentially created:

- input buffer
- program data vector (PDV)
- descriptor information

11

Input Buffer

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

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

12

Program Data Vector (PDV)

The *program data vector* is a logical area in memory where SAS builds a data set, one observation at a time.

Along with data set variables and computed variables, the PDV contains the following two automatic variables:

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

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

13

Q

Program Data Vector (PDV)

Which of the following statements is ***false*** concerning the **_N_** and **_ERROR_** variables?

A. SAS does not write the **_N_** and **_ERROR_** variables to the output data set.

B. SAS increments the **_N_** variable by 1 for each iteration of the DATA step.

C. SAS automatically generates the **_N_** and **_ERROR_** variables for every DATA step.

D. SAS sets the **_ERROR_** variable equal to the total number of errors caused by the data during execution.

14

Program Data Vector (PDV)

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Program Data Vector (PDV):

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8

variables from golf.supplies data set

variables from assignment statement

automatic variables

16

Program Data Vector (PDV)

Q

Which of the following is ***not*** one of the items in the PDV at compile time?

A. byte size of the variable
B. initial value of the variable
C. name of the variable
D. type (character or numeric) of the variable

17

Descriptor Information

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

Examples of descriptor information include the following:

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

19

Descriptor Information

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Partial Descriptor Information (Variable Attributes)

#	Variable	Type	Len
1	mfg	Char	6
3	price	Num	8
4	saleprice	Num	8
2	type	Char	8

20

Execution Phase

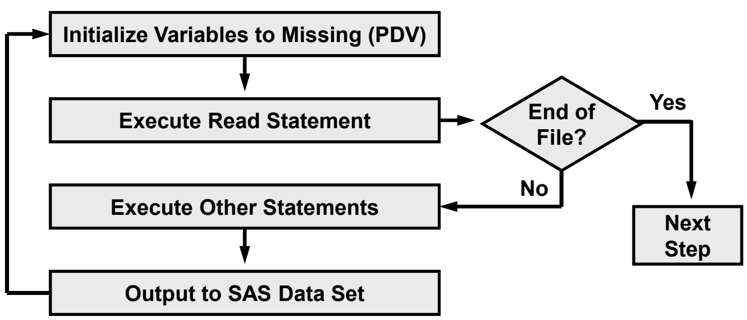
During the execution phase, SAS does the following:

- initializes the PDV to missing and sets the initial values of **_N_** and **_ERROR_**
- reads data values into the PDV
- executes any subsequent programming statements
- outputs the observation to a SAS data set
- returns to the top of the DATA step
- resets the PDV to missing for any variables not read directly from a data set and increments **_N_** by 1
- repeats the process until the end-of-file is detected

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

21

Execution Phase



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

22

Execution Phase

- initializes the PDV to missing and sets the initial values of **_N_** and **_ERROR_**

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
		.	.	0	1

23 ...

Execution Phase

- reads data values into the PDV

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Distance	8.1	.	0	1

24 ...

Execution Phase

- executes any subsequent programming statements

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Distance	8.1	6.075	0	1

25 ...

Execution Phase

- outputs the observation to a SAS data set

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

implicit output

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Distance	8.1	6.075	0	1

observation 1

26 ...

Execution Phase

- returns to the top of the DATA step

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Distance	8.1	6.075	0	1

27 ...

Execution Phase

- resets the PDV to missing for any variables not read directly from a data set and increments **_N_** by 1

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Distance	8.1	.	0	2

28 ...

Execution Phase

- reads data values into the PDV

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Spin	8.25	.	0	2

29 ...

Execution Phase

- executes any subsequent programming statements

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Spin	8.25	6.1875	0	2

30 ...

Execution Phase

- outputs the observation to a SAS data set

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

implicit output

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Spin	8.25	6.1875	0	2

observation 2

31 ...

Execution Phase

- returns to the top of the DATA step

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Spin	8.25	6.1875	0	2

32 ...

Execution Phase

- repeats the process until the end of file is detected

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Program Data Vector (PDV)

mfg $ 6	type $ 8	price N 8	saleprice N 8	_ERROR_ N 8	_N_ N 8
Crew	Spin	8.25	6.1875	0	2

33

Execution Phase

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

How many times does SAS iterate through the DATA step?

A. 0
B. 1
C. 9
D. 10

34

Execution Phase

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Next steps:
- Eliminate variables.
- Add dollar signs.
- Add column headings.
- Use the debugger.

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Output Data Set

VIEWTABLE: Work.Newprice

	mfg	type	price	saleprice
1	Crew	Distance	8.1	6.075
2	Crew	Spin	8.25	6.1875
3	Crew	Titanium	9.5	7.125
4	Hi-fly	X12000	13.75	10.3125
5	Hi-fly	X22000	14.6	10.95
6	White	Strata	10.6	7.95
7	White	Aero	12.3	9.225
8	White	XL	14.5	10.875
9	White	Flite	16.2	12.15

36

DROP and KEEP Statements

- The DROP statement specifies the names of the variables to omit from the output data set.
- The KEEP statement specifies the names of the variables to write to the output data set.

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
   drop mfg price;
run;
```

Placement of statement is irrelevant; statement is applied at output time.

Partial Data Set

Output Data Set

VIEWTABLE: Work.Newprice

	mfg	type	price	saleprice
1	~~Crew~~	Distance	~~8.1~~	6.075
2	~~Crew~~	Spin	~~8.25~~	6.1875
3	~~Crew~~	Titanium	~~9.5~~	7.125
4	~~Hi-fly~~	X12000	~~13.75~~	10.3125

37

Q

DROP and KEEP Statements

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
   drop mfg price;
run;
```

Input Data Set

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

Which KEEP statement is equivalent to the DROP statement in the above program?

A.
```
keep type;
```

B.
```
keep type saleprice;
```

38

FORMAT and LABEL Statements

Chapter 6

- The FORMAT statement associates formats to variable values.
- The LABEL statement assigns descriptive labels to variable names.

```
data work.newprice;
   set golf.supplies;
   saleprice = price * 0.75;
   drop mfg price;
   format saleprice dollar18.2;
   label type='Type of Ball'
         saleprice='Sale Price';
run;
```

40

FORMAT and LABEL Statements Chapter 6

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

```
            Alphabetic List of Variables and Attributes

 #    Variable     Type    Len    Format          Label

 2    saleprice    Num       8    DOLLAR18.2      Sale Price
 1    type         Char      8                    Type of Ball
```

- FORMAT and LABEL statements assigned in a DATA step are considered ***permanent*** attributes (stored in the descriptor portion).

41

FORMAT and LABEL Statements Chapter 6

```
proc print data=work.newprice label;
run;
```

```
             Obs    Type of Ball          Sale Price

               1    Distance                   $6.08
               2    Spin                       $6.19
               3    Titanium                   $7.13
               4    X12000                    $10.31
               5    X22000                    $10.95
               6    Strata                     $7.95
               7    Aero                       $9.23
               8    XL                        $10.88
               9    Flite                     $12.15
```

42

DATA Step Debugger

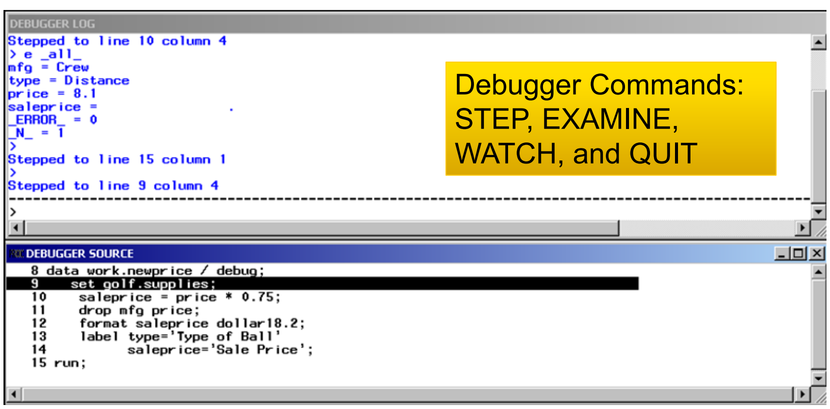
How is the DATA step debugger invoked?

A. adding a / DEBUG option to the DATA statement
B. adding a DEBUG statement after the DATA statement
C. adding the DEBUG option to the OPTIONS statement
D. adding the DEBUG=YES option to the OPTIONS statement

43

DATA Step Debugger

The DATA step debugger consists of windows and a group of commands that provide an interactive way to identify logic and data errors in DATA steps.



45

PUTLOG Statement

The PUTLOG statement can be used to write messages to the SAS log to help identify logic errors.

```
data work.newprice;
   putlog 'Top of Step';
   set golf.supplies;
   putlog type $quote12. price=;
   saleprice = price * 0.75;
   putlog _all_;
   drop mfg price;
run;
```

```
Top of Step
"Distance"  price=8.1
mfg=Crew type=Distance price=8.1 saleprice=6.075 _ERROR_=0 _N_=1
Top of Step
"Spin"      price=8.25
mfg=Crew type=Spin price=8.25 saleprice=6.1875 _ERROR_=0 _N_=2
...
```

46

Refer to Exercise 1 for Chapter 2 in Appendix A.

Implicit Output

By default, at the end of each iteration, every DATA step contains an implicit OUTPUT statement that tells SAS to write observations to the data set or data sets that are being created.

```
data work.total;
   set work.scores;
   total=test1+test2;
run;
```

implicit output

Input Data Set

VIEWTABLE: Work.Scores

	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.Total

	name	test1	test2	total
1	Kent	73	79	152
2	Mary	89	94	183
3	Sally	75	86	161
4	Thomas	92	95	187

49

Implicit Output

Yes or No: Will the following two programs produce the same results?

```
data work.total;
   set work.scores;
   total=test1+test2;
run;
```

```
data work.total;
   set work.scores;
   total=test1+test2;
   output;
run;
```

50

OUTPUT Statement

The OUTPUT statement without arguments causes the current observation to be written to all data sets that are named in the DATA statement.

```
data work.total;
   set work.scores;
   total=test1+test2;
   output;
run;
```

52

OUTPUT Statement

Multiple OUTPUT statements can be used in a DATA step.

```
data work.rotate;
   set work.scores;
   test=test1;
   output;
   test=test2;
   output;
   drop test1 test2;
run;
```

Input Data Set

VIEWTABLE: Work.Scores

	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.Rotate

	name	test
1	Kent	73
2	Kent	79
3	Mary	89
4	Mary	94
5	Sally	75
6	Sally	86
7	Thomas	92
8	Thomas	95

53

OUTPUT Statement

Q

```
data work.rotate;
   set work.scores;
   test=test1;
   output;
   test=test2;
   drop test1 test2;
run;
```

Input Data Set

VIEWTABLE: Work.Scores

	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

How many observations are in the final data set, **work.rotate**?

A. 0

B. 2

C. 4

D. 8

54

OUTPUT Statement

Placing an explicit OUTPUT statement in a DATA step overrides the implicit output, and SAS adds an observation to a data set only when an explicit OUTPUT statement is executed.

```
data work.rotate;
   set work.scores;
   test=test1;
   output;
   test=test2;
   drop test1 test2;
run;
```

no implicit output

Input Data Set

VIEWTABLE: Work.Scores

	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.Rotate

	name	test
1	Kent	73
2	Mary	89
3	Sally	75
4	Thomas	92

56

Creating Multiple Data Sets

- The DATA statement can specify multiple output data sets.
- The OUTPUT statement can specify the data set names.

```
data work.first
     work.second;
   set work.scores;
   test=test1;
   output work.first;
   test=test2;
   output work.second;
   drop test1 test2;
run;
```

57

Creating Multiple Data Sets

```
data work.first
     work.second;
   set work.scores;
   test=test1;
   output work.first;
   test=test2;
   output work.second;
   drop test1 test2;
run;
```

Input Data Set

VIEWTABLE: Work.Scores

	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.First

	name	test
1	Kent	73
2	Mary	89
3	Sally	75
4	Thomas	92

Output Data Set

VIEWTABLE: Work.Second

	name	test
1	Kent	79
2	Mary	94
3	Sally	86
4	Thomas	95

58

Q

Creating Multiple Data Sets

```
data work.total work.first work.second;
   set work.scores;
   total=test1+test2;
   output;
run;
```

Which data set(s) does the OUTPUT statement populate?

A. **work.total**
B. **work.first**
C. **work.second**
D. **work.total**, **work.first**, and **work.second**

59

Creating Multiple Data Sets

Using the OUTPUT statement without arguments causes the current observation to be written to all data sets that are named in the DATA statement.

```
data work.total
     work.first
     work.second;
   set work.scores;
   total=test1+test2;
   output;
   drop test1 test2;
run;
```

Output Data Set

VIEWTABLE: Work.Total

	name	total
1	Kent	152
2	Mary	183
3	Sally	161
4	Thomas	187

Output Data Set

VIEWTABLE: Work.First

	name	total
1	Kent	152
2	Mary	183
3	Sally	161
4	Thomas	187

Output Data Set

VIEWTABLE: Work.Second

	name	total
1	Kent	152
2	Mary	183
3	Sally	161
4	Thomas	187

61

Creating Multiple Data Sets

```
data work.total
     work.first
     work.second;
   set work.scores;
   total=test1+test2;
   output work.total;
   test=test1;
   output work.first;
   test=test2;
   output work.second;
   drop test1 test2;
run;
```

The DROP and KEEP statements apply to all output data sets.

Output Data Set

VIEWTABLE: Work.Total

	name	total	test
1	Kent	152	.
2	Mary	183	.
3	Sally	161	.
4	Thomas	187	.

Output Data Set

VIEWTABLE: Work.First

	name	total	test
1	Kent	152	73
2	Mary	183	89
3	Sally	161	75
4	Thomas	187	92

Output Data Set

VIEWTABLE: Work.Second

	name	total	test
1	Kent	152	79
2	Mary	183	94
3	Sally	161	86
4	Thomas	187	95

62

DROP= and KEEP= Options

- The DROP= data set option excludes the variables for writing to a specific output data set.
- The KEEP= data set option specifies the variables for writing to a specific output data set.

```
data work.total(keep=name total test1 test2)
     work.first(drop=test1 test2)
     work.second(keep=name total test);
   set work.scores;
   total=test1+test2;
   output work.total;
   test=test1;
   output work.first;
   test=test2;
   output work.second;
run;
```

63

DROP= and KEEP= Options

```
data work.total(keep=name total test1 test2)
     work.first(drop=test1 test2)
     work.second(keep=name total test);
   set work.scores;
   total=test1+test2;
   output work.total;
   test=test1;
   output work.first;
   test=test2;
   output work.second;
run;
```

Output Data Set

VIEWTABLE: Work.First

	name	total	test
1	Kent	152	73
2	Mary	183	89
3	Sally	161	75
4	Thomas	187	92

Output Data Set

VIEWTABLE: Work.Total

	name	test1	test2	total
1	Kent	73	79	152
2	Mary	89	94	183
3	Sally	75	86	161
4	Thomas	92	95	187

Output Data Set

VIEWTABLE: Work.Secon

	name	total	test
1	Kent	152	79
2	Mary	183	94
3	Sally	161	86
4	Thomas	187	95

64

Other Statements Using the OUTPUT Statement

The OUTPUT statement can stand alone or be part of an IF-THEN or SELECT/WHEN statement or be in DO loop processing. Chapters 4 and 5

Example with the IF-THEN statement:

```
data female
     male
     all(keep=name weight height);
   set sashelp.class;
   if sex='F' then output female all;
   else if sex='M' then output male all;
run;
```

- Multiple data sets can be specified in the OUTPUT statement.

65

Other Statements Using the OUTPUT Statement

Q

sashelp.class contains five variables.

```
data female
     male
     all(keep=name weight height);
   set sashelp.class;
   if sex='F' then output female all;
   else if sex='M' then output male all;
run;
```

How many variables are in the output data set **female**?

A. 2
B. 3
C. 5
D. 6

66

Refer to Exercise 2 for Chapter 2 in Appendix A.

2.2 Selecting Observations

Selecting Observations

By default, all observations of the input data set are written to the output data set.

```
data work.all;
   set sashelp.retail;
run;
```

Input data set **sashelp.retail** has 58 observations. → Output data set **work.all** has 58 observations.

71

Selecting Observations

The FIRSTOBS= and OBS= data set options can be used to control which observations are read from the input data set.

```
data work.ten;
   set sashelp.retail(obs=10);
run;
```

Input data set **sashelp.retail** has 58 observations. → Output data set **work.ten** has 10 observations.

FIRSTOBS= and OBS= are valid for input processing only. That is, they are not valid for output processing.

72

Selecting Observations

Q

The following program is submitted:

```
data work.portion;
   set sashelp.retail(firstobs=5 obs=10);
run;
```

How many observations are in the output data set?

A. 5
B. 6
C. 10
D. 14

73

FIRSTOBS= and OBS= Options

- The FIRSTOBS= data set option specifies a starting point for processing an input data set.
- The OBS= data set option specifies an ending point for processing an input data set.

```
data work.portion;
   set sashelp.retail(firstobs=5 obs=10);
run;
```

Input data set **sashelp.retail** has 58 observations. → Output data set **work.portion** has 6 observations (obs # 5, 6, 7, 8, 9, and 10).

The OBS= option specifies the number of the last observation, and not how many observations there are to process.

75

FIRSTOBS= and OBS= Options

Q

Which step has invalid syntax?

A.

```
data shoes(firstobs=101 obs=200);
   set sashelp.shoes;
run;
```

B.

```
data shoes;
   set sashelp.shoes
       (firstobs=101 obs=200);
run;
```

C.

```
proc print data=sashelp.shoes
           (firstobs=101 obs=200);
run;
```

76

Selecting Observations Based on an Expression

The following statements can be used to select observations based on an expression:

- WHERE statement
- subsetting IF statement
- IF-THEN DELETE statement

All three of the statements reference an ***expression***.

78

Expression

An *expression* is a sequence of operands and operators that forms a set of instructions that define a condition for selecting observations.

- *Operands* are the following:
  - constants (character or numeric)
  - variables (character or numeric)
  - SAS functions
- *Operators* are symbols that request a comparison, logical operation, or arithmetic calculation.

79

Operands

- A *constant* is a fixed value such as a number, quoted character string, or date constant.
  - If the value is numeric, do not use quotation marks.
  - If the value is character, use quotation marks.
  - A SAS date constant is a date (DDMMMYYYY) in quotation marks followed by the letter D.
- A *variable* is a variable coming from a data set, a variable created in an assignment statement, or an automatic variable created by the DATA step.
- A SAS *function* is a routine that performs a computation or system manipulation on arguments and returns a value. Chapter 5

80

Comparison Operators

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

Operators		Definition
EQ	=	equal to
NE	^= ~= ¬=	not equal to
GT	>	greater than
GE	>=	greater than or equal to
LT	<	less than
LE	<=	less than or equal to
IN		equal to one of a list

81

Operands and Comparison Operators

Which of the following is ***not*** a valid expression?

A. `qtr1 <= qtr2`
B. `address = ' '`
C. `sales gt 6400`
D. `name ne Mary Ann`

82

Logical Operators

Logical operators combine or modify expressions.

Operators		Definition
AND	&	logical and
OR	|	logical or
NOT	^	logical not

84

Arithmetic Operators

Arithmetic operators indicate that an arithmetic calculation is performed.

Operators	Definition
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction

If a missing value is an operand for an arithmetic operator, the result is a missing value.

85

Logical and Arithmetic Operators

Q

Which of the following is ***not*** a valid expression?

A. `X * 5 / A – C eq Y ** 2`
B. `level = 'up' | type = 'low'`
C. `january + february le 90000`
D. `salary > 50000 title not = 'Manager'`

86

Special WHERE Operators

The WHERE statement can use special WHERE operators.

Operators		Definition
BETWEEN – AND		an inclusive range
CONTAINS	?	a character string
LIKE		a character pattern
SOUNDS LIKE	=*	spelling variation
IS NULL		missing value
IS MISSING		missing value
SAME AND ALSO		augments an expression

88

Special WHERE Operators

```
name like 'M__k%'
```

two underscores

Which names will be selected based on the above expression?

1. Mark
2. Marcia
3. Mickey
4. Matthew
5. Michael

89

Expression Examples

```
sales > 100000
sales eq .
name = 'Smith'
name = ' '
sales gt 100000 and name = 'Smith'
sales gt 100000 or name = 'Smith'
revenue >= 150 and revenue <= 999
revenue between 150 and 999
revenue not between 150 and 999
month contains 'uary'
birthdate > '11JUL1968'd
upcase(state) = 'TX'
```

91

Selecting Observations Based on an Expression

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Which statement must be added to the above program to create an output data set with observations having **saleprice** greater than $10?

A. `if not (saleprice>10) then delete;`

B. `if saleprice>10;`

C. Either statement will work.

92

Selecting Observations Based on an Expression

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
run;
```

Which statement must be added to the above program to create an output data set with observations having **saleprice** greater than $10?

A. `where saleprice>10;`

B. `if saleprice>10;`

C. Either statement will work.

94

Selecting Observations Based on an Expression

There are three ways to select an observation based on an expression:

```
where expression;

if expression;

if not (expression) then delete;
```

not always equivalent

equivalent

96

WHERE Statement

The WHERE statement causes the DATA step to process only those observations from a data set that meet the condition of the expression.

```
data work.newprice;
   set golf.supplies;
   where mfg='White';
   saleprice=price*0.75;
   if saleprice > 10;
run;
```

Placement of statement is irrelevant; statement is applied at input time.

The expression in the WHERE statement

- can reference variables that are from the input data set
- cannot reference variables created from an assignment statement or automatic variables (**_N_** or **_ERROR_**).

97

Q

WHERE Statement

```
data subset;
   set sales;
   difference=actual-predict;
   <insert statement here>
run;
```

Variables in Input Data Set

#	Variable	Type	Len
1	DATE	Num	8
2	STATE	Char	8
3	PRODUCT	Char	10
4	ACTUAL	Num	8
5	PREDICT	Num	8

Which WHERE statement will create an error when submitted, if inserted in the above program?

A. `where actual > predict;`

B. `where difference ge 1000;`

C. `where product in ('CHAIR','SOFA');`

D. `where state='Texas' and date<'01JAN1998'd;`

98

Subsetting IF Statement

The subsetting IF statement causes the DATA step to continue processing only those observations in the program data vector that meet the condition of the expression.

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
   if saleprice > 10;
run;
```

100

Subsetting IF Statement

- If the expression is ***true*** for the observation, SAS continues to execute the remaining statements in the DATA step, including the implicit OUTPUT statement at the end of the DATA step. The resulting SAS data set (or data sets) contains a subset of the original SAS data set.
- If the expression is ***false***, no further statements are processed for that observation, the current observation is not written to the data set, the remaining program statements in the DATA step are not executed, and SAS immediately returns to the beginning of the DATA step.

101

Subsetting IF Statement

Which program will create an error when submitted?

Variables in Input Data Set

#	Variable	Type	Len
1	DATE	Num	8
2	STATE	Char	8
3	PRODUCT	Char	10
4	ACTUAL	Num	8
5	PREDICT	Num	8

A.
```
data subset;
   set sales;
   if difference < 500;
   difference=actual-predict;
   if state='Texas';
run;
```

B.
```
data subset;
   set sales;
   difference=actual-predict;
   if difference between 500 and 1000;
run;
```

C.
```
data subset;
   set sales;
   difference=actual-predict;
   if product='CHAIR' and difference ge 100;
run;
```

102

WHERE Statement versus Subsetting IF Statement

- The WHERE statement selects observations before they are brought into the program data vector.
- The subsetting IF statement selects observations that were read into the program data vector.

```
data work.newprice;
   set golf.supplies;
   where mfg='White';
   saleprice=price*0.75;
   if saleprice > 10;
run;
```

104

IF-THEN DELETE Statement

The IF-THEN DELETE statement causes the DATA step to stop processing those observations in the program data vector that meet the condition of the expression.

```
data work.newprice;
   set golf.supplies;
   saleprice=price*0.75;
   if saleprice <= 10 then delete;
run;
```

If the expression is ***true*** for the observation, the current observation is not written to a data set, and SAS returns immediately to the beginning of the DATA step for the next iteration.

105

Refer to Exercise 3 for Chapter 2 in Appendix A.

2.3 Sorting Observations with the SORT Procedure

SORT Procedure

The SORT procedure does the following:

- orders SAS data set observations by the values of one or more character or numeric variables
- either replaces the original data set or creates a new data set
- produces only an output data set, but no report
- arranges the data set by the values in ascending order by default

```
proc sort data=sashelp.shoes
          out=shoes;
   by descending region product;
run;
```

109

Refer to Exercise 4 for Chapter 2 in Appendix A.

PROC SORT Statement

Examples:

```
proc sort data=sashelp.shoes;
```

```
proc sort data=sashelp.shoes
          out=shoes;
```

```
proc sort data=sashelp.shoes
          out=sasuser.sort;
```

- The DATA= option identifies the input SAS data set.
- The OUT= option names the output data set.
- Without the OUT= option, the SORT procedure overwrites the original data set.

112

BY Statement

The BY statement specifies the sorting variables.

Examples:

```
by region;
```

```
by region product;
```

```
by region subsidiary product;
```

Ascending is the default order.

- PROC SORT first arranges the data set by the values of the first BY variable.
- PROC SORT then arranges any observations that have the same value of the first BY variable by the values of the second BY variable.
- This sorting continues for every specified BY variable.

113

BY Statement

By default, the SORT procedure orders the values by ascending order.

The DESCENDING option reverses the sort order for the variable that immediately follows in the statement.

Examples:

```
by region descending product;
```

```
by descending region product;
```

```
by descending region descending product;
```

114

BY Statement

Q

Yes or No: Will the following program run successfully?

```
proc sort data=sashelp.shoes
          out=shoes;
   by descending region ascending product;
run;
```

115

BY Statement

In addition to the SORT procedure, a BY statement can be used in the DATA step and other PROC steps.

The data sets used in the DATA step and other PROC steps must be sorted by the values of the variables that are listed in the BY statement or have an appropriate index.

```
proc sort data=personnel;
   by descending empid lastname;
run;
proc print data=personnel;
   by descending empid;
run;
```

```
proc sort data=one;
   by id;
run;
proc sort data=two;
   by id;
run;
data both;
   merge one two;
   by id;
run;
```

117

Q

BY Statement

What are the two problems associated with the following program?

```
proc sort data=sashelp.shoes
          out=shoes;
   by descending region product;
run;

data new;
   set sashelp.shoes;
   by region product;
run;
```

118

2.4 Combining Data Sets

Concatenating

If more than one data set name appears in the SET statement, the resulting output data set is a concatenation of all the data sets that are listed.

```
data work.thirdqtr;
   set work.oct
       work.nov
       work.dec;
run;
```

SAS reads all observations from the first data set, then all from the second data set, and so on, until all observations from all the data sets are read.

121

Concatenating

How many observations and variables are in the output data set **work.thirdqtr**?

```
data work.thirdqtr;
   set work.oct
       work.nov
       work.dec;
run;
```

Input Data Set

VIEWTABLE: Work.Oct

	region	date	sales
1	East	17075	1600000
2	West	17075	2100000

Input Data Set

VIEWTABLE: Work.Nov

	region	date	sales
1	East	17106	2500000
2	West	17106	2600000

Input Data Set

VIEWTABLE: Work.Dec

	region	date	sales
1	East	17136	3100000
2	West	17136	3000000

122

Concatenating

How many observations and variables are in the output data set **work.thirdqtr1**?

```
data work.thirdqtr1;
   set work.oct1
       work.nov1
       work.dec1;
run;
```

Input Data Set

VIEWTABLE: Work.Oct1

	region	sales
1	East	1600000
2	West	2100000

Input Data Set

VIEWTABLE: Work.Nov1

	region	sales
1	East	2500000
2	West	2600000

Input Data Set

VIEWTABLE: Work.Dec1

	area	sales
1	East	3100000
2	West	3000000

124

Concatenating

At compile time, SAS puts the variable information from the first data set into the PDV, and then puts the variable information from the second into the PDV, and so on.

```
data company;
   set divisionA divisionB;
run;
```

divisionA

#	Variable	Type	Len
1	name	Char	10
2	state	Char	2

divisionB

#	Variable	Type	Len
1	name	Char	15
2	location	Char	2

name $ 10	state $ 2	location $ 2	_ERROR_ N 8	_N_ N 8

126

Concatenating

```
data company;
   set divisionA divisionB;
run;
```

VIEWTABLE: Work.Company

	name	state	location
1	Joy	SC	
2	Tony	NC	
3	Michael	GA	
4	Margaret	FL	
5	Kelly		CA
6	Roger		NV
7	Mary Marga		TX

Output Data Set

#	Variable	Type	Len
1	name	Char	10
2	state	Char	2
3	location	Char	2

Next steps:
- Control byte size of name.
- Combine state and location.

127

LENGTH Statement

The LENGTH statement specifies the number of bytes for storing variables.

```
data company;
   length name $ 15;
   set divisionA divisionB;
run;
```

VIEWTABLE: Work.Company

	name	state	location
1	Joy	SC	
2	Tony	NC	
3	Michael	GA	
4	Margaret	FL	
5	Kelly		CA
6	Roger		NV
7	Mary Margaret		TX

Output Data Set

#	Variable	Type	Len
1	name	Char	15
2	state	Char	2
3	location	Char	2

128

LENGTH Statement

Q

Yes or No: Will you get the same results if the LENGTH statement is after the SET statement?

```
data company;
   set divisionA divisionB;
   length name $ 15;
run;
```

129

RENAME= Option

The RENAME= data set option changes the names of variables.

```
data company;
   length name $ 15;
   set divisionA(rename=(state=location))
       divisionB;
run;
```

- The RENAME= option specifies the variable that you want to rename equal to the new name of the variable.
- The list of variables to rename must be enclosed in parentheses.

131

RENAME= Option

Which of the following statements has proper syntax for the RENAME= option?

A.
```
set divisionA(rename=name=first,
                          state=location)
    divisionB(rename=name=first);
```

B.
```
set divisionA(rename=(name=first
                           state=location))
    divisionB(rename=(name=first));
```

C.
```
set divisionA(rename=(name=first)
                          (state=location))
    divisionB(rename=(name=first));
```

D.
```
set divisionA(rename=(name=first),
                          (state=location))
    divisionB(rename=(name=first));
```

132

Interleaving

Use a single SET statement with multiple data sets and a BY statement to interleave the specified data sets.

```
data company;
   length name $ 15;
   set divisionA(rename=(state=location))
       divisionB;
   by name;
run;
```

The observations in the new data set are arranged by the values of the BY variable or variables. Then, within each BY group, they are arranged by the order of the data sets in which they occur.

134

Interleaving

The data sets that are listed in the SET statement must be sorted by the values of the variables that are listed in the BY statement, or they must have an appropriate index.

```
data company;
  length name $ 15;
  set divisionA(rename=(state=location))
      divisionB;
  by name;
run;

ERROR: BY variables are not properly sorted on data
       set WORK.DIVISIONB.
name=Roger location=NV FIRST.name=1 LAST.name=1
_ERROR_=1 _N_=3
NOTE: The SAS System stopped processing this step
      because of errors.
```

135

Interleaving

```
proc sort data=divisionA;
   by name;
run;
proc sort data=divisionB;
   by name;
run;
data company;
   length name $ 15;
   set divisionA(rename=(state=location))
       divisionB;
   by name;
run;
```

VIEWTABLE: Work.Company

	name	location
1	Joy	SC
2	Kelly	CA
3	Margaret	FL
4	Mary Margaret	TX
5	Michael	GA
6	Roger	NV
7	Tony	NC

divisionA

divisionB

136

Interleaving

```
proc sort data=divisionA;
   by descending state;
proc sort data=divisionB;
   by descending location;
data company;
   length name $ 15;
   set divisionA(rename=(state=location))
       divisionB;
   by location;
run;
```

What is the result of this program?

A. There is an error because BY variables are not properly sorted.

B. There is an error because the variable **location** is not found in **work.divisionA**.

C. The data set **work.company** is created with seven observations and two variables.

137

Merging

- The MERGE statement joins observations from two or more SAS data sets into single observations.
- The BY statement specifies the common variables to match-merge observations. The variables in the BY statement must be common to all data sets.

```
data combine;
   merge revenue expense;
   by name;
   profit=revenue-expense;
run;
```

- The data sets listed in the MERGE statement must be sorted in the order of the values of the variables that are listed in the BY statement, or they must have an appropriate index.

