

# STAT604 SAS Lesson 10

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Producing Basic Reports

The PRINT Procedure – Prep Guide Chapter 6

# PRINT Procedure

By default, PROC PRINT displays all observations, all variables, and an Obs column on the left side.

```
proc print data=cert.therapy;  
run;
```

Partial PROC PRINT Output

| Obs | Date    | AerClass | WalkJogRun | Swim |
|-----|---------|----------|------------|------|
| 1   | JAN2012 | 56       | 78         | 14   |
| 2   | FEB2012 | 32       | 109        | 19   |
| 3   | MAR2012 | 35       | 106        | 22   |
| 4   | APR2012 | 47       | 115        | 24   |
| 5   | MAY2012 | 55       | 121        | 31   |

Statements and options can be added to the PRINT procedure to modify the default behavior.

# Viewing the Output

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

| Obs | Customer_ID        | Customer_Name           | Customer_Gender | Customer_Country | Customer_Group          |
|-----|--------------------|-------------------------|-----------------|------------------|-------------------------|
| 37  | 79                 | Najma Hicks             | F               | US               | Orion Club members      |
| 58  | 11171              | Bill Cuddy              | M               | CA               | Orion Club Gold members |
| 66  | 46966              | Lauren Krasowski        | F               | CA               | Orion Club members      |
| ... |                    |                         |                 |                  |                         |
| 76  | 70210              | Alex Santinello         | M               | CA               | Orion Club members      |
| Obs | Customer_Age_Group | Customer_Type           |                 |                  |                         |
| 37  | 15-30 years        | Orion Club members      | medium activity |                  |                         |
| 58  | 15-30 years        | Orion Club Gold members | low activity    |                  |                         |
| 66  | 15-30 years        | Orion Club members      | high activity   |                  |                         |
| ... |                    |                         |                 |                  |                         |
| 76  | 15-30 years        | Orion Club members      | medium activity |                  |                         |

The Obs column helps identify observations that span multiple lines in a report.

# VAR Statement

The VAR statement selects variables to include in the report and specifies their order.

```
proc print data=cert.admit;
  var age height weight fee;
run;
```

VAR *variable(s)*;

Partial PROC PRINT Output

| Obs | Age | Height | Weight | Fee    |
|-----|-----|--------|--------|--------|
| 1   | 27  | 72     | 168    | 85.20  |
| 2   | 34  | 66     | 152    | 124.80 |
| 3   | 31  | 61     | 123    | 149.75 |
| 4   | 43  | 63     | 137    | 149.75 |
| 5   | 51  | 71     | 158    | 124.80 |

# Suppressing the Obs Column

Use the NOOBS option in the PROC PRINT statement to suppress the Obs column.

```
proc print data=cert.admit noobs;  
  var age height weight fee;  
run;
```

PROC PRINT DATA=SAS-data-set NOOBS;

PROC PRINT Partial Output

| Age | Height | Weight | Fee    |
|-----|--------|--------|--------|
| 27  | 72     | 168    | 85.20  |
| 34  | 66     | 152    | 124.80 |
| 31  | 61     | 123    | 149.75 |
| 43  | 63     | 137    | 149.75 |
| 51  | 71     | 158    | 124.80 |

# ID Statement

The *ID statement* specifies the variable or variables to print at the beginning of each row instead of an observation number.

```
proc print data=cert.reps;
  id lastname idnum;
run;
```

ID variables;

| LastName  | IDnum | FirstName | City       | State | Sex | JobCode | Salary   | Birth   | Hired   | HomePhone    |
|-----------|-------|-----------|------------|-------|-----|---------|----------|---------|---------|--------------|
| CASTON    | 1269  | FRANKLIN  | STAMFORD   | CT    | M   | NA1     | 41690.00 | 06MAY60 | 01DEC80 | 203/781-3335 |
| FERNANDEZ | 1935  | KATRINA   | BRIDGEPORT | CT    |     | NA2     | 51081.00 | 31MAR42 | 19OCT69 | 203/675-2962 |
| NEWKIRK   | 1417  | WILLIAM   | PATERSON   | NJ    | ,   | NA2     | 52270.00 | 30JUN52 | 10MAR77 | 201/732-6611 |
| NORRIS    | 1839  | DIANE     | NEW YORK   | YN    | F   | NA1     | 43433.00 | 02DEC58 | 06JUL81 | 718/384-1767 |



Choose ID variables that uniquely identify observations.

# ID Statement

| IDnum | LastName   | FirstName | City       | State | Sex | JobCode | Salary   | Birth   | Hired   |
|-------|------------|-----------|------------|-------|-----|---------|----------|---------|---------|
| 1269  | CASTON     | FRANKLIN  | STAMFORD   | CT    | M   | NA1     | 41690.00 | 06MAY60 | 01DEC80 |
| 1935  | FERNANDEZ  | KATRINA   | BRIDGEPORT | CT    |     | NA2     | 51081.00 | 31MAR42 | 19OCT69 |
| 1417  | NEWKIRK    | WILLIAM   | PATERSON   | NJ    | .   | NA2     | 52270.00 | 30JUN52 | 10MAR77 |
| 1839  | NORRIS     | DIANE     | NEW YORK   | YN    | F   | NA1     | 43433.00 | 02DEC58 | 06JUL81 |
| 1111  | RHODES     | JEREMY    | PRINCETON  | NJ    | M   | NA1     | 40586.00 | 17JUL61 | 03NOV80 |
| 1352  | RIVERS     | SIMON     | NEW YORK   | NY    | M   | NA2     | 5379.80  | 05DEC48 | 19OCT74 |
| 1332  | STEPHENSON | ADAM      | BRIDGEPORT | CT    | M   | NA1     | 42178.00 | 20SEP58 | 07JUN79 |
| 1443  | WELLS      | AGNES     | STAMFORD   | CT    | F   | NA1     | 422.74   | 20NOV56 | 01SEP79 |

| IDnum | LastName   | HomePhone    | birth_month | area |
|-------|------------|--------------|-------------|------|
| 1269  | CASTON     | 203/781-3335 | 5           | 203  |
| 1935  | FERNANDEZ  | 203/675-2962 | 3           | 203  |
| 1417  | NEWKIRK    | 201/732-6611 | 6           | 201  |
| 1839  | NORRIS     | 718/384-1767 | 12          | 718  |
| 1111  | RHODES     | 201/812-1837 | 7           | 201  |
| 1352  | RIVERS     | 718/383-3345 | 12          | 718  |
| 1332  | STEPHENSON | 203/675-1497 | 9           | 203  |
| 1443  | WELLS      | 203/781-5546 | 11          | 203  |

# ID Statement with VAR

The *ID* and *VAR* statements can be used together.

```
proc print data=cert.reps;
  id idnum lastname;
  var idnum sex jobcode salary;
run;
```

| IDnum | LastName  | IDnum | Sex | JobCode | Salary   |
|-------|-----------|-------|-----|---------|----------|
| 1269  | CASTON    | 1269  | M   | NA1     | 41690.00 |
| 1935  | FERNANDEZ | 1935  |     | NA2     | 51081.00 |
| 1417  | NEWKIRK   | 1417  | ,   | NA2     | 52270.00 |
| 1839  | NORRIS    | 1839  | F   | NA1     | 43433.00 |



Columns in both statements will be repeated.

# WHERE Statement (Review)

The *WHERE statement* selects observations that meet the criteria specified in the WHERE expression.

```
proc print data=cert.reps;
  var LastName FirstName City Salary;
  where city contains 'OR';
run;
```

WHERE WHERE-expression;

# Viewing the Log

Only 6 of the 8 observations from **cert.reps** were selected by the WHERE statement.

```
25 proc print data=cert.reps;  
26   var LastName FirstName City Salary;  
27   where city contains 'OR';  
28 run;
```

NOTE: There were 6 observations read from the data set CERT.REPS.  
 WHERE city contains 'OR';

# Viewing the Output

## PROC PRINT Output

| Obs | LastName   | FirstName | City       | Salary   |
|-----|------------|-----------|------------|----------|
| 1   | CASTON     | FRANKLIN  | STAMFORD   | 41690.00 |
| 2   | FERNANDEZ  | KATRINA   | BRIDGEPORT | 51081.00 |
| 4   | NORRIS     | DIANE     | NEW YORK   | 43433.00 |
| 6   | RIVERS     | SIMON     | NEW YORK   | 5379.80  |
| 7   | STEPHENSON | ADAM      | BRIDGEPORT | 42178.00 |
| 8   | WELLS      | AGNES     | STAMFORD   | 422.74   |



original observation numbers

# Controlling Which Observations Are Read

- By default, SAS processes every observation in a SAS data set, from the first observation to the last.
- The FIRSTOBS= and OBS= data set options can be used to control which observations are processed.
- The FIRSTOBS= and OBS= options are used with input data sets on DATA and PROC steps.
- You cannot use either option with output data sets.



These temporarily override system option values FIRSTOBS=1 and OBS=max

# The OBS= Data Set Option

The OBS= data set option specifies an ending point for processing an input data set.

General form of OBS= data set option:

**SAS-data-set(OBS=n)**

This option specifies the number of the last observation to process, not how many observations should be processed.

# Using the OBS= Data Set Option

This OBS= data set option causes the DATA step to stop processing after observation 100.

```
data australia;
  set orion.employee_addresses (obs=100);
  if Country='AU' then output;
run;
```

Partial SAS Log

```
NOTE: There were 100 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.AUSTRALIA has 24 observations and
      9 variables.
```

# The FIRSTOBS= Data Set Option

The FIRSTOBS= data set option specifies a starting point for processing an input data set. This option specifies the number of the first observation to process.

General form of the FIRSTOBS= data set option:

**SAS-data-set (FIRSTOBS=*n*)**

FIRSTOBS= and OBS= are often used together to define a range of observations to be processed.

# Using OBS= and FIRSTOBS= Data Set Options

The FIRSTOBS= and OBS= data set options cause the SET statement below to read 51 observations from **orion.employee\_addresses**. Processing begins with observation 50 and ends after observation 100.

```
data australia;
  set orion.employee_addresses
    (firstobs=50 obs=100);
  if Country='AU' then output;
run;
```

# Check the SAS Log

## Partial SAS Log

```
640 data australia;
641     set orion.employee_addresses(firsttobs=50 obs=100);
642     if Country='AU' then output;
643 run;
```

NOTE: There were 51 observations read from the data set  
ORION.EMPLOYEE\_ADDRESSES.

NOTE: The data set WORK.AUSTRALIA has 13 observations and  
9 variables.

