Chapter 5: Reading SAS Data Sets

Objectives

- Define the concept of reading from a data source to create a SAS data set.
- Define the business scenario that will be used when reading from a SAS data set, an Excel worksheet, and a raw data file.

Business Scenario

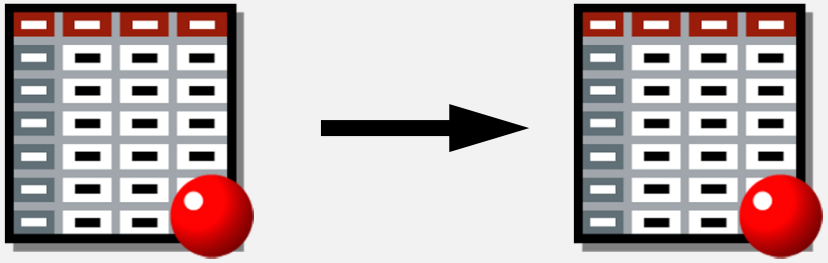
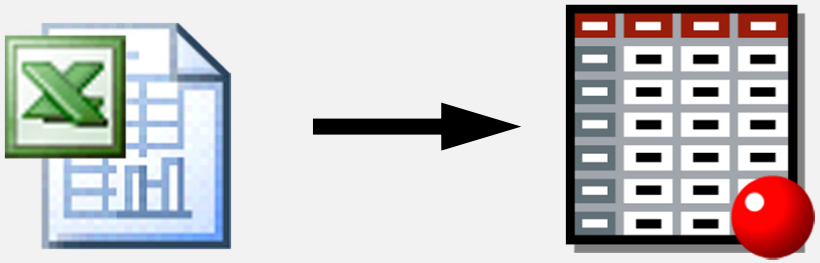
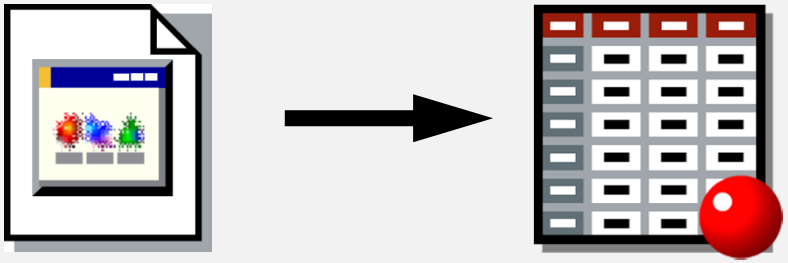
An existing data source contains information on Orion Star sales employees from Australia and the United States.

A new SAS data set needs to be created that contains a subset of this existing data source.

This new SAS data set must contain the following:

- only the employees from Australia who are Sales Representatives
- the employee's first name, last name, salary, job title, and hired date
- labels and formats in the descriptor portion

Business Scenario

Reading SAS Data Sets	
Reading Excel Worksheets	
Reading Delimited Raw Data Files	

Business Scenario

Reading SAS Data Sets	`libname _____;`
	`data _____;`
	`  set _____;`
	`  ...`
	`run;`

Reading Excel Worksheets	`libname _____;`
	`data _____;`
	`  set _____;`
	`  ...`
	`run;`

Reading Delimited Raw Data Files	`data _____;`
	`  infile _____;`
	`  input _____;`
	`  ...`
	`run;`

5.01 Multiple Answer Poll

Which types of files will you read into SAS?

