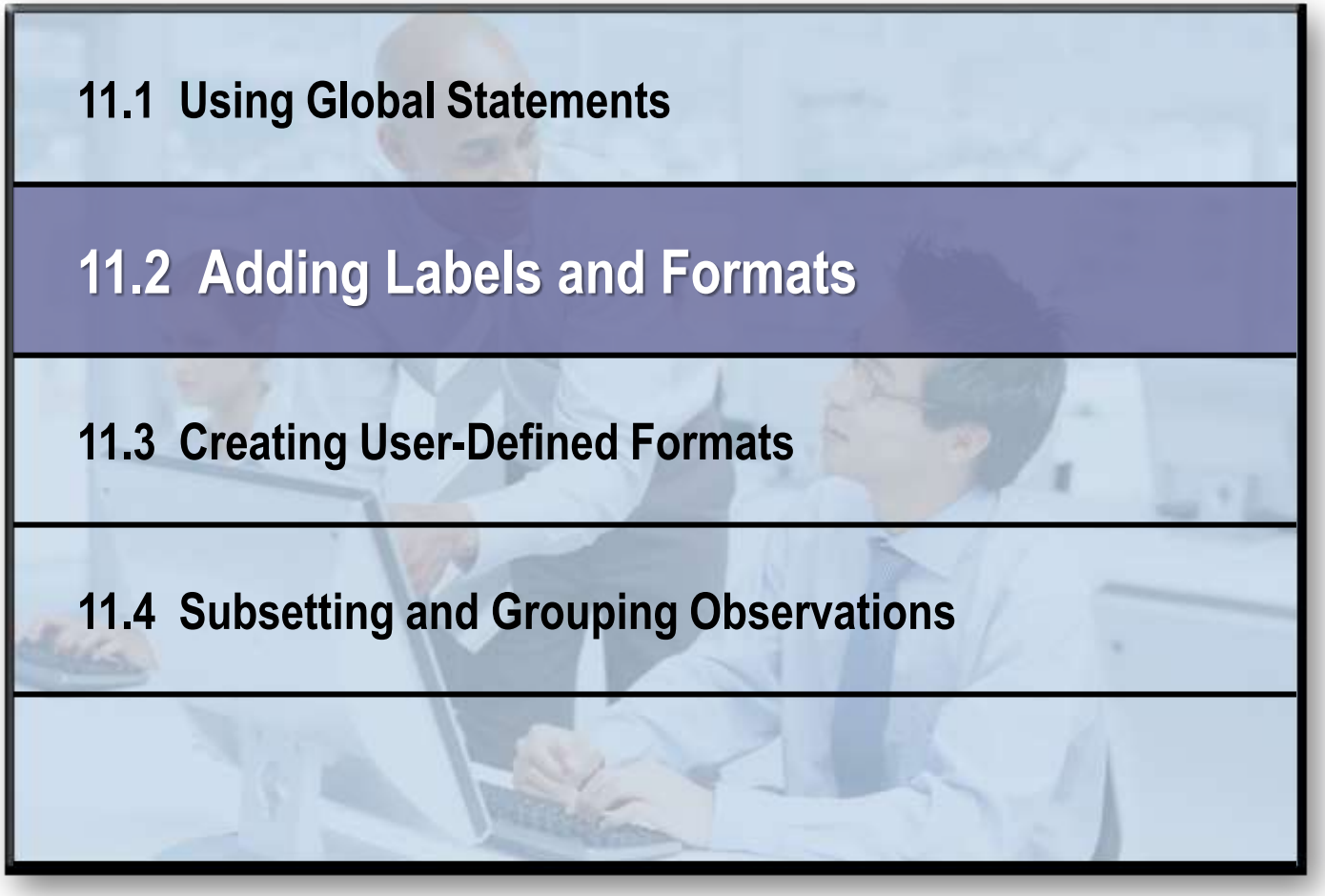
STAT604

Lesson SAS 14

Chapter 11: Enhancing Reports



Objectives

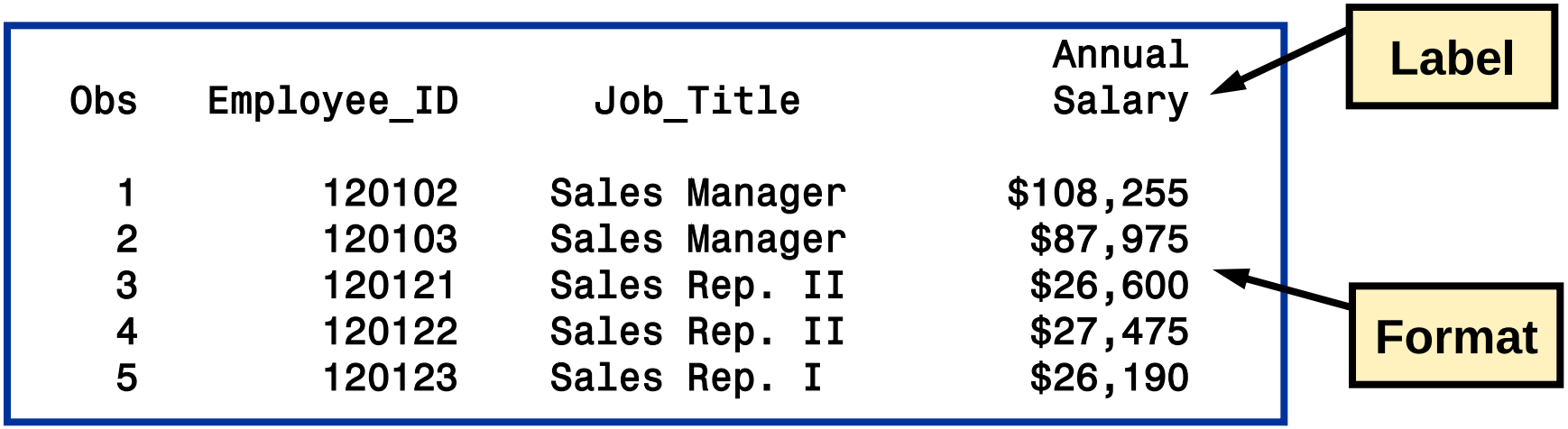
- Display descriptive column headings using the LABEL statement.
- Display formatted values using the FORMAT statement.

Labels and Formats (Review)

When displaying reports,

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

Obs	Employee_ID	Job_Title	Annual Salary
1	120102	Sales Manager	$108,255
2	120103	Sales Manager	$87,975
3	120121	Sales Rep. II	$26,600
4	120122	Sales Rep. II	$27,475
5	120123	Sales Rep. I	$26,190



The LABEL Statement (Review)

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 be up to 256 characters.
- Labels are used automatically by many procedures.
- The PRINT procedure uses labels when the LABEL or SPLIT= option is specified in the PROC PRINT statement.

Assigning Temporary Labels

PROC FREQ automatically uses labels.

```
proc freq data=orion.sales;
   tables Gender;
   label Gender='Sales Employee Gender';
run;
```

The FREQ Procedure

Sales Employee Gender

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	68	41.21	68	41.21
M	97	58.79	165	100.00

p111d05

Assigning Temporary Labels

PROC PRINT does not automatically use labels.

```
proc print data=orion.sales;
   var Employee_ID Job_Title Salary;
   label Employee_ID='Sales ID'
         Job_Title='Job Title'
         Salary='Annual Salary';
run;
```

Partial PROC PRINT Output

Obs	Employee_ID	Job_Title	Salary
1	120102	Sales Manager	108255
2	120103	Sales Manager	87975
3	120121	Sales Rep. II	26600
4	120122	Sales Rep. II	27475
5	120123	Sales Rep. I	26190

Assigning Temporary Labels

The LABEL option tells PROC PRINT to use labels.

```
proc print data=orion.sales label;
   var Employee_ID Job_Title Salary;
   label Employee_ID='Sales ID'
         Job_Title='Job Title'
         Salary='Annual Salary';
run;
```

Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary
1	120102	Sales Manager	108255
2	120103	Sales Manager	87975
3	120121	Sales Rep. II	26600
4	120122	Sales Rep. II	27475
5	120123	Sales Rep. I	26190

p111d05

Assigning Temporary Labels

Instead of the LABEL option in PROC PRINT, the SPLIT= option can be used.

The *SPLIT= option* specifies the split character, which controls line breaks in column headers.

General form of the SPLIT= option:

SPLIT='*split-character***'**

Assigning Temporary Labels

The SPLIT= option makes PROC PRINT use labels.

```
proc print data=orion.sales split='*';
   var Employee_ID Job_Title Salary;
   label Employee_ID='Sales ID'
         Job_Title='Job*Title'
         Salary='Annual*Salary';
run;
```

Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary
1	120102	Sales Manager	108255
2	120103	Sales Manager	87975
3	120121	Sales Rep. II	26600
4	120122	Sales Rep. II	27475
5	120123	Sales Rep. I	26190

p111d05

Assigning Permanent Labels (Review)

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.

```
data orion.bonus;
   set orion.sales;
   Bonus=Salary*0.10;
   label Salary='Annual*Salary'
         Bonus='Annual*Bonus';
   keep Employee_ID First_Name
        Last_Name Salary Bonus;
run;

proc print data=orion.bonus split='*';
run;
```

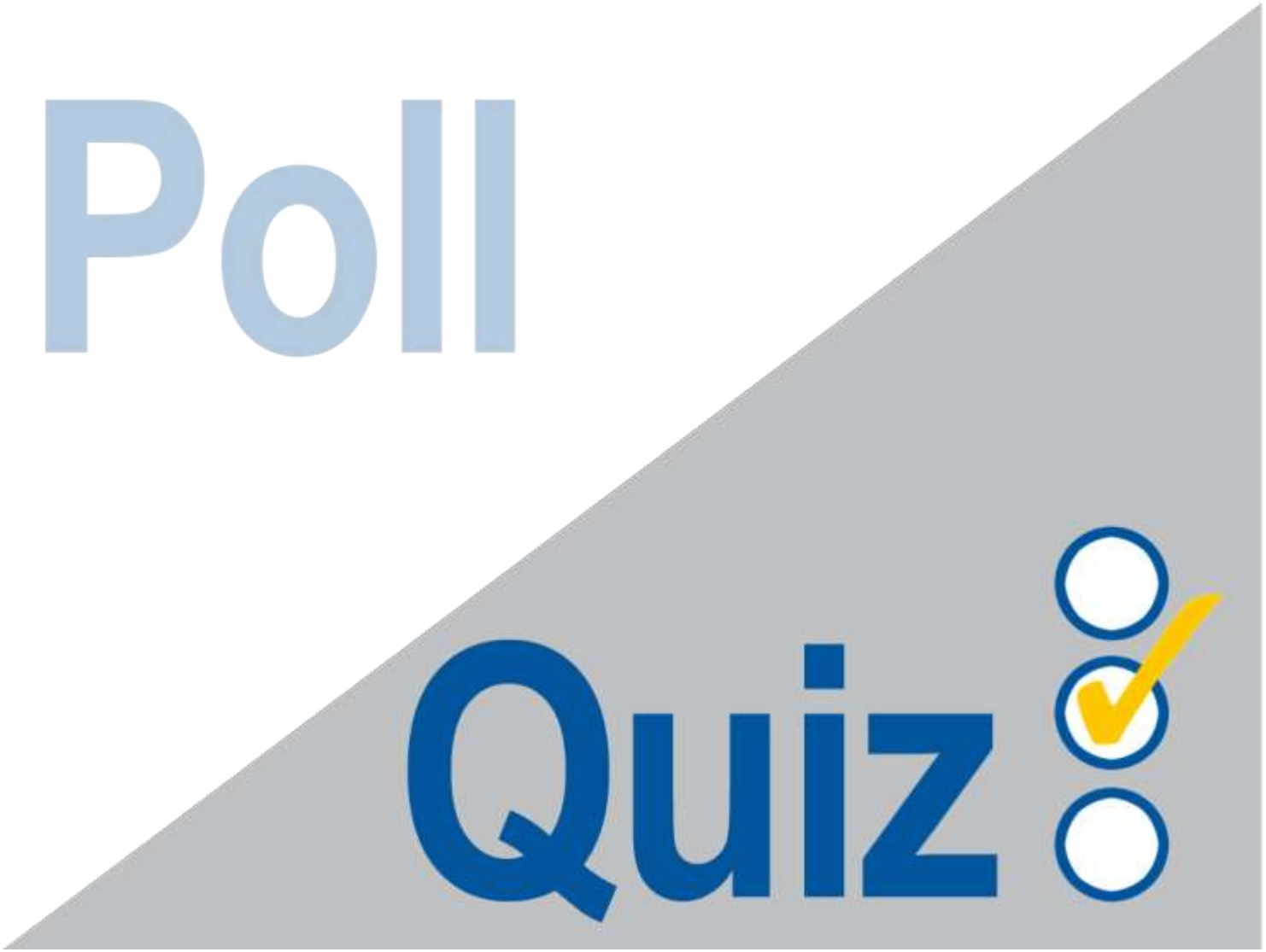

p111d05

Assigning Permanent Labels (Review)

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Annual Salary	Annual Bonus
1	120102	Tom	Zhou	108255	10825.5
2	120103	Wilson	Dawes	87975	8797.5
3	120121	Irenie	Elvish	26600	2660.0
4	120122	Christina	Ngan	27475	2747.5
5	120123	Kimiko	Hotstone	26190	2619.0
6	120124	Lucian	Daymond	26480	2648.0
7	120125	Fong	Hofmeister	32040	3204.0
8	120126	Satyakam	Denny	26780	2678.0
9	120127	Sharryn	Clarkson	28100	2810.0
10	120128	Monica	Kletschkus	30890	3089.0

Poll



11.03 Quiz

Which statement is true concerning the PROC PRINT output for **Bonus**?

a. Annual Bonus will be the label.

b. Mid-Year Bonus will be the label.

```
data orion.bonus;
   set orion.sales;
   Bonus=Salary*0.10;
   label Bonus='Annual Bonus';
run;

proc print data=orion.bonus label;
   label Bonus='Mid-Year Bonus';
run;
```


p111d05

11.03 Quiz – Correct Answer

Which statement is true concerning the PROC PRINT output for **`Bonus`**?

a. Annual Bonus will be the label.
b. Mid-Year Bonus will be the label.

Temporary labels override permanent labels.