# Using OBS= and FIRSTOBS= in a PROC Step

The FIRSTOBS= and OBS= data set options can also be used in SAS procedures. The PROC PRINT step below begins processing at observation 10 and ends after observation 15.

```
proc print data=cert.heart (firstobs=10 obs=15) ;  
run;
```

# Check the Output

## Partial SAS Log

```
29 proc print data=cert.heart (firstobs=10 obs=15);  
30 run;  
NOTE: There were 6 observations read from the data set  
CERT.HEART.
```

## PROC PRINT Output

| Obs | Patient | Sex | Survive | Shock    | Arterial | Heart | Cardiac | Urinary |
|-----|---------|-----|---------|----------|----------|-------|---------|---------|
| 10  | 509     | 2   | SURV    | OTHER    | 79       | 84    | 256     | 90      |
| 11  | 742     | 1   | DIED    | HYPOVOL  | 100      | 54    | 135     | 0       |
| 12  | 609     | 2   | DIED    | NONSHOCK | 93       | 101   | 260     | 90      |
| 13  | 318     | 2   | DIED    | OTHER    | 72       | 81    | 410     | 405     |
| 14  | 412     | 1   | SURV    | BACTER   | 61       | 87    | 296     | 44      |
| 15  | 601     | 1   | DIED    | BACTER   | 84       | 101   | 260     | 377     |

original observation numbers

# Adding a WHERE Statement

When the FIRSTOBS= or OBS= option and the WHERE statement are used together, the following occurs:

- the subsetting WHERE is applied first
- the FIRSTOBS= and OBS= options are applied to the resulting observations.

The following step includes a WHERE statement and an OBS= option.

```
proc print data=sashelp.cars  
          (obs=10);  
  where origin='Asia';  
  var Make Model MSRP Origin;  
run;
```

# Check the Output

## Partial SAS Log

```
31 proc print data=sashelp.cars(obs=10);  
32   where origin='Asia';  
32   var Make Model MSRP Origin;  
34 run;
```

NOTE: There were 10 observations read from the data set SASHELP.CARS.  
WHERE origin='Asia';

## PROC PRINT Output

| Obs | Make  | Model                                  | MSRP     | Origin |
|-----|-------|--|----------|--------|
| 1   | Acura | MDX                                    | \$36,945 | Asia   |
| 2   | Acura | RSX Type S 2dr                         | \$23,820 | Asia   |
| 3   | Acura | TSX 4dr                                | \$26,990 | Asia   |
| 4   | Acura | TL 4dr                                 | \$33,195 | Asia   |
| 5   | Acura | 3.5 RL 4dr                             | \$43,755 | Asia   |
| 6   | Acura | 3.5 RL w/Navigation 4dr                | \$46,100 | Asia   |
| 7   | Acura | NSX coupe 2dr manual S                 | \$89,765 | Asia   |
| 150 | Honda | Civic Hybrid 4dr manual (gas/electric) | \$20,140 | Asia   |
| 151 | Honda | Insight 2dr (gas/electric)             | \$19,110 | Asia   |
| 152 | Honda | Pilot LX                               | \$27,560 | Asia   |

The WHERE statement is applied first, and then 10 observations are processed.



The subsetting variable does not need to be included in the report.

# SUM Statement

The *SUM statement* calculates and displays report totals for the requested *numeric* variables.

```
proc print data=cert.insure;
  var Name Policy;
  where pctinsured=50;
  sum BalanceDue Total;
run;
```

SUM variable(s);

PROC PRINT Output

| Obs | Name         | Policy | BalanceDue | Total  |
|-----|--------------|--------|------------|--------|
| 4   | Johnson, R   | 39022  | 61.04      | 122.07 |
| 5   | LaMance, K   | 63265  | 43.68      | 87.35  |
| 10  | Eberhardt, S | 81589  | 173.17     | 346.33 |
| 14  | Quigley, M   | 97048  | 99.01      | 198.01 |
| 15  | Cameron, L   | 42351  | 111.41     | 222.82 |
|     |              |        | 488.31     | 976.58 |

# Producing Basic Reports

Sorting and Grouping Data

# Business Scenario

Display observations from cert.admit in ascending order by the variable actlevel.

| <b>actlevel</b> | <b>age</b> | <b>height</b> | <b>weight</b> |
|-----------------|------------|---------------|---------------|
| xxxx            | 99         | 99            | 999           |
| xxxx            | 99         | 99            | 999           |
| xxxx            | 99         | 99            | 999           |



# Creating a Sorted Report

## Step 1

Use the SORT procedure to create a new data set, **work.admit**. Order the observations by the value of **actlevel**.

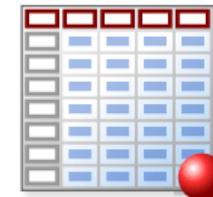
**Cert.admit**



PROC SORT



**work.admit**

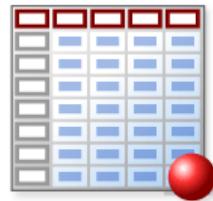


# Creating a Sorted Report

## Step 2

Use the PRINT procedure to display the sorted data set, **work.admit**.

**work.admit**



PROC PRINT



# Step 1: SORT Procedure

The *SORT procedure* rearranges the observations in the input data set based on the values of the variable or variables listed in the BY statement.

```
proc sort data=cert.admit  
          out=work.admit;  
    by actlevel;  
run;
```

```
PROC SORT DATA=input-SAS-data-set  
          <OUT=output-SAS-data-set>;  
    BY <DESCENDING> variables;  
    RUN;
```

The BY statement in a PROC SORT step specifies the sort variables, and if you indicate it, the sort order.

# SORT Procedure

The SORT procedure

- replaces the original data set or creates a new one
- can sort on multiple variables
- sorts in ascending (default) or descending order
- does not generate printed output.



The input data set is overwritten unless the  
OUT= option is used to specify an output data set.

# Viewing the Log

The SORT procedure does not produce a report. Check the log for errors or warnings.

## Partial SAS Log

```
34 proc sort data=cert.admit  
35         out=work.admit;  
36     by actlevel;  
37 run;
```

NOTE: There were 21 observations read from the data set CERT.ADMIT.

NOTE: The data set WORK.ADMIT has 21 observations and 9 variables.

## Step 2: Viewing the Output

```
proc print data=work.admit (firstobs=7 obs=15) ;  
    var actlevel age height weight;  
run;
```

PROC PRINT Output

| Obs | ActLevel | Age | Height | Weight |
|-----|----------|-----|--------|--------|
| 7   | HIGH     | 41  | 67     | 141    |
| 8   | LOW      | 31  | 61     | 123    |
| 9   | LOW      | 51  | 71     | 158    |
| 10  | LOW      | 34  | 73     | 154    |
| 11  | LOW      | 49  | 64     | 172    |
| 12  | LOW      | 28  | 62     | 118    |
| 13  | LOW      | 60  | 71     | 191    |
| 14  | LOW      | 22  | 63     | 139    |
| 15  | MOD      | 43  | 63     | 137    |

# Short Answer Poll

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

a. 

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

b. 

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

c. 

```
proc sort data=work.EmpsAU;
        by First;
run;
```

# Short Answer Poll – Correct Answer

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

a. 

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

b. 

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

C. 

```
proc sort data=work.EmpsAU;
    by First;
run;
```

# Business Scenario

Produce a report that lists therapy patients grouped by **ActLevel**, in descending **Age** order within ActLevel.

ActLevel=HIGH

| Obs | ActLevel | Age | Height | Weight |
|-----|----------|-----|--------|--------|
| 1   | HIGH     | 44  | 66     | 140    |
| 2   | HIGH     | 41  | 67     | 141    |
| 3   | HIGH     | 40  | 69     | 163    |
| 4   | HIGH     | 34  | 66     | 152    |
| 5   | HIGH     | 29  | 76     | 193    |
| 6   | HIGH     | 27  | 72     | 168    |
| 7   | HIGH     | 25  | 75     | 188    |

ActLevel=LOW

| Obs | ActLevel | Age | Height | Weight |
|-----|----------|-----|--------|--------|
| 8   | LOW      | 60  | 71     | 191    |
| 9   | LOW      | 51  | 71     | 158    |
| 10  | LOW      | 49  | 64     | 172    |

# Creating a Grouped Report

## *Step 1*

Use the SORT procedure to group data in a data set.  
This scenario requires two variables to be sorted:

- ActLevel
- descending Age within ActLevel

## *Step 2*

Use a BY statement in PROC PRINT to display the sorted observations grouped by ActLevel.

# Step 1: Sort the Data

Sort the data set to group the observations.

```
proc sort data=cert.admit  
          out=work.admit;  
  by ActLevel descending Age;  
run;
```

BY <DESCENDING> variables;

# Specifying Sort Order

The *DESCENDING* option reverses the sort order for the variable that immediately follows it. The observations are sorted from the largest value to the smallest value.

Examples:

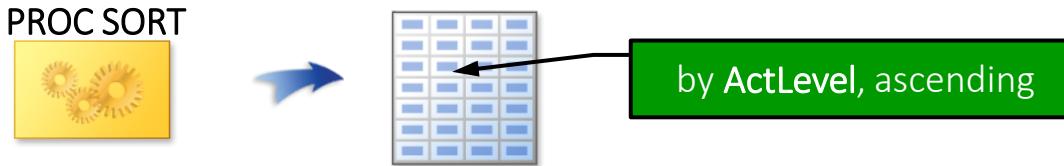
```
by descending Last  
First;
```

```
by Last descending  
First;
```

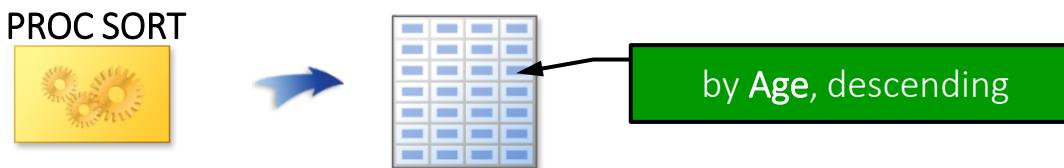
```
by descending Last descending First;
```

# Specifying Multiple BY Variables

- 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 as the first BY variable by the values of the second BY variable.



- This sorting continues for every specified BY variable.

## Step 2: Specify Report Groupings

The BY statement in a PROC PRINT step specifies the variable or variables to use to form *BY groups*.

```
proc print data=work.admit (obs=10);
    var actlevel age height weight;
    by actlevel;
run;
```

BY <DESCENDING> *variables*;

- The variables in the BY statement are called *BY variables*.
- The observations in the data set **must** be in order according to the order of the BY variable (or variables).

# Viewing the Output

## PROC PRINT Output

| ActLevel=HIGH |          |     |        |        |
|---------------|----------|-----|--------|--------|
| Obs           | ActLevel | Age | Height | Weight |
| 1             | HIGH     | 44  | 66     | 140    |
| 2             | HIGH     | 41  | 67     | 141    |
| 3             | HIGH     | 40  | 69     | 163    |
| 4             | HIGH     | 34  | 66     | 152    |
| 5             | HIGH     | 29  | 76     | 193    |
| 6             | HIGH     | 27  | 72     | 168    |
| 7             | HIGH     | 25  | 75     | 188    |

| ActLevel=LOW |          |     |        |        |
|--------------|----------|-----|--------|--------|
| Obs          | ActLevel | Age | Height | Weight |
| 8            | LOW      | 60  | 71     | 191    |
| 9            | LOW      | 51  | 71     | 158    |
| 10           | LOW      | 49  | 64     | 172    |

# Short Answer Poll

Why does this program fail?