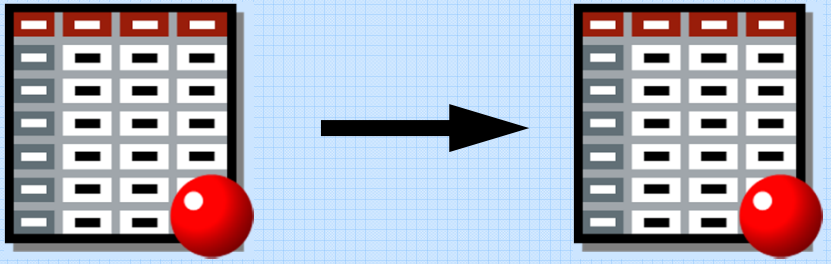
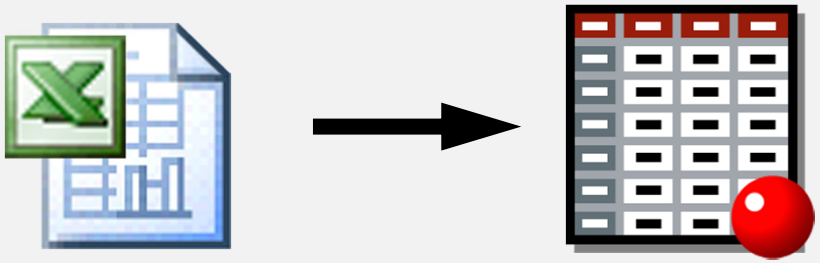
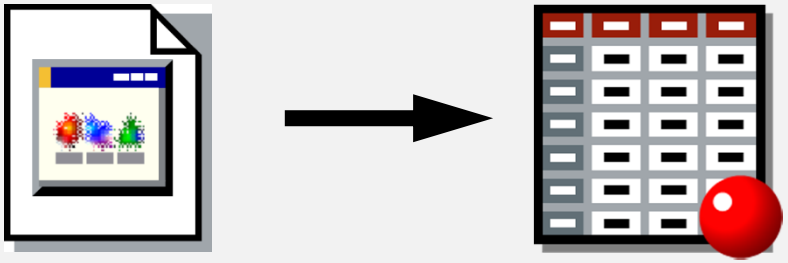
a. SAS data sets
b. Excel worksheets
c. raw data files
d. other
e. not sure

Chapter 5: Reading SAS Data Sets

Objectives

- Use the DATA step to create a SAS data set from an existing SAS data set.

Business Scenario

Reading SAS Data Sets	
Reading Excel Worksheets	
Reading Delimited Raw Data Files	

Business Scenario

Reading SAS Data Sets	`libname _____;`
	`data _____;`
	`  set _____;`
	`  ...`
	`run;`

Reading Excel Worksheets	`libname _____;`
	`data _____;`
	`  set _____;`
	`  ...`
	`run;`

Reading Delimited Raw Data Files	`data _____;`
	`  infile _____;`
	`  input _____;`
	`  ...`
	`run;`

Business Scenario Syntax

Use the following statements to complete the scenario:

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
                variable = 'label'
                variable = 'label';
      FORMAT variable(s) format;
RUN;
```


continued...

Business Scenario Syntax

Use the following statements to complete the scenario:

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Part 1

Part 2

Part 3

The LIBNAME Statement (Review)

A library reference name (libref) is needed if a permanent data set is being read or created.

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      <additional SAS statements>
RUN;
```

The *LIBNAME statement* assigns a libref to a SAS data library.

The DATA Statement

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

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      <additional SAS statements>
RUN;
```

The DATA statement can create temporary or permanent data sets.

The SET Statement

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

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      <additional SAS statements>
RUN;
```

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

Business Scenario Part 1

Create a temporary SAS data set named **`Work.subset1`** from the permanent SAS data set named **`orion.sales`**.

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

data work.subset1;
   set orion.sales;
run;
```

Partial SAS Log

```
9    data work.subset1;
10      set orion.sales;
11   run;

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.SUBSET1 has 165 observations and 9 variables.
```

Both data sets contain 165 observations and 9 variables

p105d01

Business Scenario Part 1

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

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_ Date
1	120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
3	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114
4	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-1984	6756
5	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	1732	9405
6	120124	Lucian	Daymond	M	26480	Sales Rep. I	AU	-233	6999
7	120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU	-1852	6999
8	120126	Satyakam	Denny	M	26780	Sales Rep. II	AU	10490	17014
9	120127	Sharryn	Clarkson	F	28100	Sales Rep. II	AU	6943	14184
10	120128	Monica	Kletschkus	F	30890	Sales Rep. IV	AU	9691	17106
11	120129	Alvin	Roebuck	M	30070	Sales Rep. III	AU	1787	9405
12	120130	Kevin	Lyon	M	26955	Sales Rep. I	AU	9114	16922
13	120131	Marinus	Surawski	M	26910	Sales Rep. I	AU	7207	15706
14	120132	Fancine	Kaiser	F	28525	Sales Rep. III	AU	-3923	6848
15	120133	Petrea	Soltau	F	27440	Sales Rep. II	AU	9608	17075

p105d01

Setup for the Poll

- Retrieve program **p105a01**.
- Submit the program and confirm that a new SAS data set was created with 77 observations and 12 variables.

5.02 Poll

The DATA step reads a temporary SAS data set to create a permanent SAS data set.

- ❍ True
- ❍ False

Chapter 5: Reading SAS Data Sets

Objectives

- Subset observations by using the WHERE statement.
- Subset variables by using the DROP and KEEP statements.

Business Scenario Syntax

Use the following statements to complete the scenario:

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Part 1

Part 2

Part 3

Subsetting Observations and Variables

By default, the SET statement reads **all observations** and **all variables** from the input data set.