139

One-to-One Match-Merging

```
data combine;
   merge revenue expense;
   by name;
   profit=revenue-expense;
run;
```

Input Data Set

VIEWTABLE: Work.Revenue

	name	revenue
1	Joy	15000
2	Margaret	26000
3	Michael	18000
4	Tony	25000

Input Data Set

VIEWTABLE: Work.Expense

	name	expense
1	Joy	18000
2	Margaret	20000
3	Michael	19000
4	Tony	19000

Output Data Set

VIEWTABLE: Work.Combine

	name	revenue	expense	profit
1	Joy	15000	18000	-3000
2	Margaret	26000	20000	6000
3	Michael	18000	19000	-1000
4	Tony	25000	19000	6000

140

Many-to-One Match-Merging

```
data moncombine;
   merge monrevenue monexpense;
   by month;
   profit=revenue-expense;
run;
```

Input Data Set

VIEWTABLE: Work.Mon

	name	month	revenue
1	Billy	Apr	5500
2	Janet	Apr	6000
3	Billy	May	5500
4	Janet	May	5000

Input Data Set

VIEWTABLE: Work.Monexpense

	month	expense
1	Apr	5000
2	May	5500

Output Data Set

VIEWTABLE: Work.Moncombine

	name	month	revenue	expense	profit
1	Billy	Apr	5500	5000	500
2	Janet	Apr	6000	5000	1000
3	Billy	May	5500	5500	0
4	Janet	May	5000	5500	-500

141

Nonmatches

```
data combine1;
   merge revenue1 expense1;
   by name;
   profit=revenue-expense;
run;
```

How many observations are in the output data set **work.combine1**?

Input Data Set

VIEWTABLE: Work.Reven...

	name	revenue
1	Caroline	25000
2	Jack	18000
3	Jenna	26000
4	Thomas	15000

Input Data Set

VIEWTABLE: Work.Expen...

	name	expense
1	Caroline	19000
2	Jenna	20000
3	Susan	19000
4	Thomas	18000

142

IN= Option

The IN= option creates a variable that indicates whether the data set contributed data to the current observation.

```
data combine1;
   merge revenue1(in=rev)
         expense1(in=exp);
   by name;
   profit=revenue-expense;
run;
```

Within the DATA step, the value of the variable is 1 if the data set contributed to the current observation, and 0 if the data set did not contribute to the current observation.

144

Q

IN= Option

Which of the following statements is ***false*** concerning the IN= option?

A. The IN= variables are included in the SAS data set that is being created.

B. The values of the IN= variables are available to program statements during the DATA step.

C. When a data set contributes an observation for the current BY group, the IN= value is a numeric 1.

D. The IN= data set option is specified in parentheses after a SAS data set name in the SET and MERGE statements.

145

IN= Option

```
data combine1;
   merge revenue1(in=rev)
         expense1(in=exp);
   by name;
   profit=revenue-expense;
run;
```

name $ 10	revenue N 8	expense N 8	profit N 8	rev N 8	exp N 8	_ERROR_ N 8	_N_ N8
Caroline	25000	19000	6000	1	1	0	1
Jack	18000	.	.	1	0	0	2
Jenna	26000	20000	6000	1	1	0	3
Susan	.	19000	.	0	1	0	4
Thomas	15000	18000	-3000	1	1	0	5

147

IN= Option

Possible Scenarios	
`if rev=1 and exp=1;` `if rev and exp;`	Matches only
`if rev=1 and exp=0;` `if rev and not exp;`	Nonmatches from **revenue1**
`if rev=0 and exp=1;` `if not rev and exp;`	Nonmatches from **expenses1**
`if rev=1;` `if rev;`	All observations from **revenue1** (matches and nonmatches)
`if exp=1;` `if exp;`	All observations from **expenses1** (matches and nonmatches)
`if rev=0 or exp=0;` `if not rev or not exp;`	Nonmatches from **revenue1** and **expenses1**

148

Q

IN= Option

```
data combine1;
   merge revenue1(in=rev)
         expense1(in=exp);
   by name;
   profit=revenue-expense;
run;
```

Which statement will give all observations from the **revenue1** data set regardless of matches or nonmatches?

A. `if rev=1;`

B. `if rev=1 and exp=1;`

C. `if rev=1 and (exp=1 and exp=0);`

D. `if (rev=1 and exp=1) and (rev=1 and exp=0);`

149

IN= Option

```
data revexp revonly exponly;
   merge revenue1(in=rev)
         expense1(in=exp);
   by name;
   if rev=1 and exp=1 then output revexp;
   else if rev=1 and exp=0 then output revonly;
   else if rev=0 and exp=1 then output exponly;
run;
```

Input Data Set

VIEWTABLE: Work.Revenue

	name	revenue
1	Caroline	25000
2	Jack	18000
3	Jenna	26000
4	Thomas	15000

Input Data Set

VIEWTABLE: Work.Expense

	name	expense
1	Caroline	19000
2	Jenna	20000
3	Susan	19000
4	Thomas	18000

Output Data Set

VIEWTABLE: Work.Revonly

	name	revenue	expense
1	Jack	18000	.

Output Data Set

VIEWTABLE: Work.Revexp

	name	revenue	expense
1	Caroline	25000	19000
2	Jenna	26000	20000
3	Thomas	15000	18000

Output Data Set

VIEWTABLE: Work.Exponly

	name	revenue	expense
1	Susan	.	19000

151

Refer to Exercise 5 for Chapter 2 in Appendix A.

2.5 Answers to Questions

Question Slide Number	Answer
3	1. golf.supplies 2. work.newprice
8	B.
14	D.
17	B.
34	C.
38	B.
43	A.
50	Yes
54	C.
59	D.
66	C.
73	B.
76	A.
82	D.
86	D.
89	1. and 3.
92	C.
94	B.
98	B.
102	B.
115	No
118	• DATA step is not using the sorted data set. • The BY statement of the DATA step is not specifying the correct sort order.

(Continued on the next page.)

Question Slide Number	Answer
122	Six observations and three variables
124	Six observations and three variables
129	No
132	B.
137	A.
142	Five observations (three matches and two nonmatches)
145	A.
149	A.

Chapter 3 Working with Raw Data and Microsoft Excel Files

3.1 Reading Raw Data Files: Part 1 3-3

3.2 Reading Raw Data Files: Part 2 3-16

3.3 Controlling When a Record Loads 3-32

3.4 Reading Microsoft Excel Files 3-38

3.5 Answers to Questions 3-42

3.1 Reading Raw Data Files: Part 1

Reading Raw Data Files

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

1. What is the name of the raw data file being read?

2. What is the name of the data set being created?

4

DATA Statement

The DATA statement begins a DATA step and provides names for any output SAS data sets that are created.

Output Data Set

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

- The DATA statement can create temporary or permanent data sets.

6

INFILE Statement

With an INPUT statement, the INFILE statement identifies the physical name of the external file to read.

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

Input Raw Data File

- The physical name is the name that the operating environment uses to access the file.

7

INPUT Statement

The INPUT statement describes the arrangement of values in the input data record and assigns input values to the corresponding SAS variables.

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

8

INPUT Statement

Q

Which of the following is ***not*** an input style for the INPUT statement?

A. list input
B. column input
C. delimited input
D. formatted input

9

INPUT Statement

```
----|----10---|----20---|----30---|----40---|----50
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

The following three ways can describe a record's values in the INPUT statement:

- column input
- formatted input
- list input

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

column

formatted

list

11

Column Input

With column input, the column numbers that contain the value follow a variable name in the INPUT statement.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with column input, data values

- must be in the same columns in all the input data records
- must be in standard form.

12

Formatted Input

With formatted input, an informat follows a variable name and defines how SAS reads the values of this variable. An informat gives the data type and the field width of an input value.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with formatted input, data values

- must be in the same columns in all the input data records
- can be in standard or nonstandard form.

13

List Input

With list input, variable names in the INPUT statement are specified in the same order that the fields appear in the input data records.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with list input, data values

- must be separated with a delimiter
- can be in standard or nonstandard form.

14

Column, Formatted, and List Input

Q

```
input @45 name $10;
```

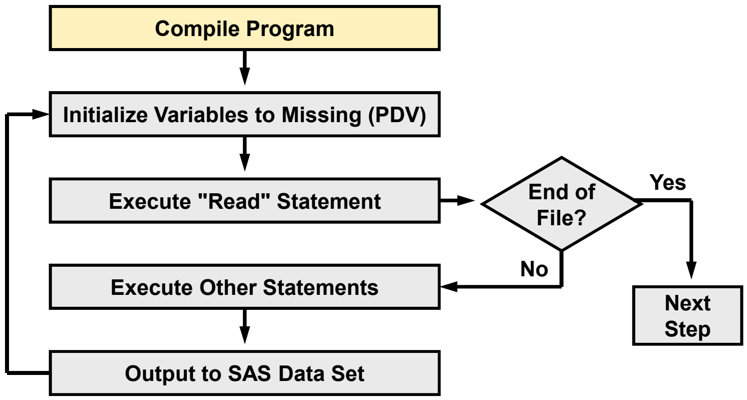
Which input technique is used in the above statement?

A. column input
B. formatted input
C. list input

15

DATA Step Execution

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



17

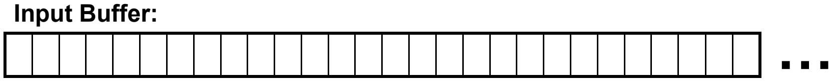
Compilation Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
        @12 bdate mmddyy10.
        @23 allowance comma2.
        hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:



PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8

18

Compilation Phase

Q

What third item is created at compile time in addition to the input buffer and the program data vector (PDV)?

A. report
B. data values
C. raw data file
D. descriptor information

19

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
	.	.	.				0	1

- initializes the PDV to missing and sets the initial values of **_N_** and **_ERROR_**

21 ...

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
        @12 bdate mmddyy10.
        @23 allowance comma2.
        hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

C	h	l	o	e					2		1	1	/	1	0	/	1	9	9	5		$	5	R	u	n	n	...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
Chloe	2	13097	5	Running	Music	Gymnasti	0	1

- reads data values into the input buffer and then the PDV
- executes any subsequent programming statements

22 ...

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
        @12 bdate mmddyy10.
        @23 allowance comma2.
        hobby1 $ hobby2 $ hobby3 $;
run;
```

implicit output

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

C	h	l	o	e					2		1	1	/	1	0	/	1	9	9	5		$	5	R	u	n	n	...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
Chloe	2	13097	5	Running	Music	Gymnasti	0	1

- outputs the observation to a SAS data set

observation 1

23 ...

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
        @12 bdate mmddyy10.
        @23 allowance comma2.
        hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe   2 11/10/1995 $5Running Music Gymnastics
Travis  2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

C	h	l	o	e					2		1	1	/	1	0	/	1	9	9	5		$	5	R	u	n	n

...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
Chloe	2	13097	5	Running	Music	Gymnasti	0	1

- returns to the top of the DATA step

24 ...

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
        @12 bdate mmddyy10.
        @23 allowance comma2.
        hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe   2 11/10/1995 $5Running Music Gymnastics
Travis  2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

C	h	l	o	e					2		1	1	/	1	0	/	1	9	9	5		$	5	R	u	n	n

...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
	.	.	.				0	2

- resets the PDV to missing for any variables not being read directly from a data set and increments **_N_** by 1

25 ...

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Input Buffer:

C	h	l	o	e					2		1	1	/	1	0	/	1	9	9	5		$	5	R	u	n	n

...

PDV:

name $ 8	siblings N 8	bdate N 8	allowance N 8	hobby1 $ 8	hobby2 $ 8	hobby3 $ 8	_ERROR_ N 8	_N_ N 8
	.	.	.				0	2

- repeats the process until the end of file is detected

26

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

How many times does SAS iterate through the DATA step?

A. 0   B. 1   C. 3   D. 4

27

Execution Phase

```
data work.kids;
   infile 'kids.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

```
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
```

Input Raw Data File

Output Data Set

VIEWTABLE: Work.Kids

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	2	13097	5	Running	Music	Gymnasti
2	Travis	2	13909	2	Basebal	Nintendo	Reading
3	Jennifer	0	14477	0	Soccer	Painting	Dancing

29

Data Errors

A data error is when the INPUT statement encounters invalid data in a field.

When SAS encounters a data error, these events occur:

- A note that describes the error is printed in the SAS log.
- The input record (contents of the input buffer) being read is displayed in the SAS log.
- The values in the SAS observation (contents of the PDV) being created are displayed in the SAS log.
- A missing value is assigned to the appropriate SAS variable.
- Execution continues.

30

Data Errors

```
data work.kids1;
   infile 'kids1.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;
```

Input Raw Data File

```
Chloe     2 11/10/1995 $5Running Music Gymnastics
Travis    X 1/30/1998  $2Baseball Nintendo Reading
Jennifer  0 8/21/1999  $0Soccer Painting Dancing
```

The siblings values are read in as numeric.

31

```
data work.kids1;
   infile 'kids1.dat';
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
run;

NOTE: The infile 'kids1.dat' is:

      File Name=c:\workshop\winsas\lwcrb\kids1.dat,
      RECFM=V,LRECL=256

NOTE: Invalid data for siblings in line 2 10-10.
RULE:       ----+----1----+----2----+----3----+----4----+----5----
2           Travis    X 1/30/1998  $2Baseball Nintendo Reading 49
name=Travis siblings=. bdate=13909 allowance=2 hobby1=Baseball
hobby2=Nintendo hobby3=Reading _ERROR_=1 _N_=2
NOTE: 3 records were read from the infile 'kids1.dat'.
      The minimum record length was 47.
      The maximum record length was 49.
NOTE: The data set WORK.KIDS1 has 3 observations and 7 variables.
```

Output Data Set

VIEWTABLE: Work.Kids1

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	2	13097	5	Running	Music	Gymnasti
2	Travis	.	13909	2	Baseball	Nintendo	Reading
3	Jennifer	0	14477	0	Soccer	Painting	Dancing

32

Q

Data Errors

What is the reason for this invalid data?

```
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 hobby2 $ hobby3 $;
  run;

NOTE: The infile 'kids.dat' is:

      File Name=c:\workshop\winsas\lwcrb\kids.dat,
      RECFM=V,LRECL=256

NOTE: Invalid data for hobby1 in line 1 25-31.
RULE:     ----+----1----+----2----+----3----+----4----+----5----
1         Chloe    2 11/10/1995 $5Running Music Gymnastics 48
name=Chloe siblings=2 bdate=13097 allowance=5 hobby1=.
hobby2=Music hobby3=Gymnasti _ERROR_=1 _N_=1
NOTE: Invalid data for hobby1 in line 2 25-32.
2         Travis   2 1/30/1998  $2Baseball Nintendo Reading 49
name=Travis siblings=2 bdate=13909 allowance=2 hobby1=.
hobby2=Nintendo hobby3=Reading _ERROR_=1 _N_=2
NOTE: Invalid data for hobby1 in line 3 25-30.
3         Jennifer 0 8/21/1999  $0Soccer Painting Dancing 47
name=Jennifer siblings=0 bdate=14477 allowance=0 hobby1=.
hobby2=Painting hobby3=Dancing _ERROR_=1 _N_=3
```

33

DATALINES Statement

The DATALINES statement can be used with an INPUT statement to read data directly from the program, rather than data stored in a raw data file.

```
data work.kids;
   input name $ 1-8 siblings 10
         @12 bdate mmddyy10.
         @23 allowance comma2.
         hobby1 $ hobby2 $ hobby3 $;
   datalines;
Chloe    2 11/10/1995 $5Running Music Gymnastics
Travis   2 1/30/1998  $2Baseball Nintendo Reading
Jennifer 0 8/21/1999  $0Soccer Painting Dancing
;
run;
```

35

DATALINES Statement

Q

Which statement is ***false*** concerning the DATALINES statement?

A. Multiple DATALINES statements can be used in a DATA step.

B. A null statement (a single semicolon) is needed to indicate the end of the input data.

C. The DATALINES statement is the last statement in the DATA step and immediately precedes the first data line.

36

Refer to Exercise 1 for Chapter 3 in Appendix A.

3.2 Reading Raw Data Files: Part 2

Column Input

With column input, the column numbers that contain the value follow a variable name in the INPUT statement.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with column input, data values

- must be in the same columns in all the input data records
- must be in standard form.

41

Column Input

- Standard data is any data that SAS can read without any special instructions.
- Examples of standard numeric data:

  `58   -23   67.23   00.99   5.67E5   1.2E-2`

A column INPUT statement can contain the following:

variable	Names a variable that is assigned input values.
$	Indicates to store a variable value as a character value rather than as a numeric value.
start-column	Specifies the first column of the input record that contains the value to read.
- end-column	Specifies the last column of the input record that contains the value to read.

42

Column Input

```
----|----10---|----20---|----30---|----40---|----50---|----
Chloe    2 11/10/1995 $5 Running    Music     Gymnastics
Travis   2 1/30/1998  $2 Baseball   Nintendo  Readin
Jennifer 0 8/21/1999  $0 Soccer     Painting  Dancing
```

Input Raw Data File

```
data work.kids2;
   infile 'kids2.dat';
   input name $ 1-8
         siblings 10
         bdate $ 12-21
         allowance $ 23-24
         hobby1 $ 26-35
         hobby2 $ 36-45
         hobby3 $ 46-55;
run;
```

43

Q

Column Input

```
input name $ 1-8
      siblings 10
      bdate $ 12-21
      allowance $ 23-24
      hobby1 $ 26-35
      hobby2 $ 36-45
      hobby3 $ 46-55;
```

How many variables are numeric and, ideally, how many variables should be numeric?

A. 1 numeric variable and 2 variables should be numeric.
B. 1 numeric variable and 3 variables should be numeric.
C. 2 numeric variables and 2 variables should be numeric.
D. 3 numeric variables and 3 variables should be numeric.

44

Formatted Input

With formatted input, an informat follows a variable name and defines how SAS reads the values of this variable. An informat gives the data type and the field width of an input value.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with formatted input, data values

- must be in the same columns in all the input data records
- can be in standard or nonstandard form.

46

Formatted Input

- *Nonstandard data* is any data that SAS cannot read without a special instruction.
- Examples of nonstandard numeric data:
  `5,823   (23)   $67.23   01/30/1999   12MAY06`

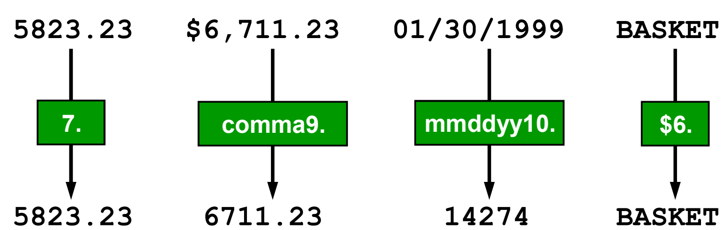
A formatted INPUT statement can contain the following:

pointer-control	Moves the input pointer to a specified column in the input buffer. @n moves the pointer to column n. +n moves the pointer n columns.
variable	Names a variable that is assigned input values.
informat	Specifies a SAS informat to use to read the variable values.

47

Informats

An *informat* is an instruction that SAS uses to read data values into a variable.

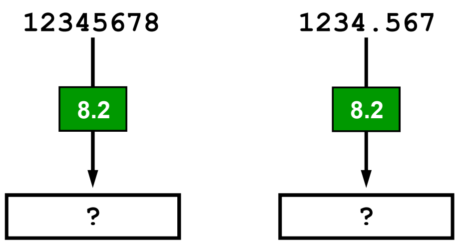


SAS uses the informat to determine the following:

- whether the variable is numeric or character
- the length of character variables

48

Informats



A.	123456.78	1234.567
B.	123456.78	1234.56
C.	123456.7	1234.567
D.	123456.7	1234.56

49

Informats

Which statement is ***false*** regarding informats?

A. When you use an informat, the informat contains a period (.) as a part of the name.

B. The $ indicates a character informat, and the absence of a $ indicates a numeric informat.

C. An informat has a default width or specifies a width, which is the number of columns to read in the input data.

D. When a problem occurs with a valid informat, SAS writes a note to the SAS log, assigns a missing value to the variable, and terminates the DATA step.

51

Informats

Raw Data Value	Informat	SAS Data Value
$12,345	COMMA7. DOLLAR7.	12345
$12.345	COMMAX7. DOLLARX7.	12345
€12.345	EUROX7.	12345
Australia	$11.	Australia
Australia	$CHAR11.	Australia
au	$UPCASE2.	AU
01/01/1960	MMDDYY10.	0
31/12/60	DDMMYY8.	365
31DEC1959	DATE9.	-1

53

Formatted Input

```
----|----10---|----20---|----30---|----40---|----50---|----
Chloe     2 11/10/1995 $5 Running   Music     Gymnastics
Travis    2 1/30/1998  $2 Baseball  Nintendo  Readin
Jennifer  0 8/21/1999  $0 Soccer    Painting  Dancing
```

Input Raw Data File

```
data work.kids2;
   infile 'kids2.dat';
   input @1 name $8.
         @10 siblings 1.
         @12 bdate mmddyy10.
         @23 allowance comma2.
         @26 hobby1 $10.
         @36 hobby2 $10.
         @46 hobby3 $10.;
run;
```

54

Q

Formatted Input

```
input @1 name $8.
      @10 siblings 1.
      @12 bdate mmddyy10.
      @23 allowance comma2.
      @26 hobby1 $10.
      @36 hobby2 $10.
      @46 hobby3 $10.;
```

What is the byte size of **bdate** and **hobby1**?

A. **bdate** = 8 and **hobby1** = 8
B. **bdate** = 8 and **hobby1** = 10
C. **bdate** = 10 and **hobby1** = 8
D. **bdate** = 10 and **hobby1** = 10

55

List Input

With list input, variable names in the INPUT statement are specified in the same order that the fields appear in the input data records.

```
input name $ 1-8 siblings 10
      @12 bdate mmddyy10.
      @23 allowance comma2.
      hobby1 $ hobby2 $ hobby3 $;
```

To read with list input, data values

- must be separated with a delimiter
- can be in standard or nonstandard form.

57

List Input

- You must specify the variables in the order that they appear in the raw data file, left to right.
- The default length for variables is 8 bytes.
- A space (blank) is the default delimiter.

pointer control	Moves the input pointer to a specified column in the input buffer.
variable	Names a variable that is assigned input values.
$	Indicates to store a variable value as a character value rather than as a numeric value.
:	Reads data values that need the additional instructions that informats can provide but are not aligned in columns.
informat	Specifies an informat to use to read the variable values.

58

List Input

```
Chloe 2 11/10/1995 $5 Running Music Gymnastics
Travis 2 1/30/1998 $2 Baseball Nintendo Reading
Jennifer 0 8/21/1999 $0 Soccer Painting Dancing
```

Input Raw Data File

```
data work.kids3;
   length hobby1 hobby2 hobby3 $ 10;
   infile 'kids3.dat';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 $
         hobby2 $
         hobby3 $;
run;
```

The LENGTH statement specifies the number of bytes for storing variables.

59 ...

List Input

```
Chloe 2 11/10/1995 $5 Running Music Gymnastics
Travis 2 1/30/1998 $2 Baseball Nintendo Reading
Jennifer 0 8/21/1999 $0 Soccer Painting Dancing
```

Input Raw Data File

```
data work.kids3;
   length hobby1 hobby2 hobby3 $ 10;
   infile 'kids3.dat';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 $
         hobby2 $
         hobby3 $;
run;
```

character

numeric

character

60

...

List Input

```
Chloe 2 11/10/1995 $5 Running Music Gymnastics
Travis 2 1/30/1998 $2 Baseball Nintendo Reading
Jennifer 0 8/21/1999 $0 Soccer Painting Dancing
```

Input Raw Data File

```
data work.kids3;
   length hobby1 hobby2 hobby3 $ 10;
   infile 'kids3.dat';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 $
         hobby2 $
         hobby3 $;
run;
```

The : modifier with an informat is used to read numeric values that contain nonstandard values.

61

List Input

```
Chloe 2 11/10/1995 $5 Running Music Gymnastics
Travis 2 1/30/1998 $2 Baseball Nintendo Reading
Jennifer 0 8/21/1999 $0 Soccer Painting Dancing
```

Input Raw Data File

```
data work.kids3;
   infile 'kids3.dat';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

The : modifier with an informat can also be used to control the byte size of character variables.

62

: Modifier with an Informat

A : modifier with an informat enables SAS to do the following:

- treat the current field as a delimited field
- apply an informat to the field, ignoring the width

```
input name $ salary:comma10. state $;
```

list input

```
input name $ salary comma10. state $;
```

formatted input

63

: Modifier with an Informat

```
----|----10---|----20
Ted $2,345 Georgia
Sam $222,345 Florida
```

Input Raw Data File

```
input name $ salary:comma10. state $;
```

VIEWTABLE: Work.Example

	name	salary	state
1	Ted	2345	Georgia
2	Sam	222345	Florida

```
input name $ salary comma10. state $;
```

VIEWTABLE: Work.Example

	name	salary	state
1	Ted	.	rgia
2	Sam	.	lorida

64

: Modifier with an Informat

Q

```
----|----10---|----20
Ted 1DEC07 Georgia
Sam 21DEC2007 Florida
```

Input Raw Data File

```
input name $ bdate date9. state $;
```

What is the result of **state** based on the above INPUT statement?

A.
```
Georgia
Florida
```

B.
```
orgia
Florida
```

C.
```
rgia
Florida
```

65

Q

: Modifier with an Informat

```
----|----10---|----20
Ted 1DEC07 Georgia
Sam 21DEC2007 Florida
```

Input Raw Data File

```
input name $ bdate:date9. state $;
```

What is the result of **state** based on the above INPUT statement?

A.
```
Georgia
Florida
```

B.
```
orgia
Florida
```

C.
```
rgia
Florida
```

67

DLM= Option

The DLM= option specifies a delimiter to be used for list input. Blank is the default delimiter.

```
Chloe,2,11/10/1995,$5,Running,Music,Gymnastics
Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading
Jennifer,0,8/21/1999,$0,Soccer,Painting,Dancing
```

Input Raw Data File

```
data work.kids4;
   infile 'kids4.dat' dlm=',';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

69

Missing Data

By default, SAS treats two consecutive delimiters as one, not as a missing value between the delimiters.

```
Chloe,,11/10/1995,,Running,Music,Gymnastics
Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading
Jennifer,0,8/21/1999,$0,Soccer,Painting,Dancing
```

Input Raw Data File

```
data work.kids5;
   infile 'kids5.dat' dlm=',';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

70

Missing Data

```
Chloe,,11/10/1995,,Running,Music,Gymnastics
Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading
Jennifer,0,8/21/1999,$0,Soccer,Painting,Dancing
```

Input Raw Data File

VIEWTABLE: Work.Kids5

Output Data Set

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	.	.	.	Gymnastics	Travis	2
2	Jennifer	0	14477	0	Soccer	Painting	Dancing

```
NOTE: Invalid data for siblings in line 1 8-17.
NOTE: Invalid data for bdate in line 1 20-26.
NOTE: Invalid data for allowance in line 1 28-32.
RULE:     ----+----1----+----2----+----3----+----4----+----5----
2         Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading 47
NOTE: Invalid data errors for file ''kids5.dat'' occurred
      outside the printed range.
NOTE: Increase available buffer lines with the INFILE n= option.
name=Chloe siblings=. bdate=. allowance=. hobby1=Gymnastics
hobby2=Travis hobby3=2 _ERROR_=1 _N_=1
NOTE: 3 records were read from the infile 'kids5.dat'.
      The minimum record length was 43.
      The maximum record length was 47.
NOTE: SAS went to a new line when INPUT statement reached past
      the end of a line.
```

71

DSD Option

The DSD option can do the following:

- treat two consecutive delimiters as a missing value
- remove quotation marks from strings and treat any delimiter inside the quotation marks as a valid character
- set the default delimiter to a comma

```
data work.kids5;
   infile 'kids5.dat' dsd;
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

72

DSD Option

```
Chloe,,11/10/1995,,Running,Music,Gymnastics
Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading
Jennifer,0,8/21/1999,$0,Soccer,Painting,Dancing
```

Input Raw Data File

Output Data Set

VIEWTABLE: Work.Kids5

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	.	13097	.	Running	Music	Gymnastics
2	Travis	2	13909	2	Baseball	Nintendo	Reading
3	Jennifer	0	14477	0	Soccer	Painting	Dancing

```
NOTE: 3 records were read from the infile 'kids5.dat'.
      The minimum record length was 43.
      The maximum record length was 47.
NOTE: The data set WORK.KIDS5 has 3 observations and 7
      variables.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.02 seconds
```

73

Q

DSD Option

Input Raw Data File

```
Chloe/2/"11/10/1995"/$5/Running/Music/Gymnastics
Travis/2/"1/30/1998"/$2/Baseball/Nintendo/Reading
Jennifer/0/"8/21/1999"//Soccer/Painting/Dancing
```

quoted values | missing data | / delimiter

Which statement will correctly read the raw data file?

A. `infile 'kids5a.dat' dsd;`

B. `infile 'kids5a.dat' dlm='/';`

C. `infile 'kids5a.dat' dsd dlm='/';`

D. `infile 'kids5a.dat' dsd, dlm='/';`

74

Varying Number of Fields per Record

A raw data file might have a varying number of fields per record.

Input Raw Data File

```
Chloe 2 11/10/1995 $5 Running Music Gymnastics
Travis 2 1/30/1998 $2 Baseball Nintendo
Jennifer 0 8/21/1999 $0 Soccer
```

Output Data Set

VIEWTABLE: Work.Kids6

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	2	13097	5	Running	Music	Gymnastics
2	Travis	2	13909	2	Baseball	Nintendo	Jennifer

```
NOTE: 3 records were read from the infile 'kids6.dat'.
      The minimum record length was 30.
      The maximum record length was 46.
NOTE: SAS went to a new line when INPUT statement reached past
      the end of a line.
NOTE: The data set WORK.KIDS6 has 2 observations and 7
      variables.
```

76

MISSOVER Option

The MISSOVER option prevents an INPUT statement from reading a new input data record if it does not find values in the current input line for all the variables in the statement.

```
data work.kids6;
   infile 'kids6.dat' missover;
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

77

MISSOVER Option

When an INPUT statement reaches the end of the current input data record, variables without any values assigned are set to missing with the MISSOVER option.

Output Data Set

VIEWTABLE: Work.Kids6

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	2	13097	5	Running	Music	Gymnastics
2	Travis	2	13909	2	Baseball	Nintendo	
3	Jennifer	0	14477	0	Soccer		

78

Refer to Exercise 2 for Chapter 3 in Appendix A.

3.3 Controlling When a Record Loads

Multiple INPUT Statements

By default, SAS advances the pointer to column 1 of the next input record when SAS encounters an INPUT statement.

Input Raw Data File

```
Chloe    2
11/10/1995 $5
Running Music Gymnastics
Travis   2
1/30/1998  $2
Baseball Nintendo Reading
Jennifer 0
8/21/1999  $0
Soccer Painting Dancing
```

```
data work.kids8;
   infile 'kids8.dat';
   input name $ 1-8
         siblings 10;
   input @1 bdate mmddyy10.
         @12 allowance comma2.;
   input hobby1:$10.
         hobby2:$10.
         hobby3:$10.;
run;
```

82

Multiple INPUT Statements

```
data work.kids8;
   infile 'kids8.dat';
   input name $ 1-8
         siblings 10;
   input @1 bdate mmddyy10.
         @12 allowance comma2.;
   input hobby1:$10.
         hobby2:$10.
         hobby3:$10.;
run;
```

Input Raw Data File

```
Chloe    2
11/10/1995 $5
Running Music Gymnastics
Travis   2
1/30/1998  $2
Baseball Nintendo Reading
Jennifer 0
8/21/1999  $0
Soccer Painting Dancing
```

Which statement is ***false***?