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

proc print data=work.sorted;
   by Gender;
run;
```

# Short Answer Poll – Correct Answer

Why does this program fail?

The input data set was not sorted by Gender.

```
188 proc sort data=orion.sales  
189         out=work.sorted;  
190     by Country Gender;  
191 run;
```

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

```
192  
193 proc print data=work.sorted;  
194     by Gender;  
195 run;
```

```
ERROR: Data set WORK.SORTED is not sorted in ascending sequence. The current  
      BY group has Gender = M and the next BY group has Gender = F.
```

```
NOTE: The SAS System stopped processing this step because of errors.  
NOTE: There were 64 observations read from the data set WORK.SORTED.
```

# Business Scenario

Modify the previous report to display selected variables, the Fee subtotal for each ActLevel, and the Fee grand total.

| ----- ActLevel=High ----- |         |        |       |
|---------------------------|---------|--------|-------|
| Age                       | Height  | Weight | Fee   |
| XXXX                      | XXXXXXX | X      | 99999 |
| XXXX                      | XXXXXXX | X      | 99999 |
| ----- ActLevel -----      |         |        |       |
| ----- ActLevel=Low -----  |         |        |       |
| Age                       | Height  | Weight | Fee   |
| XXXXXX                    | XXXXXXX | X      | 99999 |
| XXXXXX                    | XXXXXXX | X      | 99999 |
| ----- ActLevel -----      |         |        |       |

subtotals

grand total

The diagram illustrates the flow of data aggregation. Arrows point from the 'subtotals' box to the two '99999' entries under the 'ActLevel=High' section, and another arrow points from the 'grand total' box to the final '9999999' entry at the bottom right of the report.

# Generating Subtotals

Use a BY statement and a SUM statement in a PROC PRINT step.

```
proc sort data=cert.admit
           out=work.admit;
   by actlevel descending age;
run;

proc print data=work.admit noobs;
   var actlevel age height weight;
   by actlevel;
   sum Fee;
run;
```

# Viewing the Output

ActLevel=HIGH

| ActLevel | Age | Height | Weight | Fee    |
|----------|-----|--------|--------|--------|
| HIGH     | 44  | 66     | 140    | 149.75 |
| HIGH     | 41  | 67     | 141    | 149.75 |
| HIGH     | 40  | 69     | 163    | 124.80 |
| HIGH     | 34  | 66     | 152    | 124.80 |
| HIGH     | 29  | 76     | 193    | 124.80 |
| HIGH     | 27  | 72     | 168    | 85.20  |
| HIGH     | 25  | 75     | 188    | 85.20  |
| ActLevel |     |        |        | 844.30 |

subtotal for HIGH

ActLevel=LOW

| ActLevel | Age | Height | Weight | Fee    |
|----------|-----|--------|--------|--------|
| LOW      | 60  | 71     | 191    | 149.75 |
| LOW      | 51  | 71     | 158    | 124.80 |
| LOW      | 49  | 64     | 172    | 124.80 |
| LOW      | 34  | 73     | 154    | 124.80 |
| LOW      | 31  | 61     | 123    | 149.75 |
| LOW      | 28  | 62     | 118    | 85.20  |
| LOW      | 22  | 63     | 139    | 85.20  |
| ActLevel |     |        |        | 844.30 |

subtotal for LOW

ActLevel=MOD

| ActLevel | Age | Height | Weight | Fee     |
|----------|-----|--------|--------|---------|
| MOD      | 54  | 71     | 183    | 149.75  |
| MOD      | 47  | 72     | 173    | 124.80  |
| MOD      | 43  | 63     | 137    | 149.75  |
| MOD      | 43  | 65     | 123    | 124.80  |
| MOD      | 35  | 70     | 173    | 149.75  |
| MOD      | 32  | 67     | 151    | 149.75  |
| MOD      | 30  | 69     | 147    | 149.75  |
| ActLevel |     |        |        | 998.35  |
|          |     |        |        | 2686.95 |

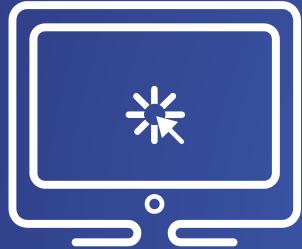
grand total

# Generating Subtotals

Add a PAGEBY statement to print each BY group on a separate page.

```
proc sort data=cert.admit
           out=work.admit;
   by actlevel descending age;
run;

proc print data=work.admit noobs;
   var actlevel age height weight;
   by actlevel;
   sum Fee;
   pageby actlevel;
run;
```



# Using PAGEBY

This demonstration illustrates the use  
of PAGEBY to print BY groups on  
separate pages.

# Setup for the Poll

Modify the previous report to display only employees earning less than 25,500. Which WHERE statement (or statements) results in the most efficient processing?

```
proc sort data=orion.sales
           out=work.sales;
  /* where Salary<25500; */
  by Country descending Salary;
run;
proc print data=work.sales noobs;
  by Country;
  sum Salary;
  /*      where Salary<25500; */
  var First_Name Last_Name Gender Salary;
run;
```

# Multiple Choice Poll

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

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

## 4.07 Multiple Choice Poll – Correct Answer

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

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

Subsetting in PROC SORT is more efficient. It selects and sorts only the required observations.



Be sure to use the OUT= option when you subset in PROC SORT or you will overwrite your original data set with the subset.

# Business Scenario

Enhance the payroll report by adding titles, footnotes, and descriptive column headings.

| Obs | Employee_ID | Last_Name  | Salary |
|-----|-------------|------------|--------|
| 1   | 9999        | xxxxxxxxxx | 99999  |
| 2   | 9999        | xxxxxxxxxx | 99999  |
| 3   | 9999        | xxxxxxxxxx | 99999  |

Orion Star Sales Staff  
Salary Report

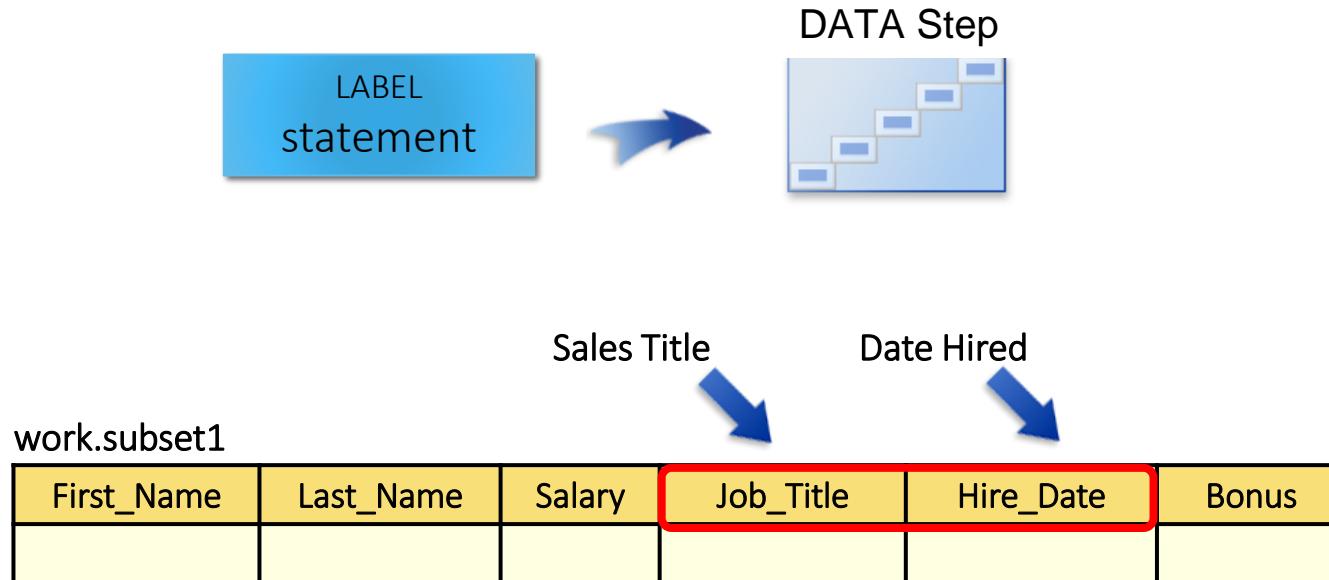
| Obs | Employee ID | Last Name  | Annual Salary |
|-----|-------------|------------|---------------|
| 1   | 9999        | xxxxxxxxxx | 99999         |
| 2   | 9999        | xxxxxxxxxx | 99999         |
| 3   | 9999        | xxxxxxxxxx | 99999         |

Confidential



# LABEL Statement on a DATA Step

The LABEL statement assigns permanent descriptive labels to variables.



# LABEL Statement

The LABEL statement assigns descriptive labels to variables.

- A label can be up to 256 characters and include any characters, including blanks.
- Labels are used automatically by most procedures.
- The PRINT procedure uses labels only when the LABEL or SPLIT= option is specified.

# Defining Permanent Labels

Use a LABEL statement in a DATA step to permanently assign labels to variables. The labels are stored in the descriptor portion of the data set.

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

LABEL variable='label'  
<variable='label' ...>;

# Viewing the Output

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

Partial PROC CONTENTS Output

| Alphabetic List of Variables and Attributes |            |      |     |             |
|---|------------|------|-----|-------------|
| #   | Variable   | Type | Len | Label       |
| 6   | Bonus      | Num  | 8   |             |
| 1   | First_Name | Char | 12  |             |
| 5   | Hire_Date  | Num  | 8   | Date Hired  |
| 4   | Job_Title  | Char | 25  | Sales Title |
| 2   | Last_Name  | Char | 18  |             |
| 3   | Salary     | Num  | 8   |             |

# Viewing the Output: Displaying Labels

To use labels in the PRINT procedure, use the LABEL option in the PROC PRINT statement.

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

Partial PROC PRINT Output

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

# LABEL Statement and PROC Print Option

Use a LABEL statement and the LABEL option to display temporary descriptive column headings instead of variable names.

```
title1 'Orion Star Sales Staff';
title2 'Salary Report';
footnote1 'Confidential';

proc print data=orion.sales label;
  var Employee_ID Last_Name Salary;
  label Employee_ID='Sales ID'
        Last_Name='Last Name'
        Salary='Annual Salary';
run;

title;
footnote;
```

LABEL variable-1='label'  
...  
variable-n='label ';

# Viewing the Output

| Orion Star Sales Staff<br>Salary Report |          |             |               |
|---|----------|-------------|---------------|
| Obs                                     | Sales ID | Last Name   | Annual Salary |
| 1                                       | 120102   | Zhou        | 108255        |
| 2                                       | 120103   | Dawes       | 87975         |
| 3                                       | 120121   | Elvish      | 26600         |
| ...                                     |          |             |               |
| 164                                     | 121144   | Capachietti | 83505         |
| 165                                     | 121145   | Lansberry   | 84260         |

Confidential

# SPLIT= Option

The SPLIT= option in PROC PRINT specifies a split character to control line breaks in column headings.