```
9    data work.subset1;
10      set orion.sales;
11   run;

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.SUBSET1 has 165 observations and 9 variables.
```

By adding statements to the DATA step, the number of observations and variables can be reduced.

```
NOTE: The data set WORK.SUBSET1 has 61 observations and 5 variables.
```

The WHERE Statement

The *WHERE statement* subsets observations that meet a particular condition.

General form of the WHERE statement:

WHERE *where-expression* **;**

The *where-expression* is a sequence of operands and operators that form a set of instructions that define a condition for selecting observations.

- Operands include constants and variables.
- Operators are symbols that request a comparison, arithmetic calculation, or logical operation.

Operands

A *constant operand* is a fixed value.

- Character values must be enclosed in quotation marks and are case sensitive.
- Numeric values do not use quotation marks.

A *variable operand* must be a variable coming from an input data set.

Examples:

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

variable constant

```
where Salary > 50000;
```

variable constant

Comparison Operators

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

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

Comparison Operators

Examples:

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

```
where Gender eq ' ';
```

```
where Salary ne .;
```

```
where Salary >= 50000;
```

```
where Country in ('AU','US');
```

```
where Country in ('AU' 'US');
```

Values must be separated by commas or blanks.

Arithmetic Operators

Arithmetic operators indicate that an arithmetic calculation is performed.

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

Arithmetic Operators

Examples:

```
where Salary / 12 < 6000;
```

```
where Salary / 12 * 1.10 >= 7500;
```

```
where (Salary / 12 ) * 1.10 >= 7500;
```

```
where Salary + Bonus <= 10000;
```

Logical Operators

Logical operators combine or modify expressions.

Symbol	Mnemonic	Definition
&	AND	logical and
|	OR	logical or
^ ¬ ~	NOT	logical not

Logical Operators

Examples:

```
where Gender ne 'M' and Salary >=50000;
```

```
where Gender ne 'M' or Salary >= 50000;
```

```
where Country = 'AU' or Country = 'US';
```

```
where Country not in ('AU' 'US');
```

5.03 Quiz

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

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

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

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

Special WHERE Operators

Special WHERE operators are operators that can only be used in a where-expression.

Symbol	Mnemonic	Definition
	BETWEEN-AND	an inclusive range
	IS NULL	missing value
	IS MISSING	missing value
?	CONTAINS	a character string
	LIKE	a character pattern

BETWEEN-AND Operator

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

Examples:

```
where salary between 50000 and 100000;
```

```
where salary not between 50000 and 100000;
```

Equivalent Expressions:

```
where salary between 50000 and 100000;
```

```
where 50000 <= salary <= 100000;
```

IS NULL and IS MISSING Operators

The *IS NULL* and *IS MISSING operators* select observations in which the value of a variable is missing.

- The operator can be used for both character and numeric variables.
- You can combine the NOT logical operator with IS NULL or IS MISSING to select nonmissing values.

Examples:

```
where Employee_ID is null;
```

```
where Employee_ID is missing;
```

CONTAINS Operator

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

- The position of the substring within the variable's values is not important.
- The operator is case sensitive when you make comparisons.

Example:

```
where Job_Title contains 'Rep';
```

5.04 Quiz

Which value will not be returned based on the WHERE statement?

a. Office Rep
b. Sales Rep. IV
c. service rep III
d. Representative

```
where Job_Title contains 'Rep';
```

LIKE Operator

The *LIKE operator* selects observations by comparing character values to specified patterns.

There are two special characters available for specifying a pattern:

- A percent sign (%) replaces any number of characters.
- An underscore (_) replaces one character.

Consecutive underscores can be specified.

A percent sign and an underscore can be specified in the same pattern.

The operator is case sensitive.

LIKE Operator

Examples:

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

This WHERE statement selects observations that begin with any number of characters and end with an N.