A. The DATA step iterates nine times.

B. The data set **work.kids8** will have three observations.

C. Three records are read with each DATA step iteration.

D. One observation is created with each DATA step iteration.

83

Multiple INPUT Statements

```
data work.kids8;
   infile 'kids8.dat';
   input name $ 1-8
         siblings 10;
   input @1 bdate mmddyy10.
         @12 allowance comma2.;
   input hobby1:$10.
         hobby2:$10.
         hobby3:$10.;
run;
```

Input Raw Data File

```
Chloe    2
11/10/1995 $5
Running Music Gymnastics
Travis   2
1/30/1998  $2
Baseball Nintendo Reading
Jennifer 0
8/21/1999  $0
Soccer Painting Dancing
```

Output Data Set

VIEWTABLE: Work.Kids8

	name	siblings	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	2	13097	5	Running	Music	Gymnastics
2	Travis	2	13909	2	Baseball	Nintendo	Reading
3	Jennifer	0	14477	0	Soccer	Painting	Dancing

85

Line-Pointer Controls

The / line-pointer control advances the pointer to column 1 of the next input record.

```
data work.kids8;
   infile 'kids8.dat';
   input name $ 1-8
         siblings 10
       / @1 bdate mmddyy10.
         @12 allowance comma2.
       / hobby1:$10.
         hobby2:$10.
         hobby3:$10.;
run;
```

86

Line-Pointer Controls

The #*n* line-pointer control advances the pointer to column 1 of record *n*.

```
data work.kids8;
   infile 'kids8.dat';
   input #1 name $ 1-8
            siblings 10
         #2 @1 bdate mmddyy10.
            @12 allowance comma2.
         #3 hobby1:$10.
            hobby2:$10.
            hobby3:$10.;
run;
```

87

Line-Pointer Controls

Q

Input Raw Data File

```
Chloe    2
11/10/1995 $5
Running Music Gymnastics
Travis   2
1/30/1998  $2
Baseball Nintendo Reading
Jennifer 0
8/21/1999  $0
Soccer Painting Dancing
```

Yes or No: Do the following three sections produce the same results?

```
input name $ 1-8 siblings 10 //
      hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input #1 name $ 1-8 siblings 10
      #3 hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input name $ 1-8 siblings 10;
input;
input hobby1:$10. hobby2:$10. hobby3:$10.;
```

88

Line-Pointer Controls

The second record is skipped in each iteration of the DATA step.

```
input name $ 1-8 siblings 10 //
      hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input #1 name $ 1-8 siblings 10
      #3 hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input name $ 1-8 siblings 10;
input;
input hobby1:$10. hobby2:$10. hobby3:$10.;
```

Output Data Set

VIEWTABLE: Work.Kids8

	name	siblings	hobby1	hobby2	hobby3
1	Chloe	2	Running	Music	Gymnastics
2	Travis	2	Baseball	Nintendo	Reading
3	Jennifer	0	Soccer	Painting	Dancing

90

Single Trailing @

The single trailing @ has the following characteristics:

- holds an input record for the execution of the next INPUT statement within the same iteration of the DATA step
- is useful when you must read from a record multiple times

Input Raw Data File

```
Chloe    IN 11/10/1995 $5Running Music Gymnastics
Travis   IL Baseball Nintendo Reading
Jennifer IN 8/21/1999  $0Soccer Painting Dancing
```

This raw data file has different layouts depending on the value of state.

91

Single Trailing @

Input Raw Data File

```
Chloe    IN 11/10/1995 $5Running Music Gymnastics
Travis   IL Baseball Nintendo Reading
Jennifer IN 8/21/1999  $0Soccer Painting Dancing
```

```
data work.kids9;
   infile 'kids9.dat';
   input name $ 1-8 state $ 10-11 @;
   if state='IN' then
      input @13 bdate mmddyy10. @24 allowance comma2.
            hobby1:$10. hobby2:$10. hobby3:$10.;
   else input @13 hobby1:$10. hobby2:$10. hobby3:$10.;
run;
```

Output Data Set

VIEWTABLE: Work.Kids9

	name	state	bdate	allowance	hobby1	hobby2	hobby3
1	Chloe	IN	13097	5	Running	Music	Gymnastics
2	Travis	IL	.	.	Baseball	Nintendo	Reading
3	Jennifer	IN	14477	0	Soccer	Painting	Dancing

92

Single Trailing @

Q

Which statement is ***false*** concerning the single trailing @?

A. In the INPUT statement, the single trailing @ must be the last item before the semicolon.

B. SAS releases a record held by a single trailing @ when an INPUT statement without a trailing @ executes.

C. The single trailing @ prevents the next INPUT statement from automatically releasing the current input record.

D. The single trailing @ holds the input record for the execution of the next INPUT statement across iterations of the DATA step.

93

Double Trailing @

The double trailing @ has the following characteristics:

- holds the input record for the execution of the next INPUT statement across iterations of the DATA step
- is useful when each input line contains values for several observations

Input Raw Data File

```
Chloe IN Travis IL Jennifer IN
Brian IL Mark IN Kurt IN Hannah IL
```

95

Double Trailing @

Input Raw Data File

```
Chloe IN Travis IL Jennifer IN
Brian IL Mark IN Kurt IN Hannah IL
```

```
data work.kids10;
   infile 'kids10.dat';
   input name $ state $ @@;
run;
```

Output Data Set

VIEWTABLE: Work.Kids10

	name	state
1	Chloe	IN
2	Travis	IL
3	Jennifer	IN
4	Brian	IL
5	Mark	IN
6	Kurt	IN
7	Hannah	IL

```
NOTE: 2 records were read from the infile 'kids10.dat'.
      The minimum record length was 30.
      The maximum record length was 34.
NOTE: SAS went to a new line when INPUT statement reached past the
      end of a line.
NOTE: The data set WORK.KIDS10 has 7 observations and 2 variables.
```

96

Q

Double Trailing @

Which statement is ***false*** concerning the double trailing @?

A. The double trailing @ must be the first item in the INPUT statement.

B. The double trailing @ is useful when one record contains several observations.

C. The double trailing @ holds a record for the next INPUT statement across iterations of the DATA step.

D. SAS releases the record that is held by a double trailing @ if the pointer moves past the end of the input record.

97

Refer to Exercise 3 for Chapter 3 in Appendix A.

3.4 Reading Microsoft Excel Files

SAS/ACCESS LIBNAME Statement

With the SAS/ACCESS Interface to PC File Formats, the LIBNAME statement can be used to access Microsoft Excel workbooks.

```
libname myxls 'customers.xls';

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

This enables you to reference a worksheet directly in a DATA step or SAS procedure, and to read from and write to an Excel worksheet as if it were a SAS data set.

102

SAS/ACCESS LIBNAME Statement

How do you reference the **females** worksheet in the Excel workbook **customer** based on the following LIBNAME statement?

```
libname myxls 'customers.xls';
```

A. `myxls.females$`

B. `customer.females`

C. `myxls.'females$'n`

D. `customer.'females'n`

103

SAS/ACCESS LIBNAME Statement

Worksheet names appear with a dollar sign at the end of the name.

```
                The CONTENTS Procedure

                       Directory

           Libref          MYXLS
           Engine          EXCEL
           Physical Name   customers.xls
           User            Admin


                                  DBMS
                         Member   Member
           #  Name       Type     Type

           1  Females$   DATA     TABLE
           2  Males$     DATA     TABLE
```

105

SAS/ACCESS LIBNAME Statement

SAS name literals enable special characters to be included in data set names.

```
                          The CONTENTS Procedure

Data Set Name         MYXLS.'Females$'n    Observations          .
Member Type           DATA                 Variables             6
Engine                EXCEL                Indexes               0
Created               .                    Observation Length    0
Last Modified         .                    Deleted Observations  0
Protection                                 Compressed            NO
Data Set Type                              Sorted                NO
Label
Data Representation   Default
Encoding              Default
```

A *SAS name literal* is a name token that is expressed as a string within quotation marks, followed immediately by the letter n.

106

SAS/ACCESS LIBNAME Statement

If SAS has a library reference name assigned to an Excel workbook, the workbook cannot be opened in Excel.

```
libname myxls 'customers.xls';

proc contents data=myxls._all_;
run;

proc print data=myxls.'females$'n;
run;

data work.usfemales;
   set myxls.'females$'n;
   where country='US';
run;
```

107

SAS/ACCESS LIBNAME Statement

Q

Which statement disassociates the **MYXLS** library reference name?

A.
```
libname myxls end;
```

B.
```
libname myxls clear;
```

C.
```
libname myxls close;
```

D.
```
libname myxls disassociate;
```

108

SAS/ACCESS LIBNAME Statement

The default Excel engine can be specified when the bit count of SAS and Microsoft Office are the same (both are 32-bit or both are 64-bit).

```
libname myxls excel 'customers.xls';
```

The PC Files Server engine must be specified when the bit counts differ.

```
libname myxls pcfiles path='customers.xls';
```

110

Refer to Exercise 4 for Chapter 3 in Appendix A.

3.5 Answers to Questions

Question Slide Number	Answer
4	1. **kids.dat** 2. **work.kids**
9	C.
15	A.
19	D.
27	C.
33	The variable **hobby1** is being read in as numeric.
36	A.
44	B.
49	A.
51	D.
55	B.
65	B.
67	A.
74	C.
83	A.
88	Yes
93	D.
97	A.
103	C.
108	B.

Chapter 4 Creating Variables

4.1 Creating Variables with the Assignment Statement 4-3

4.2 Creating Variables Conditionally 4-8

4.3 Creating Accumulator Variables 4-19

4.4 Answers to Questions 4-25

4.1 Creating Variables with the Assignment Statement

Assignment Statement

The assignment statement evaluates an expression and stores the result in a new variable or an existing variable.

Examples:

```
   name = 'Jane Doe';
revenue = 157900;
   date = '10MAY2007'd;
  total = price * quantity;
 cityst = city !! state;
product = upcase(product);
average = mean(jan, feb, mar);
```

variable expression

Assignment statements evaluate the expression on the right side of the equal sign and store the result in the variable that is specified on the left side of the equal sign.

3

Assignment Statement

Q

Which assignment statement is overwriting a variable used in the expression?

1. `name = 'Jane Doe';`
2. `revenue = 157900;`
3. `date = '10MAY2007'd;`
4. `total = price * quantity;`
5. `cityst = city !! state;`
6. `product = upcase(product);`
7. `average = mean(jan, feb, mar);`

4

Expression

The *expression* is a sequence of operands and operators that form a set of instructions that produce a value.

- *Operands* are
  - constants (character or numeric)
  - variables (character or numeric).
- *Operators* are
  - symbols that represent an arithmetic calculation or concatenation
  - a SAS function.

6

Operands

A *constant* is a number or a character string that indicates a fixed value.

```
   name = 'Jane Doe';
revenue = 157900;
   date = '10MAY2007'd;
```

Character constants must be enclosed in quotation marks.

Character constants are enclosed in quotation marks, but names of variables are not enclosed in quotation marks.

```
  total = price * quantity;
 cityst = city !! state;
product = upcase(product);
average = mean(jan, feb, mar);
```

7

Operators

- Arithmetic operators indicate that an arithmetic calculation is performed.
- A concatenation operator concatenates character values.
- A SAS function performs a computation or system manipulation on arguments and returns a value.

```
    total = price * quantity;
   cityst = city !! state;
  product = upcase(product);
  average = mean(jan, feb, mar);
```

8

Arithmetic Operators

Possible arithmetic operators:

Symbol	Definition
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction

- If a missing value is an operand for an arithmetic operator, the result is a missing value.

9

Example

```
data newprice;
   set golf.supplies;
   saleprice = price * 0.75;
   saletype = '25% off';
   format price saleprice dollar8.2;
run;
```

VIEWTABLE: Golf.Supplies

	mfg	type	price
1	Crew	Distance	8.1
2	Crew	Spin	8.25
3	Crew	Titanium	9.5
4	Hi-fly	X12000	13.75
5	Hi-fly	X22000	14.6
6	White	Strata	10.6
7	White	Aero	12.3
8	White	XL	14.5
9	White	Flite	16.2

VIEWTABLE: Work.Newprice

	mfg	type	price	saleprice	saletype
1	Crew	Distance	$8.10	$6.08	25% off
2	Crew	Spin	$8.25	$6.19	25% off
3	Crew	Titanium	$9.50	$7.13	25% off
4	Hi-fly	X12000	$13.75	$10.31	25% off
5	Hi-fly	X22000	$14.60	$10.95	25% off
6	White	Strata	$10.60	$7.95	25% off
7	White	Aero	$12.30	$9.23	25% off
8	White	XL	$14.50	$10.88	25% off
9	White	Flite	$16.20	$12.15	25% off

10

Example

```
data newprice;
   set golf.supplies;
   saleprice = price * 0.75;
   saletype = '25% off';
   format price saleprice dollar8.2;
run;
```

Which statement is ***true*** concerning the new variables?

A. **saleprice** and **saletype** are numeric (8 bytes).

B. **saleprice** and **saletype** are character (7 bytes).

C. **saleprice** is numeric (8 bytes) and **saletype** is character (7 bytes).

D. **saleprice** is numeric (8 bytes) and **saletype** is character (8 bytes).

11

Refer to Exercise 1 for Chapter 4 in Appendix A.

DROP and KEEP Statements

- The DROP statement specifies the names of the variables to omit from the output data set.
- The KEEP statement specifies the names of the variables to write to the output data set.

```
data newprice;
   set golf.supplies;
   drop type price;
   saleprice = price * 0.75;
   saletype = '25% off';
   format saleprice dollar8.2;
run;
```

Placement of statement is irrelevant; statement is applied at output time.

Partial Data Set

VIEWTABLE: Work.Newprice

	mfg	type	price	saleprice	saletype
1	Crew	Distance	$8.10	$6.08	25% off
2	Crew	Spin	$8.25	$6.19	25% off
3	Crew	Titanium	$9.50	$7.13	25% off

16

DROP= and KEEP= Options

Which program is equivalent to the previous program?

A.

```
data newprice(drop=type price);
   set golf.supplies;
   saleprice = price * 0.75;
   saletype = '25% off';
   format saleprice dollar8.2;
run;
```

B.

```
data newprice;
   set golf.supplies(drop=type price);
   saleprice = price * 0.75;
   saletype = '25% off';
   format saleprice dollar8.2;
run;
```

17

DROP= and KEEP= Options

The DROP= data set option excludes the variables for

- processing if in the SET statement
- writing to the output data set if in the DATA statement.

The KEEP= data set option specifies the variables for

- processing if in the SET statement
- writing to the output data set if in the DATA statement.

```
data newprice(drop=price);
   set golf.supplies(keep=mfg price);
   saleprice = price * 0.75;
   saletype = '25% off';
   format saleprice dollar8.2;
run;
```

19

4.2 Creating Variables Conditionally

IF-THEN / ELSE Statements

- The IF-THEN statement executes ***a statement*** for observations that meet specific conditions.
- The optional ELSE statement gives an alternative action if the THEN clause is not executed.

```
data newprice;
   set golf.supplies;
   if mfg='Crew' then saleprice=price*0.75;
   else if mfg='Hi-fly' then saleprice=price*0.70;
   else if mfg='White' then saleprice=price*0.90;
   format price saleprice dollar8.2;
run;
```

22

Expression

```
if mfg='Crew' then saleprice=price*0.75;
```

expression

- The expression is any valid SAS expression and is a required argument.
- SAS evaluates the expression in an IF-THEN statement to produce a result that is either nonzero, zero, or missing.
- A nonzero and nonmissing result causes the expression to be true; a result of zero or missing causes the expression to be false.

23

Expression

Q

Examples:

```
city in ('New York','Boston')
```

```
salary ge 125000
```

```
upcase(last)='SMITH'
```

```
birth='12DEC1999'd
```

```
city='Atlanta' and salary<50000
```

```
revenue ne goal
```

Which expression is invalid in the IF-THEN statement?

A. `if total then ...`

B. `if 12000<=revenue<=25000 then ...`

C. `if city='Reno' state='NV' then ...`

D. `if salary>75000 or bonus<5000 then ...`

24

Statement

```
if mfg='Crew' then saleprice=price*0.75;
```

statement

The statement can be any executable SAS statement.

Examples:

```
status = 'Unknown'
```

```
count + 1
```

```
total = sum(num1, num2, num3)
```

```
delete
```

```
anniversary = '15AUG2006'd
```

```
output
```

26

ELSE Statements

Q

```
if mfg='Crew' then saleprice=price*0.75;
else if mfg='Hi-fly' then saleprice=price*0.70;
else if mfg='White' then saleprice=price*0.90;
```

Yes or No: Is the word ELSE required each time in the above example?

27

ELSE Statements

```
if mfg='Crew' then saleprice=price*0.75;
else if mfg='Hi-fly' then saleprice=price*0.70;
else if mfg='White' then saleprice=price*0.90;
```

- Using IF-THEN statements ***without*** the ELSE statement causes SAS to evaluate all IF-THEN statements.
- Using IF-THEN statements ***with*** the ELSE statement causes SAS to execute the IF-THEN statements until SAS encounters the first true statement. Subsequent IF-THEN statements are not evaluated.

29

ELSE Statements

```
if mfg='Crew' then saleprice=price*0.75;
else if mfg='Hi-fly' then saleprice=price*0.70;
else if mfg='White' then saleprice=price*0.90;
```

Based on the partial program above, what is the result if an observation has a value of **mfg='X-treme'**?

A. The observation will be deleted.
B. The observation will have a missing value for **saleprice**.
C. The observation will have a **saleprice** value equal to **price** multiplied by 0.75.
D. The observation will have a **saleprice** value equal to **price** multiplied by 0.90.

30

ELSE Statements

```
if mfg='Crew' then saleprice=price*0.75;
else if mfg='Hi-fly' then saleprice=price*0.70;
else saleprice=price*0.90;
```

The final ELSE statement can be coded without an IF-THEN statement to direct all previous false conditions into the final condition.

32

IF-THEN DO / ELSE DO Statements

The IF-THEN DO statement executes ***a group of statements*** for observations that meet specific conditions.

```
...
  if mfg='Crew' then do;
     pct=0.75;
     saleprice = price * pct;
     saletype = '25% off';
  end;
  else if mfg='Hi-fly' then do;
     pct=0.70;
     saleprice = price * pct;
     saletype = '30% off';
  end;
  else do;
     pct=0.90;
     saleprice = price * pct;
     saletype = '10% Storewide Sale';
  end;
...
```

33

Refer to Exercise 2 for Chapter 4 in Appendix A.

IF-THEN DO / ELSE DO Statements

```
data newprice;
  infile 'raw-data-file';
  input mfg $ type $ price;
  length saletype $ 18;
  if mfg='Crew' then do;
     pct=0.75;
     saleprice = price * pct;
     saletype = '25% off';
  end;
  else if mfg='Hi-fly' then do;
     pct=0.70;
     saleprice = price * pct;
     saletype = '30% off';
  end;
  else do;
     pct=0.90;
     saleprice = price * pct;
     saletype = '10% Storewide Sale';
  end;
  format price saleprice dollar8.2;
run;
```

Conditional statements can be used in DATA steps that read raw data files or data sets.

1

2

3

36

IF-THEN DO / ELSE DO Statements

```
data newprice;
  infile 'raw-data-file';
  input mfg $ type $ price;
  length saletype $ 18;
  if mfg='Crew' then do;
     pct=0.75;
     saleprice = price * pct;
     saletype = '25% off';
  end;
  else if mfg='Hi-fly' then do;
     pct=0.70;
     saleprice = price * pct;
     saletype = '30% off';
  end;
  else do;
     pct=0.90;
     saleprice = price * pct;
     saletype = '10% Storewide Sale';
  end;
  format price saleprice dollar8.2;
run;
```

A LENGTH statement controls the byte size of saletype. Without the statement, byte size would be 7 (25% off).

Three variables are being created. pct and saleprice are numeric (8 bytes). saletype is character (18 bytes).

37

IF-THEN DO / ELSE DO Statements

```
data newprice;
  infile 'raw-data-file';
  input mfg $ type $ price;
  length saletype $ 18;
  if mfg='Crew' then do;
     pct=0.75;
     saleprice = price * pct;
     saletype = '25% off';
  end;
  else if mfg='Hi-fly' then do;
     pct=0.70;
     saleprice = price * pct;
     saletype = '30% off';
  end;
  else do;
     pct=0.90;
     saleprice = price * pct;
     saletype = '10% Storewide Sa
  end;
  format price saleprice dollar8.2;
run;
```

All previous false conditions fall into the final condition.

38

Q

IF-THEN DO / ELSE DO Statements

Which syntax is valid for IF-THEN DO statements?

A.
```
if payclass='monthly' then do amt=salary;
else if payclass='hourly' then do amt=hrlywage*40;
```

B.
```
if 12000<=revenue<=24000 then do; x='goal' end;
else if revenue<12000 then do; x='below' end;
else if revenue>24000 then do; x='above' end;
```

C.
```
if salary>=100000 then do;
   category='Exec'; range='Above 100K'; end;
else if salary<100000 then do;
   category='Non-Exec'; range='Below 100K'; end;
```

D.
```
if mon in ('JUN', 'JUL', 'AUG') then do;
     status='SUMMER';
else if mon in ('MAR', 'APR', 'MAY') then do;
     status='SPRING';
else do;
     status='FALL OR WINTER';
```

39

SELECT / WHEN / OTHERWISE Statements

An alternative to IF-THEN statements is SELECT / WHEN / OTHERWISE statements.

- The SELECT statement begins a SELECT group.
- SELECT groups contain WHEN statements that identify SAS statements that are executed when a particular condition is true.
- A SELECT group must use at least one WHEN statement.
- An optional OTHERWISE statement specifies a statement to be executed if no WHEN condition is met.
- An END statement ends a SELECT group.

41

SELECT / WHEN / OTHERWISE Statements

IF THEN ELSE

```
data newprice;
   set golf.supplies;
   if mfg='Crew' then
      saleprice=price*0.75;
   else if mfg='Hi-fly' then
           saleprice=price*0.70;
   else if mfg='White' then
           saleprice=price*0.90;
   format price saleprice dollar8.2;
run;
```

SELECT group

```
data newprice;
   set golf.supplies;
   select(mfg);
      when('Crew') saleprice=price*0.75;
      when('Hi-fly') saleprice=price*0.70;
      when('White') saleprice=price*0.90;
   end;
   format price saleprice dollar8.2;
run;
```

42

SELECT / WHEN / OTHERWISE Statements

Q

```
select(mfg);
   when('Crew') saleprice=price*0.75;
   when('Hi-fly') saleprice=price*0.70;
   when('White') saleprice=price*0.90;
end;
```

Based on the partial program above, what is the result if an observation has a value of **mfg='X-treme'**?

43

SELECT / WHEN / OTHERWISE Statements

A null OTHERWISE statement prevents SAS from issuing an error message when all WHEN conditions are false.

```
ERROR: Unsatisfied WHEN clause and no
OTHERWISE clause at line 618 column 3.
mfg=X-treme type=Strata price=$10.60
saleprice=. _ERROR_=1 _N_=10
```

```
select(mfg);
   when('Crew') saleprice=price*0.75;
   when('Hi-fly') saleprice=price*0.70;
   when('White') saleprice=price*0.90;
end;
```

```
select(mfg);
   when('Crew') saleprice=price*0.75;
   when('Hi-fly') saleprice=price*0.70;
   when('White') saleprice=price*0.90;
   otherwise;
end;
```

no ERROR

45

```
...
  if mfg='Crew' then do;
     pct=0.75;
     saleprice = price * pct;
     saletype = '25% off';
  end;
  else if mfg='Hi-f
     pct=0.70;
     saleprice = pr
     saletype = '30
  end;
  else do;
     pct=0.90;
     saleprice = pr
     saletype = '10
  end;
...
```

```
...
  select(mfg);
     when('Crew') do;
        pct=0.75;
        saleprice = price * pct;
        saletype = '25% off';
     end;
     when('Hi-fly') do;
        pct=0.70;
        saleprice = price * pct;
        saletype = '30% off';
     end;
     otherwise do;
        pct=0.90;
        saleprice = price * pct;
        saletype='10% Storewide Sale';
     end;
  end;
...
```

SELECT group

46

```
data newprice;
   infile 'raw-data-file';
   input mfg $ type $ price;
   length saletype $ 18;
   select(mfg);
      when('Crew') do;
         pct=0.75;
         saleprice = price * pct;
         saletype = '25% off';
      end;
      when('Hi-fly') do;
         pct=0.70;
         saleprice = price * pct;
         saletype = '30% off';
      end;
      otherwise do;
         pct=0.90;
         saleprice = pri
         saletype = '10%
      end;
   end;
   format price saleprice dollar8.2;
run;
```

SELECT group

1

2

3

All previous false conditions fall into the final condition.

47

SELECT / WHEN / OTHERWISE Statements

Additional examples:

```
select(payclass);
   when('monthly') amt=salary;
   when('hourly') amt=hrlywage*40;
   otherwise;
end;
```

```
select;
   when(12000<=revenue<=24000) target='goal';
   when(revenue<12000) target='below';
   when(revenue>24000) target='above';
   otherwise;
end;
```

```
select;
   when(mon in ('JUN', 'JUL', 'AUG') and temp>70)
        status='SUMMER';
   when(mon in ('MAR', 'APR', 'MAY'))
        status='SPRING';
   otherwise status='FALL OR WINTER';
end;
```

48

SELECT / WHEN / OTHERWISE Statements

Q

```
data new;
   set rev;
   length target $ 5;
   select;
     when (12000<=revenue<=24000) target='goal';
     when (revenue<12000) target='below';
     when (revenue>=24000) target='above';
     otherwise;
   end;
run;
```

What is the value of **target** when **revenue=24000**?

A. missing

B. goal

C. below

D. above

49

4.3 Creating Accumulator Variables

Accumulator Variables

An *accumulator variable* is a variable that adds on an expression.

Partial Output

Maine County	Population 2000	Total Population	Total Counties
Androscoggin	103,793	103,793	1
Aroostook	73,938	177,731	2
Cumberland	265,612	443,343	3
Franklin	29,467	472,810	4
Hancock	51,791	524,601	5
Kennebec	117,114	641,715	6
Knox	39,618	681,333	7
Lincoln	33,616	714,949	8
Oxford	54,755	769,704	9
Penobscot	144,919	914,623	10

accumulator variables

52

Sum Statement

The *sum statement* adds the result of an expression to an accumulator variable.

Examples:

```
TotCounties    + 1;
```

```
TotPopulation + Population2000;
```

accumulator variable **+** expression

53

Sum Statement

The accumulator variable has the following characteristics:

- must be a numeric variable
- is automatically set to 0 before SAS reads the first observation
- is retained from one iteration to the next

The expression is defined with the following features:

- is any SAS expression
- is evaluated and the result added to the accumulator variable
- is ignored if missing

54

Sum Statement

The sum statement is equivalent to using the RETAIN statement and the SUM function.

```
TotPopulation + Population2000;
```

```
retain TotPopulation 0;
TotPopulation =
   sum(TotPopulation,Population2000);
```

- The RETAIN statement causes a variable to retain its value from one iteration of the DATA step to the next and specifies an initial value for the variable.
- The SUM function returns the sum of the nonmissing arguments.

55

Sum and RETAIN Statements

Q

Yes or No: Is the RETAIN statement ever needed with the sum statement?

56

RETAIN Statement

To initialize an accumulator variable to a value other than zero, include the accumulator variable in a RETAIN statement with an initial value.

	Population	Year
1	914950	1950
2	940841	1955
3	970689	1960
4	993236	1965
5	997357	1970
6	1062640	1975
7	1125027	1980
8	1163850	1985
9	1227928	1990

```
data FiveYearPop;
   set FiveYearPop;
   retain year 1945;
   year+5;
run;
```

58

Accumulator Variable for BY Groups

In order to create an accumulator variable for BY groups, the beginning and end of each BY group must be determined.

County	Town	Population	Total Population
Androscoggin	Auburn	23203	23203
Androscoggin	Lewiston	35690	58893
Cumberland	Brunswick	21172	21172
Cumberland	Portland	64249	85421
Cumberland	Scarborough	16970	102391
Cumberland	South Portland	23324	125715
Kennebec	Augusta	18560	18560
Penobscot	Bangor	31473	31473
York	Biddeford	20942	20942
York	Sanford	20806	41748

accumulator variable by County

59

BY-Group Processing

In the DATA step, SAS identifies the beginning and end of each BY group by creating two temporary variables for each BY variable: the **FIRST.** and **LAST.** variables.

```
data TopCounties;
   set Top10Town;
   by County;
run;
```

Data must be sorted or indexed by County.

First.County and Last.County are created.

These temporary variables are available for DATA step programming but are not added to the output data set.

60

BY-Group Processing

- The **FIRST.** variable is set to 1 when an observation is the first in a BY group. Otherwise, it equals 0.
- The **LAST.** variable is set to 1 when an observation is the last in a BY group. Otherwise, it equals 0.

County	Population	first.County	last.County
Androscoggin	23203	1	0
Androscoggin	35690	0	1
Cumberland	21172	1	0
Cumberland	64249	0	0
Cumberland	16970	0	0
Cumberland	23324	0	1
Kennebec	18560	1	1

61

BY-Group Processing

Which of the following is ***false***?

A. **First.County**=1 and **Last.County**=0 means the observation is the first one in the BY group.

B. **First.County**=0 and **Last.County**=1 means the observation is the last one in the BY group.

C. **First.County**=0 and **Last.County**=0 means the observation is the first and the last one in the BY group.

D. **First.County**=0 and **Last.County**=0 means the observation is neither the first nor the last one in the BY group.

62

BY-Group Processing

The following program resets the accumulator variable at the beginning of each BY group and outputs only at the end of each BY group.

```
data TopCounties;
   set Top10Town;
   by County;
   if first.County then TotalPopulation=0;
   TotalPopulation+Population;
   if last.County=1;
   keep County TotalPopulation;
run;
```

	County	TotalPopulation
1	Androscoggin	58893
2	Cumberland	125715
3	Kennebec	18560
4	Penobscot	31473
5	York	41748

64

Multiple BY-Group Variables

A **FIRST.** variable and a **LAST.** variable is created for each variable in the BY statement.

```
data CityDonate;
  set Donations;
  by State City;
run;
```

Data must be sorted or indexed by State and City.

First.State, Last.State, First.City, and Last.City are created.

When you use more than one variable in the BY statement, a change in the primary variable forces the **LAST.** variable=1 for the secondary variable.

65

Refer to Exercise 3 for Chapter 4 in Appendix A.

Multiple BY-Group Variables

```
data CityDonate;
   set Donations;
   by State City;
   if first.City=1 then TotalDonation=0;
   TotalDonation+Donation;
   if last.City=1;
run;
```

	state	city	donation	TotalDonation
1	NC	Charlotte	4000	15000
2	NC	Greenville	3000	9000
3	SC	Greenville	2000	7000
4	SC	Pelzer	5000	5000

68

4.4 Answers to Questions

Question Slide Number	Answer
4	6.
11	C.
17	A.
24	C.
27	No
30	B.
39	C.
43	ERROR: Unsatisfied WHEN clause and no OTHERWISE clause.
49	B.
56	Yes
62	C.

Chapter 5 Manipulating Data

5.1 Using Functions to Manipulate Data..5-3

5.2 Converting Character and Numeric Data 5-34

5.3 Processing Data with DO Loops 5-41

5.4 Processing Data with Arrays 5-51

5.5 Answers to Questions..5-59

5.1 Using Functions to Manipulate Data

Functions

A SAS *function* performs a computation or system manipulation on arguments and returns a value.

SAS functions can be used in DATA step programming statements, in a WHERE expression, and in the REPORT procedure.

Functions by Category

- Character
- Date and Time
- Truncation
- Descriptive Statistics
- Special

3

Character Functions

Character		
SUBSTR	CAT	LOWCASE
SCAN	CATS	UPCASE
LEFT	CATT	PROPCASE
RIGHT	CATX	FIND
TRIM	TRANWRD	COMPRESS
STRIP	LENGTH	COMPBL

4

Refer to Exercise 1 for Chapter 5 in Appendix A.

SUBSTR Function

The *SUBSTR function* used to the right of the equal sign extracts a substring from an argument.

specifies a numeric expression that is the beginning character position

```
Features = substr( VIN , 4 , 5 );
```

specifies a numeric expression that is the length of the substring to extract

The SUBSTR function can be used to the left of the equal sign to replace character value constants.

7

SUBSTR Function

If the SUBSTR function returns a value to a variable that was not yet assigned a length, by default, the variable length is determined by the length of the first argument.

VIN (17 bytes)	Assignment Statement Using SUBSTR Function	New Variable (17 bytes)
1F1JF27W86J178227	`Make = substr(VIN,2,1);`	F
	`Features = substr(VIN,4,5);`	JF27W
	`SequenceNumber = substr(VIN,12);`	178227

If you omit the length, SAS extracts the remainder of the expression.

8

SUBSTR Function

Q

The tenth position of a vehicle identification number represents the model year.

Which assignment statement correctly extracts only the tenth position of the vehicle identification number?

A. `ModelYear = substr(VIN,1);`

B. `ModelYear = substr(VIN,10);`

C. `ModelYear = substr(VIN,1,10);`

D. `ModelYear = substr(VIN,10,1);`

9

LENGTH Function

The *LENGTH function* returns the length of a non-blank character string, excluding trailing blanks, and returns 1 for a blank character string.