```
proc print data=orion.sales split='*' ;
  var Employee_ID Last_Name Salary;
  label Employee_ID='Sales ID'
    Last_Name='Last*Name'
    Salary='Annual*Salary';
run;
```

SPLIT='split-character'

The SPLIT= option can be used instead of the LABEL option in a PROC PRINT step.

# Viewing the Output

Partial PROC PRINT Output

| Orion Star Sales Staff<br>Salary Report |          |             |               |
|---|----------|-------------|---------------|
| Obs                                     | Sales ID | Last Name   | Annual Salary |
| 1                                       | 120102   | Zhou        | 108255        |
| 2                                       | 120103   | Dawes       | 87975         |
| 3                                       | 120121   | Elvish      | 26600         |
| ...                                     |          |             |               |
| 164                                     | 121144   | Capachietti | 83505         |
| 165                                     | 121145   | Lansberry   | 84260         |

Confidential

# Enhancing Variables with Labels and Formats

- A label changes the way a column name appears when displayed or printed.
- A format changes the way a value appears when displayed or printed.
- Neither change the actual underlying data.

# Creating Custom Formats

Creating and Using Custom Formats – Chapter 12

# Formatting Data Values

| Name    | Gender | Age | Height | Weight | Birthdate |
|---------|--------|-----|--------|--------|-----------|
| Alfred  | M      | 14  | 69     | 112.5  | 16370     |
| Alice   | F      | 13  | 56.5   | 84     | 16756     |
| Barbara | F      | 13  | 65.3   | 98     | 16451     |
| Carol   | F      | 14  | 62.8   | 102.5  | 16256     |



| Name    | Gender | Age | Height | Weight | Birthdate |
|---------|--------|-----|--------|--------|-----------|
| Alfred  | M      | 14  | 69     | 113    | 26OCT2004 |
| Alice   | F      | 13  | 57     | 84     | 16NOV2005 |
| Barbara | F      | 13  | 65     | 98     | 15JAN2005 |
| Carol   | F      | 14  | 63     | 103    | 04JUL2004 |

```
proc print data=pg2.class_birthdate noobs;
  format Height Weight 3.0 Birthdate date9.;
run;
```

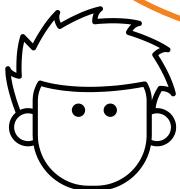
# Formatting Data Values

| Name   | Gender | Height |
|--------|--------|--------|
| James  | M      | 57.3   |
| Jane   | F      | 59.8   |
| John   | M      | 59     |
| Louise | F      | 56.3   |
| Robert | M      | 64.8   |



| Name   | Gender | Height        |
|--------|--------|---------------|
| James  | Male   | Below Average |
| Jane   | Female | Average       |
| John   | Male   | Average       |
| Louise | Female | Below Average |
| Robert | Male   | Above Average |

SAS doesn't always have a predefined format that meets your needs.



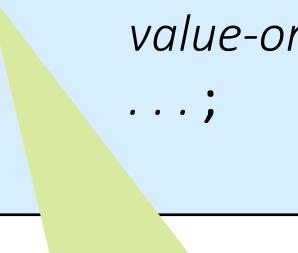
```
format Gender ? Height ?;
```

# FORMAT Procedure

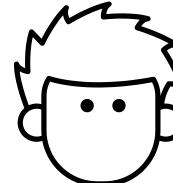
```
PROC FORMAT;
```

```
  VALUE format-name value-or-range-1 = 'formatted-value'  
        value-or-range-2 = 'formatted-value'  
        ...;  
RUN;
```

- The name can be up to 32 characters in length.
- Character formats must begin with a \$ followed by a letter or underscore.
- Numeric formats must begin with a letter or underscore.
- The name cannot end in a number or match an existing SAS format.



You can use the  
FORMAT procedure  
to create your own  
format.



# FORMAT Procedure

```
PROC FORMAT;
```

```
  VALUE format-name value-or-range-1 = 'formatted-value'  
        value-or-range-2 = 'formatted-value'  
        ... ;
```

```
RUN;
```

individual value or  
range of values that  
you want to format

format that you want to  
apply to the individual  
value or range of values

# Creating and Using Custom Formats

create  
format

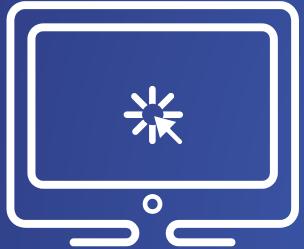
```
proc format;  
  value $genfmt 'F'='Female'  
                 'M'='Male';  
run;
```

no period in format name

apply  
format

```
proc print data=pg2.class_birthdate;  
  format Gender $genfmt.;  
run;
```

period in format name



# Creating and Using Custom Formats

The demonstration illustrates using the FORMAT procedure to create custom numeric and character formats based on single values and a range of values.

# Defining a Continuous Range

Put < before the ending value in a range to exclude the value.

```
value hrange 50-<58 = 'Below Average'  
                     58-60  = 'Average'  
                     60<-70 = 'Above Average';
```

Put < after the starting value in a range to exclude the value.

# Using Keywords

lowest possible value

```
value hrange low-<58 = 'Below Average'  
      58-60   = 'Average'  
      60<-high = 'Above Average';
```

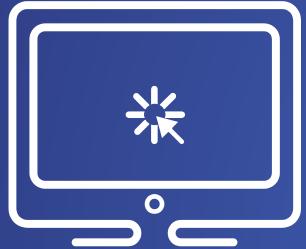
highest possible value

```
value $genfmt 'F' = 'Female'  
      'M'   = 'Male'  
      other = 'Miscoded';
```

all values that do not match any other value

# STAT604 SAS Lesson 11

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.



# Using Custom Formats

The demonstration illustrates permanent data sets that have a custom format applied.

# Using the NOFMTERR System Option

By default, the FMTERR system option is in effect. If you use a format that SAS cannot load, SAS issues an error message and stops processing the step.

To prevent the default action, change the system option FMTERR to NOFMTERR.

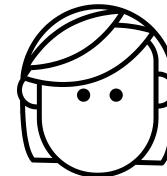
```
OPTIONS FMTERR | NOFMTERR;
```

# Location of Custom Formats

```
proc format;  
  value $sttype  
    'TS'='Tropical Storm'  
    'SS'='Subtropical Storm'  
    'ET'='Extratropical Storm'  
    'DS'='Disturbance'  
    'NR'='Not Reported';  
  
run;
```

NOTE: Format \$STTYPE has been output.

NOTE: Format \$STTYPE is already on the library WORK.FORMATS.  
NOTE: Format \$STTYPE has been output.



By default, custom formats are stored in the temporary **Work** library in a catalog named **formats**.

# Creating Permanent Custom Formats

```
proc format library=pg2.myfmts;
```

```
proc format library=pg2;
```

```
proc format lib=pg2;
```

Use the LIBRARY= option to save formats in a permanent location.

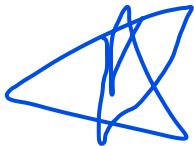


# Using the FMTSEARCH= System Option

To use permanent formats or to search multiple catalogs, use the FMTSEARCH= system option to identify the catalog(s) to be searched for the format(s).

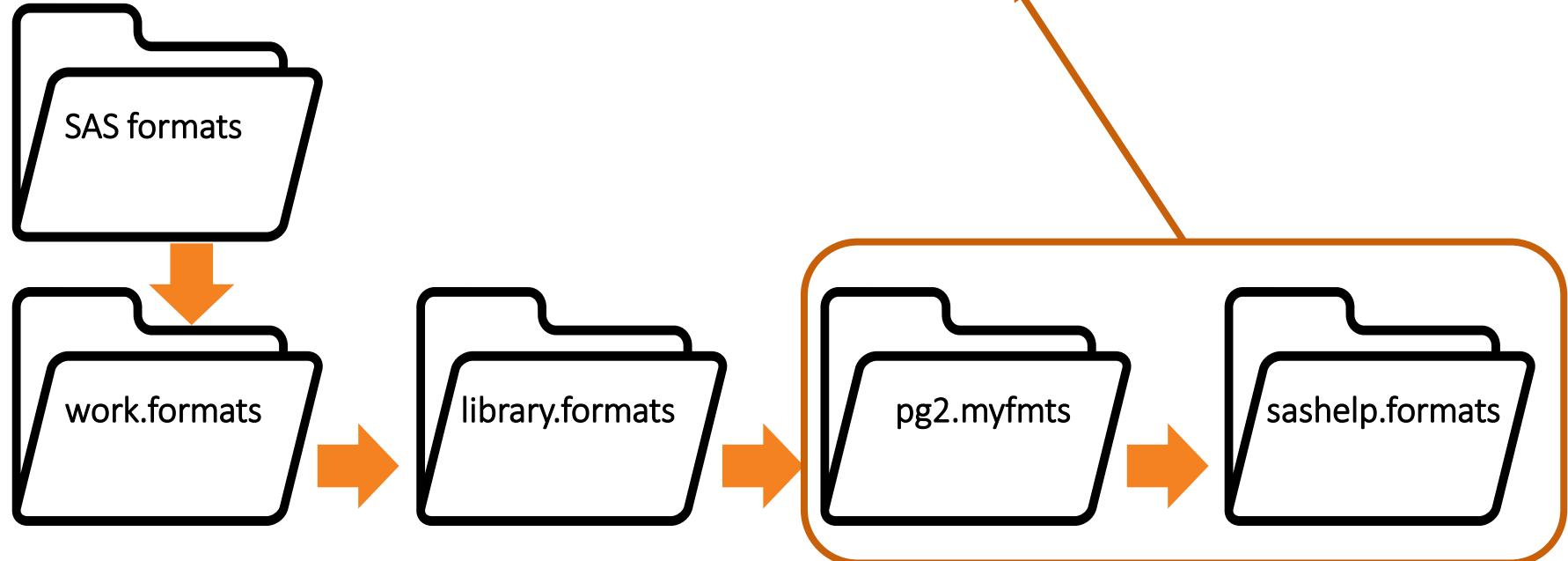
General form of the FMTSEARCH= system option:

```
OPTIONS FMTSEARCH = (item-1 item-2...item-n);
```



# Searching for Custom Formats

```
options fmtsearch=(pg2.myfmts sashelp);
```





# Using Permanent Custom Formats

The demonstration illustrates the storage and use of permanent formats.

# Documenting Formats

You can use the FMTLIB option in the PROC FORMAT statement to document the format.

General form of the FMTLIB option:

```
PROC FORMAT LIBRARY = libref.catalog
                  FMTLIB;
      <SELECT format-name format-name...;>
      <EXCLUDE format-name format-name...;>
RUN;
```

# Documenting Formats

```
proc format library=mylib fmtlib;  
    select $col;  
run;
```

| FORMAT NAME: \$COL LENGTH: 15 NUMBER OF VALUES: 4<br>MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 15 FUZZ: 0 |     |                                       |  |
|--|-----|---------------------------------------|--|
| START  | END | LABEL (VER. V7 V8 01NOV2020:15:59:52) |  |
| B  | B   | Sky Blue                              |  |
| G  | G   | Rain Cloud Gray                       |  |
| W  | W   | Moon White                            |  |
| Y  | Y   | Sunburst Yellow                       |  |



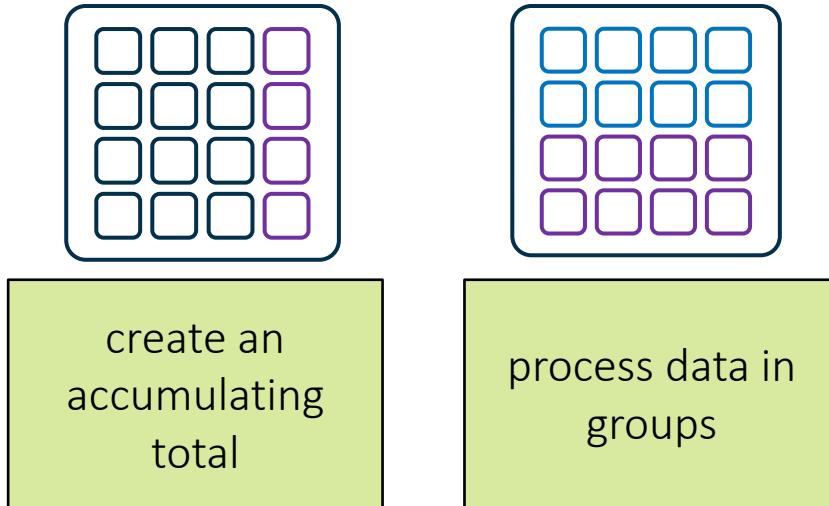
# Using Permanent Custom Formats

The demonstration illustrates the documentation of formats.

# Summarizing Data

Creating an Accumulating Column – Prep Guide Page 148

# Lesson Overview



Understanding DATA step processing is critical as you perform more complex data manipulations.



# Creating an Accumulating Column

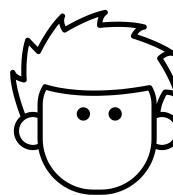
houston\_rain

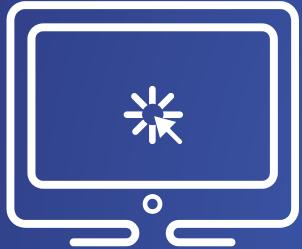
| Date      | DailyRain | YTDRain |
|-----------|-----------|---------|
| 01JAN2017 | 0.01      | 0.01    |
| 02JAN2017 | 1.29      | 1.3     |
| 03JAN2017 | 0         | 1.3     |
| 04JAN2017 | 0         | 1.3     |
| 05JAN2017 | 0         | 1.3     |
| 06JAN2017 | 0.27      | 1.57    |
| 07JAN2017 | 0         | 1.57    |

Create a new column that stores an accumulating total.

```
data houston_rain;
  set pg2.weather_houston;
  keep Date DailyRain YTDRain;
  YTDRain=YTDRain+DailyRain;
run;
```

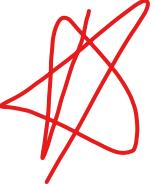
Will this code produce the desired results?  
How will SAS process this assignment statement?





# Creating an Accumulating Column

This demonstration illustrates using the DATA step debugger in PC SAS to observe how the default behavior of the PDV affects summary expressions.



# Retaining Values in the PDV

| Variable  | Value                 |
|-----------|-----------------------|
| Station   | USW00012960           |
| Name      | HOUSTON INTERCONTI... |
| Date      | 02JAN2017             |
| Month     | 1                     |
| DailyRain | 1.29                  |
| AvgWind   | 6.26                  |
| TempMin   | 52                    |
| TempMax   | 74                    |
| YTDRain   | 1.3                   |
| _ERROR_   | 0                     |
| _N_       | 2                     |

To successfully create an accumulating column:

- 1) Set the initial value to 0.
- 2) Retain the value each time that the PDV reinitializes.

# ~~A~~ Retaining Values in the PDV

**RETAIN** column <initial-value>;

```
data houston2017;  
  set pg2.weather_houston;  
  retain YTDRain 0;  
  YTDRain=YTDRain+DailyRain;  
run;
```

- 1) retains the value each time that the PDV reinitializes
- 2) assigns an initial value

| Date      | DailyRain | YTDRain |
|-----------|-----------|---------|
| 01JUN2017 | 0.01      | 0.01    |
| 02JUN2017 | 0.22      | 0.23    |
| 03JUN2017 | 0.79      | 1.02    |
| 04JUN2017 | 0.97      | 1.99    |
| 05JUN2017 | 0.2       | 2.19    |
| 06JUN2017 | 0.02      | 2.21    |
| 07JUN2017 | 0         | 2.21    |

# Activity

Return to the SAS program used in the demo and perform the following tasks:

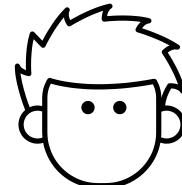
1. Modify the program to retain **YTD\_Fees** and set the initial value to 0.
2. Run the program and examine the results. Run a similar program using the **amounts** data set?
3. Change the assignment statement to use the SUM function instead of the plus symbol. Run the program again. Why are the results different?

# Creating an Accumulating Column

```
data zurich2017;  
  set pg2.weather_zurich;  
  retain TotalRain 0;  
  TotalRain=sum(TotalRain,Rain_mm);  
run;
```

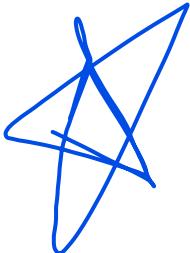
| Date      | Rain_mm | TotalRain |
|-----------|---------|-----------|
| 02JAN2017 | .       | 0         |
| 03JAN2017 | 4.8     | 4.8       |
| 04JAN2017 | 39.9    | 44.7      |
| 05JAN2017 | 1.8     | 46.5      |
| 06JAN2017 | 4.3     | 50.8      |
| 07JAN2017 | 0       | 50.8      |

The SUM  
function ignores  
missing values!



# Using the Sum Statement

*Shortcut to what we just saw.*



*column+expression;*

```
data zurich2017;  
  set pg2.weather_zurich;  
  TotalRain+Rain_mm;  
run;
```

- creates **TotalRain** and sets the initial value to 0
- retains **TotalRain**
- adds **Rain\_mm** to **TotalRain** for each row
- ignores missing values

# Question

What sum statement would you add to this program to create the column named DayNum?

```
data zurich2017;  
  set pg2.weather_zurich;  
  YTDRain_mm+Rain_mm;  
  ???  
run;
```

| PRECIP_MM | YTDPrecip_mm | DayNum |
|-----------|--------------|--------|
| .         | 0            | 1      |
| 4.8       | 4.8          | 2      |
| 39.9      | 44.7         | 3      |
| 1.8       | 46.5         | 4      |
| 4.3       | 50.8         | 5      |
| 0         | 50.8         | 6      |

# Correct Answer

What sum statement would you add to this program to create the column named DayNum?

```
data zurich2017;  
  set pg2.weather_zurich;  
  YTDRain mm+Rain_mm;  
  DayNum+1;  
run;
```

| (1) PRECIP<br>_MM | (1) YTDPrecip<br>_mm | (1) DayNum |
|-------------------|----------------------|------------|
| .                 | 0                    | 1          |
| 4.8               | 4.8                  | 2          |
| 39.9              | 44.7                 | 3          |
| 1.8               | 46.5                 | 4          |
| 4.3               | 50.8                 | 5          |
| 0                 | 50.8                 | 6          |

# Activity

Return to the SAS program used in the demo and replace the previously used accumulating code with a single statement.

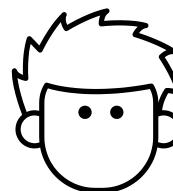
# Summarizing Data

Processing Data in Groups – Prep Guide Chapter 8

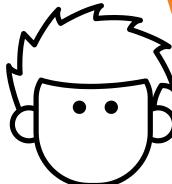
# Processing Data in Groups

| ⚠️ Basin | ⚠️ Name  | ⌚ MaxWindMPH | 📅 StartDate |
|----------|----------|--------------|-------------|
| NA       | NATE     | 90           | 04OCT2017   |
| NA       | OPHELIA  | 115          | 09OCT2017   |
| NA       | PHILIPPE | 60           | 28OCT2017   |
| NA       | RINA     | 60           | 06NOV2017   |
| NI       | MAARUTHA | 45           | 15APR2017   |
| NI       | MORA     | 70           | 28MAY2017   |
| NI       | OCKHI    | 100          | 29NOV2017   |
| SI       | ALFRED   | 50           | 16FEB2017   |
| SI       | BLANCHE  | 65           | 02MAR2017   |
| SI       | CALEB    | 50           | 23MAR2017   |
| SI       | ERNIE    | 140          | 05APR2017   |
| SI       | FRANCES  | 75           | 21APR2017   |
| SI       | GREG     | 40           | 29APR2017   |
| SI       | CEMPAKA  | 40           | 22NOV2017   |
| SI       | DAHLIA   | 60           | 24NOV2017   |
| SI       | HILDA    | 60           | 24DEC2017   |
| SP       | DEBBIE   | 120          | 23MAR2017   |
| SP       | BART     | 45           | 19FEB2017   |
| SP       | COOK     | 100          | 06APR2017   |
| SP       | DONNA    | 125          | 01MAY2017   |
| SP       | ELLA     | 70           | 07MAY2017   |

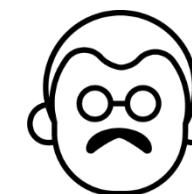
If your data is sorted into groups, the DATA step can identify when each group begins and ends.



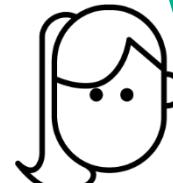
# Processing Data in Groups



What is the maximum wind measurement for each storm?



Which storm names are used more than once within a season?



When did the first storm occur in each basin?

# Processing Data in Groups

```
PROC SORT DATA=input-table
    <OUT=sorted-output-table>;
    BY <DESCENDING> col-name(s);
RUN;
```

sorts the table  
into groups

```
DATA output-table;
    SET sorted-output-table;
    BY <DESCENDING> col-name(s);
RUN;
```

processes the data  
in the sorted table  
by groups

# Processing Data in Groups

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  
run;
```

First.bycol

Last.bycol

The BY statement creates First./Last. variables in the PDV that can be used to identify when each BY group begins and ends.