```
where Name like 'T_M%';
```

This WHERE statement selects observations that begin with a T, followed by a single character, followed by an M, followed by any number of characters.

5.05 Quiz

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

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

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

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

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

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

first name

last name

Business Scenario Part 2

Include only the employees from Australia who have the word `Rep` in their job title.

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

Partial SAS Log

```
NOTE: There were 61 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep';
NOTE: The data set WORK.SUBSET1 has 61 observations and 9 variables.
```


p105d02

Business Scenario Part 2

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

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_ Date
1	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114
2	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-1984	6756
3	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	1732	9405
4	120124	Lucian	Daymond	M	26480	Sales Rep. I	AU	-233	6999
5	120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU	-1852	6999
6	120126	Satyakam	Denny	M	26780	Sales Rep. II	AU	10490	17014
7	120127	Sharryn	Clarkson	F	28100	Sales Rep. II	AU	6943	14184
8	120128	Monica	Kletschkus	F	30890	Sales Rep. IV	AU	9691	17106
9	120129	Alvin	Roebuck	M	30070	Sales Rep. III	AU	1787	9405
10	120130	Kevin	Lyon	M	26955	Sales Rep. I	AU	9114	16922
11	120131	Marinus	Surawski	M	26910	Sales Rep. I	AU	7207	15706
12	120132	Fancine	Kaiser	F	28525	Sales Rep. III	AU	-3923	6848
13	120133	Petrea	Soltau	F	27440	Sales Rep. II	AU	9608	17075
14	120134	Sian	Shannan	M	28015	Sales Rep. II	AU	-3861	5114
15	120135	Alexei	Platts	M	32490	Sales Rep. IV	AU	3313	13788

p105d02

The DROP and KEEP Statements

The *DROP statement* specifies the names of the variables to omit from the output data set(s).

DROP *variable-list*;

The *KEEP statement* specifies the names of the variables to write to the output data set(s).

KEEP *variable-list*;

The *variable-list* specifies the variables to drop or keep, respectively, in the output data set.

The DROP and KEEP Statements

Examples:

```
drop Employee_ID Gender
     Country Birth_Date;
```

```
keep First_Name Last_Name
     Salary Job_Title
     Hire_Date;
```

Business Scenario Part 2

Include only the employee's first name, last name, salary, job title, and hired date in the data set **`Work.subset1`**.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
run;
```

Partial SAS Log

```
NOTE: There were 61 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and Job_Title contains 'Rep';
NOTE: The data set WORK.SUBSET1 has 61 observations and 5 variables.
```

Business Scenario Part 2

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

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Job_Title	Hire_ Date
1	Irenie	Elvish	26600	Sales Rep. II	5114
2	Christina	Ngan	27475	Sales Rep. II	6756
3	Kimiko	Hotstone	26190	Sales Rep. I	9405
4	Lucian	Daymond	26480	Sales Rep. I	6999
5	Fong	Hofmeister	32040	Sales Rep. IV	6999
6	Satyakam	Denny	26780	Sales Rep. II	17014
7	Sharryn	Clarkson	28100	Sales Rep. II	14184
8	Monica	Kletschkus	30890	Sales Rep. IV	17106
9	Alvin	Roebuck	30070	Sales Rep. III	9405
10	Kevin	Lyon	26955	Sales Rep. I	16922
11	Marinus	Surawski	26910	Sales Rep. I	15706
12	Fancine	Kaiser	28525	Sales Rep. III	6848

p105d03

Chapter 5: Reading SAS Data Sets

Objectives

- Add labels to the descriptor portion of a SAS data set by using the LABEL statement.
- Add formats to the descriptor portion of a SAS data set by using the FORMAT statement.

Business Scenario Syntax

Use the following statements to complete the scenario:

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Part 1

Part 2

Part 3

Adding Permanent Attributes

The descriptor portion of the SAS data set stores variable attributes including the name, type (character or numeric), and length of the variable.

Labels and formats can also be stored in the descriptor portion.