```
Level=substr(Title,length(Title),1);
```

In this example, the **Level** variable is equal to the last character of the **Title** variable.

```
Name        Title           Level

Sue         Trainer 1         1
Pat         Consultant 3      3
Tim         Analyst 2         2
Lou         Consultant 2      2
```

11

SCAN Function

The *SCAN function* selects a given word from a character expression.

specifies the number of the word, *n*, in the character string to select

```
Six = scan(Sentence , 6 , ' ');
```

specifies the delimiter to use as a word separator in the character string

12

SCAN Function

If the SCAN function returns a value to a variable that was not yet assigned a length, by default, the variable is assigned a length of 200.

Sentence (34 bytes)
`This is a test, a very short test.`

Assignment Statement Using the SCAN Function	New Variable (200 bytes)
`Six = scan(Sentence,6,' ');`	`very`
`Two = scan(Sentence,2,',');`	` a very short test.`

leading blank

13

SCAN Function

What must be added to the following program to control the byte size of the variables **Six** and **Two**?

```
data test;
   Sentence=
     'This is a test, a very short test.';
   Six=scan(Sentence,6,' ');
   Two=scan(Sentence,2,',');
run;
```

14

SCAN Function

If you omit the delimiter, a default list of delimiters is used.

ASCII environment: blank . < (+ & ! $ *) ; ^ - / , % |

EBCDIC environment: blank . < (+ & ! $ *) ; ¬ - / , % | ¢

Sentence (36 bytes)
`This+is an(ex-tremely)**crazy**test!`

Assignment Statement Using the SCAN Function	New Variable (200 bytes)
`Two = scan(Sentence,2);`	`is`
`Four = scan(Sentence,4);`	`ex`

16

SCAN Function

- Leading delimiters before the first word in the character string do not affect the SCAN function.
- If there are two or more contiguous delimiters, the SCAN function treats them as one.
- If *n* is greater than the number of words in the character string, the SCAN function returns a blank value.
- If *n* is negative, the SCAN function selects the word in the character string starting from the end of the string.

17

SCAN Function

What are the new variable values?

Sentence (38 bytes)
`$%This+is an(ex-tremely)**crazy**test!`

Assignment Statement Using the SCAN Function	New Variable (200 bytes)
`One = scan(Sentence,1);`	
`Six = scan(Sentence,6);`	
`Eight = scan(Sentence,8);`	
`MinusTwo = scan(Sentence,-2);`	
`Two = scan(Sentence,2,'*');`	

18

LEFT and RIGHT Functions

- The *LEFT function* left-aligns a character expression.

  LEFT returns an argument with leading blanks moved to the end of the value.

- The *RIGHT function* right-aligns a character expression.

  RIGHT returns an argument with trailing blanks moved to the start of the value.

20

LEFT and RIGHT Functions

If the LEFT or RIGHT function returns a value to a variable that was not yet assigned a length, the variable length is determined by the length of the argument.

Var (13 bytes)	Assignment Statement Using the LEFT or RIGHT Function	New Variable (13 bytes)
3 leading blanks `ZOOLOGY` 3 trailing blanks	`NewVar1 = left(Var);`	`ZOOLOGY` 6 trailing blanks
	`NewVar2 = right(Var);`	6 leading blanks `ZOOLOGY`

21

LEFT and RIGHT Functions

What is the value of the new variable?

Var (13 bytes)	Assignment Statement Using the LEFT or RIGHT Function	New Variable (13 bytes)
3 leading blanks `ZOOLOGY` 3 trailing blanks	`NewVar3 = substr(left(Var),1,3);`	

22

Concatenation Operator

The *concatenation operator* (|| - two vertical bars, ¦¦ - two broken vertical bars, or !! - two exclamation points) concatenates character values.

```
FullName = First || Middle || Last;
```

- The length of the resulting variable is the sum of the lengths of each variable or constant in the concatenation operation, unless you use a LENGTH statement to specify a different length for the new variable.
- The concatenation operator does not trim leading or trailing blanks. If variables are padded with trailing blanks, use the TRIM function to trim trailing blanks from values before concatenating them.

24

Concatenation Operator

```
data name;
   length Name $ 20 First Middle Last $ 10;
   Name = 'Jones, Mary Ann, Sue';
   First = left(scan(Name,2,','));
   Middle = left(scan(Name,3,','));
   Last = scan(name,1,',');
   FullName = First || Middle || Last;
run;
```

First (10 bytes)	Middle (10 bytes)	Last (10 bytes)
Mary Ann	Sue	Jones

FullName (30 bytes)
Mary Ann Sue   Jones

2 blanks 7 blanks 5 blanks

25

TRIM Function

The *TRIM function* removes trailing blanks from character expressions and returns one blank if the expression is missing.

```
FullName = trim(First) || trim(Middle) || Last;
```

- The TRIM function is useful for concatenating because concatenation does not remove trailing blanks.
- If the TRIM function returns a value to a variable that was not yet assigned a length, by default, the variable length is determined by the length of the argument.

26

TRIM Function

```
data name;
   length Name $ 20 First Middle Last $ 10;
   Name = 'Jones, Mary Ann, Sue';
   First = left(scan(Name,2,','));
   Middle = left(scan(Name,3,','));
   Last = scan(name,1,',');
   FullName = trim(First)||trim(Middle)||Last;
run;
```

First (10 bytes)	Middle (10 bytes)	Last (10 bytes)
Mary Ann	Sue	Jones

FullName (30 bytes)
Mary AnnSueJones

14 blanks

27

TRIM Function

The following program is submitted:

```
data name;
   length Name $ 20 First Middle Last $ 10;
   Name = 'Jones, Mary Ann, Sue';
   First = left(scan(Name,2,','));
   Middle = left(scan(Name,3,','));
   Last = scan(name,1,',');
   Name1=trim(Middle);
   Name2=trim(First)||' '||trim(Middle)||' '||Last;
run;
```

What is the byte size of **Name1** and **Name2**?

A. Name1=3 and Name2=30

B. Name1=3 and Name2=32

C. Name1=10 and Name2=30

D. Name1=10 and Name2=32

28

STRIP Function

The *STRIP function* removes leading and trailing blanks from character expressions.

```
data name;
   length Name $ 20 Last First Middle $ 10;
   Name = 'Jones, Mary Ann, Sue';
   Last = scan(name,1,',');
   First = scan(Name,2,',');
   Middle = scan(Name,3,',');
   Name2=strip(First)||' '||strip(Middle)||
         ' '||Last;
run;
```

✎ This example does not use the LEFT function when creating the **First** and **Middle** variables.

30

CAT Functions

The following functions concatenate character strings:

CAT	Does not remove leading or trailing blanks.	`FullName1 = cat(First, Middle, Last);`
CATS	Removes leading and trailing blanks.	`FullName2 = cats(First, Middle, Last);`
CATT	Removes trailing blanks.	`FullName3 = catt(First, Middle, Last);`
CATX	Removes leading and trailing blanks and inserts separators.	`FullName4 = catx(' ', First, Middle, Last);`

31

```
data name;
   length First Middle Last $ 10;
   First = 'Tony';
   Middle = '  Albert';
   Last = 'Smith';
   Name1 = cat(First, Middle, Last);
   Name2 = cats(First, Middle, Last);
   Name3 = catt(First, Middle, Last);
   Name4 = catx(' ', First, Middle, Last);
run;
```

First (10 bytes)	Middle (10 bytes)	Last (10 bytes)
Tony	Albert	Smith

New Variables (200 bytes)		
Name1	`Tony      Albert   Smith`	CAT does not remove blanks.
Name2	`TonyAlbertSmith`	CATS removes leading and trailing blanks.
Name3	`Tony  AlbertSmith`	CATT removes trailing blanks.
Name4	`Tony Albert Smith`	CATX is CATS plus adds separators.

32

CAT Functions

The following program is submitted:

```
data location;
   length City State $ 15 CityState $ 30;
   City='Princeton';
   State='New Jersey';
   <insert statement here>
run;
```

Which statement will give the **CityState** variable a value of Princeton, New Jersey?

A. `CityState = cats(City, State);`

B. `CityState = cats(',', City, State);`

C. `CityState = catx(',', City, State);`

D. `CityState = catx(', ', City, State);`

33

TRANWRD Function

The *TRANWRD function* replaces or removes all occurrences of a word in a character string.

The target specifies the string searched for in the source.

```
NewSentence1=tranwrd(Sentence,'hard','easy');
```

The replacement specifies the string that replaces the target.

35

TRANWRD Function

If the TRANWRD function returns a value to a variable that was not yet assigned a length, by default, the variable is assigned a length of 200.

Sentence (31 bytes)
`Functions are very hard to use.`

Assignment Statement Using the TRANWRD Function	New Variable (200 bytes)
`NewSentence1 = tranwrd(Sentence,'hard','easy');`	`Functions are very easy to use.`
`NewSentence2 = tranwrd(Sentence,'hard','difficult');`	`Functions are very difficult to use.`

36

TRANWRD Function

Q

The following program is submitted:

```
data function;
   Sentence='Functions are very hard to use.';
   NewSentence3=tranwrd(Sentence,'HARD','FUN');
run;
```

What is the result of **NewSentence3**?

A. Functions are very hard to use.
B. Functions are very HARD to use.
C. Functions are very fun to use.
D. Functions are very FUN to use.

37

FIND Function

The *FIND function* searches a character expression for a string of characters.

```
num = find(string,substring);
```

- The FIND function searches the string, from left to right, for the first occurrence of the substring, and returns the position in the string of the substring's first character.
- If the substring is not found in the string, the FIND function returns a value of 0.
- If there are multiple occurrences of the substring, the FIND function returns only the position of the first occurrence.

39

FIND Function

```
num = find(string,substring);
```

string character 37 bytes	substring character 4 bytes	num numeric 8 bytes
How much WOOD would a woodchuck chuck	WOOD	10
	wood	23

10 23

40

FIND Function

The following program is submitted:

```
data tonguetwister;
   string='How much WOOD would a woodchuck chuck';
   num=find(string,'wood ');
run;
```

What is the value of **num**?

A. 0
B. 10
C. 14
D. 23

41

FIND Function

The following program is submitted:

```
data tonguetwister;
   string='How much WOOD would a woodchuck chuck';
   num=find(string,' wo');
run;
```

What is the value of **num**?

A. 0

B. 10

C. 14

D. 23

43

FIND Function

The FIND function can have modifiers and specify a starting position.

modifiers

```
num5 = find(string,'WOOD ','it',15);
```

starting position

45

FIND Function

A *modifier* is a character constant, variable, or expression that specifies one or more modifiers. The following modifiers can be in uppercase or lowercase:

- i - ignores character case during the search.
- t - trims trailing blanks from string and substring.

46

continued...

FIND Function

The *starting position* is an integer that specifies the position at which the search should start and the direction of the search.

- Greater than 0 - starts the search at the starting position, and the direction of the search is to the right. If the starting position is greater than the length of the string, the FIND function returns a value of 0.
- Less than 0 - starts the search at the starting position, and the direction of the search is to the left. If the starting position is greater than the length of the string, the search starts at the end of the string.

47

FIND Function

Ignore case and trim trailing blanks.

```
num5 = find(string,'WOOD ','it',15);
```

Start at position 15 and search to the right.

string character 37 bytes	num5 numeric 8 bytes
How much WOOD would a woodchuck chuck	23

15 → 23

48

Refer to Exercise 2 for Chapter 5 in Appendix A.

Case Functions

The *LOWCASE function* converts all letters in an argument to lowercase.

```
name1 = lowcase(name);
```

The *UPCASE function* converts all letters in an argument to uppercase.

```
name2 = upcase(name);
```

51

Case Functions

The *PROPCASE function* converts all words in an argument to proper case.

```
name3 = propcase(name);
```

- The PROPCASE function first converts all letters to lowercase letters and then converts the first character of words to uppercase.
- The first character of a word is the first letter of a string or any letter preceded by a default list of delimiters.
  Default Delimiter List: blank / - (. tab
- Delimiters can be specified as a second argument, instead of using the default list.

```
name4 = propcase(name,' ');
```

52

Case Functions

If the case functions return a value to a variable that was not yet assigned a length, by default, the variable length is determined by the length of the first argument.

name (16 bytes)	Assignment Statement Using Case Functions	New Variable (16 bytes)
Jane SMITH-JONES	`name1 = lowcase(name);`	`jane smith-jones`
	`name2 = upcase(name);`	`JANE SMITH-JONES`
	`name3 = propcase(name);`	`Jane Smith-Jones`
	`name4 = propcase(name,' ');`	`Jane Smith-jones`

53

Case Functions

Q

The following program is submitted:

```
data example;
   Var='r&d, u.s. division (not puerto rico)';
   NewVar=propcase(Var);
run;
```

The following is the desired value of **NewVar**:

```
R&D, U.S. Division (Not Puerto Rico)
```

Yes or No: Does the program create the desired value of **NewVar**?

54

COMPBL and COMPRESS Functions

- The *COMPBL function* removes multiple blanks in a character string by translating each occurrence of two or more consecutive blanks into a single blank.
- The *COMPRESS function* returns a character string with specified characters removed from the string.

var (13 bytes)	Assignment Statement	New Variable (13 bytes)
ABC - DEF GH	Var1=compbl(Var);	ABC - DEF GH
	Var2=compress(Var);	ABC-DEFGH
	Var3=compress(Var,'-');	ABC DEF GH
	Var4=compress(Var,'- ');	ABCDEFGH

56

Compressing Characters

Q

Var (13 bytes)	Assignment Statement	Var5 (13 bytes)
`ABC - DEF  GH`	`Var5=expression;`	`ABC DEF GH`

Which expression creates the desired **Var5** variable?

A. `compbl(compress(Var,'-'))`

B. `compress(compbl(Var),'-')`

57

Date Functions

Date	
WEEKDAY	YEAR
DAY	TODAY or DATE
MONTH	MDY
QTR	YRDIF

59

Date Functions

The SAS date value for Wednesday, June 13, 1962, is 894.

```
data birthday;
   BirthDate=894;
   BirthWeekDay=weekday(BirthDate);
   BirthDay=day(BirthDate);
   BirthMonth=month(BirthDate);
   BirthQtr=qtr(BirthDate);
   BirthYear=year(BirthDate);
   output;
run;
```

What is the value of **BirthQtr**?

Birth Date	Birth WeekDay	Birth Day	Birth Month	Birth Qtr	Birth Year
894	4	13	6		1962

60

Date Functions

- The *WEEKDAY function* returns the day of the week (1=Sunday, … 7=Saturday) from a SAS date value.
- The *DAY function* returns the day of the month (1-31) from a SAS date value.
- The *MONTH function* returns the month (1-12) from a SAS date value.
- The *QTR function* returns the quarter of the year (1-4) from a SAS date value.
- The *YEAR function* returns the four-digit year from a SAS date value.

62

Date Functions

Q

The following program is submitted:

```
data birthday;
   BirthDate='13JUN1962'd;
   BirthWeekDay=weekday(BirthDate);
   output;
run;
```

Which of the following statements is ***true***?

A. **BirthDate** and **BirthWeekDay** are both character.

B. **BirthDate** and **BirthWeekDay** are both numeric.

C. **BirthDate** is character and **BirthWeekDay** is numeric.

D. **BirthDate** is numeric and **BirthWeekDay** is character.

63

TODAY and DATE Functions

The *TODAY and DATE functions* returns the current date as a SAS date value.

- A SAS date value is the number of days since January 1, 1960.

If today were December 15, 2007:

	Current (Numeric 8 bytes)
`Current = today();` `Current = date();`	17515

65

MDY Function

The *MDY function* returns a SAS date value from month (integer 1 through 12), day (integer 1 through 31), and year (two-digit integer or four-digit integer) values.

	Date (Numeric 8 bytes)
`Date = mdy(12,15,2007);`	17515
`Date = mdy(m,d,y);`	8121

m	d	y	Date
3	27	1982	8121

66

YRDIF Function

The *YRDIF function* returns the difference in years between two dates.

specifies a SAS date value that identifies the starting date

identifies a basis for how SAS calculates the date difference

```
age = yrdif(birth,'02feb2007'd,'AGE');
```

specifies a SAS date value that identifies the ending date

If the value of basis is AGE, then YRDIF computes the age. The age computation takes into account leap years.

67

YRDIF Function

```
data age;
   set birthday;
   age = yrdif(birth,'02feb2007'd,'AGE');
run;
```

	birth	age
1	3985	36.178082192
2	3673	37.032876712

3985 is November 29, 1970.
3673 is January 21, 1970.

68

YRDIF Function

Given the following assignment statement:

```
age=yrdif('05MAY1999'd,'10NOV1999'd,'AGE');
```

What is an approximate value of **age**?

69

Truncation Functions

Truncation
CEIL
FLOOR
INT
ROUND

71

Refer to Exercise 3 for Chapter 5 in Appendix A.

CEIL and FLOOR Functions

The *CEIL function* returns the smallest integer that is ***greater*** than or equal to the argument.

The *FLOOR function* returns the largest integer that is ***less*** than or equal to the argument.

num	ceil(num)	floor(num)
2.75	3	2
-2.75	-2	-3
23.1234	24	23
-23.1234	-23	-24

If the argument is within 1E-12 of an integer, the function returns that integer.

74

CEIL and FLOOR Functions

Q

The following program is submitted:

```
data trunc;
   num=29;  A=ceil(num);  B=floor(num);
   output;
run;
```

What is the value of **A** and **B** in the final data set?

A. **A**=29 and **B**=29
B. **A**=30 and **B**=28
C. **A**=30 and **B**=29
D. **A**=29 and **B**=28

75

INT Function

The *INT function* returns the integer value.

num	int(num)
2.75	2
-2.75	-2
23.1234	23
-23.1234	-23

If the argument is within 1E-12 of an integer, the INT function returns that integer.

77

INT Function

Yes or No: Are both of the following statements ***true***?

- The INT function has the same result as the FLOOR function if the value of the argument is positive.
- The INT function has the same result as the CEIL function if the value of the argument is negative.

num	ceil(num)	floor(num)	int(num)
2.75	3	2	2
-2.75	-2	-3	-2
23.1234	24	23	23
-23.1234	-23	-24	-23

78

ROUND Function

The *ROUND function* rounds the first argument to the nearest integer when the second argument is omitted.

num	round(num)
2.75	3
-2.75	-3
23.1234	23
-23.1234	-23

80

ROUND Function

The ROUND function rounds the first argument to the nearest multiple of the second argument.

```
data rounding;
   d1 = round(1234.56789 , 100);
   d2 = round(1234.56789 , 10);
   d3 = round(1234.56789 , 1);
   d4 = round(1234.56789 , .1);
   d5 = round(1234.56789 , .01);
   d6 = round(1234.56789 , .001);
run;
```

d1	d2	d3	d4	d5	d6
1200	1230	1235	1234.6	1234.57	1234.568

81

ROUND Function

Q

The following program is submitted:

```
data round;
   num=986.151;
   C=round(num,100);  D=round(num,.01);
   output;
run;
```

What is the value of **C** and **D** in the final data set?

A. **C**=900 and **D**=986.2
B. **C**=1000 and **D**=986.2
C. **C**=900 and **D**=986.15
D. **C**=1000 and **D**=986.15

82

Descriptive Statistics Functions

MAX	Returns the largest value.
MEAN	Returns the arithmetic mean (average).
MIN	Returns the smallest value.
SUM	Returns the sum of the nonmissing arguments.
N	Returns the number of nonmissing numeric values.
NMISS	Returns the number of missing numeric values.
CMISS	Returns the number of missing numeric and character values.

84

Descriptive Statistics Functions

```
data math;
   var1=2;
   var2=6;
   var3=.;
   var4=4;
   maximum = max(var1,var2,var3,var4);
   average = mean(var1,var2,var3,var4);
   minimum = min(var1,var2,var3,var4);
   total = sum(var1,var2,var3,var4);
run;
```

maximum	average	minimum	total
6	4	2	12

85

Descriptive Statistics Functions

The argument list can consist of a variable list, which is preceded by OF.

```
data math;
   var1=2;
   var2=6;
   var3=.;
   var4=4;
   maximum = max(of var1-var4);
   average = mean(of var1-var4);
   minimum = min(of var1-var4);
   total = sum(of var1-var4);
run;
```

maximum	average	minimum	total
6	4	2	12

86

SAS Variable Lists		
Numbered range lists	`x1-xn`	Specifies all variables from **x1** to **x*n*** inclusive. You can begin with any number and end with any number as long as you do not violate the rules for user-supplied variable names and the numbers are consecutive.
Name range lists	`x--a`	Specifies all variables ordered as they are ordered in the program data vector, from **x** to **a** inclusive.
	`x-numeric-a`	Specifies all numeric variables from **x** to **a** inclusive.
	`x-character-a`	Specifies all character variables from **x** to **a** inclusive.
Name prefix lists	`REV:`	Specifies all the variables that begin with **REV**, such as **REVJAN**, **REVFEB**, and **REVMAR**.
Special SAS name lists	`_ALL_`	Specifies all variables that are already defined in the current DATA step.
	`_NUMERIC_`	Specifies all numeric variables that are already defined in the current DATA step.
	`_CHARACTER_`	Specifies all character variables that are already defined in the current DATA step.

Descriptive Statistics Functions

Q

The following program is submitted:

```
data math;
   var1=30;
   var2=15;
   var3=10;
   total = sum(var1-var3);
run;
```

What is the value of **total**?

A. 0

B. 20

C. 40

D. 55

87

5.2 Converting Character and Numeric Data

Converting Character and Numeric Data

Data can be converted with the following two methods:

- automatic conversion
- explicit conversion with a SAS function

Special	
INPUT	PUT
character-to-numeric	numeric-to-character

90

Automatic Character-to-Numeric Conversion

Automatic character-to-numeric conversion happens when a character value is used in a numeric context.

For example:

- assignment to a numeric variable

```
num=char;
```

- an arithmetic operation

```
num2=num1+char;
```

- logical comparison with a numeric value

```
if num>char;
```

- a function that takes numeric arguments

```
num2=mean(num1,char);
```

91

Automatic Character-to-Numeric Conversion

Automatic character-to-numeric conversion

- uses the W. informat
- produces a numeric missing if the character value does not conform to the W. informat
- writes a message to the SAS log stating that the conversion occurred.

```
819  data numeric;
820    num1=5;
821    char='6';
822    num2=num1+char;
823  run;

NOTE: Character values have been converted to numeric
      values at the places given by: (Line):(Column).
      822:13
NOTE: The data set WORK.NUMERIC has 1 observations and 3 variables.
```

92

Automatic Numeric-to-Character Conversion

Automatic numeric-to-character conversion happens when a numeric value is used in a character context.

For example:

- assignment to a character variable

```
char=num;
```

- a concatenation operation

```
char2=char1||num;
```

- a function that takes character arguments

```
char=substr(num,3,1);
```

93

Automatic Numeric-to-Character Conversion

Automatic numeric-to-character conversion

- uses the BEST12. format
- right-aligns the resulting character value
- writes a message to the SAS log stating that the conversion occurred.

```
828  data character;
829    num=1234567;
830    char=substr(num,3,1);
831  run;

NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
      830:15
NOTE: The data set WORK.CHARACTER has 1 observations and 2 variables.
```

94

Refer to Exercise 4 for Chapter 5 in Appendix A.

Explicit Conversion Using SAS Functions

Explicit conversion using a SAS function

- produces desirable results
- does ***not*** write a message to the SAS log stating that the conversion occurred.

Special	
INPUT	PUT
character-to-numeric	numeric-to-character

98

Explicit Conversion Using SAS Functions

The INPUT function converts a character value to a numeric value.

- The second argument is a numeric informat.
- If the INPUT function returns a value to a variable that was not yet assigned a length, by default, the variable length is 8 bytes.

The PUT function converts a numeric value to a character value.

- The second argument is a numeric format.
- If the PUT function returns a value to a variable that was not yet assigned a length, by default, the variable length is determined by the width of the format.

99

INPUT Function

Value of Character Variable		Value of Numeric Variable (8 bytes)
`162400`	`input('162400',6.)`	`162400`
`$162,400`	`input('$162,400',comma8.)`	`162400`
`49275.937`	`input('49275.937',9.)`	`49275.937`
`+24`	`input('+24',3.)`	`24`
`-73.5`	`input('-73.5',5.)`	`-73.5`
`01234`	`input('01234',5.)`	`1234`
`52E3`	`input('52E3',4.)`	`52000`
`01/01/1960`	`input('01/01/1960',mmddyy10.)`	`0`

100

PUT Function

Value of Numeric Variable (8 bytes)		Value of Character Variable
`162400`	`put(162400,dollar8.);`	`$162,400`
`49275.937`	`put(49275.937,comma10.3);`	`49,275.937`
`-73.5`	`put(-73.5,5.1);`	`-73.5`
`52E3`	`put(52E3,5.);`	`52000`
`0`	`put(0,date9.);`	`01JAN1960`

101

Data Set **Personnel**:

	Hired	First	Last	SSN
1	27MAR2003	Samatha	Jones	444444444
2	01SEP2006	Timothy	Peters	999999999

Hired is character.
SSN is numeric.

The following program is submitted:

```
data NewPersonnel;
   set Personnel;
   NewHired=input(Hired,date9.);
   TempSSN=put(SSN,9.);
   NewSSN=catx('-',substr(TempSSN,1,3),
                   substr(TempSSN,4,2),
                   substr(TempSSN,6));
run;
```

Data Set **NewPersonnel**:

	Hired	First	Last	SSN	NewHired	TempSSN	NewSSN
1	27MAR2003	Samatha	Jones	444444444	15791	444444444	444-44-4444
2	01SEP2006	Timothy	Peters	999999999	17045	999999999	999-99-9999

The SSN*w*. format could be used to display the numeric **SSN** with dashes.

102

INPUT and PUT Functions

Q

Data Set **Personnel**:

	Hired	First	Last	SSN
1	27MAR2003	Samatha	Jones	444444444
2	01SEP2006	Timothy	Peters	999999999

Hired is character.
SSN is numeric.

The following program is submitted:

```
data NewPersonnel;
   set Personnel;
   Hired=input(Hired,date9.);
   SSN=put(SSN,9.);
run;
```

Yes or No: Does this modified program create a numeric **Hired** and a character **SSN**?

103

INPUT and PUT Functions

After a variable type is established, it cannot be changed.

cannot be numeric because already defined as character

```
Hired=input(Hired,date9.);
```

defined in input data set as character

105

INPUT and PUT Functions

Solution:

```
data NewPersonnel;
   set Personnel
      (rename=(Hired=TempH SSN=TempS));
   Hired=input(TempH,date9.);
   SSN=put(TempS,9.);
   drop TempH TempS;
run;
```

- The RENAME= data set option changes the names of the original variables with unwanted data types.
- The DROP statement excludes the original variables with unwanted data types from the output SAS data set.

106

5.3 Processing Data with DO Loops

DO Loops

The *DO loop* executes statements between DO and END repetitively based on the value of an index variable.

```
data training;
   miles=30;
   do weeks = 1 to 6 by 1;
      miles+miles*0.04;
   end;
run;
```

- Sally ran 30 miles per week.
- She plans to increase her mileage by 4% each week for six weeks.

108

Index Variable

The *index variable* names a variable whose value governs execution of the DO group.

index variable

```
do weeks = 1 to 6 by 1;
   miles+miles*0.04;
end;
```

- The index variable argument is required.
- Unless you specify to drop it, the index variable is included in the data set that is being created.

109

Specification

The *specification* denotes an expression or a series of expressions.

specification

```
do weeks = 1 to 6 by 1;
   miles+miles*0.04;
end;
```

Possible specifications:

- ***start*** **TO** ***stop*** **BY** ***increment***
- ***start1*****,** ***start2*****, ...**
- **WHILE(*****expression*****)**
- **UNTIL(*****expression*****)**

110

Specification

start stop increment

```
do weeks = 1 to 6 by 1;
   miles+miles*0.04;
end;
```

- *start* specifies the initial value of the index variable.
- *stop* is an optional value that specifies the ending value of the index variable.
- *increment* is an optional value that specifies a positive or negative number to control the incrementing of the index variable.

  If no increment is specified, the index variable is increased by 1.

111

Specification

Which of the following is ***not*** a valid DO statement?

A.
```
do 2 to 10 by 2;
```

B.
```
do year=2006 to 2000 by -2;
```

C.
```
do count=1 to num2+num3 by num4;
```

D.
```
do date='15JUN2007'd to '31DEC2007'd;
```

112

DO Loop Output

The following program is submitted:

```
data training;
   miles=30;
   do weeks = 1 to 6;
      miles+miles*0.04;
   end;
run;
proc print data=training noobs;
run;
```

What is the result?

A.

miles	weeks
31.2000	1
32.4480	2
33.7459	3
35.0958	4
36.4996	5
37.9596	6

B.

miles	weeks
37.9596	6

C.

miles	weeks
37.9596	7

114

DO Loop Output

miles	weeks
37.9596	7

```
data training;
   miles=30;
   do weeks = 1 to 6;
      miles+miles*0.04;
   end;
   output;
run;
```

miles	weeks
31.2000	1
32.4480	2
33.7459	3
35.0958	4
36.4996	5
37.9596	6

```
data training;
   miles=30;
   do weeks = 1 to 6;
      miles+miles*0.04;
      output;
   end;
run;
```

miles	weeks
37.9596	6

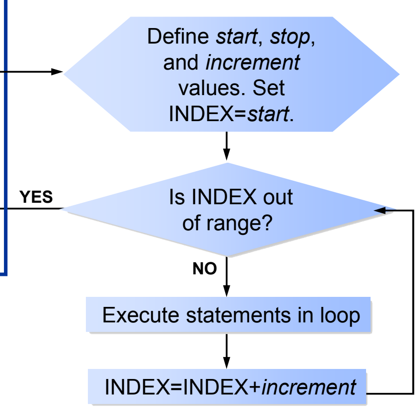
```
data training;
   miles=30;
   do weeks = 1 to 6;
     miles+miles*0.04;
     if weeks=6 then output;
   end;
run;
```

116

DO Loop Output

The value of increment is evaluated before the execution of the loop.

```
data training;
   miles=30;
   do weeks = 1 to 6;
      miles+miles*0.04;
   end;
   output;
run;
```



117

DO Loop with SET Statement

```
data training;
   set runners;
   do weeks=1 to 6;
     miles+miles*pct;
     output;
   end;
run;
```

VIEWTABLE: Work.Runners

	name	miles	pct
1	Jill	24	0.05
2	Ray	28	0.04
3	Mark	30	0.05

1. How many times does SAS loop through the DATA step?
2. How many times does SAS loop through the DO loop per each DATA step iteration?
3. How many observations are created?

118

DO Loop with SET Statement

Three observations are in the data set **work.runners**.

```
data training;
  set runners;
  do weeks=1 to 6;
    miles+miles*pct;
    output;
  end;
run;
```

3 times

6 times

18 observations

120

DO Loop with INFILE/INPUT Statements

Q

Yes or No: Can a DO loop be in a DATA step that is reading in a raw data file?