The FORMAT Statement (Review)

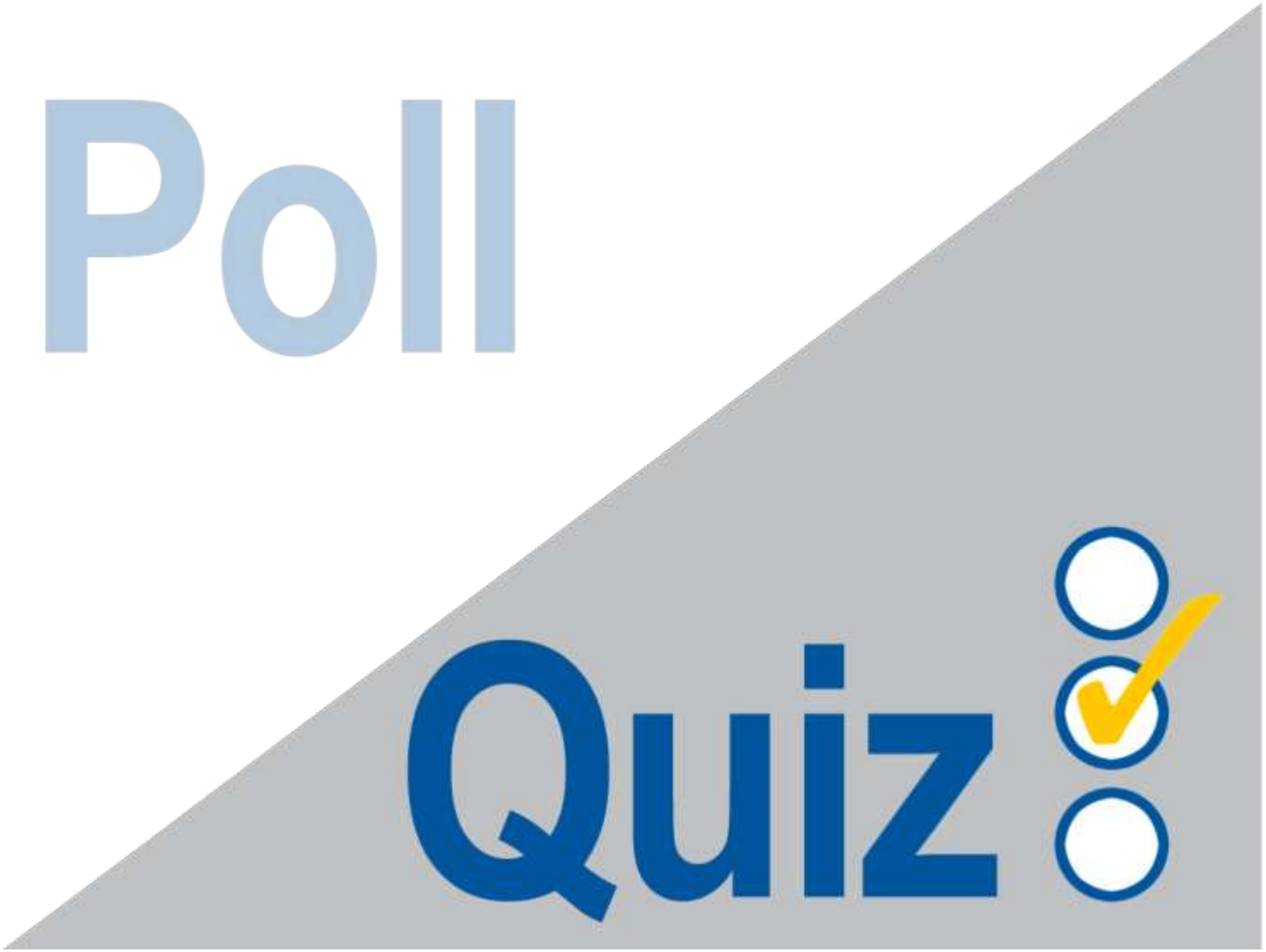
The *FORMAT statement* assigns formats to variable values.

General form of the FORMAT statement:

FORMAT *variable(s) format*;

- A *format* is an instruction that SAS uses to write data values.
- Values in the data set are not changed.

Poll



11.04 Quiz

Which displayed value is incorrect for the given format?

Format	Stored Value	Displayed Value
$3.	Wednesday	Wed
6.1	1234.345	1234.3
COMMAX5.	1234.345	1.234
DOLLAR9.2	1234.345	$1,234.35
DDMMYY8.	0	01/01/1960
DATE9.	0	01JAN1960
YEAR4.	0	1960

11.04 Quiz – Correct Answer

Which displayed value is incorrect for the given format?

Format	Stored Value	Displayed Value
$3.	Wednesday	Wed
6.1	1234.345	1234.3
COMMAX5.	1234.345	1.234
DOLLAR9.2	1234.345	$1,234.35
DDMMYY8.	0	01/01/1960
DATE9.	0	01JAN1960
YEAR4.	0	1960

DDMMYY8. produces 01/01/60.

Assigning Temporary Formats

```
proc print data=orion.sales label;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   . . .
   format Salary dollar10.0
          Birth_Date Hire_Date monyy7.;
run;
```

Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
1	120102	Sales Manager	$108,255	AU	AUG1969	JUN1989
2	120103	Sales Manager	$87,975	AU	JAN1949	JAN1974
3	120121	Sales Rep. II	$26,600	AU	AUG1944	JAN1974
4	120122	Sales Rep. II	$27,475	AU	JUL1954	JUL1978
5	120123	Sales Rep. I	$26,190	AU	SEP1964	OCT1985

p111d06

Assigning Temporary Formats

```
proc freq data=orion.sales;
   tables Hire_Date;
   format Hire_Date year4.;
run;
```

Partial PROC FREQ Output

The FREQ Procedure

Hire_Date	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1974	23	13.94	23	13.94
1975	2	1.21	25	15.15
1976	4	2.42	29	17.58
1977	3	1.82	32	19.39
1978	7	4.24	39	23.64
1979	3	1.82	42	25.45

p111d06

Assigning Permanent and Temporary Formats

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.

```
data orion.bonus;
   set orion.sales;
   Bonus=Salary*0.10;
   format Salary Bonus comma8.;
   keep Employee_ID First_Name
        Last_Name Salary Bonus;
run;

proc print data=orion.bonus;
   format Bonus dollar8.;
run;
```

- Temporary formats override permanent formats.

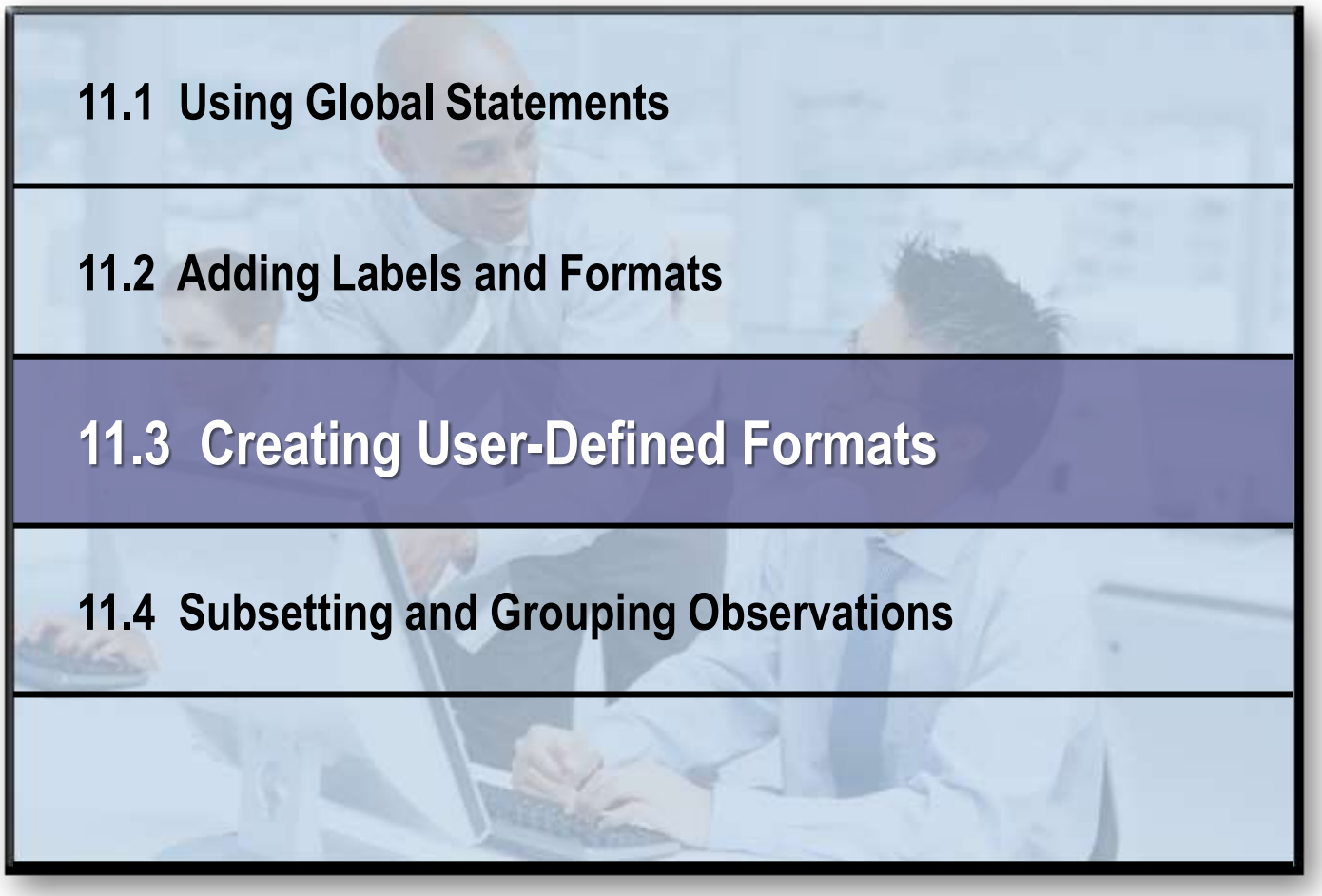
p111d06

Assigning Permanent and Temporary Formats

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Salary	Bonus
1	120102	Tom	Zhou	108,255	$10,826
2	120103	Wilson	Dawes	87,975	$8,798
3	120121	Irenie	Elvish	26,600	$2,660
4	120122	Christina	Ngan	27,475	$2,748
5	120123	Kimiko	Hotstone	26,190	$2,619
6	120124	Lucian	Daymond	26,480	$2,648
7	120125	Fong	Hofmeister	32,040	$3,204
8	120126	Satyakam	Denny	26,780	$2,678
9	120127	Sharryn	Clarkson	28,100	$2,810
10	120128	Monica	Kletschkus	30,890	$3,089

Chapter 11: Enhancing Reports



Objectives

- Create user-defined formats using the FORMAT procedure.
- Apply user-defined formats to variables in reports.

User-Defined Formats

A user-defined format needs to be created for **`Country`**.

Current Report (partial output)

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
61	120179	Sales Rep. III	$28,510	AU	MAR1974	JAN2004
62	120180	Sales Rep. II	$26,970	AU	JUN1954	DEC1978
63	120198	Sales Rep. III	$28,025	AU	JAN1988	DEC2006
64	120261	Chief Sales Officer	$243,190	US	FEB1969	AUG1987
65	121018	Sales Rep. II	$27,560	US	JAN1944	JAN1974
66	121019	Sales Rep. IV	$31,320	US	JUN1986	JUN2004

Desired Report (partial output)

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
61	120179	Sales Rep. III	$28,510	Australia	MAR1974	JAN2004
62	120180	Sales Rep. II	$26,970	Australia	JUN1954	DEC1978
63	120198	Sales Rep. III	$28,025	Australia	JAN1988	DEC2006
64	120261	Chief Sales Officer	$243,190	United States	FEB1969	AUG1987
65	121018	Sales Rep. II	$27,560	United States	JAN1944	JAN1974
66	121019	Sales Rep. IV	$31,320	United States	JUN1986	JUN2004

User-Defined Formats

To create and use your own formats, do the following:

Part 1 Use the FORMAT procedure to create the user-defined format.

Part 2 Apply the format to a specific variable(s) by using a FORMAT statement in the reporting procedure.

The FORMAT Procedure

The *FORMAT procedure* is used to create user-defined formats.

General form of the FORMAT procedure with the VALUE statement:

```
PROC FORMAT;
      VALUE format-name range1 = 'label '
                         range2 = 'label '
                                  . . . ;
RUN;
```

The FORMAT Procedure

A *format-name*

- names the format that you are creating
- cannot be more than 32 characters in SAS®9
- for character values, must have a dollar sign ($) as the first character, and a letter or underscore as the second character
- for numeric values, must have a letter or underscore as the first character
- cannot end in a number
- cannot be the name of a SAS format
- does not end with a period in the VALUE statement.


Poll
Quiz

11.05 Multiple Answer Poll

Which user-defined format names are invalid?

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

11.05 Multiple Answer Poll – Correct Answer

Which user-defined format names are invalid?

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

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

User-defined formats cannot be the name of a SAS supplied format.

The FORMAT Procedure

```
PROC FORMAT;
      VALUE format-name range1 = 'label '
                       range2 = 'label '
                                . . . ;
RUN;
```

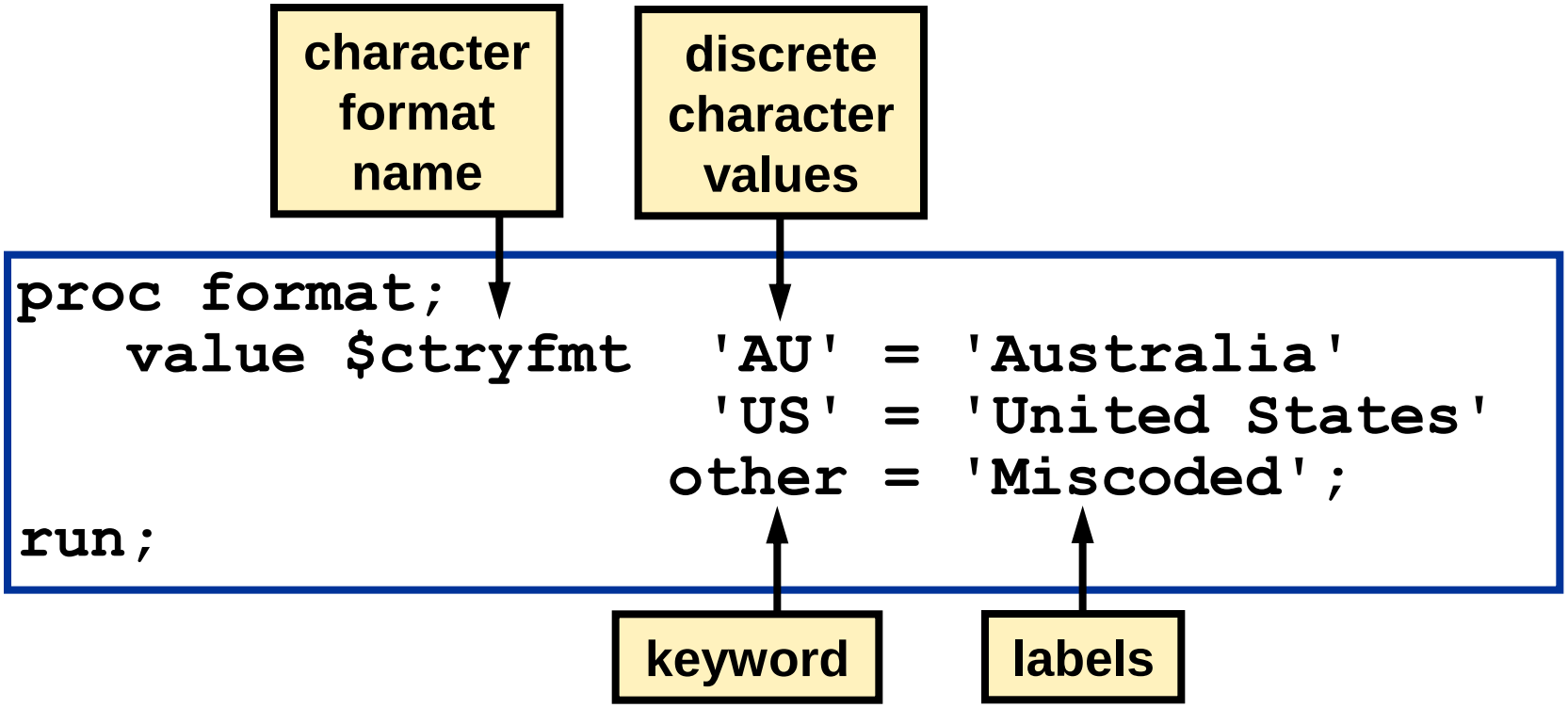
Range(s) can be

- single values
- ranges of values
- lists of values.

Labels

- can be up to 32,767 characters in length
- are typically enclosed in quotation marks, ~~although it is not required~~.