Partial PROC CONTENTS Output

```
                Alphabetic List of Variables and Attributes

#    Variable      Type    Len    Format       Label

1    First_Name    Char     12
5    Hire_Date     Num       8    DDMMYY10.    Date Hired
4    Job_Title     Char     25                 Sales Title
2    Last_Name     Char     18
3    Salary        Num       8    COMMAX8.
```

Adding Permanent Attributes

When displaying reports,

- a *label* changes the appearance of a variable name
- a *format* changes the appearance of variable value.

Partial PROC PRINT Output

Label

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired
1	Irenie	Elvish	26.600	Sales Rep. II	01/01/1974
2	Christina	Ngan	27.475	Sales Rep. II	01/07/1978
3	Kimiko	Hotstone	26.190	Sales Rep. I	01/10/1985
4	Lucian	Daymond	26.480	Sales Rep. I	01/03/1979
5	Fong	Hofmeister	32.040	Sales Rep. IV	01/03/1979

Format

The LABEL Statement

The *LABEL statement* assigns descriptive labels to variable names.

General form of the LABEL statement:

```
LABEL variable = 'label'
      variable = 'label'
      variable = 'label';
```

- A label can have as many as 256 characters.
- Any number of variables can be associated with labels in a single LABEL statement.
- Using a LABEL statement in a DATA step permanently associates labels with variables by storing the label in the descriptor portion of the SAS data set.

Business Scenario Part 3

Include labels in the descriptor portion of **`Work.subset1`**.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
run;
```

p105d04

Business Scenario Part 3

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

Partial PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes

#    Variable      Type    Len    Label

1    First_Name    Char     12
5    Hire_Date     Num       8    Date Hired
4    Job_Title     Char     25    Sales Title
2    Last_Name     Char     18
3    Salary        Num       8
```


p105d04

Business Scenario Part 3

In order to use labels in the PRINT procedure, a LABEL option needs to be added to the PROC PRINT statement.

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

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired
1	Irenie	Elvish	26600	Sales Rep. II	5114
2	Christina	Ngan	27475	Sales Rep. II	6756
3	Kimiko	Hotstone	26190	Sales Rep. I	9405
4	Lucian	Daymond	26480	Sales Rep. I	6999
5	Fong	Hofmeister	32040	Sales Rep. IV	6999
6	Satyakam	Denny	26780	Sales Rep. II	17014
7	Sharryn	Clarkson	28100	Sales Rep. II	14184
8	Monica	Kletschkus	30890	Sales Rep. IV	17106
9	Alvin	Roebuck	30070	Sales Rep. III	9405
10	Kevin	Lyon	26955	Sales Rep. I	16922

p105d04

The FORMAT Statement

The *FORMAT statement* assigns formats to variable values.

General form of the FORMAT statement:

FORMAT *variable(s) format*;

- A *format* is an instruction that SAS uses to write data values.
- Using a FORMAT statement in a DATA step permanently associates formats with variables by storing the format in the descriptor portion of the SAS data set.

SAS Formats

SAS formats have the following form:

<$>*format*<*w*>.<*d*>

$	indicates a character format.
format	names the SAS format or user-defined format.
w	specifies the total format width including decimal places and special characters.
.	is a required delimiter.
d	specifies the number of decimal places in numeric formats.

SAS Formats

Selected SAS formats:

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

SAS Formats

Selected SAS formats:

Format	Stored Value	Displayed Value
$4.	Programming	Prog
12.	27134.2864	27134
12.2	27134.2864	27134.29
COMMA12.2	27134.2864	27,134.29
COMMAX12.2	27134.2864	27.134,29
DOLLAR12.2	27134.2864	$27,134.29
EUROX12.2	27134.2864	€27.134,29

SAS Formats

If you do not specify a format width that is large enough to accommodate a numeric value, the displayed value is automatically adjusted to fit into the width.

Format	Stored Value	Displayed Value
DOLLAR12.2	27134.2864	$27,134.29
DOLLAR9.2	27134.2864	$27134.29
DOLLAR8.2	27134.2864	27134.29
DOLLAR5.2	27134.2864	27134
DOLLAR4.2	27134.2864	27E3

5.06 Quiz

Which numeric format writes standard numeric data with leading zeros?

Documentation on formats can be found in the SAS Help and Documentation from the Contents tab (**SAS Products** ⇨ **Base SAS** ⇨ **SAS 9.2 Language Reference: Dictionary** ⇨ **Dictionary of Language Elements** ⇨ **Formats** ⇨ **Formats by Category**).