```
data training;
   infile 'raw-data-file';
   input name $ miles pct;
   do weeks=1 to 6;
      miles+miles*pct;
      output;
   end;
run;
```

121

Nested DO Loops

DO loops can be nested.

miles	pct	month	weeks	
30.9000	0.03	1	1	3% weekly increase for month 1
31.8270	0.03	1	2	
32.7818	0.03	1	3	
33.7653	0.03	1	4	
35.1159	0.04	2	1	4% weekly increase for month 2
36.5205	0.04	2	2	
37.9813	0.04	2	3	
39.5006	0.04	2	4	
41.4756	0.05	3	1	5% weekly increase for month 3
43.5494	0.05	3	2	
45.7269	0.05	3	3	
48.0132	0.05	3	4	

- Sally runs 30 miles per week.
- She plans to increase her mileage weekly for three months.

123

Nested DO Loops

```
data training;
   miles=30;
   pct=0.02;
   do month=1 to 3;
      pct+0.01;
      do weeks = 1 to 4;
         miles+miles*pct;
         output;
      end;
   end;
run;
```

3 times

4 times

miles	pct	month	weeks
30.9000	0.03	1	1
31.8270	0.03	1	2
32.7818	0.03	1	3
33.7653	0.03	1	4
35.1159	0.04	2	1
36.5205	0.04	2	2
37.9813	0.04	2	3
39.5006	0.04	2	4
41.4756	0.05	3	1
43.5494	0.05	3	2
45.7269	0.05	3	
48.0132	0.05	3	

12 observations

124

Nested DO Loops

The following program is submitted with no OUTPUT statement:

```
data training;
   miles=30;
   pct=0.02;
   do month=1 to 3;
      pct+0.01;
      do weeks = 1 to 4;
         miles+miles*pct;
      end;
   end;
run;
```

How many observations are created?

A. 0 B. 1 C. 3 D. 12

125

Additional Specifications

The specification in the DO statement can be a series of items separated by commas.

- The items can be either all numeric or all character constants, or might be variables.
- Character constants must be enclosed in quotation marks.
- The DO group is executed once for each value in the list.

Examples:

- `do month = 'JAN','FEB','MAR';`
- `do count = 2,3,5,7,11,13,17;`
- `do i = var1, var2, var3;`
- `do date = '01JAN2007'd, '25APR2007'd;`

127

Additional Specifications

```
data training;
   miles=30;
   do date=today()+30, today()+100;
      miles+miles*0.10;
      output;
   end;
run;
```

miles	date
33.0	27MAY2007
36.3	05AUG2007

- Sally runs 30 miles per week.
- Thirty days from now she plans to increase her mileage by 10%.
- One hundred days from now she plans to increase her mileage again by 10%.

128

Conditional Specifications

The DO statement can execute statements repetitively while a condition is true or until a condition is true.

DO WHILE(*expression*)

- executes while a condition is true
- is evaluated at the ***top*** of the loop
- does not execute if the expression is false the first time that it is evaluated.

DO UNTIL(*expression*)

- executes until a condition is true
- is evaluated at the ***bottom*** of the loop
- is executed at least once.

129

Conditional Specifications

```
data training;
   miles=30;
   do while(miles<=50);
     week+1;
     miles+miles*0.04;
     output;
   end;
run;
```

```
data training;
   miles=30;
   do until(miles>50);
     week+1;
     miles+miles*0.04;
     output;
   end;
run;
```

	miles	week
1	31.2	1
2	32.45	2
3	33.75	3
4	35.1	4
5	36.5	5
6	37.96	6
7	39.48	7
8	41.06	8
9	42.7	9
10	44.41	10
11	46.18	11
12	48.03	12
13	49.95	13
14	51.95	14

Sally plans to increase her mileage by 4% each week until she runs 50+ miles per week.

130

Conditional Specifications

Q

Yes or No: Will the DO loop in the following program execute?

```
data training;
   miles=45;
   do until(miles>40);
      week+1;
      miles+miles*0.04;
      output;
   end;
run;
```

131

Combined Specifications

A WHILE(*expression*) or UNTIL(*expression*) specification can be combined with the start-to-stop-by-increment specification.

```
data training;
   miles=30;
   do weeks=1 to 30 until(miles>50);
      miles+miles*0.02;
      output;
   end;
run;
```

Sally plans to increase her mileage by 2% each week until she runs 50+ miles per week ***or*** until she reaches 30 weeks.

133

In a DO UNTIL loop, the condition is checked ***before*** the index variable is incremented.

In a DO WHILE loop, the condition is checked ***after*** the index variable is incremented.

Q

Combined Specifications

The following program is submitted:

```
data training;
   miles=30;
   do weeks=1 to 30 until(miles>50);
     miles+miles*0.02;
     output;
   end;
run;
```

The **training** data set has 26 observations with the last observation resembling the following:

miles	weeks
50.202543431	26

What ended the DO loop?

A. `weeks=1 to 30`

B. `until(miles>50)`

134

5.4 Processing Data with Arrays

Arrays

An *array* is a temporary grouping of SAS variables that are arranged in a particular order and identified by an array name.

- Arrays exists only for the duration of the current DATA step.
- Arrays are referenced by the array name and a subscript.
- The array name is not a variable.

An array is only a convenient way of temporarily identifying a group of variables. Arrays are often referenced in DO loops because more than one element in an array must be processed.

138

Arrays

Examples:

Arrays	Variables				
	1	2	3	4	
`run`	`week1`	`week2`	`week3`	`week4`	
	1	2	3	4	5
`health`	`Height`	`Weight`	`BlPres`	`Pulse`	`Chol`

139

Defining an Array

An ARRAY statement defines elements of an array.

```
array run{4} week1-week4;
```

```
array health{5} Height Weight BlPres Pulse Chol;
```

array name

number of elements (variables) in the array

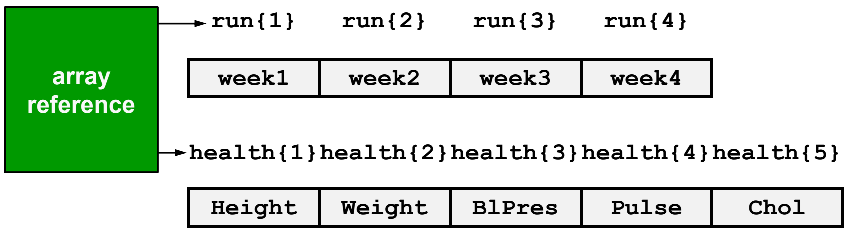
list of variables in the array

- The number of elements must be enclosed in parentheses (), braces {}, or brackets [].
- Variables defined in a given array must be all character or all numeric.

140

Referencing an Array

To reference an array that was previously defined in the same DATA step, use an array reference.



141

Defining and Referencing an Array

VIEWTABLE: Work.Weekly

	name	week1	week2	week3	week4
1	Jack	25	32	48	33
2	Susan	10	12	10	10

```
data Increase;
   set Weekly;
   array run{4} week1 - week4;
   do week = 1 to 4;
      run{week} = run{week}*1.10;
   end;
   drop week;
run;
```

VIEWTABLE: Work.Increase

	name	week1	week2	week3	week4
1	Jack	27.5	35.2	52.8	36.3
2	Susan	11	13.2	11	11

142

Q

Defining and Referencing an Array

```
data Stats2(drop=i);
   set Stats;
   count=0;
   array __________ height weight blpres pulse chol;
   do i = 1 to 5;
      if health{i} = 'AboveAve' then count+1;
   end;
run;
```

VIEWTABLE: Work.Stats2

	name	height	weight	blpres	pulse	chol	count
1	Jana	Ave	Ave	Ave	Ave	Ave	0
2	Tyler	Ave	AboveAve	AboveAve	Ave	AboveAve	3
3	William	Ave	Ave	Ave	Ave	AboveAve	1

Based on the program and the results of the final data set, what should be in the blank space?

A. `i{5}`

B. `count{5}`

C. `height{5}`

D. `health{5}`

143

Defining and Referencing an Array

When an asterisk is used to specify the number of elements, SAS is to determine the subscript by counting the variables in the array.

```
data newprices;
   set sashelp.pricedata;
   array parray{*} price:;
   total=sum(of parray{*});
   do num = 1 to dim(parray);
      parray{num} = parray{num}*1.10;
   end;
run;
```

The DIM function in the iterative DO statement returns the number of elements in an array.

145

Refer to Exercise 5 for Chapter 5 in Appendix A.

Creating Numeric Variables with an Array

An array can be based on existing variables or new variables.

```
data WeeklyDiff(drop=week);
   set Weekly;
   array run{4} week1 - week4;
   array diff{3} diff21 diff32 diff43;


   do week = 1 to 3;
      diff{week} = run{week+1}-run{week};
   end;
run;
```

array based on existing variables

array based on new variables

	name	week1	week2	week3	week4	diff21	diff32	diff43
1	Jack	25	32	48	33	7	16	-15
2	Susan	10	12	10	10	2	-2	0

148

Creating Numeric Variables with an Array

SAS creates variable names by concatenating the array name and the numbers 1, 2, 3, . . . *n*.

```
data WeeklyPct(drop=week);
   set Weekly;
   total=sum(of week1-week4);
   array run{4} week1 - week4;
   array pctwk{4};
   do week = 1 to 4;
      pctwk{week} = run{week}/total;
   end;
   format pctwk1-pctwk4 percent6.;
run;
```

array based on existing variables

array based on new variables

	name	week1	week2	week3	week4	total	pctwk1	pctwk2	pctwk3	pctwk4
1	Jack	25	32	48	33	138	18%	23%	35%	24%
2	Susan	10	12	10	10	42	24%	29%	24%	24%

149

Creating Character Variables with an Array

If an array is based on new character variables, the elements must be defined as character with a byte size.

```
data newnames(drop=i);
   set names;
   array first{3};
   array newfirst{3} $ 7;

   do i=1 to 3;
      newfirst{i}=propcase(first{i});
   end;
run;
```

array based on existing variables (`array first{3};`)

array based on new variables (`array newfirst{3} $ 7;`)

	first1	first2	first3	newfirst1	newfirst2	newfirst3
1	janet	russell	cheryl	Janet	Russell	Cheryl

150

Creating Character Variables with an Array

Q

```
data newnames(drop=i);
   set names;
   array first{3};
   array newfirst{3} $ ;
   do i=1 to 3;
      newfirst{i}=propcase(first{i});
   end;
run;
```

What is the byte size of **newfirst1-newfirst3** if a byte size is not specified in the ARRAY statement?

A. 0

B. 7 (the byte size of **first1-first3**)

C. 8

D. 18

151

Initial Values

Initial values can be specified for the corresponding elements in the array.

```
data score(drop=x);
   infile 'raw-data-file';
   input name $ q1 $ q2 $ q3 $ q4 $ q5 $;
   array answer{5} q1-q5;
   array correct{5} $ 1 ('A','B','A','D','C');
   score=0;
   do x=1 to 5;
      if answer{x}=correct{x}
         then score+1;
   end;
run;
```

initial values

	name	q1	q2	q3	q4	q5	correct1	correct2	correct3	correct4	correct5	score
1	monica	A	C	B	D	C	A	B	A	D	C	3
2	ted	A	B	A	D	C	A	B	A	D	C	5
3	kelly	A	C	C	A	C	A	B	A	D	C	2

153

TEMPORARY Option

The _TEMPORARY_ option is used to create a list of temporary data elements.

```
data score(drop=x);
   infile 'raw-data-file';
   input name $ q1 $ q2 $ q3 $ q4 $ q5 $;
   array answer{5} q1-q5;
   array correct{5} $ 1 _temporary_
                    ('A','B','A','D','C');
   score=0;
   do x=1 to 5;
      if answer{x}=correct{x}
         then score+1;
   end;
run;
```

	name	q1	q2	q3	q4	q5	score
1	monica	A	C	B	D	C	3
2	ted	A	B	A	D	C	5
3	kelly	A	C	C	A	C	2

154

Q

TEMPORARY Option

Which of the following statements is valid?

A.
```
array revenue{3} _temporary_
      rev1-rev3 (12,15,22);
```

B.
```
array revenue{3} $ rev1-rev3
      _temporary_ ('12','15','22');
```

C.
```
array revenue{3} _temporary_ (12 15 22);
```

D.
```
array revenue{3} _temporary_ 8 (12 15 22);
```

155

5.5 Answers to Questions

Question Slide Number	Answer
9	D.
14	length Six Two $ 22;
18	• One = This • Six = crazy • Eight = • MinusTwo = crazy • Two = crazy
22	ZOO (with 10 trailing blanks)
28	D.
33	D.
37	A.
41	A.
43	C.
54	No
57	A.
60	BirthQtr = 2
63	B.
69	0.5178
75	A.
78	Yes
82	D.
87	B.
103	No
112	A.
114	C.

(Continued on the next page.)

Question Slide Number	Answer
118	1. 3 2. 6 3. 18
121	Yes
125	B.
131	Yes
134	B.
143	D.
151	C.
155	C.

Chapter 6 Generating Reports

6.1 Creating Detail Reports with the PRINT Procedure..6-3

6.2 Creating Formats with the FORMAT Procedure 6-22

6.3 Creating Frequency Tables with the FREQ Procedure 6-30

6.4 Creating Summary Reports with the MEANS Procedure..6-35

6.5 Directing Reports to External Files with ODS..6-40

6.6 Answers to Questions..6-46

6.1 Creating Detail Reports with the PRINT Procedure

PRINT Procedure

The *PRINT procedure* creates a report of the variables and observations in a SAS data set. You can create a variety of reports ranging from a simple listing to a highly customized report that groups the data and calculates totals and subtotals for numeric variables.

Partial Output

```
                     Boot and Sandal Report
------------------------ Region=Asia ------------------------

                                   Total             Total
   Subsidiary    Product       Inventory             Sales

    Bangkok      Boot          $9,576.00         $1,996.00
    Bangkok      Sandal       $15,087.00         $3,230.00
    Seoul        Boot        $160,589.00        $60,712.00
    Seoul        Sandal       $21,483.00         $4,978.00
   ----------              -------------     -------------
    Region                   $206,735.00        $70,916.00

                     Created by Tony Smith
                          Chicago, IL
```

4

PRINT Procedure

Q

Which ***two*** of the following items ***cannot*** be accomplished with the PRINT procedure?

1. produce detail reports
2. produce summary reports
3. sort data values by one or more variables
4. produce column totals for numeric variables
5. replace variable values with formatted values
6. replace variable names with descriptive labels
7. choose only observations that meet a condition
8. select specific variables and control the order in which the variables appear

5

Example of the PRINT Procedure

```
options nodate nonumber ps=30 ls=64;

proc print data=sashelp.shoes noobs split='*';
   var subsidiary product inventory sales;
   where product='Boot' or product='Sandal';
   sum inventory sales;
   by region;
   pageby region;
   label inventory='Total*Inventory'
         sales='Total*Sales';
   format inventory sales dollar14.2;
   title 'Boot and Sandal Report';
   footnote 'Created by Tony Smith';
   footnote2 'Chicago, IL';
run;
```

7

Refer to Exercise 1 for Chapter 6 in Appendix A.

NOOBS Option

The *NOOBS option* suppresses the column in the output that identifies each observation by number.

```
proc print data=sashelp.shoes noobs split='*';
```

By default, the PRINT procedure gives an observation column.

10

NOOBS Option

Without NOOBS

```
                Boot and Sandal Report

-                   Region=Asia -------------------------

                                      Total            Total
  Obs    Subsidiary    Product      Inventory          Sales

   57     Bangkok      Boot         $9,576.00      $1,996.00
   59     Bangkok      Sandal      $15,087.00      $3,230.00
   62     Seoul        Boot       $160,589.00     $60,712.00
   65     Seoul        Sandal      $21,483.00      $4,978.00
------
Region
```

Partial Output

With NOOBS

```
                       Boot and Sandal Report

-                   --- Region=Asia -------------------------

                                    Total                Total
Subsidiary     Product          Inventory                Sales

 Bangkok       Boot             $9,576.00            $1,996.00
 Bangkok       Sandal          $15,087.00            $3,230.00
 Seoul         Boot           $160,589.00           $60,712.00
 Seoul         Sandal          $21,483.00            $4,978.00
----------                 --------------       --------------
 Region                       $206,735.00           $70,916.00
```

Partial Output

11

VAR Statement

The *VAR statement* selects variables that appear in the report and determines the order of the variables.

```
var subsidiary product inventory sales;
```

```
                      Boot and Sandal Report

------------------------- Region=Asia -------------------------

                                    Total                Total
Subsidiary     Product          Inventory                Sales

 Bangkok       Boot             $9,576.00            $1,996.00
 Bangkok       Sandal          $15,087.00            $3,230.00
 Seoul         Boot           $160,589.00           $60,712.00
 Seoul         Sandal          $21,483.00            $4,978.00
----------                 --------------       --------------
 Region                       $206,735.00           $70,916.00
```

By default, the PRINT procedure displays all variables in the order that the variables are stored in the data set.

12

VAR Statement

The following list report is requested:

```
Obs    Product          Stores       Returns

  1    Boot               12          $769
  2    Men's Casual        4        $2,284
  3    Men's Dress         7        $2,433
  4    Sandal             10        $1,861
```

Which VAR statement creates the desired output?

A.
```
var product stores returns;
```

B.
```
var product, stores, returns;
```

C.
```
var obs product stores returns;
```

D.
```
var obs, product, stores, returns;
```

13

WHERE Statement

The *WHERE statement* subsets the input data set by specifying certain conditions that each observation must meet before it is available for the report.

```
where product='Boot' or product='Sandal';
```

- The WHERE statement does not alter the original data set.
- Use only one WHERE statement in a step unless you use a WHERE SAME AND or WHERE ALSO statement with a WHERE statement.
- Character values are case sensitive.

15

Q

WHERE Statement

Which WHERE statement does ***not*** produce results that are identical to those of the other three WHERE statements?

A.
```
where product='Boot' or 'Sandal';
```

B.
```
where product in ('Boot' 'Sandal');
```

C.
```
where product in ('Boot','Sandal');
```

D.
```
where product='Boot' or product='Sandal';
```

16

WHERE Statement

Examples:

```
where sales > 100000;
where sales eq .;
where name = 'Smith';
where name = ' ';
where sales ge 100000 and name = 'Smith';
where sales ge 100000 or name = 'Smith';
where revenue >= 150 and revenue <= 999;
where revenue between 150 and 999;
where revenue not between 150 and 999;
where month contains 'uary';
where birthdate > '11JUL1968'd;
```

18

Q

WHERE Statement

The following SAS program is submitted:

```
proc print data=sales;
   where month=2;
   where sales>100;
run;
```

Which of the following statements is ***true*** regarding the program?

A. Only the first WHERE statement will be used.

B. Only the second WHERE statement will be used.

C. Both WHERE statements will be used with a logical OR between the statements.

D. Both WHERE statements will be used with a logical AND between the statements.

19

SUM Statement

The *SUM statement* totals values of numeric variables.

```
sum inventory sales;
```

Partial Output

```
                        Boot and Sandal Report

------------------------- Region=Asia -------------------------

                                        Total             Total
   Subsidiary    Product            Inventory             Sales

    Bangkok      Boot               $9,576.00         $1,996.00
    Bangkok      Sandal            $15,087.00         $3,230.00
    Seoul        Boot             $160,589.00        $60,712.00
    Seoul        Sandal            $21,483.00         $4,978.00
   ----------                  --------------    --------------
    Region                        $206,735.00        $70,916.00
```

The SUM statement always gives grand totals and gives subtotals if used with a BY statement.

21

BY Statement

The *BY statement* produces a separate section of the report for each BY group.

```
by region;
```

Partial Output

```
                    Boot and Sandal Report

--------------------- Region=Asia ------------------------

                                   Total              Total
Subsidiary    Product          Inventory              Sales

 Bangkok      Boot             $9,576.00          $1,996.00
 Bangkok      Sandal          $15,087.00          $3,230.00
 Seoul        Boot           $160,589.00         $60,712.00
 Seoul        Sandal          $21,483.00          $4,978.00
----------                --------------     --------------
 Region                      $206,735.00         $70,916.00
```

Data must be indexed or sorted to use a BY statement.

22

SUM and BY Statements

Partial Output

```
------------------- Region=Western Europe -----------------

                                   Total              Total
Subsidiary    Product          Inventory              Sales

Copenhagen    Boot             $4,657.00          $1,663.00
Geneva        Boot           $171,030.00         $41,341.00
Geneva        Sandal           $3,529.00            $736.00
Heidelberg    Boot           $301,779.00         $65,610.00
Heidelberg    Sandal           $4,618.00            $977.00
Lisbon        Boot           $341,911.00         $76,349.00
Lisbon        Sandal          $24,253.00          $1,650.00
London        Boot           $289,527.00         $54,449.00
London        Sandal          $11,111.00          $5,217.00
Madrid        Boot             $1,027.00          $1,179.00
Paris         Boot            $41,506.00         $19,196.00
Paris         Sandal          $23,816.00          $1,520.00
Rome          Boot           $209,271.00         $36,244.00
Rome          Sandal           $4,611.00          $1,249.00
----------                --------------     --------------
Region                     $1,432,646.00        $307,380.00
                          ==============     ==============
                          $12,956,946.00      $3,218,979.00
```

BY Group

Subtotal

Grand Total

23

PAGEBY Statement

The *PAGEBY statement* puts each separate section of a BY group on separate pages.

```
pageby region;
```

The PAGEBY statement must name a variable that appears in the BY statement.

24

PAGEBY Statement

Partial Output

```
                    Boot and Sandal Report

-------------------- Region=Asia -------------------------
```

```
                    Boot and Sandal Report

--------------- Region=Eastern Europe --------------------
```

```
                    Boot and Sandal Report

----------------- Region=Middle East ---------------------

                                    Total           Total
Subsidiary    Product           Inventory           Sales

Al-Khobar     Boot             $44,658.00      $15,062.00
Al-Khobar     Sandal           $13,343.00       $1,380.00
Dubai         Boot            $403,259.00      $90,972.00
Dubai         Sandal           $59,985.00      $17,492.00
Tel Aviv      Boot            $222,165.00      $65,248.00
Tel Aviv      Sandal           $71,094.00      $16,314.00
----------                 --------------  --------------
Region                        $814,504.00     $206,468.00
```

25

BY and PAGEBY Statements

Page 1 ------- Region=United States Subsidiary=Chicago -------
Page 2 ----- Region=United States Subsidiary=New York ------
Page 3 --- Region=United States Subsidiary=Seattle -------

Which statements give the desired layout?

A.
```
by region;
pageby subsidiary;
```

B.
```
by region subsidiary;
pageby region;
```

C.
```
by region subsidiary;
pageby subsidiary;
```

D.
```
by region subsidiary;
pageby region subsidiary;
```

26

ID Statement

The *ID statement* specifies the variable(s) to print at the beginning of each row instead of an observation number.

```
id region;
by region;
```

When used with a BY statement, the ID statement eliminates the BY line and suppresses repetitious printing of the BY variable(s).

Partial Output

Region	Subsidiary	Product	Total Inventory	Total Sales
Western Europe	Copenhagen	Boot	$4,657.00	$1,663.00
	Geneva	Boot	$171,030.00	$41,341.00
	Geneva	Sandal	$3,529.00	$736.00
	Heidelberg	Boot	$301,779.00	$65,610.00
	Heidelberg	Sandal	$4,618.00	$977.00

28

LABEL Statement

The *LABEL statement* assigns descriptive labels to variable names.

```
label inventory='Total*Inventory'
      sales='Total*Sales';
```

Partial Output

```
                     Boot and Sandal Report

------------------------ Region=Asia ------------------------

                                          Total             Total
   Obs   Subsidiary   Product         Inventory             Sales

    57    Bangkok     Boot            $9,576.00         $1,996.00
    59    Bangkok     Sandal         $15,087.00         $3,230.00
    62    Seoul       Boot          $160,589.00        $60,712.00
    65    Seoul       Sandal         $21,483.00         $4,978.00
------                            -------------     -------------
Region                              $206,735.00        $70,916.00
```

A label can be up to 256 characters.

29

LABEL Statement

A LABEL statement must also be accompanied by an option in the PROC PRINT statement in order for the labels to appear.

Which options are used in conjunction with the LABEL statement?

1. APPEAR
2. DESCRIPTION
3. LABEL
4. NOW
5. PRINT
6. SPLIT=

30

LABEL Statement

The LABEL option enforces variables' labels as column headings.

```
proc print data=sashelp.shoes noobs label;
```

The SPLIT= option specifies the split character, which controls line breaks in column headers and implies the use of labels.

```
proc print data=sashelp.shoes noobs split='*';
```

32

FORMAT Statement

The *FORMAT statement* associates formats to variable values.

```
format inventory sales dollar14.2;
```

Partial Output

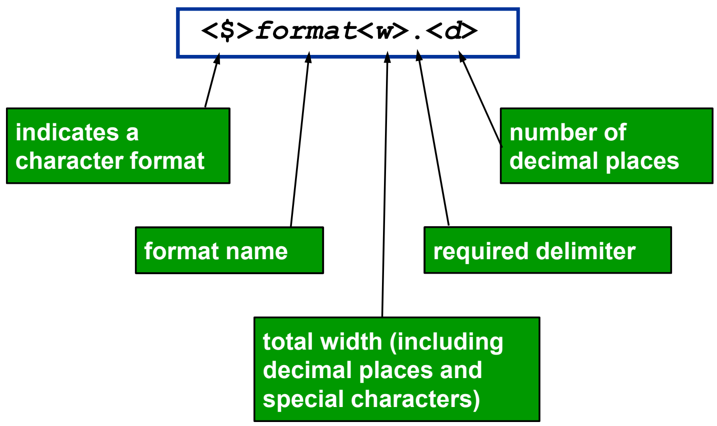
```
------------------------ Region=Asia ------------------------

                                        Total          Total
  Obs   Subsidiary   Product        Inventory          Sales

   57    Bangkok     Boot           $9,576.00      $1,996.00
   59    Bangkok     Sandal        $15,087.00      $3,230.00
   62    Seoul       Boot         $160,589.00     $60,712.00
   65    Seoul       Sandal        $21,483.00      $4,978.00
------                            -------------  -------------
Region                            $206,735.00     $70,916.00
```

A *format* is an instruction that SAS uses to write data values.

33

FORMAT Statement

Formats have the following form:



34

FORMAT Statement

Stored Value	Format	Displayed Value
`Washington`	`$4.`	`Wash`
`1234.4567`	`8.0`	`1234`
`1234.4567`	`8.2`	`1234.46`
`1234.4567`	`comma8.2`	`1,234.46`
`1234.4567`	`dollar9.2`	`$1,234.46`

35

Q

FORMAT Statement

Partial Output

```
-------------------- Region=Western Europe --------------------

                                        Total              Total
   Subsidiary    Product            Inventory              Sales

   Copenhagen    Boot               $4,657.00          $1,663.00
   Geneva        Boot             $171,030.00         $41,341.00
                   dal              $3,529.00            $736.00
                   t              $301,779.00         $65,610.00
                   dal              $4,618.00            $977.00
                   t              $341,911.00         $76,349.00
                   dal             $24,253.00          $1,650.00
                   t              $289,527.00         $54,449.00
                   dal             $11,111.00          $5,217.00
   Madrid        Boot               $1,027.00          $1,179.00
   Paris         Boot              $41,506.00         $19,196.00
   Paris         Sandal            $23,816.00          $1,520.00
   Rome          Boot             $209,271.00         $36,244.00
   Rome          Sandal             $4,611.00          $1,249.00
   ----------                  --------------     --------------
   Region                       $1,432,646.00        $307,380.00
                               ==============     ==============
                               $12,956,946.00      $3,218,979.00
```

What minimum widths are needed to complete the FORMAT statement for this desired output?

```
format inventory dollar___.2 sales dollar___.2;
```

3

Q

FORMAT Statement

The following program is submitted:

```
proc print data=sales;
   format sales dollar7.2;
run;
```

What is the result if the format width for **dollar** is too narrow to represent the value?

A. The program fails execution due to errors.

B. SAS supplies a missing value to the **sales** variable.

C. SAS fits the value into the space available in the best way that it can.

D. The program runs with warnings and eliminates the **sales** variable.

38

FORMAT Statement

Stored Value	Format	Displayed Value
17332	mmddyy6.	061507
17332	mmddyy8.	06/15/07
17332	mmddyy10.	06/15/2007
17332	date7.	15JUN07
17332	date9.	15JUN2007
17332	ddmmyy8.	15/06/07
17332	worddate.	June 15, 2007
17332	weekdate.	Friday, June 15, 2007
17332	monyy7.	JUN2007

40

Q

FORMAT Statement

Desired report:

```
Obs        birth         hired       retired

  1     02/17/1941    05/01/1976    31DEC2006
```

What two corrections must be made to the following statement to obtain the desired report?

```
format birth hired mmddyy8. retired date9;
```

41

LABEL and FORMAT Statements

LABEL and FORMAT statements assigned in a PROC step are considered ***temporary*** attributes (apply only for the duration of the step).

LABEL and FORMAT statements assigned in a DATA step are considered ***permanent*** attributes (stored in the descriptor portion).

```
          Alphabetic List of Variables and Attributes

#  Variable    Type  Len  Format     Informat   Label

6  Inventory   Num     8  DOLLAR12.  DOLLAR12.  Total Inventory
2  Product     Char   14
1  Region      Char   25
7  Returns     Num     8  DOLLAR12.  DOLLAR12.  Total Returns
5  Sales       Num     8  DOLLAR12.  DOLLAR12.  Total Sales
4  Stores      Num     8                        Number of Stores
3  Subsidiary  Char   12
```

43

TITLE Statement

The *TITLE statement* specifies up to 10 lines of text at the top of output.

```
title 'Boot and Sandal Report';
```

Partial Output

```
                       Boot and Sandal Report

   ------------------------ Region=Asia ------------------------

                                          Total             Total
     Subsidiary    Product            Inventory             Sales

      Bangkok      Boot               $9,576.00         $1,996.00
      Bangkok      Sandal            $15,087.00         $3,230.00
      Seoul        Boot             $160,589.00        $60,712.00
      Seoul        Sandal            $21,483.00         $4,978.00
     ----------                  --------------    --------------
      Region                        $206,735.00        $70,916.00
```

TITLE is the same as TITLE1.

44

FOOTNOTE Statement

The *FOOTNOTE statement* specifies up to 10 lines of text at the bottom of output.

```
footnote 'Created by Tony Smith';
footnote2 'Chicago, IL';
```

Partial Output

```
                                       Total            Total
   Subsidiary   Product            Inventory            Sales

    Bangkok     Boot               $9,576.00        $1,996.00
    Bangkok     Sandal            $15,087.00        $3,230.00
    Seoul       Boot             $160,589.00       $60,712.00
    Seoul       Sandal            $21,483.00        $4,978.00
   ----------                 --------------   --------------
    Region                       $206,735.00       $70,916.00

                          Created by Tony Smith
                               Chicago, IL
```

FOOTNOTE is the same as FOOTNOTE1.

45

TITLE and FOOTNOTE Statements

The TITLE and FOOTNOTE statements are global statements, which means that the statements stay in effect until they are canceled or changed, or you end your SAS session.

The code **`title;`** cancels all titles.

The code **`footnote;`** cancels all footnotes.

TITLE*n* or FOOTNOTE*n*

- replaces a previous title or footnote with the same number
- cancels all titles or footnotes with higher numbers.

46

TITLE and FOOTNOTE Statements

Q

The following SAS program is submitted:

```
proc print data=shoes1;
   title1 'Shoe Store';
   title2 'Report One';
   title3 'Accounting';
run;
proc print data=shoes2;
   title2 'Report Two';
run;
```

What titles appear in the second procedure output?

A.
```
Report Two
```

B.
```
Shoe Store
Report Two
```

C.
```
Report Two
Accounting
```

D.
```
Shoe Store
Report Two
Accounting
```

47

OPTIONS Statement

The *OPTIONS statement* changes the value of one or more SAS system options.

```
options nodate nonumber ps=30 ls=64;
```

The OPTIONS statement is a global statement, which means that the options remain in effect until they are canceled or changed, or you end your SAS session.

The OPTIONS statement is not usually included in a step.

Some system options change the appearance of a report.

49

OPTIONS Statement

System Option	Description
DATE | NODATE	Specifies that the date and the time that the SAS job was initialized appear in the top right corner of each page of output.
NODTRESET | DTRESET	Specifies that the current date and the current time appear in the top right corner of each page of output.
NUMBER | NONUMBER	Specifies that the page number appear in the top right corner of each page of output.
PAGENO=n	Specifies a beginning page number for the next page of output.
LS=n LINESIZE=n	Specifies the number of characters that can be printed on one page width of output.
PS=n PAGESIZE=n	Specifies the number of lines that can be printed per page of output.

50

OPTIONS Statement

Q

The following SAS program is submitted.