Character User-Defined Format



The OTHER keyword matches all values that do not match any other value or range.

p111d07

Character User-Defined Format

Part 1

```
proc format;
   value $ctryfmt   'AU' = 'Australia'
                    'US' = 'United States'
                  other = 'Miscoded';
run;

proc print data=orion.sales label;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   label Employee_ID='Sales ID'
         Job_Title='Job Title'
         Salary='Annual Salary'
         Birth_Date='Date of Birth'
         Hire_Date='Date of Hire';
   format Salary dollar10.0
          Birth_Date Hire_Date monyy7.
          Country $ctryfmt.;
run;
```

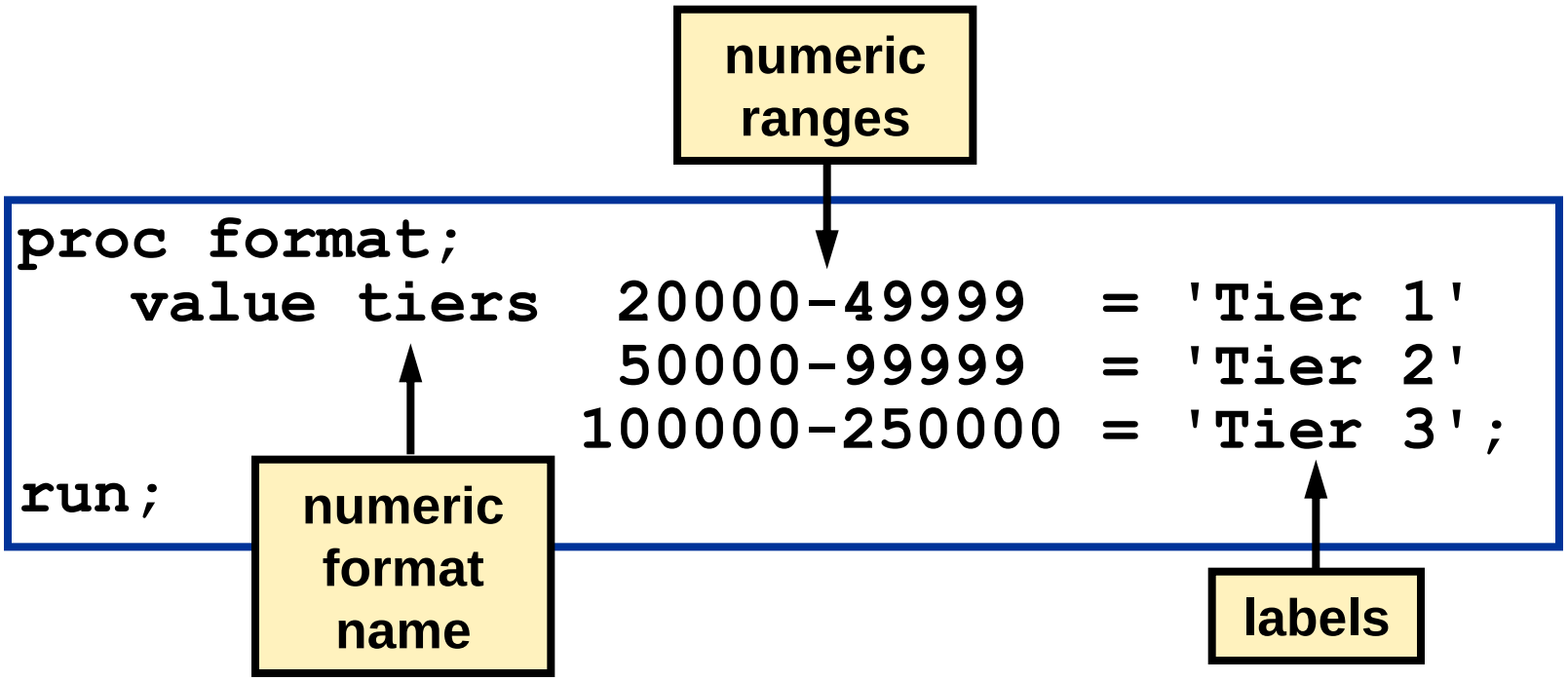
Part 2

Character User-Defined Format

Partial PROC PRINT Output

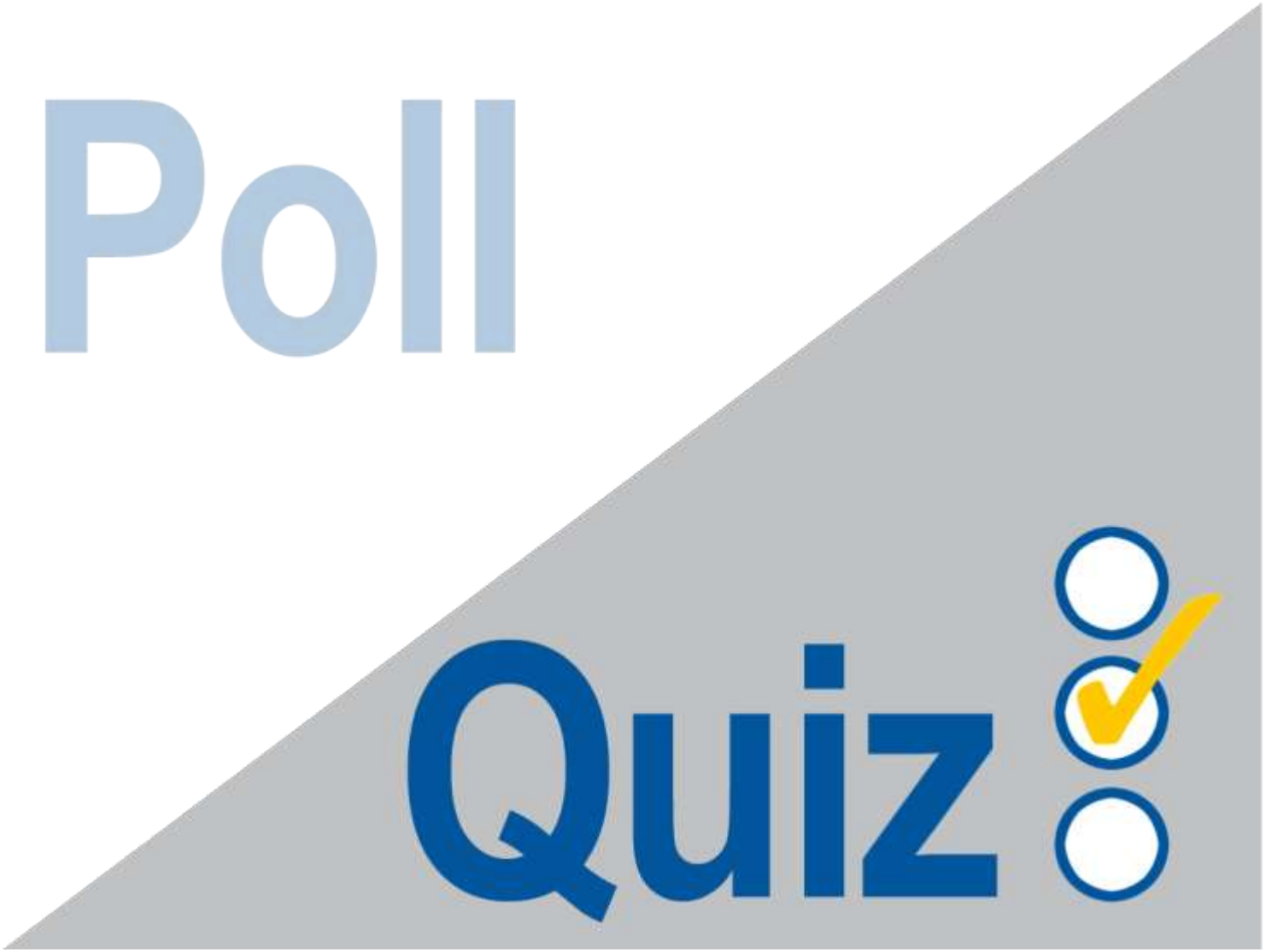
Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
60	120178	Sales Rep. II	$26,165	Australia	NOV1954	APR1974
61	120179	Sales Rep. III	$28,510	Australia	MAR1974	JAN2004
62	120180	Sales Rep. II	$26,970	Australia	JUN1954	DEC1978
63	120198	Sales Rep. III	$28,025	Australia	JAN1988	DEC2006
64	120261	Chief Sales Officer	$243,190	United States	FEB1969	AUG1987
65	121018	Sales Rep. II	$27,560	United States	JAN1944	JAN1974
66	121019	Sales Rep. IV	$31,320	United States	JUN1986	JUN2004
67	121020	Sales Rep. IV	$31,750	United States	FEB1984	MAY2002
68	121021	Sales Rep. IV	$32,985	United States	DEC1974	MAR1994
69	121022	Sales Rep. IV	$32,210	United States	OCT1979	FEB2002
70	121023	Sales Rep. I	$26,010	United States	MAR1964	MAY1989
71	121024	Sales Rep. II	$26,600	United States	SEP1984	MAY2004
72	121025	Sales Rep. II	$28,295	United States	OCT1949	SEP1975

Numeric User-Defined Format



p111d07

Poll



11.06 Quiz

If you have a value of 99999.87, how will it be displayed if the TIERS format is applied to the value?

a. Tier 2
b. Tier 3
c. a missing value
d. none of the above

```
proc format;
   value tiers  20000-49999  = 'Tier 1'
                50000-99999  = 'Tier 2'
               100000-250000 = 'Tier 3';
run;
```

11.06 Quiz – Correct Answer

If you have a value of 99999.87, how will it be displayed if the TIERS format is applied to the value?

a. Tier 2
b. Tier 3
c. a missing value
d. none of the above

```
proc format;
   value tiers  20000-49999   = 'Tier 1'
                50000-99999   = 'Tier 2'
               100000-250000  = 'Tier 3';
run;
```

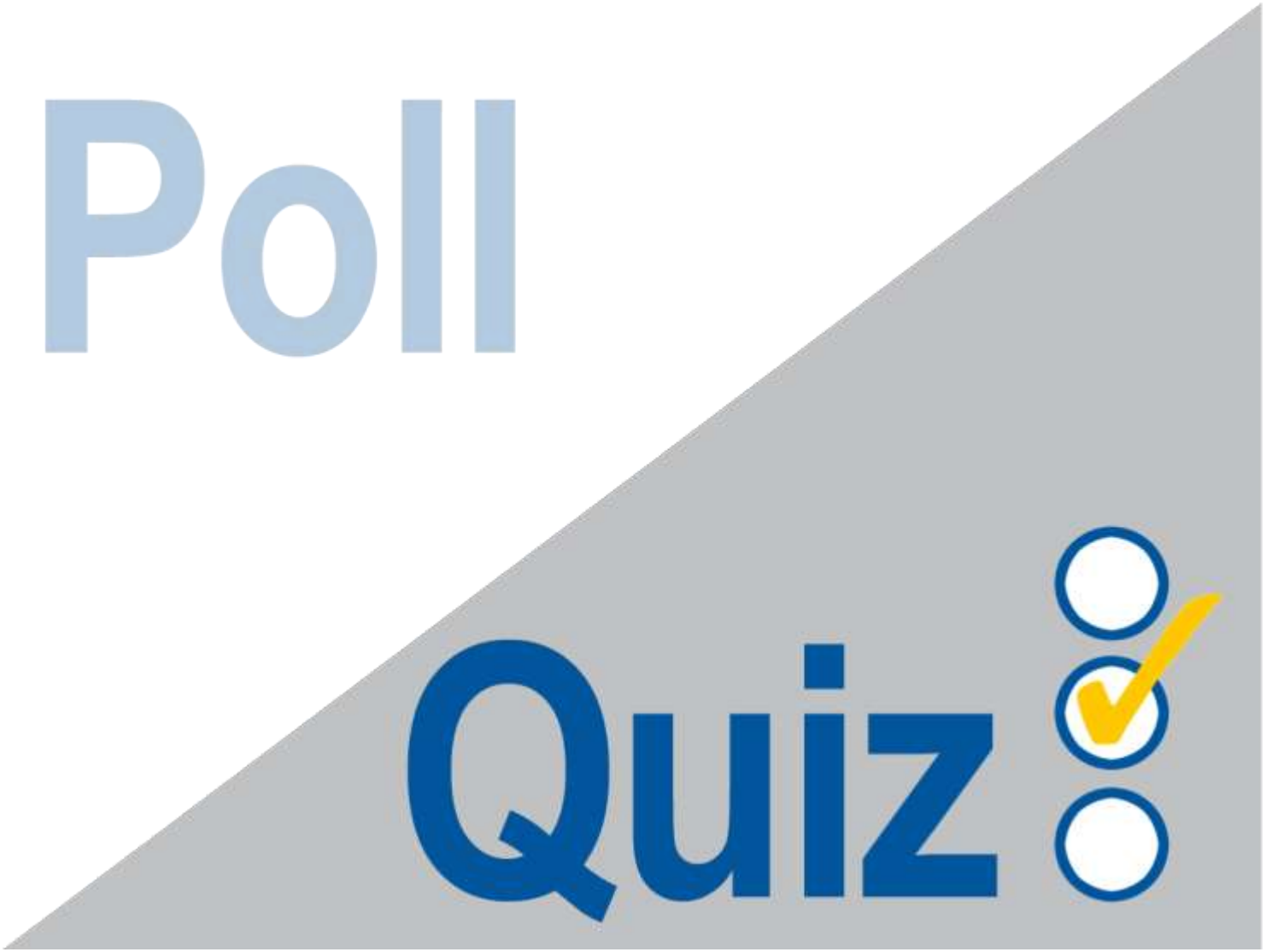
Numeric User-Defined Formats

The less than (<) symbol excludes values from ranges.

- Put < after the value if you want to exclude the first value in a range.
- Put < before the value if you want to exclude the last value in a range.

50000 - 100000	Includes 50000	Includes 100000
50000 - < 100000	Includes 50000	Excludes 100000
50000 < - 100000	Excludes 50000	Includes 100000
50000 < - < 100000	Excludes 50000	Excludes 100000

Poll



11.07 Quiz

If you have a value of 100000, how will it be displayed if the TIERS format is applied to the value?

a. Tier 2
b. Tier 3
c. 100000
d. a missing value

```
proc format;
   value tiers  20000-<50000  = 'Tier 1'
                50000- 100000 = 'Tier 2'
               100000<-250000 = 'Tier 3';
run;
```

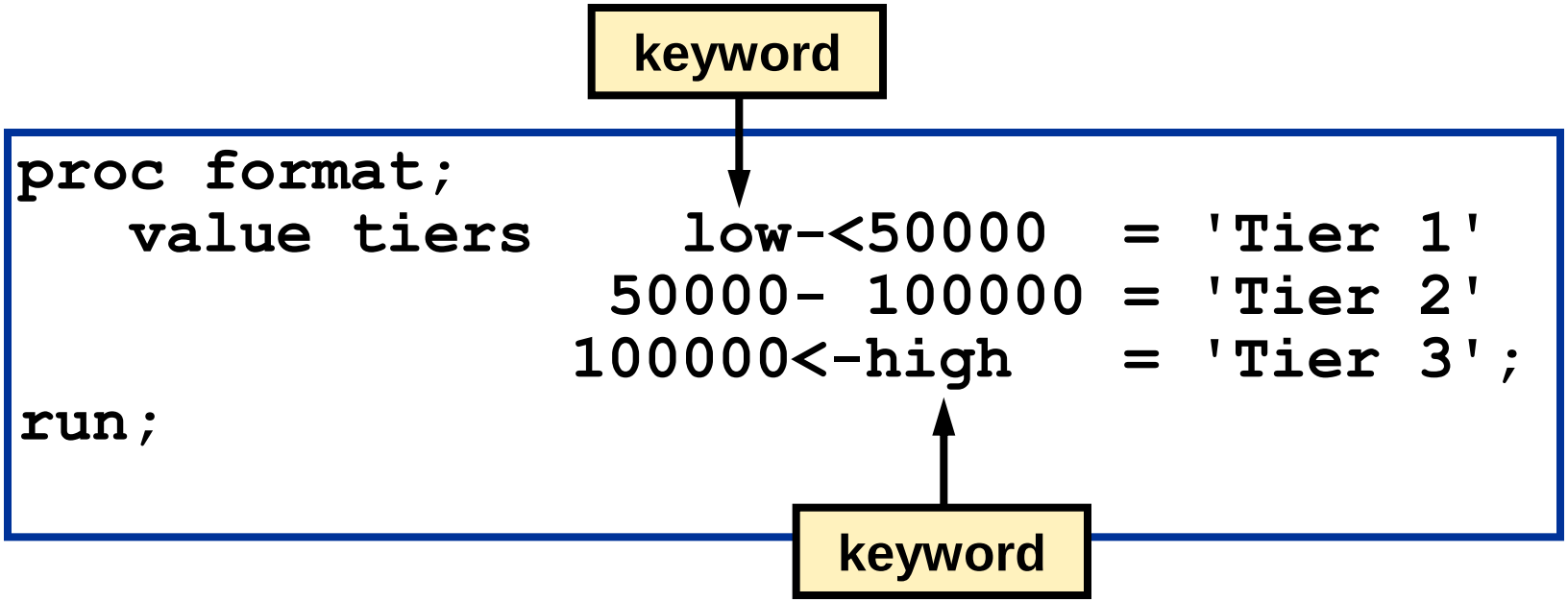
11.07 Quiz – Correct Answer

If you have a value of 100000, how will it be displayed if the TIERS format is applied to the value?

a. Tier 2
b. Tier 3
c. 100000
d. a missing value

```
proc format;
   value tiers  20000-<50000  = 'Tier 1'
                50000- 100000 = 'Tier 2'
               100000<-250000 = 'Tier 3';
run;
```

Numeric User-Defined Format



LOW encompasses the lowest possible value.