## PDV

| <i>...other columns...</i> | Basin | D | First.Basin | D | Last.Basin |
|----------------------------|-------|---|-------------|---|------------|
|                            |       |   |             |   |            |

# Processing Data in Groups

PDV

| <i>...other columns...</i> | Basin | First.Basin | Last.Basin |
|----------------------------|-------|-------------|------------|
|                            | NA    | D<br>1      | D<br>0     |

first row where  
**Basin** is NA

PDV

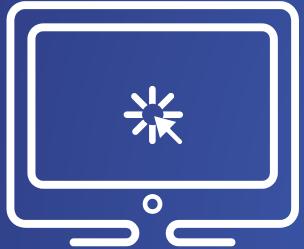
| <i>...other columns...</i> | Basin | First.Basin | Last.Basin |
|----------------------------|-------|-------------|------------|
|                            | NA    | D<br>0      | D<br>0     |

subsequent  
rows where  
**Basin** is NA

PDV

| <i>...other columns...</i> | Basin | First.Basin | Last.Basin |
|----------------------------|-------|-------------|------------|
|                            | NA    | D<br>0      | D<br>1     |

last row where  
**Basin** is NA



# Identifying the First and Last Row in Each Group

This demonstration illustrates using the DATA step debugger in PC SAS to observe how **First./Last.** variables are assigned values in the PDV during execution.

# Short Answer Poll

What are the values for **First.Dept** and **Last.Dept** when the DATA step is processing the observation indicated by the arrow?

| Dept   | Salary |
|--------|--------|
| ADMIN  | 20000  |
| ADMIN  | 100000 |
| ADMIN  | 50000  |
| ENGINR | 25000  |
| FINANC | 10000  |
| FINANC | 12000  |



| First.Dept |
|------------|
| ?          |

| Last.Dept |
|-----------|
| ?         |

# Short Answer Poll – Correct Answer

What are the values for **First.Dept** and **Last.Dept** when the DATA step is processing the observation indicated by the arrow?

| Dept   | Salary |
|--------|--------|
| ADMIN  | 20000  |
| ADMIN  | 100000 |
| ADMIN  | 50000  |
| ENGINR | 25000  |
| FINANC | 10000  |
| FINANC | 12000  |

| First.Dept |
|------------|
| 1          |



| Last.Dept |
|-----------|
| 1         |

**First.Dept** and **Last.Dept** are both 1. This happens when a group consists of a single observation.

# Quiz

Suppose we only one to keep the last row from each group. Can the where statement be successfully added to this data step?

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  where last.Basin=1;  
run;
```

# Quiz – Correct Answer

Can the where statement be successfully added to this data step?

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  where last.Basin=1;  
run;
```

No, an error is generated in the log.

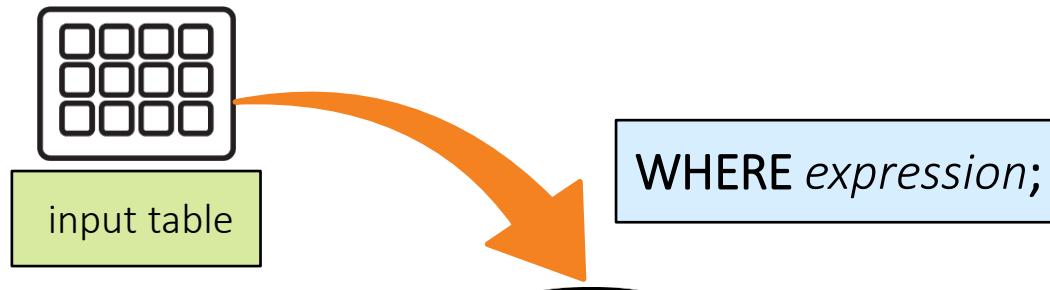
```
32      where last.Basin=1;
```

```
_____  
180
```

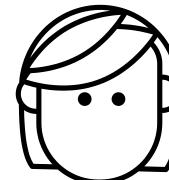
ERROR: Syntax error while parsing WHERE clause.

ERROR 180-322: Statement is not valid or it is used out of proper order.

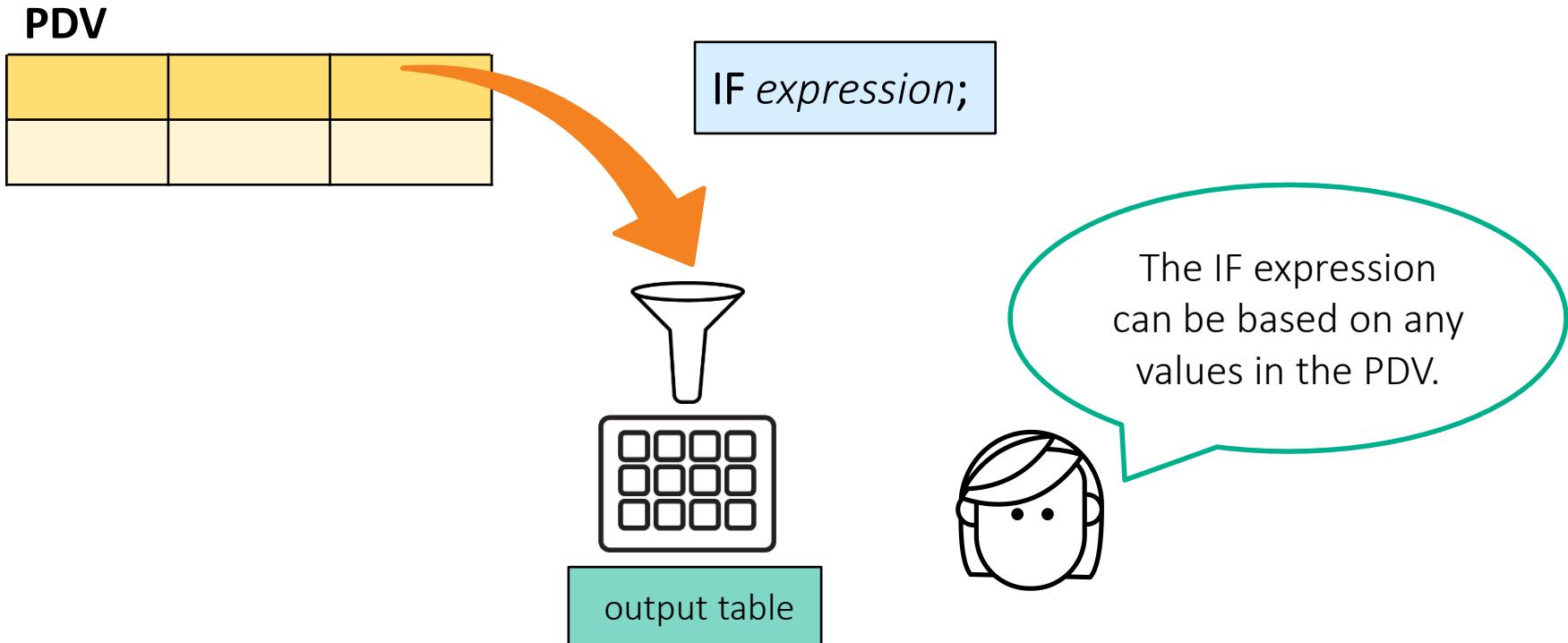
# Subsetting Rows in Execution (Review)



The WHERE expression must be based on columns in the input table.



# Subsetting Rows in Execution (Review)



# Subsetting Rows in Execution

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  if last.Basin=1;  
    StormLength=EndDate-StartDate;  
    MaxWindKM=MaxWindMPH*1.60934;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

When the IF condition is true,  
SAS continues processing  
statements for that row.

true



# Subsetting in Execution

Implicit RETURN;

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  if last.Basin=1;  
    StormLength=EndDate-StartDate;  
    MaxWindKM=MaxWindMPH*1.60934;  
  
run;
```

Implicit OUTPUT;

false

When the IF condition is false,  
SAS stops processing statements  
for that row and returns to the  
top of the DATA step.

# Quiz

What difference does it make if the IF statement were moved to the bottom of the DATA step?

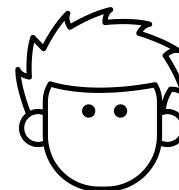
```
data storm2017_max;
  set storm2017_sort;
  by Basin;
  StormLength=EndDate-StartDate;
  MaxWindKM=MaxWindMPH*1.60934;
  if last.Basin=1;
run;
```

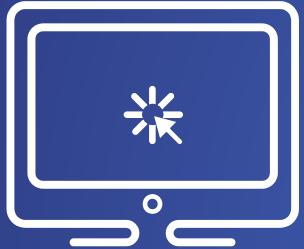
# Quiz – Correct Answer

Both places produce the same output. However, consider the sequence of the statements in the execution phase. Where is the optimal placement of the subsetting IF statement?

```
data storm2017_max;  
  set storm2017_sort;  
  by Basin;  
  if last.Basin=1;  
    StormLength=EndDate-StartDate;  
    MaxWindKM=MaxWindMPH*1.60934;  
run;
```

Use the subsetting IF statement as early as possible so that SAS processes additional statements only for rows that will be written to the output table.





## Keeping only one observation per group

This demonstration illustrates using the **Last.** variable to control output.

# What Must Happen When?

There is a three-step process for using the DATA step to summarize grouped data.

## *Step 1*

Initialize: Set the accumulating variable to zero at the start of each BY group.



## *Step 2*

Accumulate: Increment the accumulating variable with a sum statement (automatically retains).



## *Step 3*

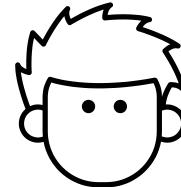
Output: Write only the last observation of each BY group.

# Accumulating Column within Groups

| Date      | Month | DailyRain | MTDRain |
|-----------|-------|-----------|---------|
| 01MAR2017 | 3     | 0         | 0       |
| 02MAR2017 | 3     | 0         | 0       |
| 03MAR2017 | 3     | 0         | 0       |
| 04MAR2017 | 3     | 0.05      | 0.05    |
| 05MAR2017 | 3     | 0.99      | 1.04    |
| 06MAR2017 | 3     | 0.24      | 1.28    |

|           |   |      |      |
|-----------|---|------|------|
| 24MAR2017 | 3 | 1.26 | 3.93 |
| 25MAR2017 | 3 | 0.02 | 3.95 |
| 26MAR2017 | 3 | 0    | 3.95 |
| 27MAR2017 | 3 | 0    | 3.95 |
| 28MAR2017 | 3 | 0    | 3.95 |
| 29MAR2017 | 3 | 1.68 | 5.63 |
| 30MAR2017 | 3 | 0    | 5.63 |
| 31MAR2017 | 3 | 0    | 5.63 |
| 01APR2017 | 4 | 0    | 0    |
| 02APR2017 | 4 | 0.09 | 0.09 |

How can we reset an accumulating column at the beginning of each group?



# Summarizing Data by Groups

## Step 1 Initialize