```
options nodate nonumber;
proc print data=shoes1;
run;
options pageno=1;
proc print data=shoes2;
run;
```

What is the result of the second report?

A. The second report has a date and no page number.
B. The second report has a date and a page number of 1.
C. The second report has no date and no page number.
D. The second report has no date and a page number of 1.

51

Common Statements

The following statements have the same function in a number of Base SAS procedures:

- WHERE
- LABEL
- FORMAT
- TITLE
- FOOTNOTE
- OPTIONS

53

Refer to Exercise 2 for Chapter 6 in Appendix A.

6.2 Creating Formats with the FORMAT Procedure

FORMAT Procedure

The *FORMAT procedure* enables you to define your own formats for variable values.

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
                 other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

Formats determine how variable values are printed.

57

Refer to Exercise 3 for Chapter 6 in Appendix A.

Naming Convention

Format names must be

- 32 characters or less
- different than the name of a format supplied by SAS.

Character formats start with a $, followed by a letter or underscore.

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
                 other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

Numeric formats start with a letter or underscore.

60

Single Values

On the left side of the equal sign, you can have single values. Character values should be enclosed in quotation marks.

case-sensitive

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
                other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

You can have multiple single values with commas separating the values.

```
value $gender 'F','FEM','FEMALE'  = 'Female'
              'M','MAL','MALE'    = 'Male';
```

61

Q

Single Values

A format needs to be created for a numeric variable where 0 means NO, 1 means YES, 3 and 4 means UNSURE, and . means MISSING.

```
proc format;
   value code      '0'   = 'NO'
                   '1'   = 'YES'
               '3','4'   = 'UNSURE'
                    .    = 'MISSING';
run;
```

What is the error in the PROC FORMAT step?

62

Ranges

On the left side of the equal sign, you can have ranges.

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
                other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

For character ranges, each string should be enclosed in quotation marks (example: **'A' – 'Z'**).

64

Ranges

Q

Given the following code:

```
proc format;
   value group  0 - 50  = 'First Half'
               51 - 100 = 'Second Half';
run;
```

What will be the formatted value of 50.5?

A. 50.5

B. First Half

C. Second Half

D. missing value

65

Ranges

The less than (<) sign excludes values from ranges.

- Put < after the value if you exclude the first value in a range.
- Put < before the value if you exclude the last value in a range.

50 - 100	Includes 50	Includes 100
50 - < 100	Includes 50	Excludes 100
50 < - 100	Excludes 50	Includes 100
50 < - < 100	Excludes 50	Excludes 100

67

Ranges

Given the following code:

```
proc format;
   value group  0 - <  50 = 'First Half'
               50 -   100 = 'Second Half';
run;
```

What will be the formatted value of 50.5?

A. 50.5

B. First Half

C. Second Half

D. missing value

68

Keywords

- OTHER matches all values that do not match any other value or range.
- LOW encompasses lowest possible value.
  LOW does not include missing for numeric variables.
  LOW does include missing for character variables.
- HIGH encompasses highest possible value.
- LOW - HIGH encompasses all values.

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
                other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

70

Keywords

What is wrong with the keyword in the following example?

```
proc format;
   value $gender  'F'    = 'Female'
                  'M'    = 'Male'
                'other' = 'Miscoded';
run;
```

71

Formatted Values

On the right side of the equal sign, you have the formatted values.

Formatted values

- are typically quoted strings
- can be up to 32,767 characters.

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
               other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

73

Creating and Using Formats

Create format (no period in format name):

```
proc format;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
               other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
```

Use format (period in format name):

```
options nodate nonumber ps=30 ls=64;
proc print data=sashelp.class;
   var name sex weight;
   format sex $gender. weight wtrange.;
run;
```

74

Creating and Using Formats

Data Set

	name	test
1	Jane	D
2	Tom	B
3	Mark	Z
4	Sue	A

Desired Report

```
Obs    name    test

 1     Jane    Needs Improvement
 2     Tom     Good
 3     Mark    Miscoded
 4     Sue     Excellent
```

```
proc format;
   value $testfmt 'a' = 'Excellent'
                  'b' = 'Good'
                  'c' = 'Average'
              'd','f' = 'Needs Improvement'
                  ' ' = 'Incomplete'
                other = 'Miscoded';
run;

proc print data=test;
   format test $testfmt;
run;
```

What two issues cause this program not to give the desired results?

75

Permanent Formats

By default, user-defined formats are stored in the **work.formats** catalog and exist only for the duration of the SAS session.

```
proc format library = sasuser.myfmts;
   value $gender 'F'   = 'Female'
                 'M'   = 'Male'
               other = 'Miscoded';
run;
```

The LIBRARY= option is used to create permanent formats. LIBRARY= can point to either a library or a catalog. If only a library is specified, then SAS uses the catalog name FORMATS.

77

Format System Options

Q

Which system option specifies the order in which format catalogs are searched?

A. FMTCATALOG
B. FMTERR
C. FMTORDER
D. FMTSEARCH

78

Format System Options

- The FMTERR system option (default) specifies that when SAS cannot find a specified variable format, it generates an error message and stops processing.
- The NOFMTERR replaces missing formats with the *w*. or $*w*. default format and continues processing.

```
options nofmterr
        fmtsearch=(sasuser sasuser.myfmts);
```

- The FMTSEARCH= system option specifies the order in which format catalogs are searched. When a library is specified without a catalog, SAS uses FORMATS as the default catalog name. The default value for FMTSEARCH is (WORK LIBRARY).

80

6.3 Creating Frequency Tables with the FREQ Procedure

FREQ Procedure

The *FREQ procedure* produces one-way to *n*-way frequency tables. By default, the procedure generates one-way frequency tables for all data set variables.

```
proc freq data=sashelp.orsales;
run;
```

One-Way Table

The FREQ Procedure

Product Line

Product_Line	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Children	176	19.30	176	19.30
Clothes & Shoes	288	31.58	464	50.88
Outdoors	112	12.28	576	63.16
Sports	336	36.84	912	100.00

The data set **sashelp.orsales** has eight variables. Therefore, eight one-way frequency tables are created.

82

TABLES Statement

The *TABLES statement* requests one-way to *n*-way frequency tables and statistics for those tables.

Product Line

Product_Line	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Children				
Clothes & Shoes				
Outdoors				
Sports				

```
proc freq data=sashelp.orsales;
   tables product_line year;
run;
```

Year

Year	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1999	228	25.00	228	25.00
2000	228	25.00	456	50.00
2001	228	25.00	684	75.00
2002	228	25.00	912	100.00

83

Two-Way Tables

An asterisk between two variables produces a two-way table.

```
proc freq data=sashelp.orsales;
   where product_line in ('Outdoors','Sports');
   tables product_line * year ;
run;
```

row

column

84

Two-Way Tables

Table of Product_Line by Year

Product_Line(Product Line)
Year(Year)

column

row

Frequency Percent Row Pct Col Pct	1999	2000	2001	2002	Total
Outdoors	28 6.25 25.00 25.00	28 6.25 25.00 25.00	28 6.25 25.00 25.00	28 6.25 25.00 25.00	112 25.00
Sports	84 18.75 25.00 75.00	84 18.75 25.00 75.00	84 18.75 25.00 75.00	84 18.75 25.00 75.00	336 75.00
Total	112 25.00	112 25.00	112 25.00	112 25.00	448 100.00

85

Q

Two-Way Tables

The following program is submitted:

```
proc freq data=sashelp.orsales ;
   tables quarter product_category
          year*product_line;
run;
```

Which of the following is ***true*** regarding the program?

A. One frequency table is produced.
B. Two frequency tables are produced.
C. Three frequency tables are produced.
D. Four frequency tables are produced.

86

CROSSLIST Option

The *CROSSLIST option* displays two-way tables in column format, instead of cell format.

```
proc freq data=sashelp.orsales ;
   where product_line in ('Outdoors','Sports');
   tables product_line*year / crosslist ;
run;
```

Options in the TABLES statement must come after a forward slash.

88

```
                 The FREQ Procedure

             Table of Product_Line by Year

Product_                                  Row    Column
Line        Year   Frequency   Percent   Percent  Percent
---------------------------------------------------------
Outdoors    1999          28      6.25     25.00    25.00
            2000          28      6.25     25.00    25.00
            2001          28      6.25     25.00    25.00
            2002          28      6.25     25.00    25.00

           Total         112     25.00    100.00
---------------------------------------------------------
Sports      1999          84     18.75     25.00    75.00
            2000          84     18.75     25.00    75.00
            2001          84     18.75     25.00    75.00
            2002          84     18.75     25.00    75.00

           Total         336     75.00    100.00
---------------------------------------------------------
Total       1999         112     25.00             100.00
            2000         112     25.00             100.00
            2001         112     25.00             100.00
            2002         112     25.00             100.00

           Total         448    100.00
---------------------------------------------------------
```

89

Statistics

	Default Statistics	Options to Eliminate Statistics
One-Way Tables	Frequency Percent Cumulative Frequency Cumulative Percent	NOPERCENT NOCUM
Two-Way Tables	Frequency Percent Row Percent Column Percent	NOFREQ NOPERCENT NOROW NOCOL

```
tables product_line*year / options;
```

90

Statistics

Q

The following frequency table is desired:

```
                    Product Line

          Product_                    Cumulative
           Line         Frequency      Frequency
          ------------------------------------
          Outdoors           112             112
          Sports             336             448
```

Which statement creates the desired report?

A. `tables product_line;`
B. `tables product_line / nocum;`
C. `tables product_line / nopercent;`
D. `tables product_line / nocum nopercent;`

91

NLEVELS Option

The *NLEVELS option* displays the number of levels for all TABLES variables.

```
proc freq data=sashelp.orsales nlevels;
   tables year quarter
          product_line product_category;
run;
```

```
                  The FREQ Procedure

             Number of Variable Levels

Variable            Label                  Levels
--------------------------------------------------
Year                Year                        4
Quarter             Quarter                    16
Product_Line        Product Line                4
Product_Category    Product Category           12
```

This output will appear before the four one-way frequency tables.

93

Refer to Exercise 4 for Chapter 6 in Appendix A.

6.4 Creating Summary Reports with the MEANS Procedure

MEANS Procedure

The *MEANS procedure* computes descriptive statistics for variables across all observations and within groups of observations.

```
proc means data=sashelp.prdsale
           maxdec=2 mean stddev;
   var predict actual;
   class country year;
run;
```

The MEANS Procedure

Country	Year	N Obs	Variable	Label	Mean	Std Dev
CANADA	1993	240	PREDICT	Predicted Sales	497.20	280.59
			ACTUAL	Actual Sales	504.25	291.83
	1994	240	PREDICT	Predicted Sales	473.71	280.33
			ACTUAL	Actual Sales	524.88	287.08

97

Statistics

Q

Which statistics does the MEANS procedure produce by default?

A. MEAN, STDDEV, SUM
B. MEAN, STDDEV, MIN, MAX
C. N, MEAN, STDDEV, MIN, MAX
D. N, MEAN, STDDEV, SUM, MIN, MAX

98

Statistics

By default, the MEANS procedure creates a report with N (number of nonmissing values), MEAN, STDDEV, MIN, and MAX.

```
proc means data=sashelp.prdsale;
run;
```

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
ACTUAL	Actual Sales	1440	507.1784722	287.0313065	3.0000000	1000.00
PREDICT	Predicted Sales	1440	490.4826389	285.7667904	0	1000.00
QUARTER	Quarter	1440	2.5000000	1.1184224	1.0000000	4.0000000
YEAR	Year	1440	1993.50	0.5001737	1993.00	1994.00
MONTH	Month	1440	12403.00	210.6291578	12054.00	12753.00

100

Statistics

In the PROC MEANS statement, you can specify which statistics to compute and the order in which to display them in the output.

```
proc means data=sashelp.prdsale sum range;
run;
```

The MEANS Procedure

Variable	Label	Sum	Range
ACTUAL	Actual Sales	730337.00	997.0000000
PREDICT	Predicted Sales	706295.00	1000.00
QUARTER	Quarter	3600.00	3.0000000
YEAR	Year	2870640.00	1.0000000
MONTH	Month	17860320.00	699.0000000

101

MAXDEC= Option

The *MAXDEC= option* in the PROC MEANS statement specifies the maximum number of decimal places to display the statistics in the output.

Without MAXDEC=

		N	Mean	Std Dev	Minimum	Maximum
ACTUAL	Actual Sales	1440	507.1784722	287.0313065	3.0000000	1000.00
PREDICT	Predicted Sales	1440	490.4826389	285.7667904	0	1000.00
QUARTER	Quarter	1440	2.5000000	1.1184224	1.0000000	4.0000000
YEAR	Year	1440	1993.50	0.5001737	1993.00	1994.00
MONTH	Month	1440	12403.00	210.6291578	12054.00	12753.00

With MAXDEC=2

		N	Mean	Std Dev	Minimum	Maximum
ACTUAL	Actual Sales	1440	507.18	287.03	3.00	1000.00
PREDICT	Predicted Sales	1440	490.48	285.77	0.00	1000.00
QUARTER	Quarter	1440	2.50	1.12	1.00	4.00
YEAR	Year	1440	1993.50	0.50	1993.00	1994.00
MONTH	Month	1440	12403.00	210.63	12054.00	12753.00

VAR Statement

The *VAR statement* identifies the analysis variables and specifies their order in the results.

```
proc means data=sashelp.prdsale
           maxdec=2 n mean;
   var predict actual;
run;
```

The MEANS Procedure

Variable	Label	N	Mean
PREDICT	Predicted Sales	1440	490.48
ACTUAL	Actual Sales	1440	507.18

The MEANS procedure analyzes all numeric variables if you omit the VAR statement.

103

CLASS Statement

The *CLASS statement* specifies one or more variables that the procedure uses to group the data.

Country	Year	N Obs	Variable	Label	N	Mean
CANADA	1993	240	PREDICT	Predicted Sales	240	497.20
			ACTUAL	Actual Sales	240	504.25
	1994	240	PREDICT	Predicted Sales	240	473.71
			ACTUAL	Actual Sales	240	524.88
GERMANY	1993	240	PREDICT	Predicted Sales	240	488.00
			ACTUAL	Actual Sales	240	530.85
	1994	240	PREDICT	Predicted Sales	240	476.81
			ACTUAL	Actual Sales	240	494.14
U.S.A.	1993	2				
	1994	2				

```
proc means data=sashelp.prdsale
           maxdec=2 n mean;
   var predict actual;
   class country year;
run;
```

104

CLASS Statement

Adding a CLASS statement adds the N Obs column.

Country	Year	N Obs	Variable	Label	N	Mean
CANADA	1993	240	PREDICT	Predicted Sales	240	497.20
			ACTUAL	Actual Sales	240	504.25
	1994	240	PREDICT	Pr		
			ACTUAL	Ac		
GERMANY	1993	240	PREDICT	Pr		
			ACTUAL	Actual Sales	240	530.85
	1994	240				476.81
						494.14
U.S.A.	1993	240				515.68
						504.39
	1994	240	PREDICT	Predicted Sales	240	491.50
			ACTUAL	Actual Sales	240	484.57

number of nonmissing values for the analysis variable

number of observations for each unique combination of the class variables

105

CLASS Statement

For a given data set, there are 20 observations with a **Country** value of GERMANY. Of those 20 observations, only 15 observations have a value for PREDICT.

Which output is correct?

A.

Country	N Obs	Variable	Label	N
GERMANY	15	PREDICT	Predicted Sales	20

B.

Country	N Obs	Variable	Label	N
GERMANY	20	PREDICT	Predicted Sales	15

106

Refer to Exercise 5 for Chapter 6 in Appendix A.

UNIVARIATE Procedure

Another procedure for creating summary reports is the UNIVARIATE procedure.

```
proc univariate data=sashelp.air;
   var air;
run;
```

The UNIVARIATE procedure can display the following sections of output:

- Moments
- Basic Statistical Measures
- Tests for Locations
- Quantiles
- Extreme Observations
- Missing Values

110

UNIVARIATE Procedure

By default, the *Extreme Observations* section includes the five lowest and five highest values for the analysis variable and the corresponding observation numbers.

```
proc univariate data=sashelp.air nextrobs=2;
   var air;
   id date;
run;
```

impacts the Extreme Observations section

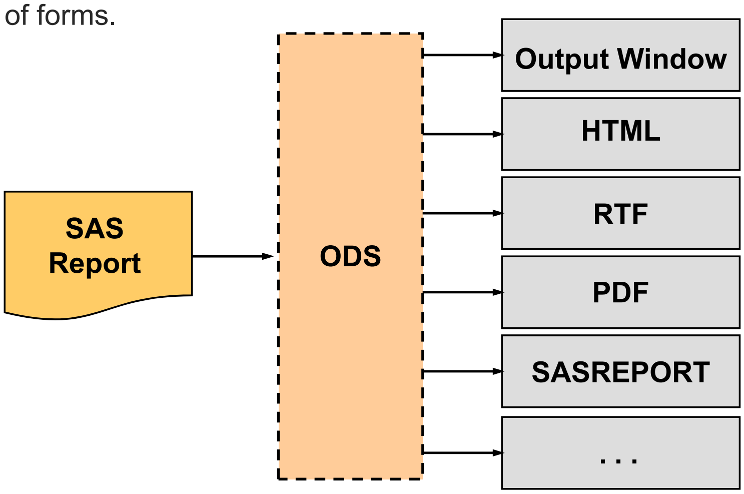
```
                     Extreme Observations

     --------Lowest--------        --------Highest-------

     Value    DATE      Obs        Value    DATE      Obs

       104    NOV49      11          606    AUG60     140
       112    JAN49       1          622    JUL60     139
```

111

6.5 Directing Reports to External Files with ODS

Output Delivery System (ODS)

ODS statements enable you to create output in a variety of forms.



113

Output Delivery System

Q

Which ODS destination is used to direct reports to the Output window?

A. LISTING
B. PRINT
C. OUTPUT
D. WINDOW

114

ODS HTML Statement

The *ODS HTML statement* opens or closes the HTML destination, which produces HyperText Markup Language files that are viewable with a web browser.

```
ods html file='gnp.html' style=default;

proc print data=sashelp.gnp;
   ...
run;

proc means data=sashelp.gnp;
   ...
run;

ods html close;
```

116

ODS HTML Statement

My Report

DATE	govt purchases of goods and services	net exports of goods and services	gross national product ($billions)
1985Q1	784.4	-53.1	3925.6
1985Q2	801.7	-74.3	3979
1985Q3	840.2	-81.2	4047
1985Q4	856.7	-103.2	4107.9

My Report

The MEANS Procedure

Analysis Variable : GNP gross national product ($billions)				
N	Mean	Std Dev	Minimum	Maximum
4	4014.88	79.4621660	3925.60	4107.90

117

ODS HTML Statement

Q

What is wrong with the following program?

```
ods html file='rent.pdf';

proc print data=sashelp.rent;
   title 'Rent Report';

ods close;

run;
```

118

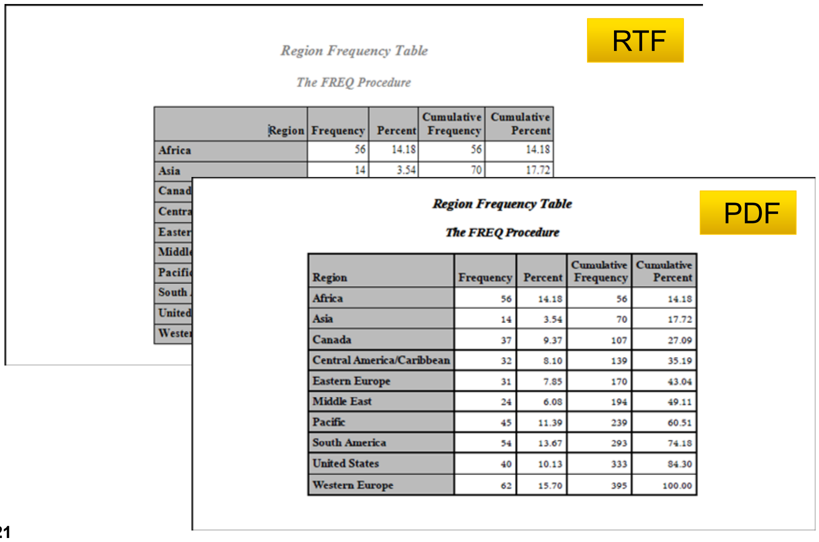
ODS RTF and PDF Statements

- The *ODS RTF statement* opens or closes the RTF destination, which produces Rich Text Format files that are viewable with a word processor.
- The *ODS PDF statement* opens or closes the PDF destination, which produces Portable Document Format files that are viewable with an Adobe product.

```
ods rtf file='shoes.rtf';
ods pdf file='shoes.pdf';

proc freq data=sashelp.shoes;
   ...
run;

ods rtf close;
ods pdf close;
```

120

ODS RTF and PDF Statements



Region Frequency Table

The FREQ Procedure

Region	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Africa	56	14.18	56	14.18
Asia	14	3.54	70	17.72
Canada	37	9.37	107	27.09
Central America/Caribbean	32	8.10	139	35.19
Eastern Europe	31	7.85	170	43.04
Middle East	24	6.08	194	49.11
Pacific	45	11.39	239	60.51
South America	54	13.67	293	74.18
United States	40	10.13	333	84.30
Western Europe	62	15.70	395	100.00

121

Multiple ODS Destinations

Which statement can be used to close multiple ODS destinations?

A. `ods close;`

B. `ods all close;`

C. `ods _all_ close;`

D. `ods _destination_ close;`

122

Multiple ODS Destinations

The ODS _ALL_ CLOSE statement closes ***all*** open destinations including the LISTING destination.

```
ods listing;
ods html file='shoes.html';
ods rtf file='shoes.rtf';
ods pdf file='shoes.pdf';

proc print data=sashelp.shoes;
  ...
run;

ods _all_ close;
ods listing;
```

124

Destinations Used with Excel

Q

Which destination ***cannot*** create a file that can be opened in Excel?

A. CSVALL
B. EXCEL
C. EXCELXP
D. MSOFFICE2K

125

Destinations Used with Excel

- The CSVALL destination creates a CSV (comma-separated value) file.
- The EXCELXP destination creates an XML (Extensible Markup Language) file.
- The MSOFFICE2K destination creates an HTML (HyperText Markup Language) file.

```
ods csvall file='shoes.csv';
ods tagsets.excelxp file='shoes.xml';
ods msoffice2k file='shoes.html';

proc print data=sashelp.shoes;
  ...
run;

ods _all_ close;
```

127

Refer to Exercise 6 for Chapter 6 in Appendix A.

6.6 Answers to Questions

Question Slide Number	Answer
5	2. and 3.
13	A.
16	A.
19	B.
26	C.
30	3. and 6.
36	`format inventory dollar14.2 sales dollar13.2;`
38	C.
41	• The MMDDYY8. format must be the MMDDYY10. format. • A period must be added to the end of the DATE9 format.
47	B.
51	C.
62	`ERROR: The quoted string '0' is not acceptable to a numeric format or informat.`
65	A.
68	C.
71	Keywords do not use quotation marks.
75	• Character values are case sensitive. • Missing period, $testfmt.
78	D.
86	C.
91	C.
98	C.
106	B.
114	A.

(Continued on the next page.)

Question Slide Number	Answer
118	• The extension of the file must be appropriate to the destination. • HTML is missing in the statement to close the file. • The RUN statement must be before the statement that closes the file.
122	C.
125	B.

Chapter 7 Additional Information

7.1 More Specifics about the SAS Base Programming Exam 7-3

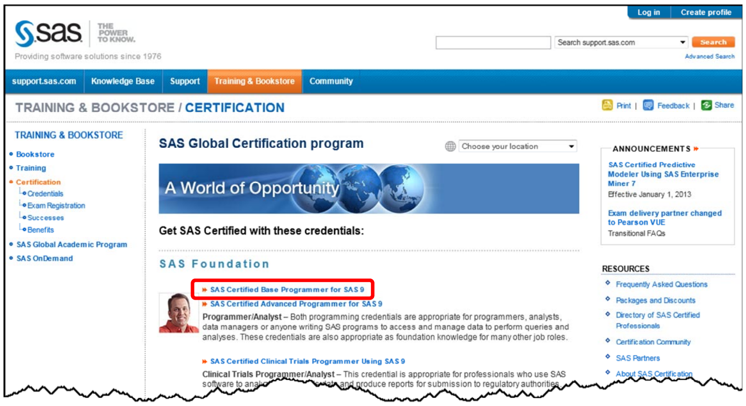
7.2 Additional Preparation Resources 7-6

7.3 Test-Taking Strategies 7-9

7.1 More Specifics about the SAS Base Programming Exam

SAS Certification Website

Remember to check the SAS certification website prior to taking your exam for up-to-date information.



http://support.sas.com/certify

3

SAS Base Programming for SAS®9

After completing the exam, you receive your score.

PASSING SCORE: % YOUR SCORE: 100% GRADE: Pass

Section Analysis	
Accessing Data	**100%**
Creating Data Structures	**100%**
Managing Data	**100%**
Generating Reports	**100%**
Handling Errors	**100%**

The score report will display the percentage of items in each section that you answered correctly for your exam.

These section scores are calculated on a per section basis and cannot be used in determining your total score.

4

GRADE: Pass

If you pass your exam and meet all requirements for this credential, you will receive an e-mail from SAS with instructions providing access to your certificate and logo.

✐ These e-mails are sent to the e-mail address provided at exam registration.

Please **allow at least one week** from your exam date to receive your e-mail.

5

SAS Foundation Credentials and Exams

After you pass the SAS Base Programming Exam, consider taking other certification exams to earn additional SAS Foundation credentials.

Credentials	Certification Exams
✓ SAS Certified Base Programmer for SAS®9	➢ SAS Base Programming for SAS®9
SAS Certified Advanced Programmer for SAS®9	➢ SAS Base Programming for SAS®9 ➢ SAS Advanced Programming for SAS®9
SAS Certified Clinical Trials Programmer Using SAS®9	➢ Clinical Trials Programming Using SAS®9 OR ➢ SAS Base Programming for SAS®9 ➢ Clinical Trials Programming Using SAS®9 – Accelerated Version

6

Recertification

- Recertification is not required for individuals holding versioned certification credentials.
- SAS Certified Base Programmer for SAS®9 is a versioned certification credential.

7

Retaking the Exam

- A candidate can retake an exam five times in a 12-month period.
- A candidate must wait a minimum of 14 calendar days between attempts.
- Exams that do not comply with the retake examination policy will be considered invalid.
- Exam retakes require payment of the full exam fee and are not discounted.

8

7.2 Additional Preparation Resources

Additional Preparation Resources

Taking this review course is not a guarantee that you will pass the exam.

This review course provides practice in multiple-choice and short-answer questions, familiarizes you with SAS terminology, and refreshes your mind on topics learned but not used.

In addition, this course helps you to determine in what areas you need additional preparation and experience.

10

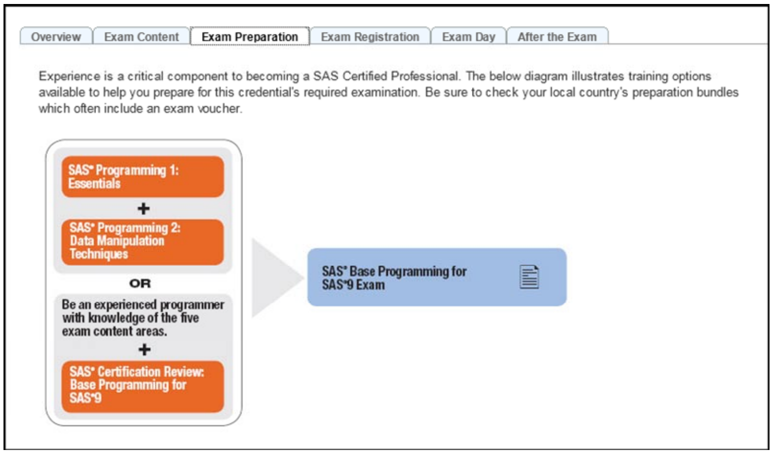
Exam Preparation Options

The following are some of the ways a candidate can prepare for the SAS Base Programming Exam:

- Instructor-led Training
- Self-paced e-learning
- Certification Prep Guide
- Online Documentation

11

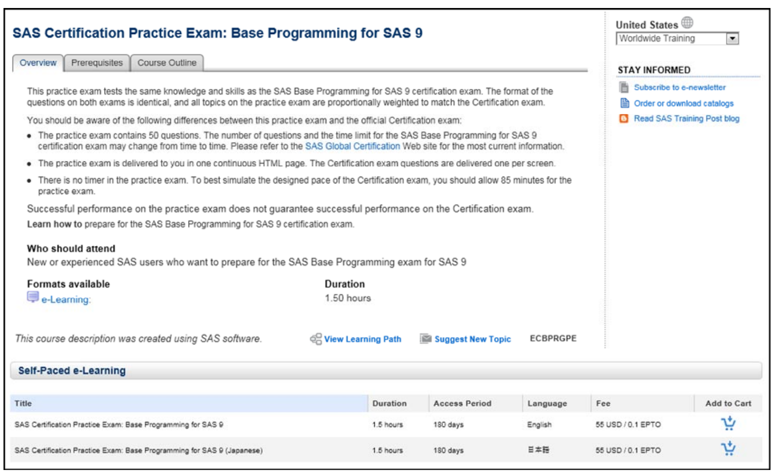
Instructor-led Training



http://support.sas.com/training

12

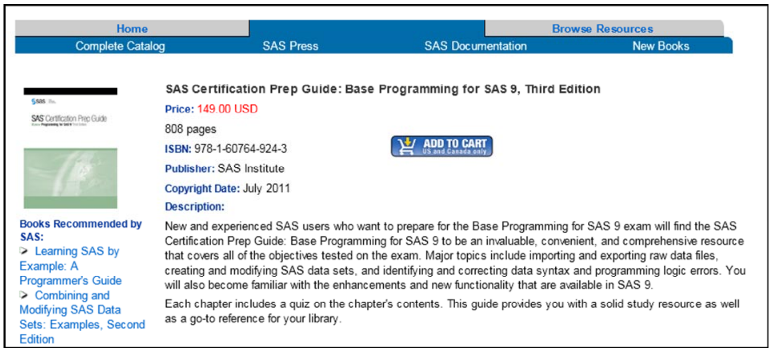
Self-paced e-Learning



http://support.sas.com/training/elearn

13

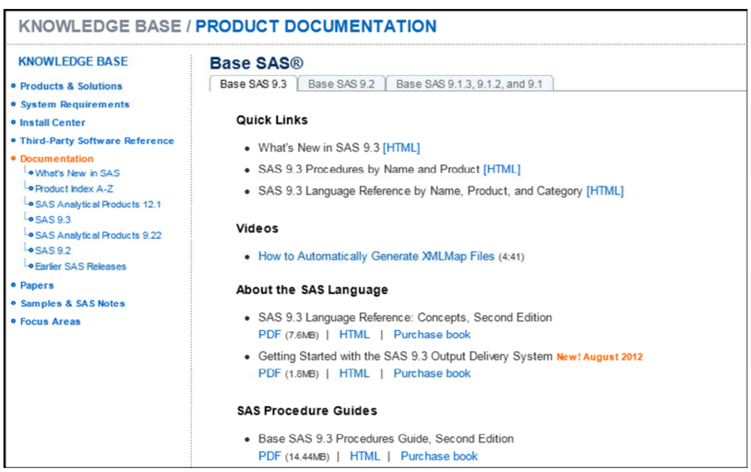
Certification Prep Guide



http://support.sas.com/publishing

14

Online Documentation



http://support.sas.com/documentation

15

Online Documentation

Recommended documentation:

- *SAS® 9.3 Language Reference: Concepts*
- *Step-by-Step Programming with Base SAS® Software*
- *Base SAS® 9.3 Procedures Guide*
- *Base SAS® 9.3 Procedures Guide: Statistical Procedures*
- *SAS® 9.3 Output Delivery System: User's Guide*
- *SAS® 9.3 Data Set Options: Reference*
- *SAS® 9.3 Formats and Informats: Reference*
- *SAS® 9.3 Functions and CALL Routines: Reference*
- *SAS® 9.3 Statements: Reference*
- *SAS® 9.3 System Options: Reference, Second Edition*
- *SAS/ACCESS® 9.3 Interface to PC Files: Reference*

16

7.3 Test-Taking Strategies

Before the Exam

- Extend studying and reviewing sessions over days or weeks.
- Do not cram the night before.
- Practice answering multiple choice SAS questions.
- Practice exam questions under timed conditions.
- Determine a plan for how you will use the allotted exam time.
- Get a full night's sleep before the exam.
- Arrive early to the exam and take a moment to relax.
- Listen attentively to the instructions given by the staff.
- Read the exam directions carefully.

18

During the Exam

- Maintain a positive attitude.
- Read each question carefully and thoroughly.
- Formulate your answer before reading the options.
- Eliminate unlikely options first.
- Be sure to read all options before selecting one.
- Pace yourself so that you have enough time to answer every question.
- Leave no questions unanswered.

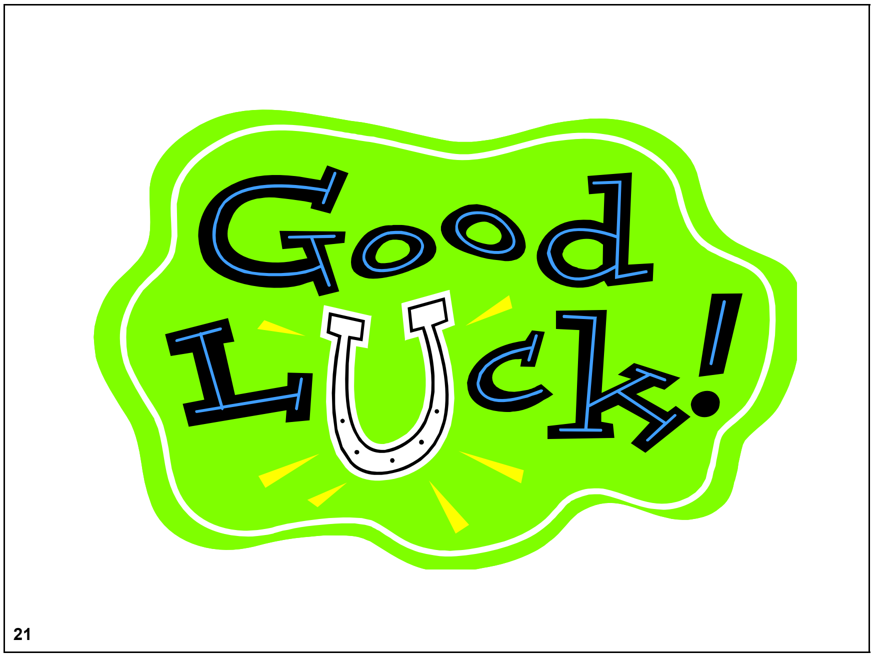
19

continued...

During the Exam

- Rely on your first impression.
- Do not be afraid to change an answer if you feel strongly about it.
- Do not be discouraged if you cannot answer a question.
- Skip questions that you cannot answer, and return to those questions after completing the remainder of the exam.
- Plan to finish early and have time for review.
- Return to difficult questions that you marked for review.

20


Good Luck!
21

Appendix A Exercises and Solutions

A.1 Exercises **A-3**

Chapter 1 .... A-3

Chapter 2 .... A-4

Chapter 3 .... A-7

Chapter 4 .... A-11

Chapter 5 .... A-14

Chapter 6 .... A-18

A.2 Solutions **A-24**

Chapter 1 .... A-24

Chapter 2 .... A-25

Chapter 3 .... A-26

Chapter 4 .... A-27

Chapter 5 .... A-29

Chapter 6 .... A-31

A.1 Exercises

Chapter 1

1. Fundamental Concepts

Answer TRUE or FALSE to the following sentences.

a. ____________ The two types of steps that can make up a SAS program are DATA and PROC.

b. ____________ A DATA step must use a SAS data set as input.

c. ____________ A statement always ends in a colon.

d. ____________ A global statement stays in effect for only the subsequent step.

e. ____________ The LIBNAME statement assigns a logical name to a SAS data library.

f. ____________ Data sets are referenced using a four-level name.

g. ____________ Data sets located in the **Sasuser** data library are considered temporary.

h. ____________ A variable name and the name of a data set can be up to 32 characters long.

i. ____________ By default, a variable name can contain special characters such as a dash (–).

j. ____________ A numeric variable is stored as 32 bytes by default.

k. ____________ A numeric variable can be stored with digits, decimal point, comma, minus sign, and E for scientific notation.

l. ____________ A character variable is stored as 1 to 32,767 bytes.

m. ____________ A SAS date value represents the number of days between January 1, 1960, and a specific date.

n. ____________ A missing numeric value is represented with a zero.

o. ____________ A missing character value is represented with a blank.

p. ____________ The DESCRIPTOR procedure views the descriptor portion of a SAS data set.

q. ____________ A statement that starts with an asterisk is a SAS comment.

r. ____________ The SAS log contains messages starting with the words NOTE, SUGGESTION, and ERROR.

Chapter 2

1. Input and Output Data Sets

The SAS data set **company.sales** has three variables (**product**, **price**, and **quantity**). A new data set **work.sales** must be created. The new data set needs to contain two variables (**product** and **total**). The variable **total** is the result of **price** multiplied by **quantity**.

Complete the following program based on the previous scenario:

```
data ____________________________________________;

   set ____________________________________________;

   keep ____________________________________________;

   total = ____________________________________________;

run;
```

2. Multiple Data Sets

The SAS data set **sashelp.class** has five variables (**name**, **sex**, **age**, **height**, and **weight**) and 19 observations (9 observations with **sex='F'** and 10 observations with **sex='M'**).

Answer the questions based on the previous information and the following program:

```
data work.female(drop=height)
     work.everyone(keep=name weight height);
   set sashelp.class;
   if sex='F' then output work.female;
   output work.everyone;
run;
```

a. What is the input data set? ____________________

b. How many output data sets are being created? ____________________

c. How many observations are in **work.female**? ____________________

d. How many observations are in **work.everyone**? ____________________

e. What variables are in **work.female**? ____________________

3. WHERE and Subsetting IF Statements

Below is a partial view of the **work.sales** data set:

VIEWTABLE: Work.Sales

	DATE	STATE	PRODUCT	actual	predict
575	14184	Florida	DESK	1852	2043
576	14214	Florida	DESK	1211	2146
577	13515	Texas	SOFA	1151	465
578	13546	Texas	SOFA	1630	103

A new data set, **work.subset**, must be created. The new data set should contain only observations with a state equal to Texas, a date less than January 1, 1998, and a difference greater than 1000.

Complete the following program as efficiently as possible based on the previous scenario:

```
data subset;
   set sales;

   where ____________________________________________________________;

   difference=actual-predict;

   if _______________________________________________________________;

run;
```

4. SORT Procedure

Answer the questions based on the following program:

```
proc sort data=sashelp.shoes
          out=shoes;
   by descending region product;
run;
```

a. What is the input data set? ______________________________

b. What is the output data set? ______________________________

c. Where is the output data set stored? ______________________________

d. How many variables are used to sort the data set? ______________________________

e. Does the DESCENDING option apply to the **region** variable? ______________________________

f. Does the DESCENDING option apply to the **product** variable? ______________________________

g. Which variable is considered the primary sort variable? ______________________________

h. What other statements can be added to the SORT procedure? ______________________________

i. Does the SORT procedure create a report? ______________________________

5. DATA Step Merge

Below is the input data set **work.employees**:

VIEWTABLE: Work.Employees

	name	id
1	Troy	12649
2	Melissa	38901
3	Larry	49255
4	Tonya	56391

Below is the input data set **work.salaries**:

VIEWTABLE: Work.Salaries

	idnum	salary
1	12649	52000
2	49255	75000
3	56391	89000
4	88376	66000

Below is the output data set **work.empsal**:

VIEWTABLE: Work.Empsal

	name	idnum	salary
1	Troy	12649	52000
2	Larry	49255	75000
3	Tonya	56391	89000

Add the appropriate statement to the following program to create the output data set based on the input data sets:

```
data empsal;

   ______________________________________________

   ______________________________________________

   by idnum;
   if emp=1 and sal=1;
run;
```

Chapter 3

1. Program Data Vector (PDV)

```
input state $ 1-2
      @5 date mmddyy10. @18 populat comma15.
      @35 city1 $ city2 $;
```

Given the previous INPUT statement, create the PDV that is created at compile time. Include the variable name, the variable type (char or num), and the variable byte size.

Name Type Size							

2. Column, Formatted, and List Input

Below is the raw data file **kids7.dat**:

```
----|----10---|----20---|----30---|----41---|
Tina     Smith    15$1,322!Tim!Tammy
Michael  McMurray 12$532!Mary
Jackie   Jones    14$1,000!Jim!John!Joyce
```

Below is the desired output data set, **work.kids7**:

VIEWTABLE: Work.Kids7

	first	last	age	savings	sibling1	sibling2	sibling3
1	Tina	Smith	15	1322	Tim	Tammy	
2	Michael	McMurray	12	532	Mary		
3	Jackie	Jones	14	1000	Jim	John	Joyce

Complete the following program based on the raw data file **kids7.dat** and the desired output data set **work.kids7**. Use column input for **first**, **last**, and **age**. Use list input for **savings**, **sibling1**, **sibling2**, and **sibling3**.

```
data work.kids7;
   infile 'kids7.dat' ____________________________________;
   input _________________________________________________
         _________________________________________________
         _________________________________________________
         _________________________________________________
         _________________________________________________
         _________________________________________________
         ________________________________________________;
run;
```

3. Definitions Applying to Raw Data Files

Place the appropriate letter before each item.

_____	DLM=	_____	: MODIFIER
_____	DSD	_____	/
_____	MISSOVER	_____	#*N*
_____	INFORMAT		

a. A line-pointer control that advances the pointer to column 1 of the next input record

b. An option that prevents an INPUT statement from reading a new input data record if it does not find values in the current input line for all the variables in the statement

c. An option that specifies a delimiter to be used for LIST input

d. Applies an informat to the field and ignores the width

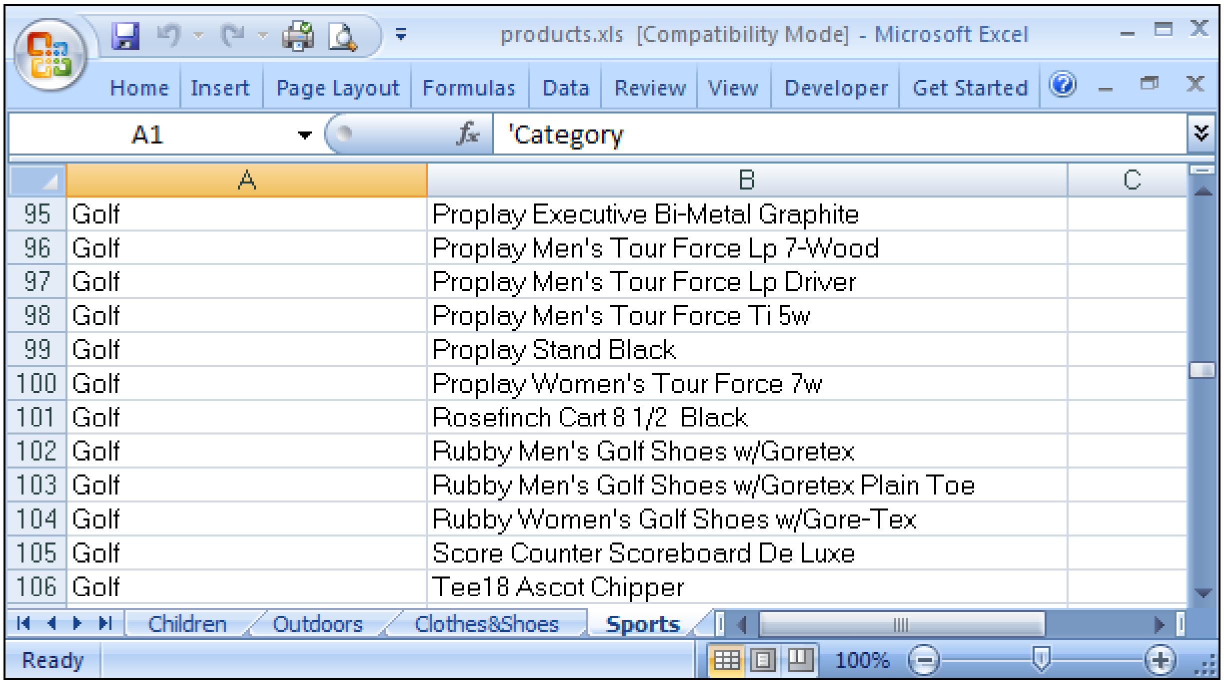
e. An option that treats two consecutive delimiters as a missing value

f. A line-pointer control that advances the pointer to column 1 of record *N*

g. An instruction that SAS uses to read data values into a variable

4. Reading Excel Files

The Excel workbook named **products.xls** contains four worksheets. Each sheet contains two columns: **category** and **name**.



	A	B	C
95	Golf	Proplay Executive Bi-Metal Graphite	
96	Golf	Proplay Men's Tour Force Lp 7-Wood	
97	Golf	Proplay Men's Tour Force Lp Driver	
98	Golf	Proplay Men's Tour Force Ti 5w	
99	Golf	Proplay Stand Black	
100	Golf	Proplay Women's Tour Force 7w	
101	Golf	Rosefinch Cart 8 1/2 Black	
102	Golf	Rubby Men's Golf Shoes w/Goretex	
103	Golf	Rubby Men's Golf Shoes w/Goretex Plain Toe	
104	Golf	Rubby Women's Golf Shoes w/Gore-Tex	
105	Golf	Score Counter Scoreboard De Luxe	
106	Golf	Tee18 Ascot Chipper	

Find the five mistakes in the following program:

```
libname prod 'products'

proc contents data=prod.all;

run;

data work.golf;

   set prod.'sports$';

   where category='Golf';

run;

libname clear;

proc print data=work.golf;

run;
```

Chapter 4

1. Assignment Statements

Answer TRUE or FALSE to the following comments based on the given assignment statement:

```
total = num1 + num2 + num3;
```

______________ The variable **total** is a numeric variable.

______________ The values **num1**, **num2**, and **num3** are constants.

______________ If **num2** is missing, then **total** will be missing.

```
fullname = 'Ms. or Mr.' || name;
```

______________ The variable **fullname** is a numeric variable.

______________ The value 'Ms. or Mr.' is a constant.

______________ The value || is an operator.

```
birthdate = '12FEB1992'd;
```

______________ The variable **birthdate** is a character variable.

______________ The byte size of **birthdate** is 9 bytes.

______________ The value '12FEB1992'd is an operator.

```
phonenumber = '888-999-0000';
```

______________ The variable **phonenumber** is a character variable.

______________ The byte size of **phonenumber** is 8 bytes.

______________ The value '888-999-0000' is an operator.

2. IF-THEN DO / ELSE DO Statements

Answer the questions based on the following program:

```
data newprice;
   infile 'raw-data-file';
   input mfg $ type $ price;
   length saletype $ 18;
   if mfg='Crew' then do;
      pct=0.75;
      saleprice = price * pct;
      saletype = '25% off';
   end;
   else if mfg='Hi-fly' then do;
      pct=0.70;
      saleprice = price * pct;
      saletype = '30% off';
   end;
   else do;
      pct=0.90;
      saleprice = price * pct;
      saletype = '10% Storewide Sale';
   end;
   format price saleprice dollar8.2;
run;
```

a. How many DO blocks are in the program? __________

b. How many variables will be assigned values if an expression is true? __________

c. How many of those variables are numeric? __________

d. What is the byte size of **pct**? __________

e. What would be the byte size of **saletype** if the LENGTH statement were not part of the program?

f. How many ELSE statements are in the program? __________

g. Why is the word ELSE used? __________

h. Why are the DO blocks needed? __________

i. What stops each DO block? __________

j. Will the value of **pct** ever be missing in the data set? __________

3. Multiple BY-Group Variables

Given the following statement:

```
by state city;
```

Fill in the following table with the correct **Total Donation** for each BY group and the **FIRST.** and **LAST.** values:

State	City	Donation	Total Donation	first. State	last. State	first. City	last. City
NC	Charlotte	2000					
NC	Charlotte	9000					
NC	Charlotte	4000					
NC	Greenville	6000					
NC	Greenville	3000					
SC	Greenville	5000					
SC	Greenville	2000					
SC	Pelzer	5000					

The variable **Total Donation** represents a running total within each BY group.

Chapter 5

1. Character Functions

Place the appropriate letter before each function.

_____	CATS	_____	PROPCASE
_____	CATX	_____	RIGHT
_____	COMPBL	_____	SCAN
_____	COMPRESS	_____	STRIP
_____	FIND	_____	SUBSTR
_____	LEFT	_____	TRANWRD
_____	LENGTH	_____	TRIM
_____	LOWCASE	_____	UPCASE

a. Concatenates character strings and removes leading and trailing blanks

b. Selects a given word from a character expression

c. Replaces or removes all occurrences of a word in a character string

d. Converts all letters in an argument to lowercase

e. Right-aligns a character expression

f. Removes multiple blanks from a character string

g. Searches a character expression for a string of characters with the capability of ignoring case and trimming trailing blanks

h. Extracts a substring from an argument

i. Converts all letters in an argument to uppercase

j. Returns a character string with specified characters removed from the original string

k. Removes trailing blanks from character expressions

l. Returns an integer that represents the position of the rightmost non-blank character in a string

m. Left-aligns a character expression

n. Concatenates character strings, removes leading and trailing blanks, and inserts separators

o. Converts all words in an argument to proper case

p. Removes leading and trailing blanks from character expressions

2. FIND Function

The following program is submitted:

```
data tonguetwister;
   length string $ 37;
   string='How much WOOD would a woodchuck chuck';
   num1=find(string,'wood');
   num2=find(string,'wood','i');
   num3=find(string,'wood ','t');
   num4=find(string,'wood   ','it');
   num5=find(string,'WOOD ','it',15);
   num6=find(string,'WOOD ','it',-15);
   num7=find(string,'wood',40);
   num8=find(string,'WOOD','i',-40);
run;
```

Fill in the following table with the correct value of **num1** through **num8**:

num1	num2	num3	num4	num5	num6	num7	num8

3. Truncation Functions

Fill in the following table based on the value of **num** and the truncation function:

num	ceil(num)	floor(num)	int(num)	round(num)
2.75				
-2.75				
23.1234				
-23.1234				

4. Automatic Data Conversions

Fill in the following tables with the converted value assuming automatic conversion:

Value of Character Variable	Value of Numeric Variable (8 bytes)
`162400`	
`$162,400`	
`49275.937`	
`+24`	
`-73.5`	
`01234`	
`52E3`	
`01/01/1960`	

Automatic character-to-numeric conversion using the *W.* informat

Value of Numeric Variable (8 bytes)	Value of Character Variable (12 bytes)
`162400`	
`49275.937`	
`-73.5`	
`52E3`	
`0`	

Automatic numeric-to-character conversion using the BEST12. format

5. DO Loops and Arrays

The following is the input data set, **Weekly**:

VIEWTABLE: Work.Weekly

	name	week1	week2	week3	week4
1	Jack	25	32	48	33
2	Susan	10	12	10	10

The following is the desired output data set, **WeeklyRotate**:

VIEWTABLE: Work.Weeklyrotate

	name	week	miles
1	Jack	1	25
2	Jack	2	32
3	Jack	3	48
4	Jack	4	33
5	Susan	1	10
6	Susan	2	12
7	Susan	3	10
8	Susan	4	10

Complete the following program to create the desired output data set:

```
data WeeklyRotate;
   set Weekly;

   array run ______________________________________________________;

   do _____________________________________________________________;

      miles = _____________________________________________________;

      output;

   ________________________________________________________________;

   drop week1 - week4;
run;
```

Chapter 6

1. PRINT Procedure

Place the appropriate letter before each item.

_____	BY Statement	_____	PAGEBY Statement
_____	FOOTNOTE Statement	_____	SUM Statement
_____	FORMAT Statement	_____	TITLE Statement
_____	LABEL Statement	_____	VAR Statement
_____	NOOBS Option	_____	WHERE Statement
_____	OPTIONS Statement		

a. Puts each separate section of a BY group on separate pages

b. Changes the value of one or more SAS system options

c. Suppresses the column in the output that identifies each observation by number

d. Specifies up to 10 lines of text at the top of output

e. Subsets the input data set by specifying certain conditions that each observation must meet

f. Assigns descriptive labels to variable names

g. Produces a separate section of the report for each BY group

h. Selects variables that appear in the report and determines the variables order

i. Specifies up to 10 lines of text at the bottom of output

j. Associates formats to variable values

k. Totals values of numeric variables

2. PRINT Procedure

The following is the desired report (partial output):

```
                     Predicted versus Actual Sales

         ----------------------- Country=Mexico -----------------------

                                       Predicted
              Country      Year            Sales    Actual Sales

              Mexico       1995             $282            $189
              Mexico       1995             $276            $266
              Mexico       1995             $807            $241
              Mexico       1995             $782            $451
              Mexico       1995             $381            $126
```

Find the six syntax mistakes in the following program:

```
options nodate numberno ps=30 ls=64;
proc print data=sashelp.prdsal2 nobs headers;
   where country=Mexico;
   var country year predict actual;
   by country;
   sum predict actual;
   label predict='Predicted Sales'
         actual='Actual Sales';
   format predict actual dollar12;
   title 'Predicted versus Actual Sales'
run;
```

3. FORMAT Procedure

Answer the questions based on the following program:

```
proc format;
   value $gender 'F'  = 'Female'
                 'M'  = 'Male'
                other = 'Miscoded';
   value wtrange low - <100 = 'Under 100 lbs'
                 100 - high = '100+ lbs';
run;
proc print data=sashelp.class;
   var name sex weight;
   format sex $gender. weight wtrange.;
run;
```

a. How many formats are created in the PROC FORMAT step? ______________________________

b. Is $GENDER a character or numeric format? ______________________________

c. Is WTRANGE a character or numeric format? ______________________________

d. What is the maximum length of a format name? ______________________________

e. What must start a character format name? ______________________________

f. What are the three keywords used in the VALUE statements? ______________________________

g. Does the less than (<) sign include or exclude values from ranges? ______________________________

h. Does the LOW keyword include a numeric missing value? ______________________________

i. What special character is used at the end of the format name when you use a format? __________

j. Are these formats temporary or permanent? ______________________________

k. Does the PROC FORMAT step refer to the data set and the variable that will use the format?

____

4. FREQ Procedure

The following is the desired report:

```
                              Report One

                          The FREQ Procedure

                      Number of Variable Levels

            Variable            Label                  Levels
            ------------------------------------------------
            Product_Category    Product Category           12


                           Product Category

           Product_Category             Frequency     Percent
           -------------------------------------------------
           Assorted Sports Articles            64        7.02
           Children Sports                    176       19.30
           Clothes                            240       26.32
           Golf                                32        3.51
           Indoor Sports                       48        5.26
           Outdoors                           112       12.28
           Racket Sports                       48        5.26
           Running - Jogging                   32        3.51
           Shoes                               48        5.26
           Swim Sports                         16        1.75
           Team Sports                         64        7.02
           Winter Sports                       32        3.51
```

Complete the following program to create the desired report:

```
options nodate nonumber ps=50 ls=74;

proc freq data=sashelp.orsales _____________________________;

   ___________________________________________________________;

   title 'Report One';
run;
```

5. MEANS Procedure

The following is the desired report:

```
                         Product Sales Report

                         The MEANS Procedure

              Analysis Variable : ACTUAL Actual Retail Sales

       State/Province            N Obs             Sum          Median
       ________________________________________________________________

       Illinois                   5760          410977              71

       New York                   1152         1705493            1445
       ________________________________________________________________


                             Confidential
```

Complete the following program to create the desired report:

```
options nodate nonumber ps=18 ls=74;

proc means data=sashelp.prdsal2 ___________________________________;

   where state in ('New York','Illinois');

   _________________________________________________________________;

   _________________________________________________________________;

   label actual='Actual Retail Sales';
   title 'Product Sales Report';
   footnote 'Confidential';
run;
```

6. Output Delivery System

Fill in the blanks in the following program to complete the program and answer the questions:

```
ods listing;

ods html _______________ = 'steel.html';

ods _______________ file = 'steel.xml';

proc sort data=sashelp.steel out=steel;
   by date;
run;

proc print data=steel;
run;

proc freq data=steel;
   tables date;
run;

ods _all_ _______________;

ods listing;
```

a. How many destinations are open in the program? ________________________________

b. How many reports are sent to the open destinations? ________________________________

A.2 Solutions

Chapter 1

1. Fundamental Concepts

- **a.** TRUE The two types of steps that can make up a SAS program are DATA and PROC.
- **b.** FALSE A DATA step must use a SAS data set as input.
- **c.** FALSE A statement always ends in a colon.
- **d.** FALSE A global statement stays in effect for only the subsequent step.
- **e.** TRUE The LIBNAME statement assigns a logical name to a SAS data library.
- **f.** FALSE Data sets are referenced using a four-level name.
- **g.** FALSE Data sets located in the **Sasuser** data library are considered temporary.
- **h.** TRUE A variable name and the name of a data set can be up to 32 characters long.
- **i.** FALSE By default, a variable name can contain special characters such as a dash (–).
- **j.** FALSE A numeric variable is stored as 32 bytes by default.
- **k.** FALSE A numeric variable can be stored with digits, decimal point, comma, minus sign, and E for scientific notation.
- **l.** TRUE A character variable is stored as 1 to 32,767 bytes.
- **m.** TRUE A SAS date value represents the number of days between January 1, 1960 and a specific date.
- **n.** FALSE A missing numeric value is represented with a zero.
- **o.** TRUE A missing character value is represented with a blank.
- **p.** FALSE The DESCRIPTOR procedure views the descriptor portion of a SAS data set.
- **q.** TRUE A statement that starts with an asterisk is a SAS comment.
- **r.** FALSE The SAS log contains messages starting with the words NOTE, SUGGESTION, and ERROR.

Chapter 2

1. Input and Output Data Sets

```
data work.sales;
   set company.sales;
   keep product total;
   total = price*quantity;
run;
```

2. Multiple Data Sets

a. What is the input data set? **sashelp.class**

b. How many output data sets are being created? two

c. How many observations are in **work.female**? nine

d. How many observations are in **work.everyone**? 19

e. What variables are in **work.female**? **name**, **sex**, **age**, and **weight**

3. WHERE and Subsetting IF Statements

```
data subset;
   set sales;
   where state='Texas' and date<'01JAN1998'd;
   difference=actual-predict;
   if difference>1000;
run;
```

4. SORT Procedure

a. What is the input data set? **sashelp.shoes**

b. What is the output data set? **shoes**

c. Where is the output data set stored? **work**

d. How many variables are used to sort the data set? two

e. Does the DESCENDING option apply to the **region** variable? yes

f. Does the DESCENDING option apply to the **product** variable? no

g. Which variable is considered the primary sort variable? **region**

h. What other statements can be added to the SORT procedure? FORMAT, LABEL, and WHERE

i. Does the SORT procedure create a report? no

5. DATA Step Merge

```
data empsal;
   merge employees(in=emp rename=(id=idnum))
         salaries(in=sal);
   by idnum;
   if emp=1 and sal=1;
run;
```

Chapter 3

1. Program Data Vector (PDV)

Name	state	date	populat	city1	city2	_ERROR_	_N_
Type	char	num	num	char	char	num	num
Size	2	8	8	8	8	8	8

2. Column, Formatted, and List Input

```
data work.kids7;
   infile 'kids7.dat' dlm='!' missover;
   input first $ 1-9
         last $ 10-19
         age 20-21
         savings : comma8.
         sibling1 $
         sibling2 $
         sibling3 $;
run;
```

3. Definitions Applying to Raw Data Files

c. DLM=

d. : MODIFIER

e. DSD

a. /

b. MISSOVER

f. *#N*

g. INFORMAT

4. Reading Excel Files

```
libname prod 'products.xls';

proc contents data=prod._all_;
run;

data work.golf;
   set prod.'sports$'n;
   where category='Golf';
run;

libname prod clear;

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

Chapter 4

1. Assignment Statements

```
total = num1 + num2 + num3;
```

TRUE The variable **total** is a numeric variable.

FALSE The values **num1**, **num2**, and **num3** are constants.

TRUE If **num2** is missing, then **total** will be missing.

```
fullname = 'Ms. or Mr.' || name;
```

FALSE The variable **fullname** is a numeric variable.

TRUE The value 'Ms. or Mr.' is a constant.

TRUE The value || is an operator.

```
birthdate = '12FEB1992'd;
```

FALSE The variable **birthdate** is a character variable.

FALSE The byte size of **birthdate** is 9 bytes.

FALSE The value '12FEB1992'd is an operator.

```
phonenumber = '888-999-0000';
```

TRUE The variable **phonenumber** is a character variable.

FALSE The byte size of **phonenumber** is 8 bytes.

FALSE The value '888-999-0000' is an operator.

2. IF-THEN DO / ELSE DO Statements

a. How many DO blocks are in the program? three

b. How many variables will be assigned values if an expression is true? three

c. How many of those variables are numeric? two

d. What is the byte size of **pct**? eight

e. What would be the byte size of **saletype** if the LENGTH statement were not part of the program? seven

f. How many ELSE statements are in the program? two

g. Why is the word ELSE used? Subsequent ELSE statements are not evaluated after a true statement.

h. Why are the DO blocks needed? executing three statements, not only one statement

i. What stops each DO block? **end;**

j. Will the value of **pct** ever be missing in the data set? no

3. Multiple BY-Group Variables

State	City	Donation	Total Donation	first. State	last. State	first. City	last. City
NC	Charlotte	2000	15000	1	0	1	0
NC	Charlotte	9000		0	0	0	0
NC	Charlotte	4000		0	0	0	1
NC	Greenville	6000	9000	0	0	1	0
NC	Greenville	3000		0	1	0	1
SC	Greenville	5000	7000	1	0	1	0
SC	Greenville	2000		0	0	0	1
SC	Pelzer	5000	5000	0	1	1	1

Chapter 5

1. Character Functions

a.	CATS	**o.**	PROPCASE
n.	CATX	**e.**	RIGHT
f.	COMPBL	**b.**	SCAN
j.	COMPRESS	**p.**	STRIP
g.	FIND	**h.**	SUBSTR
m.	LEFT	**c.**	TRANWRD
l.	LENGTH	**k.**	TRIM
d.	LOWCASE	**i.**	UPCASE

2. FIND Function

num1	num2	num3	num4	num5	num6	num7	num8
23	10	23	10	23	10	0	23

3. Truncation Functions

num	ceil(num)	floor(num)	int(num)	round(num)
2.75	3	2	2	3
-2.75	-2	-3	-2	-3
23.1234	24	23	23	23
-23.1234	-23	-24	-23	-23

4. Automatic Data Conversions

Value of Character Variable
162400
$162,400
49275.937
+24
-73.5
01234
52E3
01/01/1960

Automatic character-to-numeric conversion using the W. informat

Value of Numeric Variable (8 bytes)
162400
.
49275.937
24
-73.5
1234
52000
.