HIGH encompasses the highest possible value.

p111d07

Numeric User-Defined Format

Part 1

Part 2

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

proc print data=orion.sales label;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   label Employee_ID='Sales ID'
         Job_Title='Job Title'
         Salary='Annual Salary'
         Birth_Date='Date of Birth'
         Hire_Date='Date of Hire';
   format Birth_Date Hire_Date monyy7.
          Salary tiers.;
run;
```

Numeric User-Defined Format

Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
60	120178	Sales Rep. II	Tier 1	AU	NOV1954	APR1974
61	120179	Sales Rep. III	Tier 1	AU	MAR1974	JAN2004
62	120180	Sales Rep. II	Tier 1	AU	JUN1954	DEC1978
63	120198	Sales Rep. III	Tier 1	AU	JAN1988	DEC2006
64	120261	Chief Sales Officer	Tier 3	US	FEB1969	AUG1987
65	121018	Sales Rep. II	Tier 1	US	JAN1944	JAN1974
66	121019	Sales Rep. IV	Tier 1	US	JUN1986	JUN2004
67	121020	Sales Rep. IV	Tier 1	US	FEB1984	MAY2002
68	121021	Sales Rep. IV	Tier 1	US	DEC1974	MAR1994
69	121022	Sales Rep. IV	Tier 1	US	OCT1979	FEB2002
70	121023	Sales Rep. I	Tier 1	US	MAR1964	MAY1989
71	121024	Sales Rep. II	Tier 1	US	SEP1984	MAY2004
72	121025	Sales Rep. II	Tier 1	US	OCT1949	SEP1975

Other User-Defined Format Examples

```
proc format;
   value $grade       'A' = 'Good'
                  'B'-'D' = 'Fair'
                      'F' = 'Poor'
                  'I','U' = 'See Instructor'
                    other = 'Miscoded';
run;
```

```
proc format;
   value mnthfmt     1,2,3 = 'Qtr 1'
                     4,5,6 = 'Qtr 2'
                     7,8,9 = 'Qtr 3'
                  10,11,12 = 'Qtr 4'
                         . = 'missing'
                     other = 'unknown';
run;
```

Multiple User-Defined Formats

Multiple VALUE statements can be in a single PROC FORMAT step.

```
proc format;
   value $ctryfmt   'AU' = 'Australia'
                    'US' = 'United States'
                  other = 'Miscoded';
   value tiers      low-<50000  = 'Tier 1'
                 50000- 100000 = 'Tier 2'
                100000<-high   = 'Tier 3';
run;
```

Multiple User-Defined Formats

```
proc print data=orion.sales label;
    . . .
    format Birth_Date Hire_Date monyy7.
           Country $ctryfmt.
           Salary tiers.;
run;
```

Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
60	120178	Sales Rep. II	Tier 1	Australia	NOV1954	APR1974
61	120179	Sales Rep. III	Tier 1	Australia	MAR1974	JAN2004
62	120180	Sales Rep. II	Tier 1	Australia	JUN1954	DEC1978
63	120198	Sales Rep. III	Tier 1	Australia	JAN1988	DEC2006
64	120261	Chief Sales Officer	Tier 3	United States	FEB1969	AUG1987
65	121018	Sales Rep. II	Tier 1	United States	JAN1944	JAN1974
66	121019	Sales Rep. IV	Tier 1	United States	JUN1986	JUN2004
67	121020	Sales Rep. IV	Tier 1	United States	FEB1984	MAY2002

p111d07

Multiple User-Defined Formats

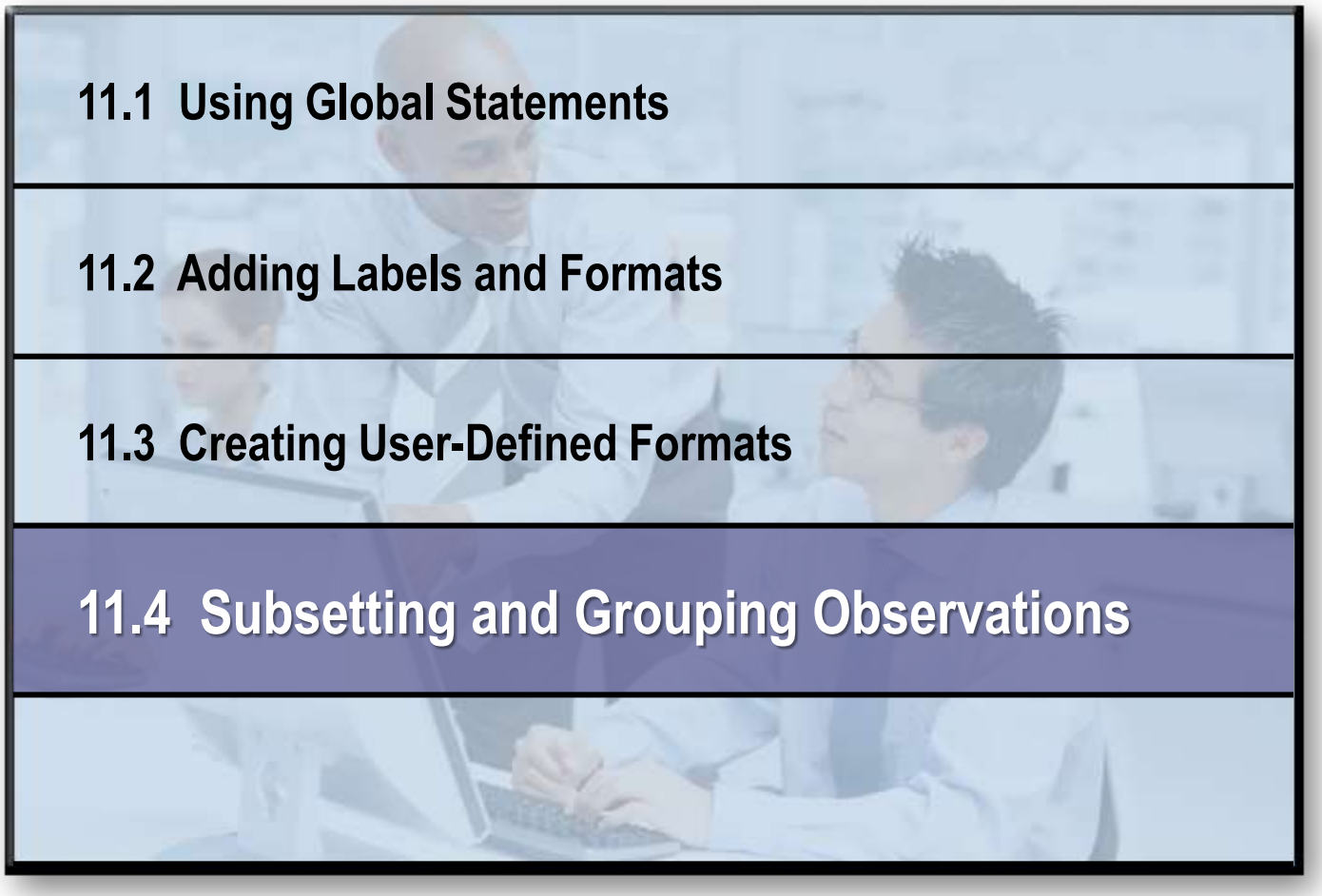
```
proc freq data=orion.sales;
   tables Country Salary;
   format Country $ctryfmt. Salary tiers.;
run;
```

The FREQ Procedure

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Australia	63	38.18	63	38.18
United States	102	61.82	165	100.00

Salary	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Tier 1	159	96.36	159	96.36
Tier 2	4	2.42	163	98.79
Tier 3	2	1.21	165	100.00

Chapter 11: Enhancing Reports



Objectives

- Display selected observations in reports by using the WHERE statement.
- Display groups of observations in reports by using the BY statement.

The WHERE Statement (Review)

For subsetting observations in a report, the WHERE statement is used to select observations that meet a certain condition.

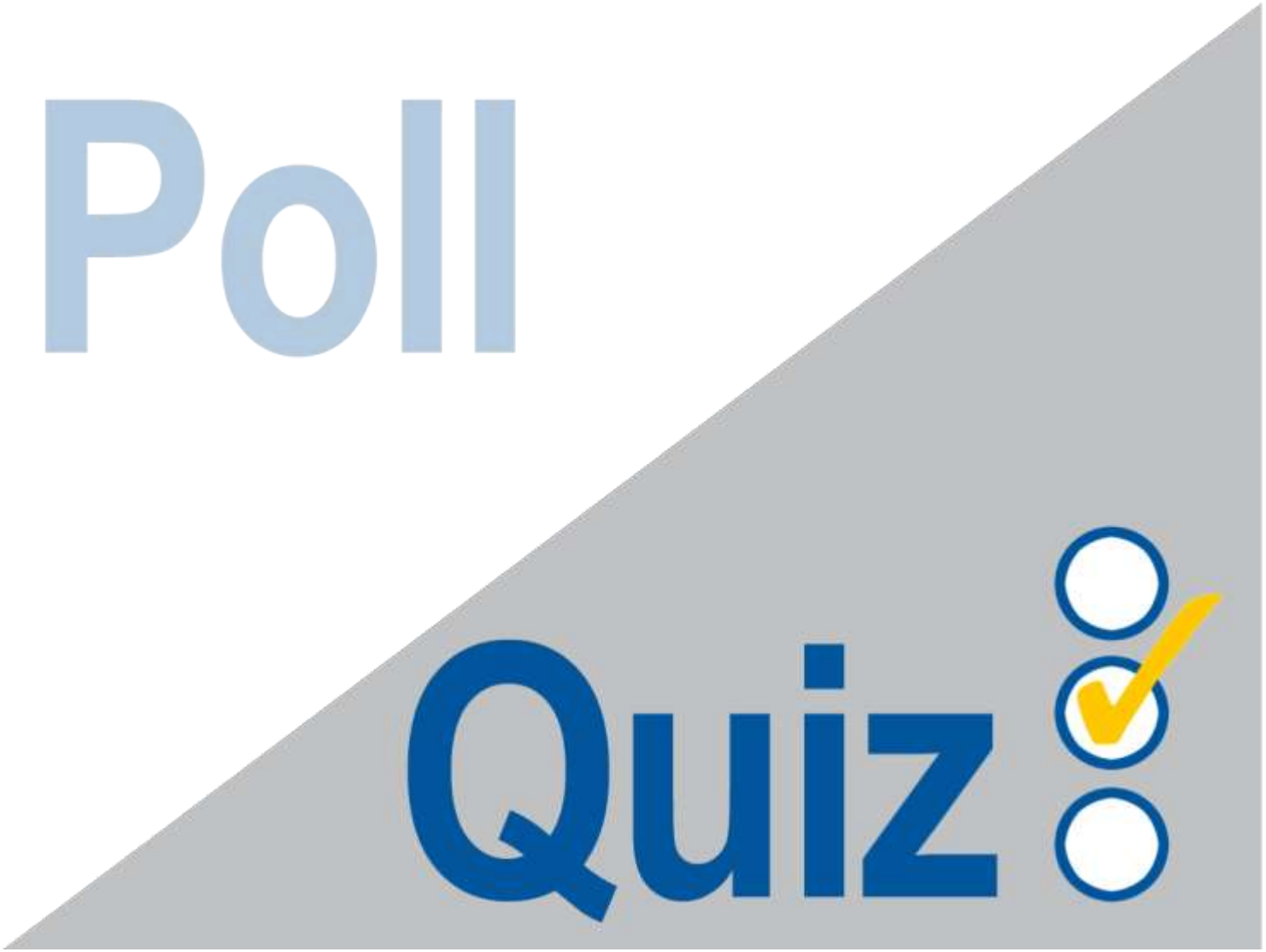
General form of the WHERE statement:

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.

Poll



11.08 Quiz

Which of the following WHERE statements have invalid syntax?

a.
```
where Salary ne .;
```

b.
```
where Hire_Date >= '01APR2008'd;
```

c.
```
where Country in (AU US);
```

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

e.
```
where Gender ne 'M' Salary >= 50000;
```

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

11.08 Quiz – Correct Answer

Which of the following WHERE statements have invalid syntax?

a.
```
where Salary ne .;
```

b.
```
where Hire_Date >= '01APR2008'd;
```

c.
```
where Country in (AU US);
```

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

e.
```
where Gender ne 'M' Salary >= 50000;
```

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

Subsetting Observations

```
proc print data=orion.sales;
   var First_Name Last_Name
       Job_Title Country Salary;
   where Salary > 75000;
run;
```

Obs	First_ Name	Last_Name	Job_Title	Country	Salary
1	Tom	Zhou	Sales Manager	AU	108255
2	Wilson	Dawes	Sales Manager	AU	87975
64	Harry	Highpoint	Chief Sales Officer	US	243190
163	Louis	Favaron	Senior Sales Manager	US	95090
164	Renee	Capachietti	Sales Manager	US	83505
165	Dennis	Lansberry	Sales Manager	US	84260

p111d08

Subsetting Observations

```
proc means data=orion.sales;
   var Salary;
   where Country = 'AU';
run;
```

The MEANS Procedure

Analysis Variable : Salary

N	Mean	Std Dev	Minimum	Maximum
63	30158.97	12699.14	25185.00	108255.00

p111d08


Poll
Quiz

Setup for the Poll

- Retrieve and submit program **p111a02**.
- View the log to determine how SAS handles multiple WHERE statements.

```
proc freq data=orion.sales;
   tables Gender;
   where Salary > 75000;
   where Country = 'US';
run;
```

11.09 Multiple Choice Poll

Which statement is true concerning the multiple WHERE statements?

a. All the WHERE statements are used.
b. None of the WHERE statements is used.
c. The first WHERE statement is used.
d. The last WHERE statement is used.

11.09 Multiple Choice Poll – Correct Answer

Which statement is true concerning the multiple WHERE statements?

a. All the WHERE statements are used.
b. None of the WHERE statements is used.
c. The first WHERE statement is used.
d. The last WHERE statement is used.