```
data deptsals (keep=Dept DeptSal) ;  
  set SalSort;  
  by Dept;  
  if First.Dept then DeptSal=0;  
  <additional SAS statements>  
run;
```

-  The condition is considered true when **First.Dept** has a value of 1.

# Summarizing Data by Groups

Step 2

Accumulate

```
data deptsals(keep=Dept DeptSal) ;  
  set SalSort;  
  by Dept;  
  if First.Dept then DeptSal=0;  
  DeptSal+Salary;  
  <additional SAS statements>  
run;
```

# Summarizing Data by Groups

Step 3

Output

| Dept   | Salary | DeptSal |
|--------|--------|---------|
| ADMIN  | 20000  | 20000   |
| ADMIN  | 100000 | 120000  |
| ADMIN  | 50000  | 170000  |
| ENGINR | 25000  | 25000   |
| ENGINR | 20000  | 45000   |
| ENGINR | 23000  | 68000   |
| ENGINR | 27000  | 95000   |
| FINANC | 10000  | 10000   |
| FINANC | 12000  | 22000   |

# Summarizing Data by Groups

Step 3

Output

```
data deptsals(keep=Dept DeptSal) ;
  set SalSort;
  by Dept;
  if First.Dept then DeptSal=0;
  DeptSal+Salary;
  if Last.Dept;
run;
```

# Summarizing Data by Groups

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

PROC PRINT Output

| Dept   | DeptSal |
|--------|---------|
| ADMIN  | 410000  |
| ENGINR | 163000  |
| FINANC | 318000  |
| HUMRES | 181000  |
| SALES  | 373000  |

Partial SAS Log

```
NOTE: There were 39 observations read  
      from the data set WORK.SALSORT.  
NOTE: The data set WORK.DEPTSALS has 5  
      observations and 2 variables.
```



# Creating an Accumulating Column within Groups

This demonstration illustrates using the `First.` variable to reset an accumulating column.

# Identifying and Removing Duplicate Rows

```
PROC SORT DATA=input-table <OUT=output-table>  
    NODUPRECS <DUPOUT=output-table>;  
    BY _ALL_;  
RUN;
```

sorts by all columns to ensure that duplicate rows are adjacent

removes adjacent rows that are entirely duplicated

output table of duplicates

# Identifying and Removing Duplicate Rows

```
proc sort data=pg1.class_test3  
          out=test_clean  
          noduprecs  
          dupout=test_dups;  
    by _all_;  
run;
```

pg1.class\_test3

| Name    | Subject | Test Score |
|---------|---------|------------|
| Judy    | Math    | 97         |
| Judy    | Reading | 91         |
| Barbara | Math    | 96         |
| Barbara | Reading | 86         |
| Barbara | Math    | 96         |
| Louise  | Math    | 92         |

test\_clean

| Name    | Subject | Test Score |
|---------|---------|------------|
| Alice   | Math    | 71         |
| Alice   | Reading | 67         |
| Barbara | Math    | 96         |
| Barbara | Reading | 86         |
| Carol   | Math    | 61         |
| Carol   | Reading | 57         |

test\_dups

| Name    | Subject | Test Score |
|---------|---------|------------|
| Barbara | Math    | 96         |

# Identifying and Removing Duplicate Key Values

```
PROC SORT DATA=input-table <OUT=output-table>
      NODUPKEY <DUPOUT=output-table>;
      BY <DESCENDING> col-name(s);
RUN;
```

keeps only the first occurrence of each unique value of the BY variable

This removes duplicate values of the column listed in the BY statement.



# Identifying and Removing Duplicate Key Values

```
proc sort data=pg1.class_test2  
          out=test_clean  
          dupout=test_dups  
          nodupkey;  
    by Name;  
run;
```

pg1.class\_test2

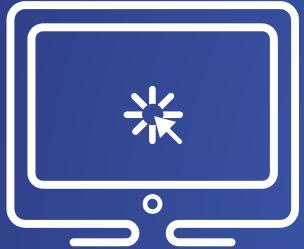
| Name    | Subject | Test Score |
|---------|---------|------------|
| Judy    | Math    | 97         |
| Judy    | Reading | 91         |
| Barbara | Math    | 96         |
| Barbara | Reading | 86         |
| Louise  | Math    | 92         |
| Louise  | Reading | 99         |
| James   | Math    | 90         |
| James   | Reading | 85         |

test\_clean

| Name    | Subject | Test Score |
|---------|---------|------------|
| Alfred  | Math    | 82         |
| Alice   | Math    | 71         |
| Barbara | Math    | 96         |
| Carol   | Math    | 61         |
| Henry   | Math    | 85         |
| James   | Math    | 90         |
| Jane    | Math    | 84         |
| Janet   | Math    | 75         |

test\_dups

| Name    | Subject | Test Score |
|---------|---------|------------|
| Alfred  | Reading | 79         |
| Alice   | Reading | 67         |
| Barbara | Reading | 86         |
| Carol   | Reading | 57         |
| Henry   | Reading | 86         |
| James   | Reading | 85         |
| Jane    | Reading | 76         |
| Janet   | Reading | 71         |



# Identifying and Removing Duplicate Values

This demonstration compares using the NODUPRECS and NODUPKEY options in PROC SORT with First. and Last. to identify and remove duplicates.

# Lesson Quiz



1. Which of the following contains valid syntax?

a.

```
value grades 'A'='Excellent'  
             'B'='Good';
```

b.

```
value qtrfmt 1,2,3='First'  
        4,5,6='Second';
```

c.

```
value $grades. 'A'='Excellent'  
             'B'='Good';
```

d.

```
value qtrfmt '1'-'3'='First'  
           '4'-'6'='Second';
```

1. Which of the following contains valid syntax?

a.

```
value grades 'A'='Excellent'  
             'B'='Good';
```

b.

```
value qtrfmt 1,2,3='First'  
        4,5,6='Second';
```

c.

```
value $grades. 'A'='Excellent'  
             'B'='Good';
```

d.

```
value qtrfmt '1'-'3'='First'  
           '4'-'6'='Second';
```

2. What is the formatted value for the value of 100 given the following step?

```
proc format;
  value rates  1-<100='low'
              100<-<200='medium'
              200-300 = 'high'
              other = 'out of range';
run;
```

- a. *low*
- b. *medium*
- c. *high*
- d. *out of range*

2. What is the formatted value for the value of 100 given the following step?

```
proc format;  
    value rates    1-<100='low'  
                  100<-<200='medium'  
                  200-300 = 'high'  
                  other = 'out of range';  
run;
```

- a. *low*
- b. *medium*
- c. *high*
- d. *out of range*

3. In the FORMAT procedure, you specify the name of the format and the name of the column that will use the custom format.
  - a. True
  - b. False

3. In the FORMAT procedure, you specify the name of the format and the name of the column that will use the custom format.
- a. True
- b. False

5. Which one of the following does not have proper syntax for specifying a range in the VALUE statement?
- a. **500>-700**
  - b. **500-<700**
  - c. **'A'-'C'**
  - d. **'Horse'-'Mouse'**

5. Which one of the following does not have proper syntax for specifying a range in the VALUE statement?

- a. **500>-700**
- b. **500-<700**
- c. **'A'-'C'**
- d. **'Horse'-'Mouse'**

8. Which option in the PROC FORMAT statement specifies a library to store a custom format?
- a. CATALOG=
  - b. FMTLIB=
  - c. LIBRARY=
  - d. STORE=

8. Which option in the PROC FORMAT statement specifies a library to store a custom format?
- a. CATALOG=
  - b. FMTLIB=
  - c. LIBRARY= 
  - d. STORE=

9. What is the default search order that is used to locate formats?
- a. LIBRARY.FORMATS  $\Rightarrow$  WORK.FORMATS
  - b. SASHELP.FORMATS  $\Rightarrow$  LIBRARY.FORMATS
  - c. SASHELP.FORMATS  $\Rightarrow$  WORK.FORMATS
  - d. WORK.FORMATS  $\Rightarrow$  LIBRARY.FORMATS

9. What is the default search order that is used to locate formats?
- a. LIBRARY.FORMATS  $\Rightarrow$  WORK.FORMATS
  - b. SASHELP.FORMATS  $\Rightarrow$  LIBRARY.FORMATS
  - c. SASHELP.FORMATS  $\Rightarrow$  WORK.FORMATS
  - d. WORK.FORMATS  $\Rightarrow$  LIBRARY.FORMATS

10. Which of the following contains valid syntax for the FMTSEARCH= option?
- a. `options fmtsearch=sashelp;`
  - b. `options fmtsearch=sashelp.formats;`
  - c. `options fmtsearch=(sashelp sashelp.fmts);`
  - d. `options fmtsearch=[sashelp.fmts sashelp];`

10. Which of the following contains valid syntax for the FMTSEARCH= option?

- a. `options fmtsearch=sashelp;`
- b. `options fmtsearch=sashelp.formats;`
- c. `options fmtsearch=(sashelp sashelp.fmts);`
- d. `options fmtsearch=[sashelp.fmts sashelp];`

1. Which statement contains valid syntax for the RETAIN statement?
  - a. **`retain year 2018;`**
  - b. **`retain year*2018;`**
  - c. **`retain year=2018;`**
  - d. **`retain year{2018};`**

1. Which statement contains valid syntax for the RETAIN statement?

- a. `retain year 2018;`
- b. `retain year*2018;`
- c. `retain year=2018;`
- d. `retain year{2018};`

2. Which statement is false concerning the sum statement?
- a. The sum statement ignores missing values.
  - b. The sum statement initially sets the accumulator column to missing.
  - c. The sum statements adds a numeric value to an accumulator column.
  - d. The sum statement automatically retains the value of the accumulating column.

2. Which statement is false concerning the sum statement?
- a. The sum statement ignores missing values.
  - b.** The sum statement initially sets the accumulator column to missing.
  - c. The sum statements adds a numeric value to an accumulator column.
  - d. The sum statement automatically retains the value of the accumulating column.

3. What is the value of Count at the end of the third DATA step iteration?

work.nums

| 123 | Tens |
|-----|------|
|     | 10   |
|     | 20   |
|     | 30   |
|     | 40   |

```
data newnums;  
  set nums;  
  retain Count 100;  
  Count+Tens;  
run;
```

- a. . (missing)
- b. 60
- c. 130
- d. 160

3. What is the value of Count at the end of the third DATA step iteration?

work.nums

| 123 | Tens |
|-----|------|
|     | 10   |
|     | 20   |
|     | 30   |
|     | 40   |

```
data newnums;  
  set nums;  
  retain Count 100;  
  Count+Tens;  
run;
```

- a. . (missing)
- b. 60
- c. 130
- d. 160

4. What is the value of Count at the end of the third DATA step iteration?

work.nums

| 123 | Tens |
|-----|------|
|     | 10   |
|     | 20   |
|     | .    |
|     | 40   |

```
data newnums;  
  set nums;  
  Count+Tens;  
run;
```

- a. . (missing)
- b. 20
- c. 30
- d. 70

4. What is the value of Count at the end of the third DATA step iteration?

work.nums

| 123 | Tens |
|-----|------|
|     | 10   |
|     | 20   |
|     | .    |
|     | 40   |

```
data newnums;  
  set nums;  
  Count+Tens;  
run;
```

- a. . (missing)
- b. 20
- c. 30
- d. 70

5. Which step executes successfully without an error, given the input table sashelp.class?

| Name   | Sex | Age | Height | Weight |
|--------|-----|-----|--------|--------|
| Alfred | M   | 14  | 69     | 112.5  |
| Alice  | F   | 13  | 56.5   | 84     |

a.