SAS Date Formats

SAS date formats display SAS date values in standard date forms.

Format	Stored Value	Displayed Value
MMDDYY6.	0	010160
MMDDYY8.	0	01/01/60
MMDDYY10.	0	01/01/1960
DDMMYY6.	365	311260
DDMMYY8.	365	31/12/60
DDMMYY10.	365	31/12/1960

SAS Date Formats

Additional date formats:

Format	Stored Value	Displayed Value
DATE7.	-1	31DEC59
DATE9.	-1	31DEC1959
WORDDATE.	0	January 1, 1960
WEEKDATE.	0	Friday, January 1, 1960
MONYY7.	0	JAN1960
YEAR4.	0	1960

5.07 Quiz

Which FORMAT statement creates the output?

a.
```
format Birth_Date Hire_Date ddmmyy9.
       Term_Date mmyy7.;
```

b.
```
format Birth_Date Hire_Date ddmmyyyy.
       Term_Date mmmyyyy.;
```

c.
```
format Birth_Date Hire_Date ddmmyy10.
       Term_Date monyy7.;
```

Output

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

SAS Date Formats

The SAS National Language Support (NLS) date formats convert SAS date values to a locale-sensitive date string.

Format	Locale	Example
NLDATE*w.*	English_UnitedStates	`January 01, 1960`
	German_Germany	`01. Januar 1960`
NLDATEMN*w.*	English_UnitedStates	`January`
	German_Germany	`Januar`
NLDATEW*w.*	English_UnitedStates	`Fri, Jan 01, 60`
	German_Germany	`Fr, 01. Jan 60`
NLDATEWN*w.*	English_UnitedStates	`Friday`
	German_Germany	`Freitag`

p105d05

5.08 Quiz

How many date and time formats start with EUR?

Documentation on NLS formats can be found in the SAS Help and Documentation from the Contents tab (**SAS Products** ⇨ **Base SAS** ⇨ **SAS 9.2 Language Reference: Dictionary** ⇨ **Dictionary of Language Elements** ⇨ **Formats** ⇨ **Formats Documented in Other SAS Publications**).

Business Scenario Part 3

Include formats in the descriptor portion of **`Work.subset1`**.

```
data work.subset1;
   set orion.sales;
   where Country='AU' and
         Job_Title contains 'Rep';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary commax8. Hire_Date ddmmyy10.;
run;
```

p105d06

Business Scenario Part 3

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

Partial PROC CONTENTS Output

```
                Alphabetic List of Variables and Attributes

   #    Variable       Type    Len    Format       Label

   1    First_Name     Char     12
   5    Hire_Date      Num       8    DDMMYY10.    Date Hired
   4    Job_Title      Char     25                 Sales Title
   2    Last_Name      Char     18
   3    Salary         Num       8    COMMAX8.
```

Business Scenario Part 3

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

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Salary	Sales Title	Date Hired
1	Irenie	Elvish	26.600	Sales Rep. II	01/01/1974
2	Christina	Ngan	27.475	Sales Rep. II	01/07/1978
3	Kimiko	Hotstone	26.190	Sales Rep. I	01/10/1985
4	Lucian	Daymond	26.480	Sales Rep. I	01/03/1979
5	Fong	Hofmeister	32.040	Sales Rep. IV	01/03/1979
6	Satyakam	Denny	26.780	Sales Rep. II	01/08/2006
7	Sharryn	Clarkson	28.100	Sales Rep. II	01/11/1998
8	Monica	Kletschkus	30.890	Sales Rep. IV	01/11/2006
9	Alvin	Roebuck	30.070	Sales Rep. III	01/10/1985
10	Kevin	Lyon	26.955	Sales Rep. I	01/05/2006
11	Marinus	Surawski	26.910	Sales Rep. I	01/01/2003
12	Fancine	Kaiser	28.525	Sales Rep. III	01/10/1978

p105d06

Chapter Review

1. What statement is used to read from a SAS data set in the DATA step?

2. What statement is used to write to a SAS data set in the DATA step?

3. What does the WHERE statement do?

4. What are examples of logical operators?

5. How can you limit the variables written to an output data set in the DATA step?