```
1000  proc freq data=orion.sales;
1001     tables Gender;
1002     where Salary > 75000;
1003     where Country = 'US';
NOTE: Where clause has been replaced.
1004  run;

NOTE: There were 102 observations read from the data set
      ORION.SALES.
      WHERE Country='US';
```

The BY Statement

For grouping observations in a report, the BY statement is used to produce separate sections of the report for each BY group.

General form of the BY statement:

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

The observations in the data set must be sorted by the variables specified in the BY statement.

Grouping Observations

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

proc print data=work.sort;
   by Country descending Gender;
run;
```


p111d09

Grouping Observations

Partial PROC PRINT Output

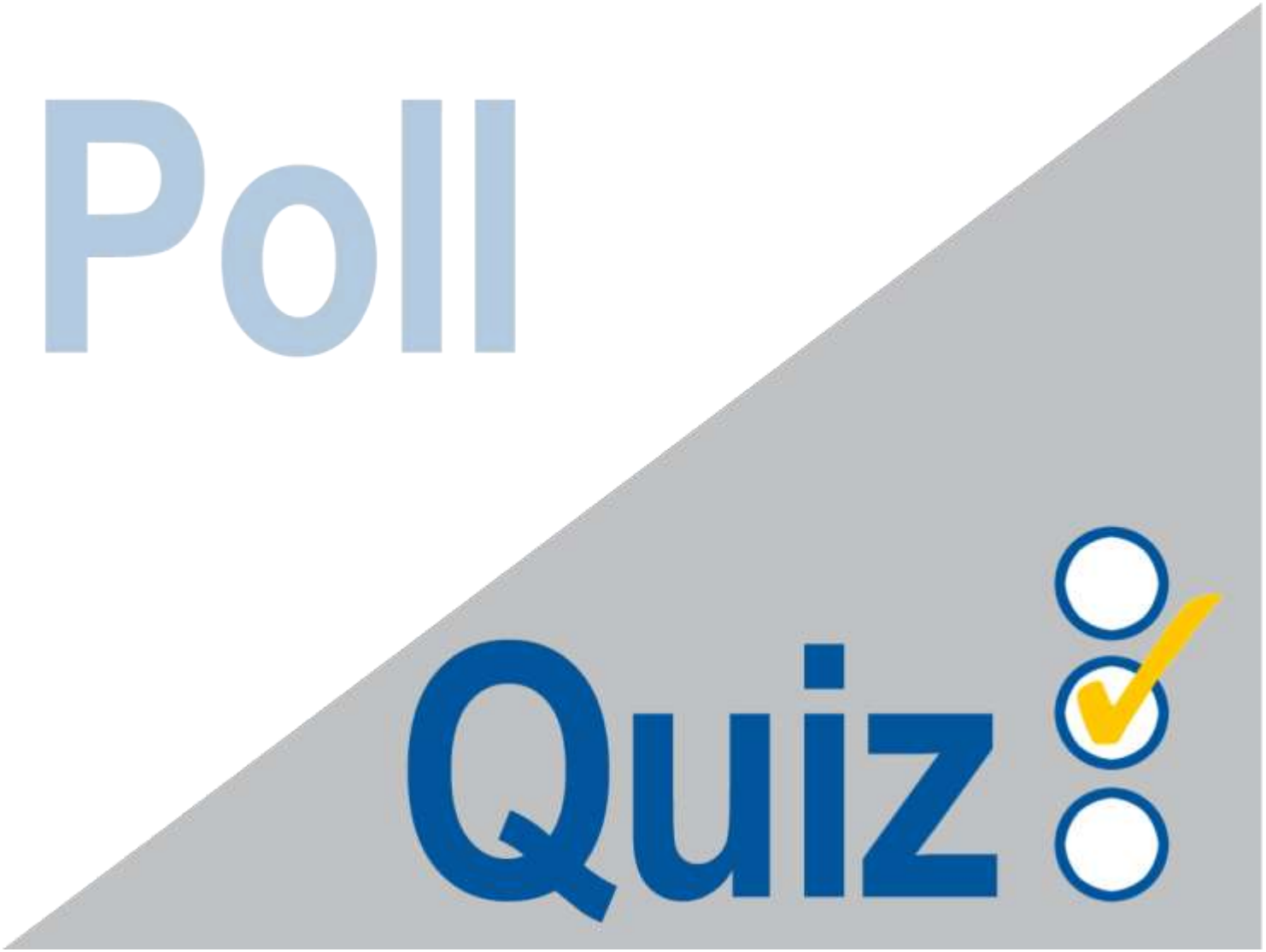
---------- Country=AU Gender=M ----------

Obs	Employee_ID	First_ Name	Last_Name	Salary
1	120145	Sandy	Aisbitt	26060
2	120144	Viney	Barbis	30265
3	120146	Wendall	Cederlund	25985

---------- Country=AU Gender=F ----------

Obs	Employee_ID	First_Name	Last_Name	Salary
37	120168	Selina	Barcoe	25275
38	120198	Meera	Body	28025
39	120149	Judy	Chantharasy	26390
40	120127	Sharryn	Clarkson	28100
41	120138	Shani	Duckett	25795
42	120121	Irenie	Elvish	26600
43	120154	Caterina	Hayawardhana	30490
44	120123	Kimiko	Hotstone	26190

Poll



11.10 Quiz

Which is a valid BY statement for the PROC FREQ step?

a. `by Country Gender;`

b. `by Gender Last_Name;`

c. `by Country;`

d. `by Gender;`

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

proc freq data=work.sort;
   tables Gender;
run;
```

11.10 Quiz – Correct Answer

Which is a valid BY statement for the PROC FREQ step?

a. `by Country Gender;`

b. `by Gender Last_Name;`

c. `by Country;`

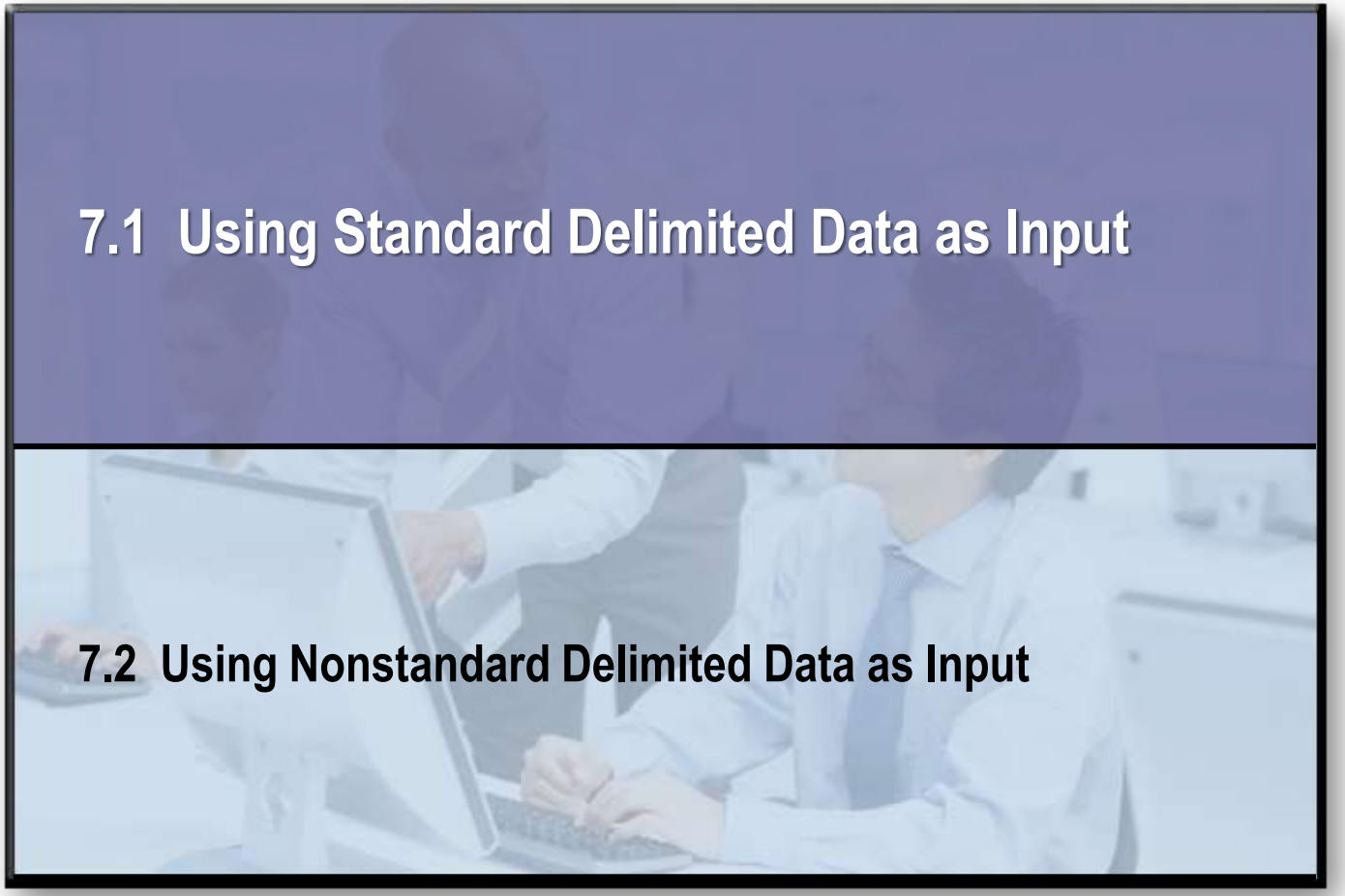
d. `by Gender;`

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

proc freq data=work.sort;
   tables Gender;
run;
```

Chapter 7: Reading Delimited Raw Data Files

Chapter 7: Reading Delimited Raw Data Files



Objectives

- Use the DATA step to create a SAS data set from a delimited raw data file.
- Examine the compilation and execution phases of the DATA step when reading a raw data file.
- Explicitly define the length of a variable by using the LENGTH statement.

Business Scenario

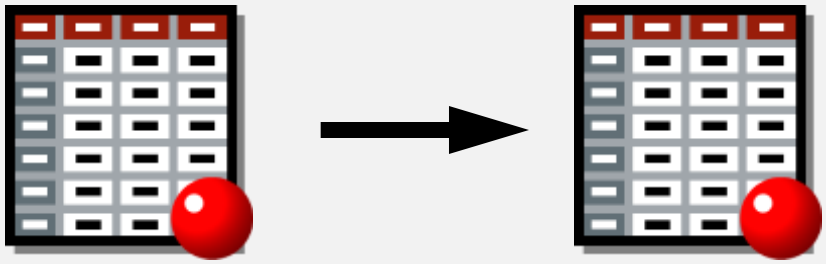
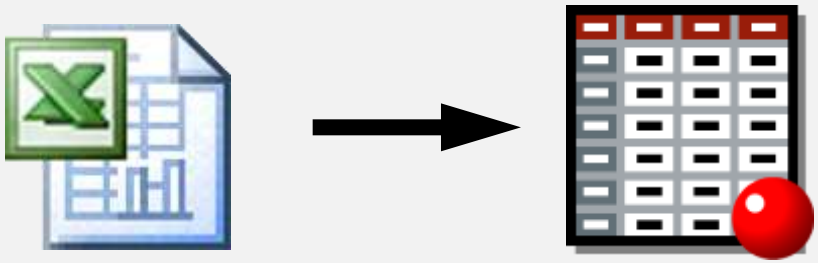
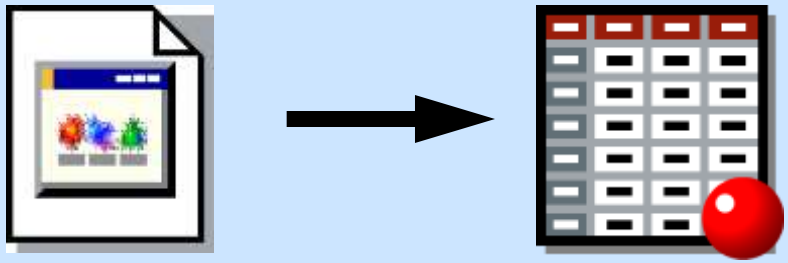
An existing data source contains information on Orion Star sales employees from Australia and the United States.

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;`

sales.csv

Partial sales.csv

comma delimited

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1954,07/01/1978
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1964,10/01/1985
120124,Lucian,Daymond,M,26480,Sales Rep. I,AU,13MAY1959,03/01/1979
120125,Fong,Hofmeister,M,32040,Sales Rep. IV,AU,06DEC1954,03/01/1979
120126,Satyakam,Denny,M,26780,Sales Rep. II,AU,20SEP1988,08/01/2006
120127,Sharryn,Clarkson,F,28100,Sales Rep. II,AU,04JAN1979,11/01/1998
120128,Monica,Kletschkus,F,30890,Sales Rep. IV,AU,14JUL1986,11/01/2006
120129,Alvin,Roebuck,M,30070,Sales Rep. III,AU,22NOV1964,10/01/1985
120130,Kevin,Lyon,M,26955,Sales Rep. I,AU,14DEC1984,05/01/2006
120131,Marinus,Surawski,M,26910,Sales Rep. I,AU,25SEP1979,01/01/2003
120132,Fancine,Kaiser,F,28525,Sales Rep. III,AU,05APR1949,10/01/1978
120133,Petrea,Soltau,F,27440,Sales Rep. II,AU,22APR1986,10/01/2006
120134,Sian,Shannan,M,28015,Sales Rep. II,AU,06JUN1949,01/01/1974
120135,Alexei,Platts,M,32490,Sales Rep. IV,AU,26JAN1969,10/01/1997
```

Business Scenario Syntax

Use the following statements to complete the scenario:

```
DATA output-SAS-data-set;
        LENGTH variable(s) $ length;
        INFILE 'raw-data-file-name';
        INPUT specifications;
        KEEP variable-list;
        LABEL variable = 'label'
                    variable = 'label'
                    variable = 'label';
        FORMAT variable(s) format;
RUN;
```

The DATA Statement (Review)

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

```
DATA output-SAS-data-set;
        INFILE 'raw-data-file-name';
        INPUT specifications;
        <additional SAS statements>
RUN;
```

The DATA statement can create temporary or permanent data sets.

The INFILE Statement

The *INFILE statement* identifies the physical name of the raw data file to read with an INPUT statement.

```
DATA output-SAS-data-set;
     INFILE 'raw-data-file-name';
     INPUT specifications;
     <additional SAS statements>
RUN;
```

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

The INFILE Statement

Examples:

Windows	`infile 's:\workshop\sales.csv';`
UNIX	`infile '/users/userid/sales.csv';`
z/OS (OS/390)	`infile '.workshop.rawdata(sales)';`
fileref	`infile rawin;`

The INPUT Statement

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

```
DATA output-SAS-data-set;
        INFILE 'raw-data-file-name';
        INPUT specifications;
        <additional SAS statements>
RUN;
```

The following are input specifications:

- column input
- formatted input
- list input


Poll
Quiz

List Input

To read with list input, data values

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

Partial **`sales.csv`**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1954,07/01/1978
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1964,10/01/1985
120124,Lucian,Daymond,M,26480,Sales Rep. I,AU,13MAY1959,03/01/1979
120125,Fong,Hofmeister,M,32040,Sales Rep. IV,AU,06DEC1954,03/01/1979
120126,Satyakam,Denny,M,26780,Sales Rep. II,AU,20SEP1988,08/01/2006
120127,Sharryn,Clarkson,F,28100,Sales Rep. II,AU,04JAN1979,11/01/1998
```

Delimiter

A space (blank) is the default delimiter.

The *DLM= option* can be added to the INFILE statement to specify an alternate delimiter.

```
DATA output-SAS-data-set;
      INFILE 'raw-data-file-name' DLM='delimiter';
      INPUT specifications;
      <additional SAS statements>
RUN;
```

Standard and Nonstandard Data

- *Standard data* is 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

- *Nonstandard data* is any data that SAS cannot read without a special instruction.

  Examples of nonstandard numeric data:

  5,823 (23) $67.23 01/12/1999 12MAY2006

List Input for Standard Data

List input specification:

INPUT *variable* <$>**;**

- Variables must be specified in the order that they appear in the raw data file, left to right.
- $ indicates to store a variable value as a character value rather than as a numeric value.
- The default length for character and numeric variables is eight bytes.

List Input for Standard Data

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1954,07/01/1978
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1964,10/01/1985
120124,Lucian,Daymond,M,26480,Sales Rep. I,AU,13MAY1959,03/01/1979
120125,Fong,Hofmeister,M,32040,Sales Rep. IV,AU,06DEC1954,03/01/1979
120126,Satyakam,Denny,M,26780,Sales Rep. II,AU,20SEP1988,08/01/2006
120127,Sharryn,Clarkson,F,28100,Sales Rep. II,AU,04JAN1979,11/01/1998
```

```
input Employee_ID First_Name $ Last_Name $
      Gender $ Salary Job_Title $ Country $;
```

Business Scenario

Create a temporary SAS data set named **Work.subset3** from the delimited raw data file named **sales.csv**.