Value of Numeric Variable (8 bytes)
162400
49275.937
-73.5
52E3
0

Automatic numeric-to-character conversion using the BEST12. format

Value of Character Variable (12 bytes)
162400
49275.937
-73.5
52000
0

5. DO Loops and Arrays

```
data WeeklyRotate;
   set Weekly;
   array run{4} week1 - week4;
   do week = 1 to 4;
      miles = run{week};
      output;
   end;
   drop week1 - week4;
run;
```

Chapter 6

1. PRINT Procedure

g.	BY Statement	**a.**	PAGEBY Statement
i.	FOOTNOTE Statement	**k.**	SUM Statement
j.	FORMAT Statement	**d.**	TITLE Statement
f.	LABEL Statement	**h.**	VAR Statement
c.	NOOBS Option	**e.**	WHERE Statement
b.	OPTIONS Statement		

2. PRINT Procedure

```
options nodate nonumber ps=30 ls=64;
proc print data=sashelp.prdsal2 noobs label;
   where country='Mexico';
   var country year predict actual;
   by country;
   sum predict actual;
   label predict='Predicted Sales'
         actual='Actual Sales';
   format predict actual dollar12.;
   title 'Predicted versus Actual Sales';
run;
```

3. FORMAT Procedure

a. How many formats are created in the PROC FORMAT step? two ($GENDER and WTRANGE)

b. Is $GENDER a character or a numeric format? character

c. Is WTRANGE a character or a numeric format? numeric

d. What is the maximum length of a format name? 32

e. What must start a character format name? dollar sign ($)

f. What are the three keywords used in the VALUE statements? OTHER, LOW, and HIGH

g. Does the less than (<) sign include or exclude values from ranges? exclude

h. Does the LOW keyword include a numeric missing value? no

i. What special character is used at the end of the format name when you use a format? period (.)

j. Are these formats temporary or permanent? temporary

k. Does the PROC FORMAT step refer to the data set and the variable that will use the format? no

4. FREQ Procedure

```
options nodate nonumber ps=50 ls=74;
proc freq data=sashelp.orsales nlevels;
   tables product_category / nocum;
   title 'Report One';
run;
```

5. MEANS Procedure

```
options nodate nonumber ps=18 ls=74;
proc means data=sashelp.prdsal2 maxdec=0 sum median;
   where state in ('New York','Illinois');
   var actual;
   class state;
   label actual='Actual Retail Sales';
   title 'Product Sales Report';
   footnote 'Confidential';
run;
```

6. Output Delivery System

```
ods listing;
ods html file = 'steel.html';
ods tagsets.excelxp file = 'steel.xml';

proc sort data=sashelp.steel out=steel;
   by date;
run;

proc print data=steel;
run;

proc freq data=steel;
   tables date;
run;

ods _all_ close;
ods listing;
```

BODY= can be used in place of FILE=.

a. How many destinations are open in the program? three (LISTING, HTML, and EXCELXP)

b. How many reports are sent to the open destinations? two (PROC PRINT and PROC FREQ)

Appendix B Practice Exam

B.1 SAS Fundamental Concepts B-3

B.2 Working with SAS Data Sets B-6

B.3 Working with Raw Data and Microsoft Excel Files B-12

B.4 Creating Variables B-20

B.5 Manipulating Data B-26

B.6 Generating Reports B-31

B.7 Scores B-39

B.8 Answers B-40

B.1 SAS Fundamental Concepts

Answer the **5 questions** in this section in **9 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in Section 8 of this appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
SAS Fundamentals Concepts	5		

Percent Correct

5 correct out of 5 = 100%

4 correct out of 5 = 80%

3 correct out of 5 = 60%

2 correct out of 5 = 40%

1 correct out of 5 = 20%

1. Which of the following LIBNAME statements has the correct syntax?

   a. **`libname monthly 'c:\monthly';`**

   b. **`libname 'c:\monthly' monthly;`**

   c. **`libname monthly='c:\monthly';`**

   d. **`libname 'c:\monthly'=monthly;`**

2. Which of the following statements is true?

   a. A libref can be 32 or fewer characters.

   b. A variable name can be 32 or fewer characters.

   c. Numeric variables are stored as 32 bytes by default.

   d. Character variables are stored as 32 bytes by default.

3. A program was submitted and the SAS log is shown below.

```
169  data work.sales;
170    set sashelp.orsales;
171    drop quarter year;
172  run;
NOTE: There were 912 observations read from the data set SASHELP.ORSALES.
NOTE: The data set WORK.SALES has 912 observations and 6 variables.
173  /*
174  proc contents data=work.sales;
175  run;
176  */
177  proc print data=work.sales;
178  run;
NOTE: There were 912 observations read from the data set WORK.SALES.
```

   Which of the following is true regarding the CONTENTS procedure?

   a. The PROC CONTENTS step failed execution.

   b. The PROC CONTENTS step did not execute.

   c. The second note applies to the PROC CONTENTS step.

   d. The last note applies to the PROC CONTENTS step and the PROC PRINT step.

4. The following program is submitted:

```
proc contents data=temp.sales;
run;
```

   Which is the result?

   a. a report showing the data portion of the temporary data set **temp.sales**
   b. a report showing the data portion of the permanent data set **temp.sales**
   c. a report showing the descriptor portion of the temporary data set **temp.sales**
   d. a report showing the descriptor portion of the permanent data set **temp.sales**

5. Which of the following statements is true concerning SAS date values?

   a. The SAS date value for 05JAN1960 is 4.
   b. The SAS date value for 03/24/1952 is a positive number.
   c. A SAS date value represents the number of days between January 1, 1950, and a specified date.
   d. The SAS date value for September 16, 1999, can be written as the SAS date constant '09/16/1999'd.

B.2 Working with SAS Data Sets

Answer the **9 questions** in this section in **15 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in Section 8 of this appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
Working with SAS Data Sets	9		

Percent Correct

9 correct out of 9 = 100%

8 correct out of 9 = 89%

7 correct out of 9 = 78%

6 correct out of 9 = 67%

5 correct out of 9 = 56%

4 correct out of 9 = 44%

3 correct out of 9 = 33%

2 correct out of 9 = 22%

1 correct out of 9 = 11%

1. The following program is submitted:

```
data work.newsales;
   set work.sales;
   sales=price*quantity;
   <insert statement here>
run;
```

   Which SAS statement will output observations with **product** equal to the character value Shorts and **sales** less than one million?

   a. **`if product eq Shorts and sales<1000000;`**

   b. **`if product='Shorts' and sales lt 1000000;`**

   c. **`where product=Shorts and sales lt 1000000;`**

   d. **`where product eq 'Shorts' and sales<1000000;`**

2. Which of the following is true regarding the SORT procedure?

   a. The SORT procedure requires the BY statement.

   b. The SORT procedure has the ability to create a report or a new data set.

   c. The SORT procedure can only sort values based on character variables.

   d. The SORT procedure sorts values by descending order unless the ASCENDING option is specified.

3. Which items are potentially created at compile time of a DATA step?

   a. input buffer, data values, and report

   b. raw data file, program data vector, and report

   c. raw data file, data values, and descriptor information

   d. input buffer, program data vector, and descriptor information

4. Given the input SAS data set **salary1**:

IDNUM	SALARY
12649	52000
49255	75000

Given the input SAS data set **salary2**:

ID	SALARY
56391	89000
88376	66000

The following program is submitted:

```
data salaryall;
   <insert statement here>
run;
```

Given the desired output SAS data set **salaryall**:

ID	SALARY
12649	52000
49255	75000
56391	89000
88376	66000

Which statement will produce the desired output data set?

a. `set salary2 salary1(rename id=idnum);`

b. `set salary1 rename=(idnum=id) salary2;`

c. `set salary1(rename=(idnum=id)) salary2;`

d. `set salary1 salary2(rename=(id=idnum));`

5. The following program is submitted:

```
data work.firsthalf work.thirdqtr work.misc;
   set sashelp.retail;
   if 1<=month<=6 then output work.firsthalf;
   else if 7<=month<=9 then output work.thirdqtr;
run;
```

Which of the following statements is true regarding the previous program with an observation having **month** equal to 12?

a. The observation will be output to the **work.firsthalf** data set.

b. The observation will be output to the **work.thirdqtr** data set.

c. The observation will be output to the **work.misc** data set.

d. The observation will not be output to any data set.

6. The following SAS program is submitted:

```
proc sort data=sashelp.class new=sortdata;
   by name descending age;
run;
```

What is the result?

a. The program fails execution due to a syntax error with the NEW= option.

b. The program fails execution due to a syntax error with the DESCENDING option.

c. The program runs without errors and creates a new data set **work.sortdata** with the sorted observations.

d. The program runs with warnings and overwrites the original data set **sashelp.class** with the sorted observations.

7. Given the input data set **products**:

```
CODE        PRODUCT
__________________
A123        Sandal
A234        Slipper
B345        Boot
B456        Sneaker
```

Given the input data set **costs**:

```
CODE         COST
__________________
A123        19.99
A234         9.99
B456        25.99
```

The following program is submitted:

```
data prodcost;
   merge products(in=p) costs(in=c);
   by code;
   if p and c;
run;
```

Which is the result?

a. The program fails execution because of invalid IN= syntax.

b. The program fails execution because the subsetting IF statement is incomplete.

c. The program runs without errors or warnings and produces a data set with three observations and three variables.

d. The program runs without errors or warnings and produces a data set with four observations and three variables.

8. The following program is submitted:

```
data work.orsales;
   set sashelp.orsales(firstobs=500 obs=700);
run;
```

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

Enter your numeric answer.

9. The following program is submitted:

```
data work.sales;
   set sashelp.orsales;
   drop quarter year;
run;
```

Which of the following statements is true regarding the previous program?

a. The variables **quarter** and **year** will not be in **work.sales**.

b. The output data set will contain the variables **_N_** and **_ERROR_**.

c. The data set **work.sales** is the input data set and **sashelp.orsales** is the output data set.

d. The output data set will have no observations because there is no OUTPUT statement.

B.3 Working with Raw Data and Microsoft Excel Files

Answer the **9 questions** in this section in **15 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in section 8 of this Appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
Working with Raw Data and Excel Files	9		

Percent Correct

9 correct out of 9 = 100%

8 correct out of 9 = 89%

7 correct out of 9 = 78%

6 correct out of 9 = 67%

5 correct out of 9 = 56%

4 correct out of 9 = 44%

3 correct out of 9 = 33%

2 correct out of 9 = 22%

1 correct out of 9 = 11%

1. Given the raw data file **address.dat**:

```
Sue      Smith
123 Main Street
San Diego   CA 92625
Julie    Brown
456 Monroe Road
Birmingham  AL 35235
```

The following SAS program is submitted:

```
data work.address;
   infile 'address.dat';
   input first $ 1-9 last $ 10-14 / street $ 1-15 /;
run;
```

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

Enter your numeric answer.

2. Given the raw data file **2005pop.dat**:

```
----|----10---|----20---|----30
 8,143,197 1 New York     NY
 3,844,829 2 Los Angeles  CA
 2,842,518 3 Chicago      IL
```

The following SAS program is submitted:

```
data work.population;
   infile '2005pop.dat';
   input  @2 POPULATION comma9.
         @12 RANK 1.
         @14 CITY $12.
         @27 STATE $2;
run;
```

Which is the output data set **work.population**?

a.

POPULATION	RANK	CITY	STATE
8,143,197	1	New York	NY
3,844,829	2	Los Angeles	CA
2,842,518	3	Chicago	IL

b.

POPULATION	RANK	CITY	STATE
8,143,197		New York	NY
3,844,829		Los Angeles	CA
2,842,518		Chicago	IL

c.

POPULATION	RANK	CITY	STATE
8143197	1	New York	8
3844829	2	Los Angeles	3
2842518	3	Chicago	2

d.

POPULATION	RANK	CITY	STATE
8143197	1	New York	NY
3844829	2	Los Angeles	CA
2842518	3	Chicago	IL

3. The Excel workbook **customers.xls** contains a worksheet named **Females** and a worksheet named **Males**.

   Which program will read the **Males** worksheet to create a SAS data set?

   a.

```
libname customer excel 'customers.xls';

data work.males;
   set customer.males;
run;

libname customer clear;
```

   b.

```
libname customer excel 'customers.xls';

data work.males;
   set customer.males.worksheet;
run;

libname customer clear;
```

   c.

```
libname customer excel 'customers.xls';

data work.males;
   set customer.males$;
run;

libname customer clear;
```

   d.

```
libname customer excel 'customers.xls';

data work.males;
   set customer.'males$'n;
run;

libname customer clear;
```

4. Which of the following is true regarding the DSD option when reading raw data files?

   a. The DSD option sets the delimiter to a blank.

   b. The DSD option treats two consecutive delimiters as a missing value.

   c. The DSD option belongs in the INPUT statement after a forward slash.

   d. The DSD option removes any delimiters located inside a set of quotation marks.

5. Given the raw data file **info.dat**:

```
John Louisville KY
Lorna Columbia MO 65203
```

   The following SAS program is submitted:

```
data info;
   infile 'info.dat';
   input name $ city $ state $ zipcode $;
run;
```

   Which is the result?

   a. The program runs without errors or warnings and produces a data set with two observations and four variables.

   b. The program produces a warning of invalid data for **zipcode** in line 1 and produces a data set with two observations and four variables.

   c. The program produces a note that SAS went to a new line when the INPUT statement reached past the end of a line and produces a data set with one observation and four variables.

   d. The program produces an error that SAS went to a new line when the INPUT statement reached past the end of a line and produces an error that SAS stopped processing the step because of errors.

6. Given the raw data file **revenue.dat**:

```
Jan $13,000 above
Feb $900 below
Mar $27,000 above
```

The following SAS program is submitted:

```
data work.target;
   infile 'revenue.dat';
   <insert statement here>
run;
```

Given the desired output data set **work.target**:

MONTH	REVENUE	TARGET
Jan	13000	above
Feb	900	below
Mar	27000	above

Which statement will produce the desired output data set?

a. **`input MONTH $ REVENUE TARGET $;`**

b. **`input MONTH $ REVENUE $ TARGET $;`**

c. **`input MONTH $ REVENUE:dollar8. TARGET $;`**

d. **`input MONTH $ REVENUE dollar7. TARGET $;`**

7. A DATA step was submitted and a portion of the SAS log is shown below.

```
601  data work.population;
602    infile '2000pop.dat';
603    input population 1-7
604          rank 9
605          state 24-25
606          city $ 11-21;
607  run;

NOTE: Invalid data for state in line 1 24-25.
RULE:     ----+----1----+----2----+----3----+----4----+----5----
1         8008278 1 New York     NZ 25
population=8008278 rank=1 state=. city=New York _ERROR_=1 _N_=1
NOTE: Invalid data for state in line 2 24-25.
2         3694820 2 Los Angeles  CA 25
population=3694820 rank=2 state=. city=Los Angeles _ERROR_=1
_N_=2
NOTE: Invalid data for state in line 3 24-25.
3         2896016 3 Chicago      IL 25
population=2896016 rank=3 state=. city=Chicago _ERROR_=1 _N_=3
```

What is the cause of the notes about invalid data?

a. NZ is not a valid **state** value.

b. The variable **state** is not numeric data.

c. The variable **state** is not located before **CITY**.

d. The variable **state** is not located in positions 24 and 25.

8. Which LIBNAME statement will access the Excel workbook **products.xl**s, which contains the worksheet **Children**?

a. `libname myexcel 'products.xls';`

b. `libname myexcel children 'products.xls';`

c. `libname myexcel 'products.xls' sheet='children';`

d. `libname myexcel workbook='products.xls' worksheet='children';`

9. Given the raw data file **1990pop.dat**:

```
New York!7,322,564!Los Angeles!3,485,398
Chicago!2,783,726!Houston!1,630,553
```

The following SAS program is submitted:

```
data work.population;
   length CITY $ 12;
   <insert statements here>
run;
```

Given the desired output data set **work.population**:

CITY	POPULATION
New York	7322564
Los Angeles	3485398
Chicago	2783726
Houston	1630553

Which statements will produce the desired output data set?

a. `infile '1990pop.dat';`
   `input CITY POPULATION:comma9. dlm='!' @;`

b. `infile '1990pop.dat';`
   `input CITY $ POPULATION:comma9. dlm='!' @@;`

c. `infile '1990pop.dat' dlm='!';`
   `input CITY $ POPULATION:comma9. @;`

d. `infile '1990pop.dat' dlm='!';`
   `input CITY POPULATION:comma9. @@;`

B.4 Creating Variables

Answer the **9 questions** in this section in **15 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in Section 8 of this appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
Creating Variables	9		

Percent Correct

9 correct out of 9 = 100%

8 correct out of 9 = 89%

7 correct out of 9 = 78%

6 correct out of 9 = 67%

5 correct out of 9 = 56%

4 correct out of 9 = 44%

3 correct out of 9 = 33%

2 correct out of 9 = 22%

1 correct out of 9 = 11%

1. The following SAS program is submitted:

```
data work.class;
   set sashelp.class(keep=name age);
   if age>=13 then group='Teen';
   if 11<=age<=13 then group='Pre-Teen';
run;
```

What is the value of **group** in the data set **work.class** if an observation has a value of **age** equal to 13?

a. missing

b. Teen

c. Pre-

d. Pre-Teen

2. A DATA step was submitted and a portion of the SAS log is shown below.

```
263  data new;
264    newvar = THIS IS A TEST;
                    --
                    388
                    76
ERROR 388-185: Expecting an arithmetic operator.

ERROR 76-322: Syntax error, statement will be ignored.

265  run;

NOTE: The SAS System stopped processing this step because of
      errors.
WARNING: The data set WORK.NEW may be incomplete.  When this
         step was stopped there were 0 observations and 1
         variables.
```

Which of the following actions resolves the error message?

a. Put quotation marks around THIS IS A TEST.

b. Put parentheses around THIS IS A TEST.

c. Add commas between the words THIS IS A TEST.

d. Add a FORMAT statement declaring **newvar** as character.

3. Which statement must be added to the DATA step in order for SAS to create the temporary **FIRST.** and **LAST.** variables?

   Enter the keyword of the statement. Do not add leading or trailing spaces to your answer.

4. The following program is submitted:

```
data work.total;
   n1 = 4;
   n2 = .;
   n3 = 10;
   n4 = n1 + n2 + n3;
run;
```

   What is the resulting value of **n4** in the data set **work.total**?

   a. .

   b. 14

   c. 4 + . + 10

   d. n1 + n2 + n3

5. Given the SAS data set **Work.Employees**:

```
Name
____
Jeff
Dawn
Mary
Gene
```

The following SAS program is submitted:

```
data Work.EmployeeCount;
   set Work.Employees;
   Count=Count+1;
run;
proc print data=Work.EmployeeCount noobs;
run;
```

What is the result of the PRINT procedure?

a.

```
Name        Count

Jeff          .
Dawn          .
Mary          .
Gene          .
```

b.

```
Name        Count

Jeff          0
Dawn          0
Mary          0
Gene          0
```

c.

```
Name        Count

Jeff          1
Dawn          1
Mary          1
Gene          1
```

d.

```
Name        Count

Jeff          1
Dawn          2
Mary          3
Gene          4
```

6. The following program is submitted:

```
data personnel;
   hired='01MAR2003'd;
   name='William Smith';
run;
```

Which of the following is true regarding the variables created with the assignment statements?

a. The variables **hired** and **name** are 8 bytes.

b. The variables **hired** and **name** are character.

c. The variable **hired** is 9 bytes and **name** is 13 bytes.

d. The variable **hired** is numeric and **name** is character.

7. Given the SAS data set **birth**:

NAME	STATE
Tim	CA
Sue	IN
Bill	NY

The following SAS program is submitted:

```
data birthregion;
   set birth;
   if state='CA' then do;
      region='West';
   end;
   else if state='NY' then do;
      region='East';
run;
```

What is the result?

a. The program fails execution because of invalid DO block syntax.

b. The program fails execution because there is not a DO block for the **state** value of IN.

c. The program runs without errors or warnings and produces a data set with two observations and three variables.

d. The program runs without errors or warnings and produces a data set with three observations and three variables.

8. Which of the following is true regarding the sum statement?

   a. The sum statement can be used only for variables being read in from a SET statement.

   b. The sum statement initializes the variable to missing before the first iteration of the DATA step.

   c. The sum statement automatically retains the variable value without using a RETAIN statement.

   d. The sum statement produces an error if a missing value is added to the accumulator variable.

9. Which of the following is valid syntax for SELECT and WHEN statements?

   a.

```
select(salary);
   when <100000 status='Non-Exec';
   when >=100000 status='Exec';
end;
```

   b.

```
select(salary);
   when(<100000) status='Non-Exec';
   when(>=100000) status='Exec';
end;
```

   c.

```
select;
   when salary<100000 status='Non-Exec';
   when salary>=100000 status='Exec';
end;
```

   d.

```
select;
   when(salary<100000) status='Non-Exec';
   when(salary>=100000) status='Exec';
end;
```

B.5 Manipulating Data

Answer the **9 questions** in this section in **15 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in Section 8 of this appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
Manipulating Data	9		

Percent Correct

9 correct out of 9 = 100%

8 correct out of 9 = 89%

7 correct out of 9 = 78%

6 correct out of 9 = 67%

5 correct out of 9 = 56%

4 correct out of 9 = 44%

3 correct out of 9 = 33%

2 correct out of 9 = 22%

1 correct out of 9 = 11%

1. Which statement has correct syntax for a DO statement?

   a. **`do year = 2000 to 2005 while(amount < 1000000);`**

   b. **`do year = 2000 to 2005, while(amount < 1000000);`**

   c. **`do year = 2000 to 2005 or while(amount < 1000000);`**

   d. **`do year = 2000 to 2005 and while(amount < 1000000);`**

2. A DATA step was submitted and a portion of the SAS log is shown below.

```
345  data numdates;
346    set chardates;
347    newhired=input(hired, date9);
                             -----
                             85
                             76
ERROR 85-322: Expecting a format name.

ERROR 76-322: Syntax error, statement will be ignored.

348  run;

NOTE: The SAS System stopped processing this step because of
      errors.
```

   Which of the following actions resolves the error message?

   a. Put quotation marks around **`date9`**.

   b. Put **`date9`** before **`hired`**.

   c. Put a period at the end of **`date9`**.

   d. Delete the comma before **`date9`**.

3. The following SAS program is submitted:

```
data orders;
   set product.orders;
   <insert statement here>
run;
```

   Which SAS statement returns observations containing the text GOLF, regardless of the case, in the variable **`product_name`**?

   a. **`if find(product_name,'golf','t')>0;`**

   b. **`if find(product_name,'golf','t')=0;`**

   c. **`if find(product_name,'golf','i')=0;`**

   d. **`if find(product_name,'golf','i')>0;`**

4. The following SAS program is submitted:

```
data salesgoals(drop=i);
   set sales(keep=jan feb mar);
   array sales{3} jan feb mar;
   array goals{3} _temporary_ (130,170,170);
   array diff{3};
   <insert DO loop here>
run;
proc print data=salesgoals noobs;
run;
```

The following report is generated:

```
jan    feb    mar    diff1    diff2    diff3

120    185    160     -10       15      -10
115    160    180     -15      -10       10
```

Which DO loop created the report?

a.

```
do i=1 to 3;
   diff{i}=sales{i}-goals{i};
end;
```

b.

```
do i=1 to 3;
   sales{i}=goals{i}-diff{i};
end;
```

c.

```
do array=1 to 3;
   sales=goals-diff;
end;
```

d.

```
do array=1 to 3;
   diff{array}=sales{array}-goals{array};
end;
```

5. Which of the following is true regarding the SCAN function?

   a. The SCAN function uses only two default delimiters (the blank and the comma) if a delimiter is not specified.

   b. The SCAN function has an optional fourth argument, which is the direction (forward or backward) to read the string.

   c. The SCAN function returns a missing value if the number of the word scanned is greater than the number of words in the character string.

   d. If the SCAN function returns a value to a variable that was not yet assigned a length, the variable length is determined by the length of the first argument.

6. The following SAS program is submitted:

```
data investment;
   do year=1 to 5;
   invest+1000;
      do month=1 to 12 by 3;
         invest+50;
         output;
      end;
   end;
run;
```

   How many observations are in the data set **investment**?

   Enter your numeric answer.

7. Which assignment statement will produce a value for **FULLNAME** with a comma between **LASTNAME** and **FIRSTNAME**?

   a. `FULLNAME = CATS(',', LASTNAME, FIRSTNAME);`

   b. `FULLNAME = CATX(',', LASTNAME, FIRSTNAME);`

   c. `FULLNAME = CATS(LASTNAME, FIRSTNAME, ',');`

   d. `FULLNAME = CATX(LASTNAME, FIRSTNAME, ',');`

8. A character array that contains three variables (**name1**, **name2**, and **name3**) with the values of Smith, Jones, and Westinghouse is requested.

   Which ARRAY statement will create the desired array?

   a. **`array name{3} $ ('Smith','Jones','Westinghouse');`**

   b. **`array name(3) $ 12 ('Smith','Jones','Westinghouse');`**

   c. **`array name1-name3 $ ('Smith' 'Jones' 'Westinghouse');`**

   d. **`array name{3} name1-name3 ('Smith' 'Jones' 'Westinghouse');`**

9. The following SAS program is submitted:

```
data personnel;
   Phone = 6667778888;
   <insert statement here>
run;
proc print data=personnel;
run;
```

   The following report is generated:

```
                         Area
     Obs        Phone    Code

       1   6667778888    666
```

   Which SAS statement created the report?

   a. **`AreaCode=substr(Phone,1,3);`**

   b. **`AreaCode=substr(Phone,2,3);`**

   c. **`AreaCode=substr(put(Phone,10.),1,3);`**

   d. **`AreaCode=substr(input(Phone,10.),1,3);`**

B.6 Generating Reports

Answer the **9 questions** in this section in **15 minutes** or less. Do not use your course notes.

After completing the questions, compare your answers with the correct answers in Section 8 of this appendix.

Complete the following table after you determine the number of questions that you answered correctly:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
Generating Reports	9		

Percent Correct

9 correct out of 9 = 100%

8 correct out of 9 = 89%

7 correct out of 9 = 78%

6 correct out of 9 = 67%

5 correct out of 9 = 56%

4 correct out of 9 = 44%

3 correct out of 9 = 33%

2 correct out of 9 = 22%

1 correct out of 9 = 11%

1. The following SAS program is submitted:

```
proc freq data=sashelp.shoes;
   where region contains 'East' and
         product in ('Sandal','Slipper');
   <insert statement here>
   title 'Sandal and Slipper Report';
run;
```

The following report is generated:

```
                          Sandal and Slipper Report

                             The FREQ Procedure

                          Table of Region by Product

                                                          Row    Column
Region          Product  Frequency   Percent  Percent  Percent
───────────────────────────────────────────────────────────────
Eastern Europe  Sandal           3     23.08    42.86    50.00
                Slipper          4     30.77    57.14    57.14

                Total            7     53.85   100.00
---------------------------------------------------------------
Middle East     Sandal           3     23.08    50.00    50.00
                Slipper          3     23.08    50.00    42.86

                Total            6     46.15   100.00
---------------------------------------------------------------
Total           Sandal           6     46.15            100.00
                Slipper          7     53.85            100.00

                Total           13    100.00
```

Which SAS statement created the report?

a. **`tables region product;`**

b. **`tables region product / list;`**

c. **`tables region*product / list;`**

d. **`tables region*product / crosslist;`**

2. Given the following SAS program:

```
options nodate nonumber;
ods listing;

proc univariate data=sashelp.shoes;
   var Sales Inventory;
   label Sales='Sales Total'
         Inventory='Number of Stores';
   title 'Univariate Report';
run;
```

How many of the statements are global statements?

Enter your numeric answer.

3. Given the SAS data set **marriage**:

```
Name      Date
________________
Thomas    15413
Susan     17000
Marsha    16529
```

The following SAS program is submitted:

```
proc print data=marriage;
   <insert statement here>
   format date mmddyy10.;
run;
```

The following report is generated:

```
 Obs    name          date

   2    Susan    07/18/2006
```

Which SAS statement created the report subsetting for **date** greater than or equal to January 1, 2006?

a. `where date >= 01/01/2006;`

b. `where date >= '01jan2006';`

c. `where date >= '01jan2006'd;`

d. `where date >= '01/01/2006't;`

4. Given the following program:

```
ods html file='report.html';
proc print data=retail;
run;
proc freq data=retail;
run;
<insert ODS statement here>
```

Which ODS statement must end the HTML file?

a. **ods end;**

b. **ods close;**

c. **ods html end;**

d. **ods html close;**

5. The following SAS program is submitted:

```
proc freq data=sashelp.cars;
   tables cylinders;
run;
```

The following report is generated:

```
                                          10:00 Thursday, November 12, 2009  29

                            The FREQ Procedure

                                                Cumulative    Cumulative
          Cylinders    Frequency     Percent     Frequency      Percent
          ---------------------------------------------------------------
                  3            1        0.23             1         0.23
                  4          136       31.92           137        32.16
                  5            7        1.64           144        33.80
                  6          190       44.60           334        78.40
                  8           87       20.42           421        98.83
                 10            2        0.47           423        99.30
                 12            3        0.70           426       100.00

                          Frequency Missing = 2
```

The following report is desired:

```
                                                                              1

                            The FREQ Procedure

                                                Cumulative    Cumulative
          Cylinders    Frequency     Percent     Frequency      Percent
          ---------------------------------------------------------------
                  3            1        0.23             1         0.23
                  4          136       31.92           137        32.16
                  5            7        1.64           144        33.80
                  6          190       44.60           334        78.40
                  8           87       20.42           421        98.83
                 10            2        0.47           423        99.30
                 12            3        0.70           426       100.00

                          Frequency Missing = 2
```

Which OPTIONS statement will produce the desired report?

a. `options nodate page=1;`

b. `options nodate pageno=1;`

c. `options notime nodate number=1;`

d. `options notime nodate pagenumber=1;`

6. The following SAS program is submitted:

```
proc print data=account;
   <insert statement here>
run;
```

The following list report containing the character variable **name** and the numeric variables **date** and **amount** is generated:

```
Obs    name        date         amount

 1     JACK    16MAY2007    $123,830.23
 2     SUE     17JUL2006     $89,654.05
```

Which SAS statement created the report?

a. **format date date9. amount dollar11.2;**

b. **format date ddmmyy9. amount dollar10.2;**

c. **format date ddmmmyyyy9. amount comma11.2;**

d. **format date date7. amount dollarcomma10.2;**

7. The following SAS program is submitted:

```
proc means data=sashelp.orsales maxdec=0;
   class product_line;
   var total_retail_price / sum mean;
run;
```

What is the result?

a. The program runs without errors or warnings and produces a summary table.

b. The program fails execution because of invalid options in the VAR statement.

c. The program fails execution because of an invalid option in the PROC statement.

d. The program runs with warnings due to the order of the CLASS and VAR statements.

8. Given the SAS data set **dept** with two character variables:

```
Department      Answer
________________________
Accounting      0
IT              1
Marketing       0
Sales           5
```

The following SAS program is submitted:

```
proc format;
   value $yesno    '0'='No'
                   '1'='Yes'
               'other'='Unknown';
run;

proc print data=dept noobs;
   format answer $yesno.;
run;
```

Which report is created?

a.

```
Department  Answer

Accounting  0
IT          1
Marketing   0
Sales       5
```

b.

```
Department  Answer

Accounting  No
IT          Yes
Marketing   No
Sales       Unknown
```

c.

```
Department  Answer

Accounting  No
IT          Yes
Marketing   No
Sales       5
```

d.

```
Department  Answer

Accounting  Unknown
IT          Unknown
Marketing   Unknown
Sales       Unknown
```

9. Which program will create a file that can be opened in Microsoft Excel?

a.

```
ods rtf file='report.rtf';
proc freq data=sashelp.shoes;
   tables region;
run;
ods rtf close;
```

b.

```
ods excel file='report.xls';
proc freq data=sashelp.shoes;
   tables region;
run;
ods excel close;
```

c.

```
ods excelxp file='report.xml';
proc freq data=sashelp.shoes;
   tables region;
run;
ods excelxp close;
```

d.

```
ods csvall file='report.csv';
proc freq data=sashelp.shoes;
   tables region;
run;
ods csvall close;
```

B.7 Scores

Complete the following table as you complete the sections:

Section	Number of Questions	Numbers of Questions Answered Correctly	Percent Correct
SAS Fundamentals Concepts	5		
Working with SAS Data Sets	9		
Working with Raw Data and Excel Files	9		
Creating Variables	9		
Manipulating Data	9		
Generating Reports	9		
Total	50		

<u>Percent Correct</u>

45-50 correct out of 50 = 90-100%

40-44 correct out of 50 = 80-89%

35-39 correct out of 50 = 70-79%

33-34 correct out of 50 = 65-69%

30-32 correct out of 50 = 60-64%

25-29 correct out of 50 = 50-59%

20-24 correct out of 50 = 40-49%

15-19 correct out of 50 = 30-39%

B.8 Answers

Question Number	B.1	B.2	B.3	B.4	B.5	B.6
1	A.	B.	2	C.	A.	D.
2	B.	A.	C.	A.	C.	3
3	B.	D.	D.	BY	D.	C.
4	D.	C.	B.	A.	A.	D.
5	A.	D.	C.	A.	C.	B.
6		A.	C.	D.	20	A.
7		C.	B.	A.	B.	B.
8		201	A.	C.	B	C.
9		A.	D.	D.	C.	D.