```
data work.subset3;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Business Scenario

```
281  data work.subset3;
282     infile 'sales.csv' dlm=',';
283     input Employee_ID First_Name $ Last_Name $
284           Gender $ Salary Job_Title $ Country $;
285  run;

NOTE: The infile 'sales.csv' is:
      File Name=S:\Workshop\sales.csv,
      RECFM=V,LRECL=256

NOTE: 165 records were read from the infile 'sales.csv'.
      The minimum record length was 61.
      The maximum record length was 80.
NOTE: The data set WORK.SUBSET3 has 165 observations and 7 variables.
```

Business Scenario

```
proc print data=work.subset3;
run;
```

Partial PROC PRINT Output

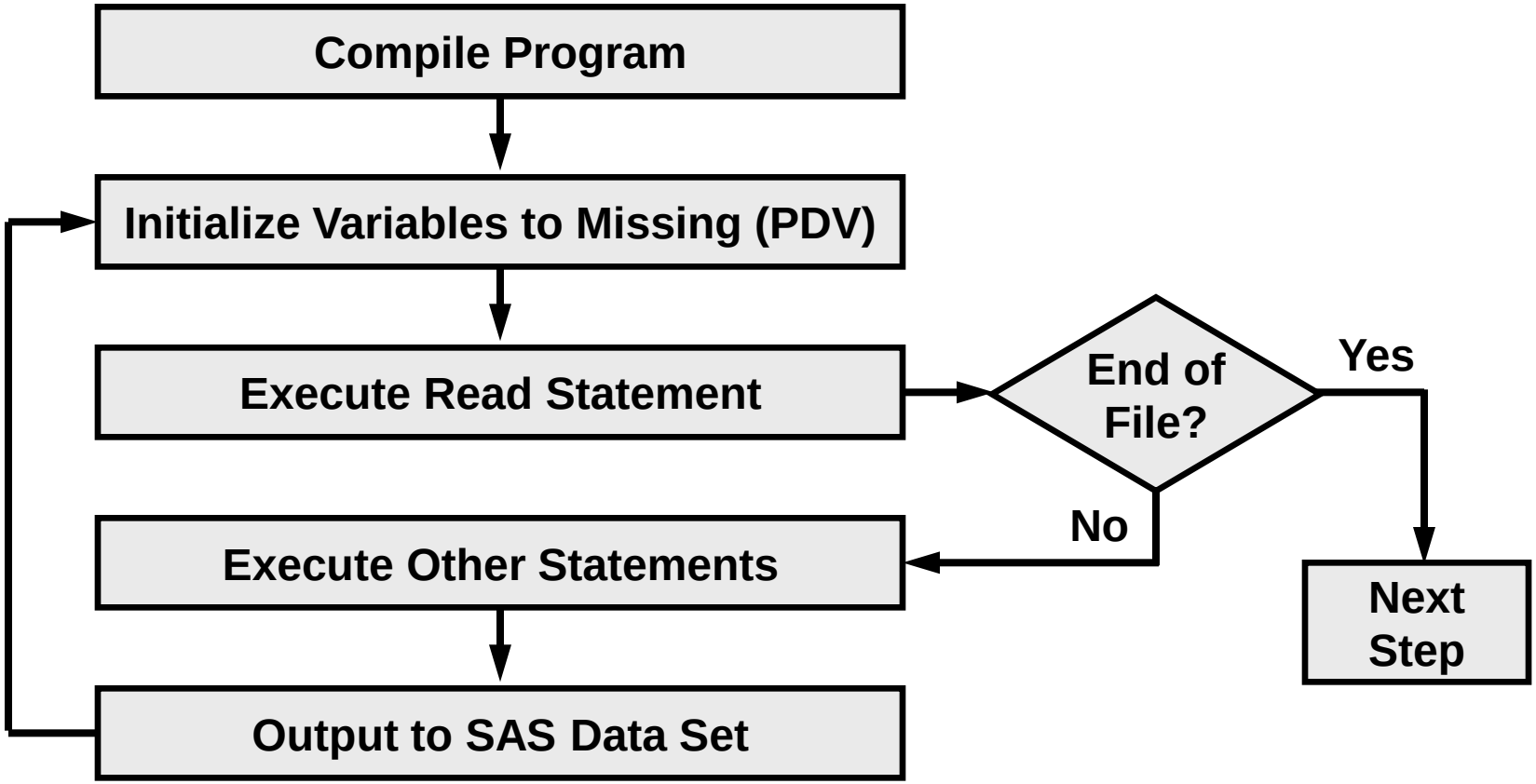
Obs	Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country
1	120102	Tom	Zhou	M	108255	Sales Ma	AU
2	120103	Wilson	Dawes	M	87975	Sales Ma	AU
3	120121	Irenie	Elvish	F	26600	Sales Re	AU
4	120122	Christin	Ngan	F	27475	Sales Re	AU
5	120123	Kimiko	Hotstone	F	26190	Sales Re	AU
6	120124	Lucian	Daymond	M	26480	Sales Re	AU
7	120125	Fong	Hofmeist	M	32040	Sales Re	AU
8	120126	Satyakam	Denny	M	26780	Sales Re	AU
9	120127	Sharryn	Clarkson	F	28100	Sales Re	AU
10	120128	Monica	Kletschk	F	30890	Sales Re	AU
11	120129	Alvin	Roebuck	M	30070	Sales Re	AU
12	120130	Kevin	Lyon	M	26955	Sales Re	AU

p107d01

DATA Step Processing

The DATA step is processed in two phases:

- compilation
- execution



Compilation

During the compilation phase, SAS

- checks the syntax of the DATA step statements
- creates an input buffer to hold the current raw data file record that is being processed
- creates a program data vector (PDV) to hold the current SAS observation
- creates the descriptor portion of the output data set.

Compilation

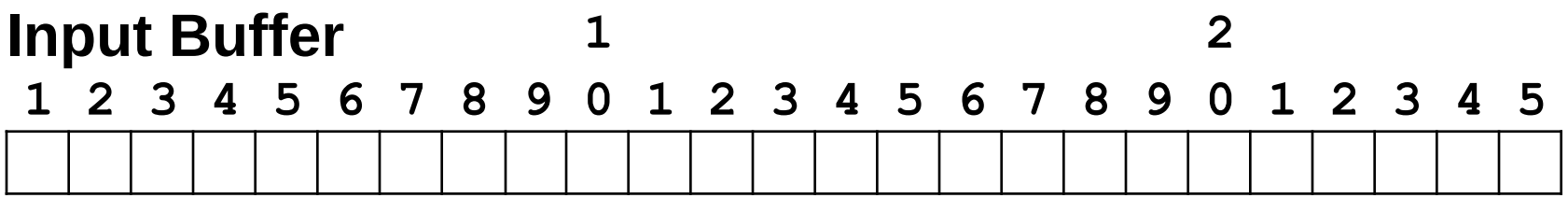
```
data work.subset3;
    infile 'sales.csv' dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```


...

Compilation

```
data work.subset3;
    infile 'sales.csv' dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer

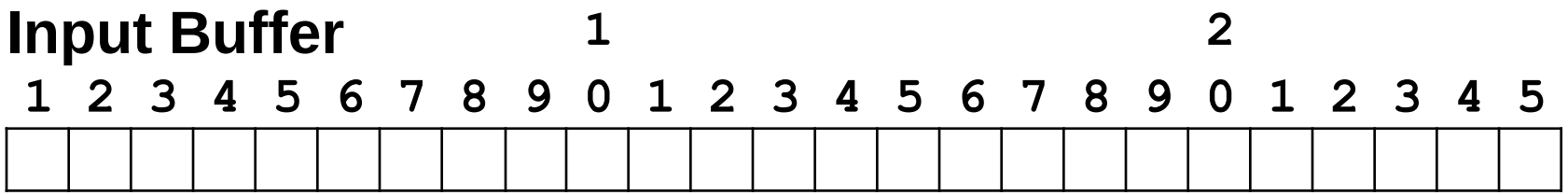


...

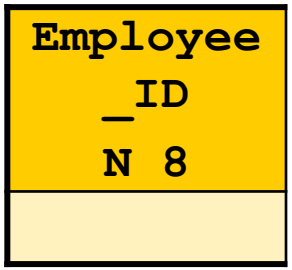
Compilation

```
data work.subset3;
    infile 'sales.csv' dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer



PDV



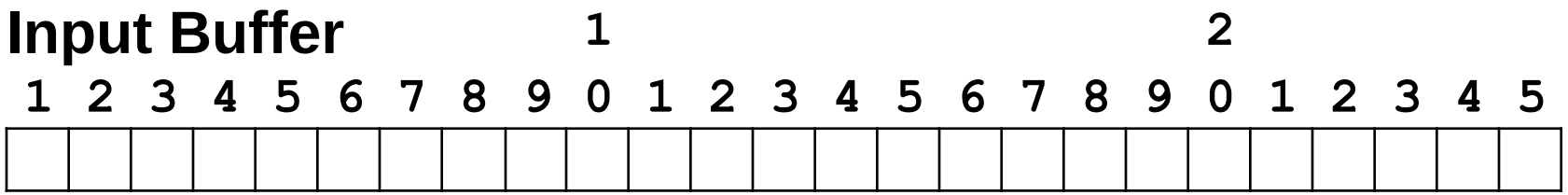
The default length for numeric variables is eight bytes.

...

Compilation

```
data work.subset3;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8

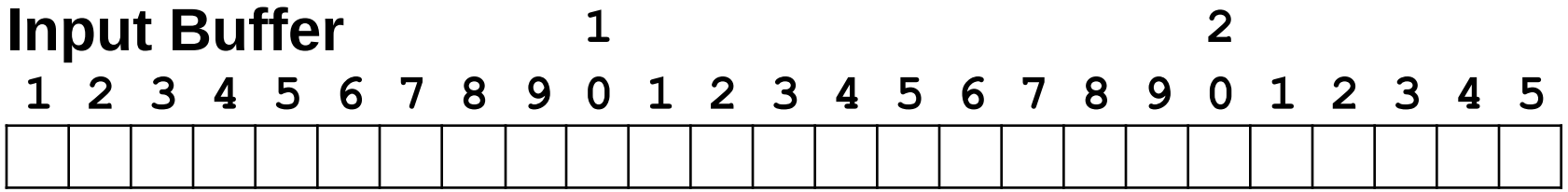
For list input, the default length for character variables is eight bytes.

...

Compilation

```
data work.subset3;
    infile 'sales.csv' dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer



PDV

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

...

Compilation

```
data work.subset3;
    infile 'sales.csv' dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Descriptor Portion Work.subset3

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


Poll
Quiz

7.02 Multiple Choice Poll

Which statement is true?

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

7.02 Multiple Choice Poll – Correct Answer

Which statement is true?

a. An input buffer is only created if you are reading data from a raw data file.

b. The PDV at compile time holds the variable name, type, byte size, and initial value.

c. The descriptor portion is the first item that is created at compile time.

Execution

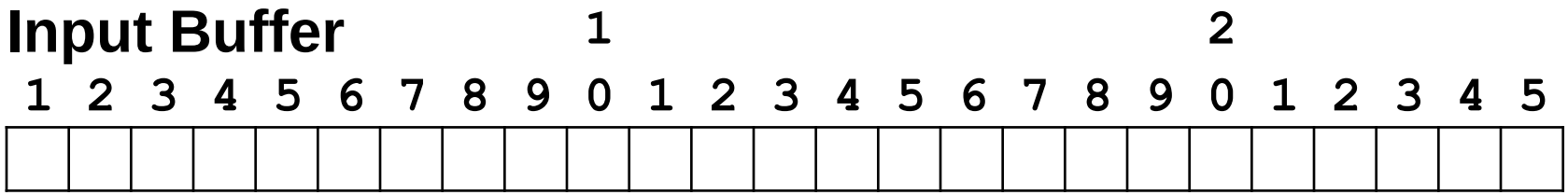
Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
   infile 'sa                    ,';
   input Empl                    Name $
         Last_                   $
         Salary Job_Title $
         Country $;
run;
```

Initialize PDV

Input Buffer



PDV

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

...

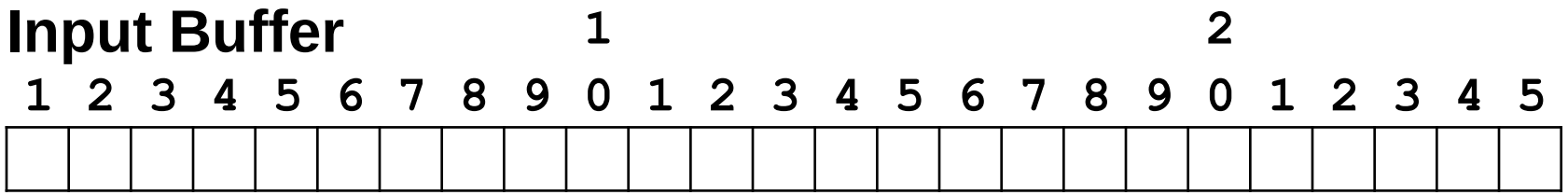
Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer



PDV

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

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

									1												2			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

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

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

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

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

```
                                  1
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
1 2 0 1 0 2 , T o m , Z h o u , M , 1 0 8 2 5 5 ,
```

PDV

Employee_ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_ Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

...

Execution

Output SAS Data Set after First Iteration of DATA Step

Work.subset3

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
   infile 'sa                    ;
   input Empl                    me $
         Last_                   |
         Salary Job_Title $
         Country $;
run;
```

Reinitialize PDV

Input Buffer

									1												2			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

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

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

									1												2			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

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

...

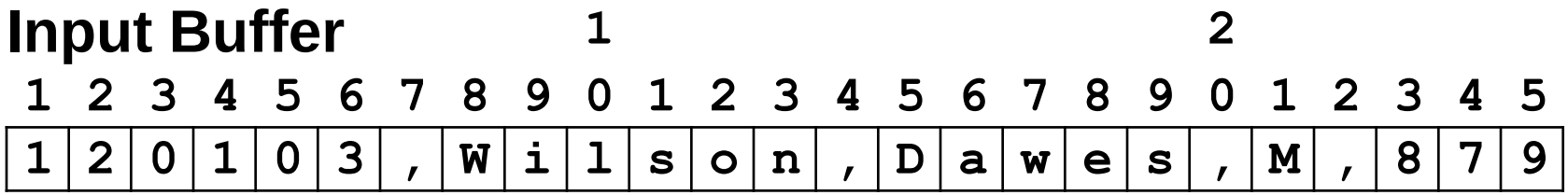
Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer



PDV

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

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer

									1												2			
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120103	Wilson	Dawes	M	87975	Sales Ma	AU

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset3;
  infile 'sales.csv' dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

									1															
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120103	Wilson	Dawes	M	87975	Sales Ma	AU

...

Execution

Output SAS Data Set after Second Iteration of DATA Step

Work.subset3

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU

...

Execution

Partial `sales.csv`

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

Continue until EOF

```
infile 'sales.csv' dlm=',';
input Employee_ID First_Name $
      Last_Name $ Gender $
      Salary Job_Title $
      Country $;
run;
```

Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
1 2 0 1 0 3 , W i l s o n , D a w e s , M , 8 7 9
```

PDV

Employee_ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_ Title $ 8	Country $ 8
120103	Wilson	Dawes	M	87975	Sales Ma	AU


Poll
Quiz

7.03 Multiple Choice Poll

Which statement is true?

a. Data is read directly from the raw data file to the PDV.
b. At the bottom of the DATA step, the contents of the PDV are output to the output SAS data set.
c. When SAS returns to the top of the DATA step, any variable coming from a SAS data set is set to missing.

7.03 Multiple Choice Poll – Correct Answer

Which statement is true?

a. Data is read directly from the raw data file to the PDV.
b. At the bottom of the DATA step, the contents of the PDV are output to the output SAS data set.
c. When SAS returns to the top of the DATA step, any variable coming from a SAS data set is set to missing.

The LENGTH Statement

The *LENGTH statement* defines the length of a variable explicitly.

General form of the LENGTH statement:

LENGTH *variable(s)* $ *length*;

Example:

```
length First_Name Last_Name $ 12
       Gender $ 1;
```

Business Scenario

Create a temporary SAS data set named **Work.subset3** from the delimited raw data file named **sales.csv**.

```
data work.subset3;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```


p107d02

Business Scenario

```
proc print data=work.subset3;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Gender	Job_Title	Country	Employee_ ID	Salary
1	Tom	Zhou	M	Sales Manager	AU	120102	108255
2	Wilson	Dawes	M	Sales Manager	AU	120103	87975
3	Irenie	Elvish	F	Sales Rep. II	AU	120121	26600
4	Christina	Ngan	F	Sales Rep. II	AU	120122	27475
5	Kimiko	Hotstone	F	Sales Rep. I	AU	120123	26190
6	Lucian	Daymond	M	Sales Rep. I	AU	120124	26480
7	Fong	Hofmeister	M	Sales Rep. IV	AU	120125	32040
8	Satyakam	Denny	M	Sales Rep. II	AU	120126	26780
9	Sharryn	Clarkson	F	Sales Rep. II	AU	120127	28100
10	Monica	Kletschkus	F	Sales Rep. IV	AU	120128	30890
11	Alvin	Roebuck	M	Sales Rep. III	AU	120129	30070
12	Kevin	Lyon	M	Sales Rep. I	AU	120130	26955

p107d02

Compilation

```
data work.subset3;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

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

...

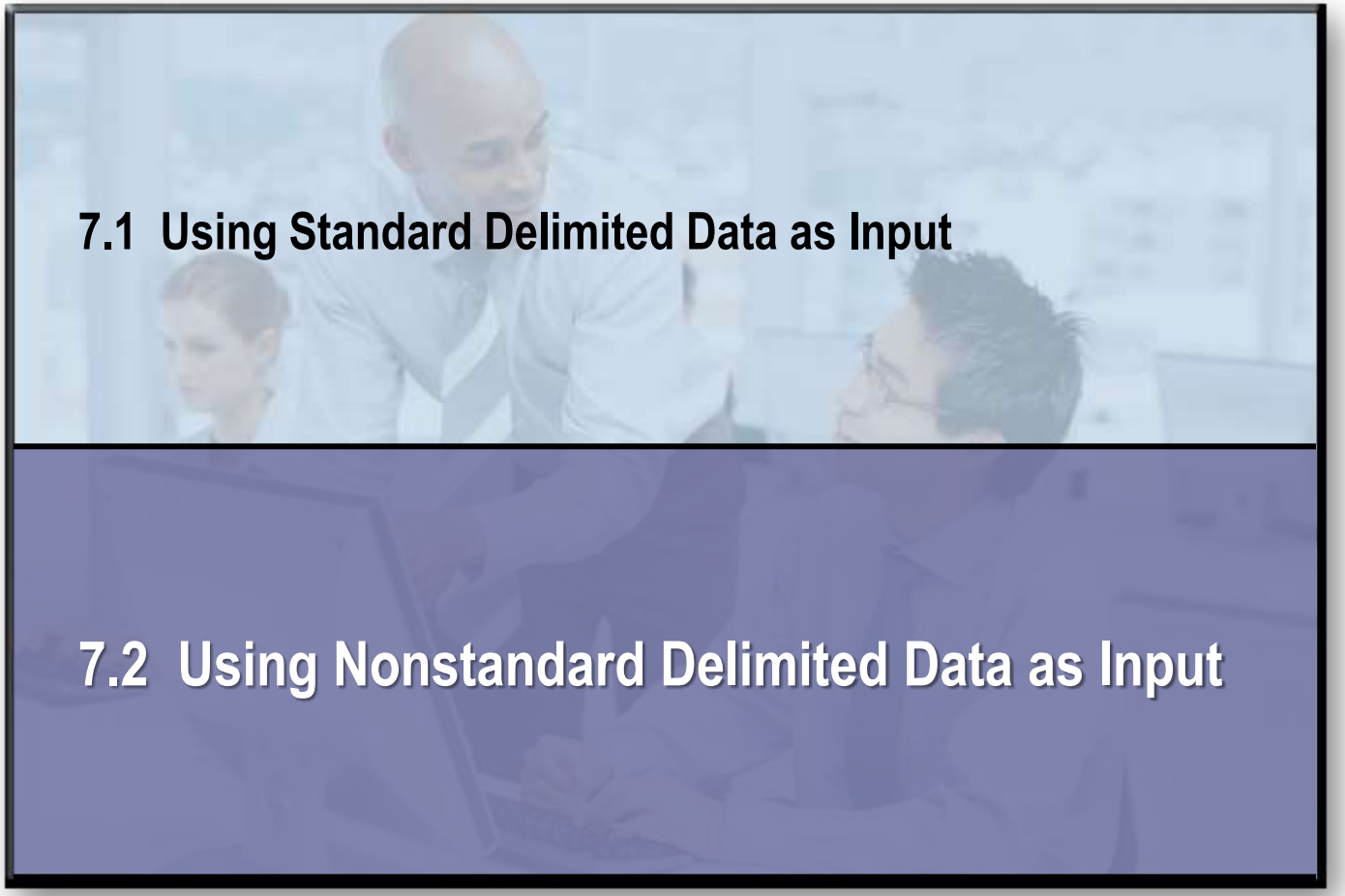
Compilation

```
data work.subset3;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

First_Name $ 12	Last_Name $ 18	Gender $ 1	Job_Title $ 25	Country $ 2	Employee_ID N 8	Salary N 8

Chapter 7: Reading Delimited Raw Data Files



Objectives

- Use informats to read nonstandard data.
- Add additional SAS statements to perform further processing in the DATA step.
- Use the DSD option with list input to read consecutive delimiters as missing values.
- Use the MISSOVER option to recognize missing values at the end of a record.

Standard and Nonstandard Data

- *Standard data* is 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

- *Nonstandard data* is any data that SAS cannot read without a special instruction.

  Examples of nonstandard numeric data:

  5,823 (23) $67.23 01/12/1999 12MAY2006

List Input for Nonstandard Data

List input specification:

INPUT *variable* <$> *variable* < *:informat* >**;**

- The : format modifier enables you to use an informat to read nonstandard delimited data.
- An *informat* is an instruction that SAS uses to read data values into a variable.
- The width of the informat can be eliminated.
- For character variables, the width of the informat determines the variable length, if it has not been previously defined.

SAS Informats

SAS informats have the following form:

<$>*informat*<*w*>.<*d*>

$	indicates a character informat.
informat	names the SAS informat or user-defined informat.
w	specifies the number of columns to read in the input data.
.	is a required delimiter.
d	specifies an optional decimal scaling factor in the numeric informats.

SAS Informats

Selected SAS Informats:

Informat	Definition
$*w*.	reads standard character data.
w.*d*	reads standard numeric data.
COMMA*w*.*d* DOLLAR*w*.*d*	reads nonstandard numeric data and removes embedded commas, blanks, dollar signs, percent signs, and dashes.
COMMAX*w*.*d* DOLLARX*w*.*d*	reads nonstandard numeric data and removes embedded periods, blanks, dollar signs, percent signs, and dashes.
EUROX*w*.*d*	reads nonstandard numeric data and removes embedded characters in European currency.

SAS Informats

In list input, informats are used to convert nonstandard numeric data to SAS numeric values.

Informat	Raw Data Value	SAS Data Value
COMMA. DOLLAR.	$12,345	12345
COMMAX. DOLLARX.	$12.345	12345
EUROX.	€12.345	12345

SAS Informats

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

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

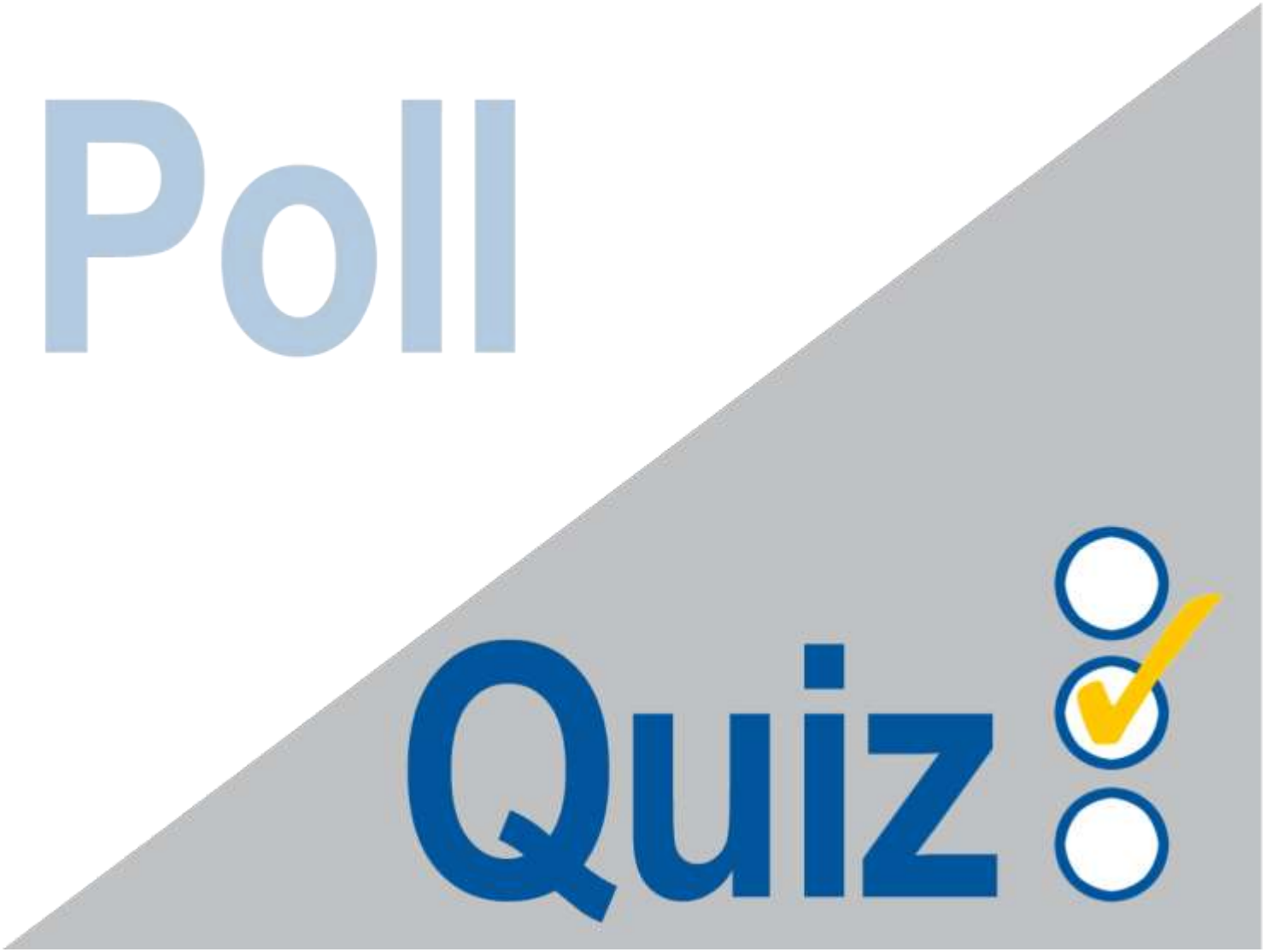
List Input for Nonstandard Data

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1954,07/01/1978
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1964,10/01/1985
120124,Lucian,Daymond,M,26480,Sales Rep. I,AU,13MAY1959,03/01/1979
120125,Fong,Hofmeister,M,32040,Sales Rep. IV,AU,06DEC1954,03/01/1979
120126,Satyakam,Denny,M,26780,Sales Rep. II,AU,20SEP1988,08/01/2006
120127,Sharryn,Clarkson,F,28100,Sales Rep. II,AU,04JAN1979,11/01/1998
```

```
input Employee_ID First_Name $ Last_Name $
      Gender $ Salary Job_Title $ Country $
      Birth_Date :date.
      Hire_Date :mmddyy.;
```

Poll



7.04 Quiz

Which INPUT statement correctly uses list input to read the space-delimited raw data file?

Raw Data

```
Donny 5MAY2008 25 FL $43,132.50
Margaret 20FEB2008 43 NC 65,150
```

a.
```
input name $ hired date. age
      state $ salary comma.;
```

b.
```
input name $ hired :date. age
      state $ salary :comma.;
```

7.04 Quiz – Correct Answer

Which INPUT statement correctly uses list input to read the space-delimited raw data file?

Raw Data

```
Donny 5MAY2008 25 FL $43,132.50
Margaret 20FEB2008 43 NC 65,150
```

a.
```
input name $ hired date. age
      state $ salary comma.;
```

b.
```
input name $ hired :date. age
      state $ salary :comma.;
```

Business Scenario

Create a temporary SAS data set named **Work.subset3** from the delimited raw data file named **sales.csv**.

```
data work.subset3;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $
         Birth_Date :date.
         Hire_Date :mmddyy.;
run;
```

Business Scenario

```
proc print data=work.subset3;
run;
```

Partial PROC PRINT Output

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

p107d03

Additional SAS Statements

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

```
data work.subset3;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile 'sales.csv' dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $
         Birth_Date :date.
         Hire_Date :mmddyy.;
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary dollar12. Hire_Date monyy7.;
run;
```

p107d04

Additional SAS Statements

```
proc print data=work.subset3 label;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Sales Title	Salary	Date Hired
1	Tom	Zhou	Sales Manager	$108,255	JUN1989
2	Wilson	Dawes	Sales Manager	$87,975	JAN1974
3	Irenie	Elvish	Sales Rep. II	$26,600	JAN1974
4	Christina	Ngan	Sales Rep. II	$27,475	JUL1978
5	Kimiko	Hotstone	Sales Rep. I	$26,190	OCT1985
6	Lucian	Daymond	Sales Rep. I	$26,480	MAR1979
7	Fong	Hofmeister	Sales Rep. IV	$32,040	MAR1979
8	Satyakam	Denny	Sales Rep. II	$26,780	AUG2006
9	Sharryn	Clarkson	Sales Rep. II	$28,100	NOV1998
10	Monica	Kletschkus	Sales Rep. IV	$30,890	NOV2006

p107d04

Additional SAS Statements

- The WHERE statement is used to obtain a subset of observations from an input data set.
- The WHERE statement cannot be used to select records from a raw data file.

The subsetting IF can subset data that is in the PDV.

Missing Values in the Middle of the Record

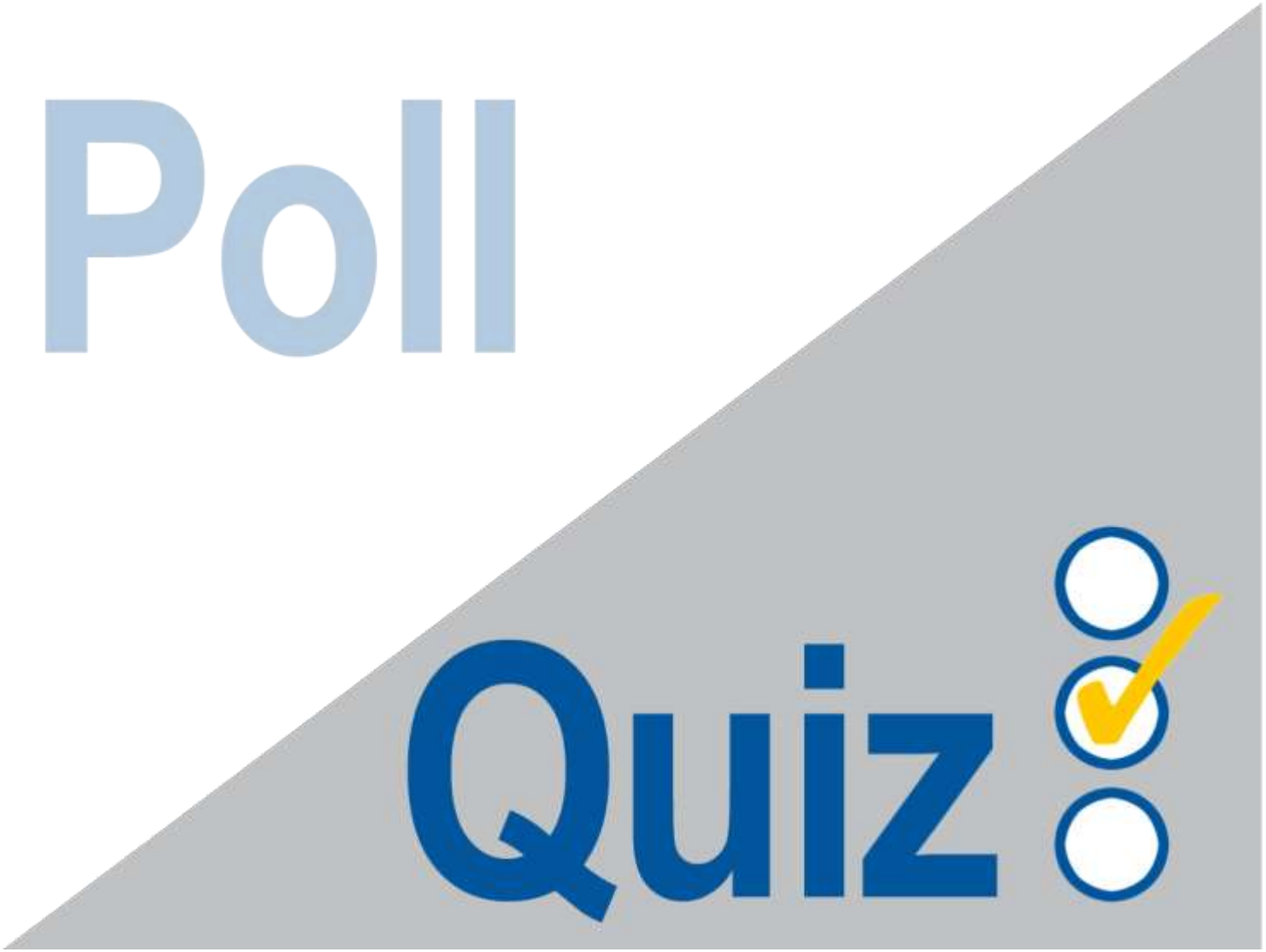
Each record in **`phone2.csv`** has a contact name, phone number, and a mobile number. The phone number is missing from some of the records.

Missing data is indicated by two consecutive delimiters.

`phone2.csv`

```
          1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,,(919) 271-4592
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,,(714) 644-9090
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

Poll



7.05 Quiz

- Open and submit **p107a01**.
- Examine the SAS log.
- How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone2.csv' dlm=',';
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

7.05 Quiz – Correct Answer

- Open and submit **p107a01**.
- Examine the SAS log.
- How many input records were read and how many observations were created?

Five records were read from the input file and three observations were created.

Unexpected Results

The missing phone numbers caused unexpected results in the output.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	Cornelia Krahl
Karen Ballinger	(714) 344-4321	Elke Wallstab

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone2.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: SAS went to a new line when INPUT statement reached
past the end of a line.
NOTE: The data set WORK.CONTACTS has 3 observations and 3
variables.
```

Consecutive Delimiters in List Input

By default, list input treats two or more consecutive delimiters as a single delimiter and not treated as a missing value.

The two consecutive commas are not being read as a missing value.

phone2.csv

```
          1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,, (919) 271-4592
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,, (714) 644-9090
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

The DSD Option

The DSD option for the INFILE statement

- sets the default delimiter to a comma
- treats consecutive delimiters as missing values
- enables SAS to read values with embedded delimiters if the value is surrounded by quotation marks.

General form of a DSD option in an INFILE statement:

INFILE '*raw-data-file-name*' DSD**;**

Using the DSD Option

Adding the DSD option will correctly read the **phone2.csv** data file.

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone2.csv' dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

- The DLM=',' option is no longer needed in the INFILE statement because the DSD option sets the default delimiter to a comma.

Results

Adding the DSD option gives the expected results.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano		(919) 271-4592
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger		(714) 644-9090
Elke Wallstab	(910) 763-5561	(910) 545-3421

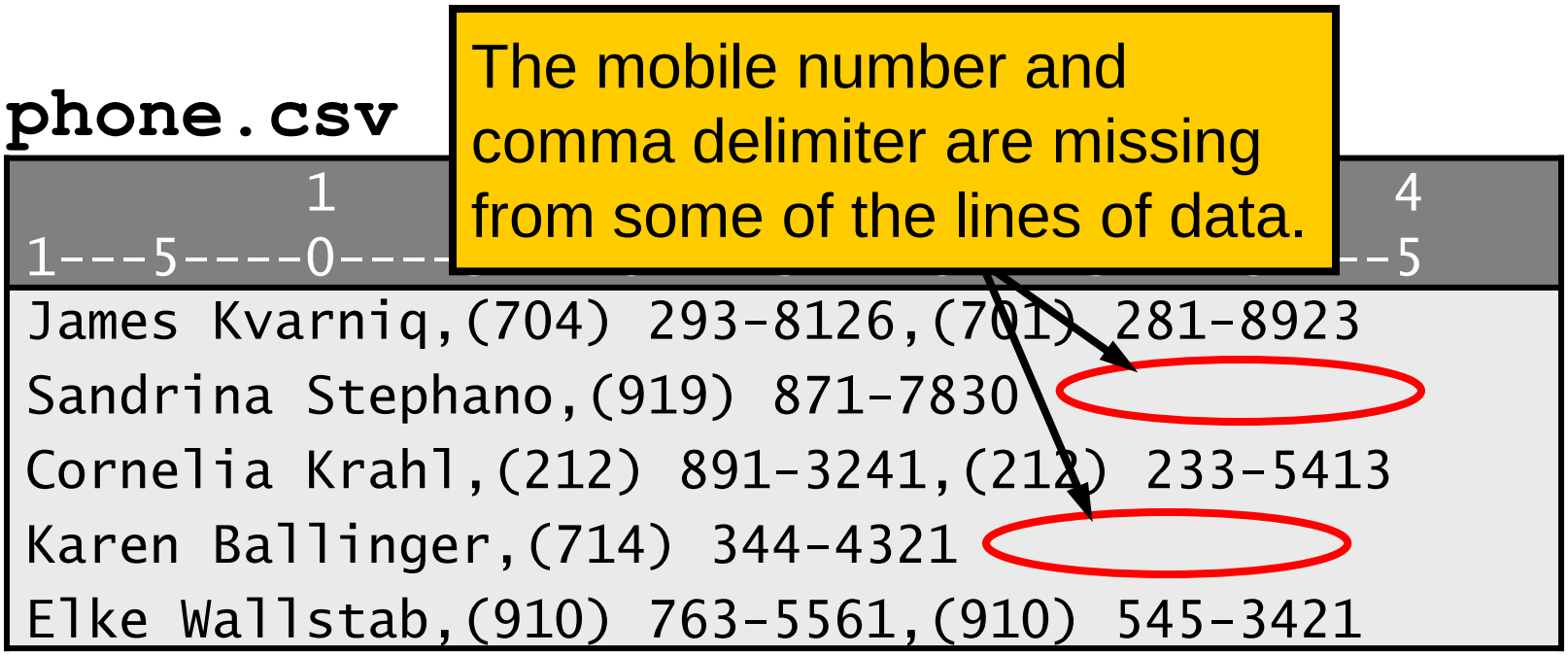
Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone2.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and
3 variables.
```

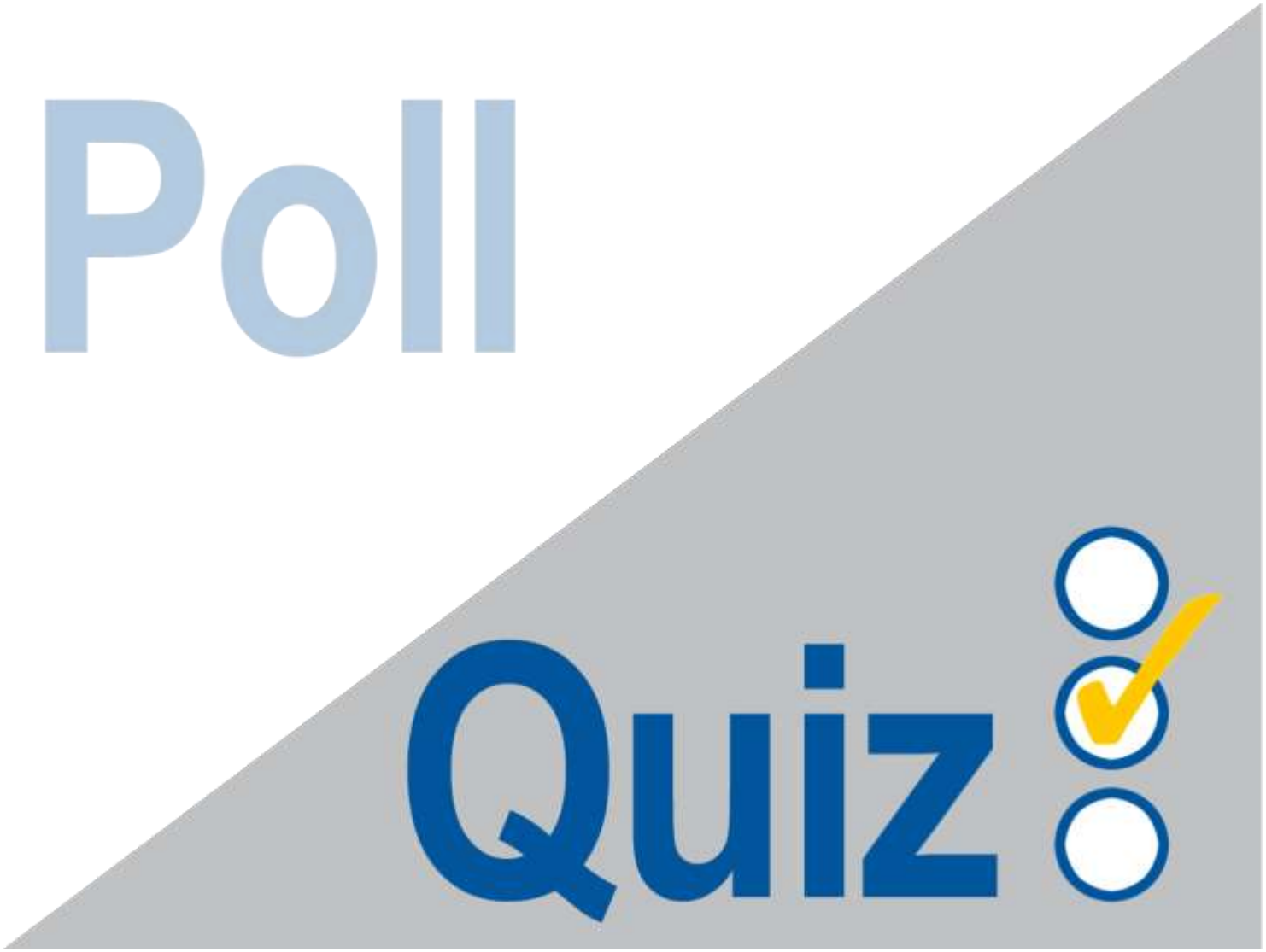
Missing Values at the End of a Record

The data values in **`phone.csv`** are separated by commas. Each record has a contact name, and then a phone number, and finally a mobile number.

`phone.csv`



Poll



7.06 Quiz

Open and submit **p107a02**. Examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone.csv' dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

7.06 Quiz – Correct Answer

Open and submit **p107a02**. Examine the SAS log. How many input records were read and how many observations were created?

Five records were read from the input file, and three observations were created.

Unexpected Results

The missing mobile phone numbers caused unexpected results in the output.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	Cornelia Krahl
Karen Ballinger	(714) 344-4321	Elke Wallstab

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: SAS went to a new line when INPUT statement
reached past the end of a line.
NOTE: The data set WORK.CONTACTS has 3 observations and
3 variables.
```

Missing Values at the End of a Record

By default, when there is missing data at the end of a row, SAS does the following:

- loads the next record to finish the observation
- writes a note to the log

The MISSOVER Option

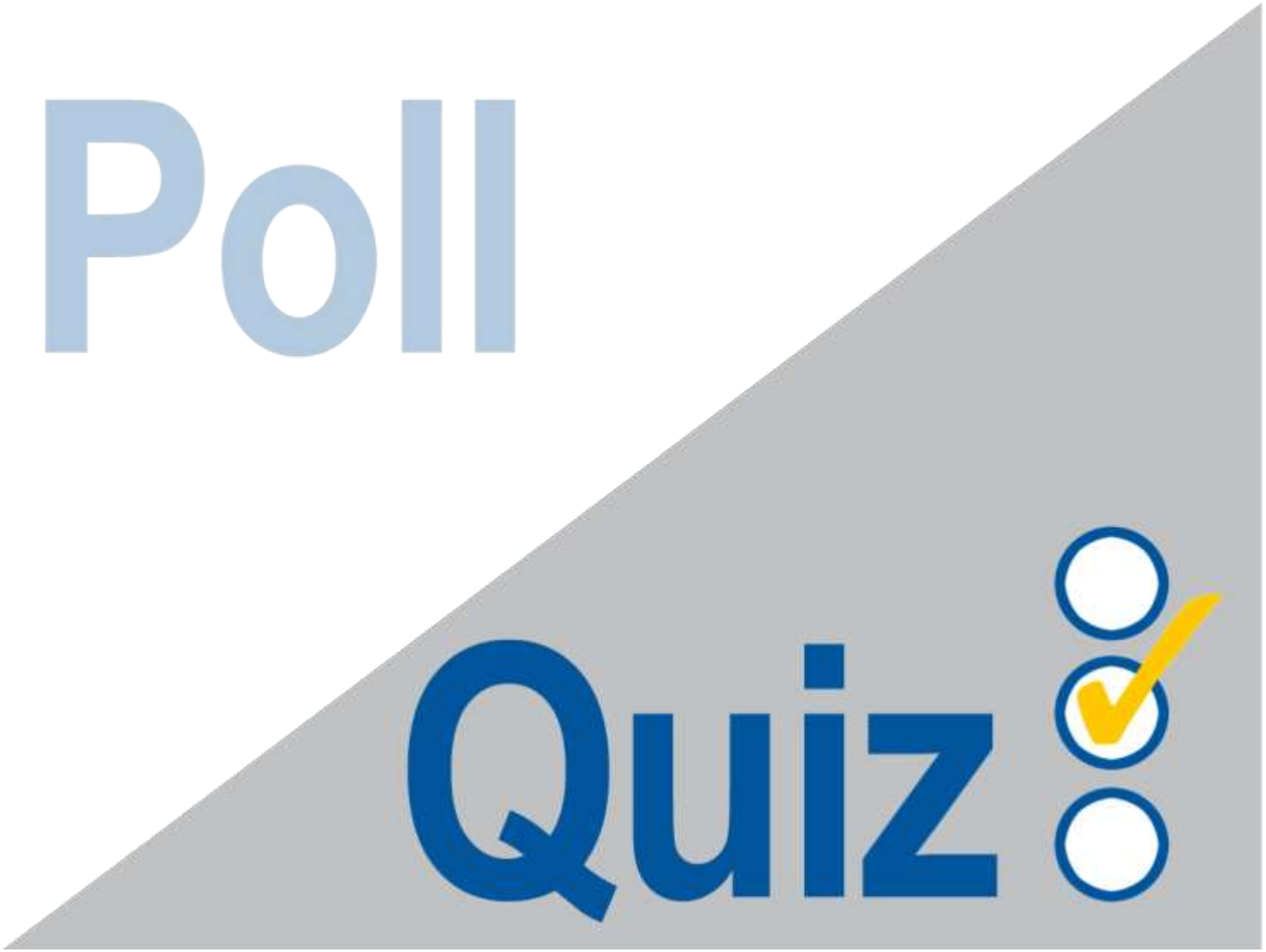
The MISSOVER option prevents SAS from loading a new record when the end of the current record is reached.

General form of an INFILE statement with a MISSOVER option:

INFILE '*raw-data-file-name*' MISSOVER**;**

If SAS reaches the end of the row without finding values for all fields, variables without values are set to missing.

Poll



7.07 Quiz

Open **p107a03** and add the MISSOVER option to the INFILE statement. Submit the program and examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone.csv' dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

7.07 Quiz – Correct Answer

Open **p107a03** and add the MISSOVER option to the INFILE statement. Submit the program and examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone.csv' dsd missover;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

Five input records were read and five observations created.

p107a03s

Results

Adding the MISSOVER option gives the expected results.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger	(714) 344-4321	
Elke Wallstab	(910) 763-5561	(910) 545-3421

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and
3 variables.
```