```
data new;
  set sashelp.class;
  Ratio=Height/Weight;
  where Sex='M' & Ratio>0.6;
run;
```

b.

```
data new;
  set sashelp.class;
  where Sex='M';
  Ratio=Height/Weight;
  where Ratio>0.6;
run;
```

c.

```
data new;
  set sashelp.class;
  where Sex='M';
  Ratio=Height/Weight;
  if Ratio>0.6;
run;
```

d.

```
data new;
  set sashelp.class;
  if Sex='M';
  Ratio=Height/Weight;
  where Ratio>0.6;
run;
```

5. Which step executes successfully without an error, given the input table sashelp.class?

| Name   | Sex | Age | Height | Weight |
|--------|-----|-----|--------|--------|
| Alfred | M   | 14  | 69     | 112.5  |
| Alice  | F   | 13  | 56.5   | 84     |

a.

```
data new;
  set sashelp.class;
  Ratio=Height/Weight;
  where Sex='M' & Ratio>0.6;
run;
```

b.

```
data new;
  set sashelp.class;
  where Sex='M';
  Ratio=Height/Weight;
  where Ratio>0.6;
run;
```

c.

```
data new;
  set sashelp.class;
  where Sex='M';
  Ratio=Height/Weight;
  if Ratio>0.6;
run;
```

d.

```
data new;
  set sashelp.class;
  if Sex='M';
  Ratio=Height/Weight;
  where Ratio>0.6;
run;
```

6. Which statement is true given the following program?

```
data work.student;
  set sashelp.class;
  by Name;
run;
```

- a. The PDV contains a temporary variable named **First.Name**.
- b. The output table **work.student** contains a column named **Last.Name**.
- c. The DATA step sorts the input table by the column **Name**.
- d. An error is produced because the BY statement is not permitted in the DATA step.

6. Which statement is true given the following program?

```
data work.student;
  set sashelp.class;
  by Name;
run;
```

- a. The PDV contains a temporary variable named **First.Name**.
- b. The output table **work.student** contains a column named **Last.Name**.
- c. The DATA step sorts the input table by the column **Name**.
- d. An error is produced because the BY statement is not permitted in the DATA step.

7. What are the correct values for **First.Name** and **Last.Name** if the value of **Name** appears only once in the input table?

```
data work.student;
  set sashelp.class;
  by Name;
run;
```

- a. **First.Name**=0 and **Last.Name**=0
- b. **First.Name**=1 and **Last.Name**=1
- c. **First.Name**=1 and **Last.Name**=0
- d. **First.Name**=0 and **Last.Name**=1

7. What are the correct values for **First.Name** and **Last.Name** if the value of **Name** appears only once in the input table?

```
data work.student;
  set sashelp.class;
  by Name;
run;
```

- a. First.Name=0 and Last.Name=0
- b. First.Name=1 and Last.Name=1
- c. First.Name=1 and Last.Name=0
- d. First.Name=0 and Last.Name=1

8. Which DATA step statement indicates to continue processing the last row of a BY group?
- a. `if First.JobTitle;`
  - b. `if Last.JobTitle;`
  - c. `where First.JobTitle=1;`
  - d. `where Last.JobTitle=1;`

8. Which DATA step statement indicates to continue processing the last row of a BY group?
- a. `if First.JobTitle;`
  - b. `if Last.JobTitle;`
  - c. `where First.JobTitle=1;`
  - d. `where Last.JobTitle=1;`

9. Which statement needs to be added to the DATA step to reset the value of **TotalWt** for each new BY group?

```
data GenderWeight;
  set Students;
  by Gender;
  ... add statement here ...
  TotalWt+Weight;
  if Last.Gender=1 then output;
run;
```

- a. **TotalWt=0;**
- b. **if Last.Gender=0 then TotalWt=0;**
- c. **if First.Gender=0 then TotalWt=0;**
- d. **if First.Gender=1 then TotalWt=0;**

9. Which statement needs to be added to the DATA step to reset the value of **TotalWt** for each new BY group?

```
data GenderWeight;  
  set Students;  
  by Gender;  
  ... add statement here ...  
  TotalWt+Weight;  
  if Last.Gender=1 then output;  
run;
```

- a. **TotalWt=0;**
- b. **if Last.Gender=0 then TotalWt=0;**
- c. **if First.Gender=0 then TotalWt=0;**
- d. **if First.Gender=1 then TotalWt=0;**

# STAT604 SAS Lesson 12

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Multiple BY Columns



| ⚠ | Name               | Date    | Rain_mm | Year | Qtr |
|---|--------------------|---------|---------|------|-----|
|   | SYDNEY AIRPORT, AS | 01-2017 | 48.4    | 2017 | 1   |
|   | SYDNEY AIRPORT, AS | 02-2017 | 158     | 2017 | 1   |
|   | SYDNEY AIRPORT, AS | 03-2017 | 189.2   | 2017 | 1   |
|   | SYDNEY AIRPORT, AS | 04-2017 | 94.4    | 2017 | 2   |
|   | SYDNEY AIRPORT, AS | 05-2017 | 32.4    | 2017 | 2   |
|   | SYDNEY AIRPORT, AS | 06-2017 | 113.6   | 2017 | 2   |
|   | SYDNEY AIRPORT, AS | 07-2017 | 18      | 2017 | 3   |
|   | SYDNEY AIRPORT, AS | 08-2017 | 27.2    | 2017 | 3   |
|   | SYDNEY AIRPORT, AS | 09-2017 | 0.2     | 2017 | 3   |
|   | SYDNEY AIRPORT, AS | 10-2017 | 59.6    | 2017 | 4   |
|   | SYDNEY AIRPORT, AS | 11-2017 | 37.8    | 2017 | 4   |
|   | SYDNEY AIRPORT, AS | 12-2017 | 44.4    | 2017 | 4   |
|   | SYDNEY AIRPORT, AS | 01-2018 | 27.6    | 2018 | 1   |
|   | SYDNEY AIRPORT, AS | 02-2018 | 88.4    | 2018 | 1   |

# Multiple BY Columns

```
data sydney_summary;  
  set pg2.weather_sydney;  
  by Year Qtr;  
run;
```

## PDV

| other columns | Year | Qtr | First.Year | Last.Year | First.Qtr | Last.Qtr |
|---------------|------|-----|------------|-----------|-----------|----------|
|               |      |     | D          | D         | D         | D        |

First./Last. variables are created for each column in the BY statement.

# Sort by Multiple Variables First

In the upcoming illustration, the data must be sorted by multiple BY variables: Proj and Dept.

```
proc sort data=orion.projsals
           out=projsort;
   by Proj Dept;
run;
```

primary sort  
variable

secondary sort  
variable

# First. / Last. Values: First DATA Step Iteration

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |



| First.Proj |
|------------|
| 1          |

| First.Dept |
|------------|
| 1          |

| Last.Proj |
|-----------|
| ?         |

| Last.Dept |
|-----------|
| ?         |

# First. / Last. Values: First DATA Step Iteration

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |
| C    | SALES  |

C SAS looks ahead at the next observation to determine the values for Last.Proj and Last.Dept.



| First.Proj |
|------------|
| 1          |

| First.Dept |
|------------|
| 1          |

| Last.Proj |
|-----------|
| 0         |

| Last.Dept |
|-----------|
| 0         |

# First. / Last. Values: Second DATA Step Iteration

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |



| First.Proj |
|------------|
| 0          |

| First.Dept |
|------------|
| 0          |

| Last.Proj |
|-----------|
| 0         |

| Last.Dept |
|-----------|
| 0         |

# First. / Last. Values: Third DATA Step Iteration

*& when over the project start changes, in this case Proj, everything resets*

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |

|            |
|------------|
| First.Proj |
| 0          |

|            |
|------------|
| First.Dept |
| 0          |

|           |
|-----------|
| Last.Proj |
| 1         |

|           |
|-----------|
| Last.Dept |
| 1         |

# First. / Last. Values: Fourth DATA Step Iteration

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |



| First.Proj |
|------------|
| 1          |

| First.Dept |
|------------|
| 1          |

| Last.Proj |
|-----------|
| 0         |

| Last.Dept |
|-----------|
| 1         |

# First. / Last. Values: Fifth DATA Step Iteration

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |



| First.Proj |
|------------|
| 0          |

| First.Dept |
|------------|
| 1          |

| Last.Proj |
|-----------|
| 0         |

| Last.Dept |
|-----------|
| 0         |

# Short Answer Poll

What are the values for First. and Last. variables when the DATA step is processing the observation indicated by the arrow?

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |
| C    | SALES  |



|            |   |   |
|------------|---|---|
| First.Proj | ? | O |
| First.Dept | ? | O |
| Last.Proj  | ? | I |
| Last.Dept  | ? | I |

# Short Answer Poll – Correct Answer

What are the values for First. and Last. variables when the DATA step is processing the observation indicated by the arrow?

| Proj | Dept   |
|------|--------|
| A    | ADMIN  |
| A    | ADMIN  |
| A    | ADMIN  |
| B    | ADMIN  |
| B    | ENGINR |
| B    | ENGINR |
| C    | ENGINR |
| C    | SALES  |
| C    | SALES  |



| First.Proj |
|------------|
| 0          |

| First.Dept |
|------------|
| 0          |

| Last.Proj |
|-----------|
| 1         |

| Last.Dept |
|-----------|
| 1         |

✗

# First. and Last. for Multiple BY Variables

When you use more than one variable in the BY statement,  
`Last.BY-variable=1` for the *primary variable* forces `Last.BY-variable=1` for the *secondary variable(s)*.

| Proj | Dept   | First. | Last. | First. |           |
|------|--------|--------|-------|--------|-----------|
|      |        | Proj   | Proj  | Dept   | Last.Dept |
| A    | ADMIN  | 1      | 0     | 1      | 0         |
| A    | ADMIN  | 0      | 0     | 0      | 0         |
| A    | ADMIN  | 0      | 1     | 0      | 1         |
| B    | ADMIN  | 1      | 0     | 1      | 1         |
| B    | ENGINR | 0      | 0     | 1      | 0         |

change in  
Primary

change in  
Secondary



## Using Multiple By Groups

This demonstration compares PROC SORT with the DATA step from removing duplicates and illustrates using multiple by groups to create multiple accumulating variables.

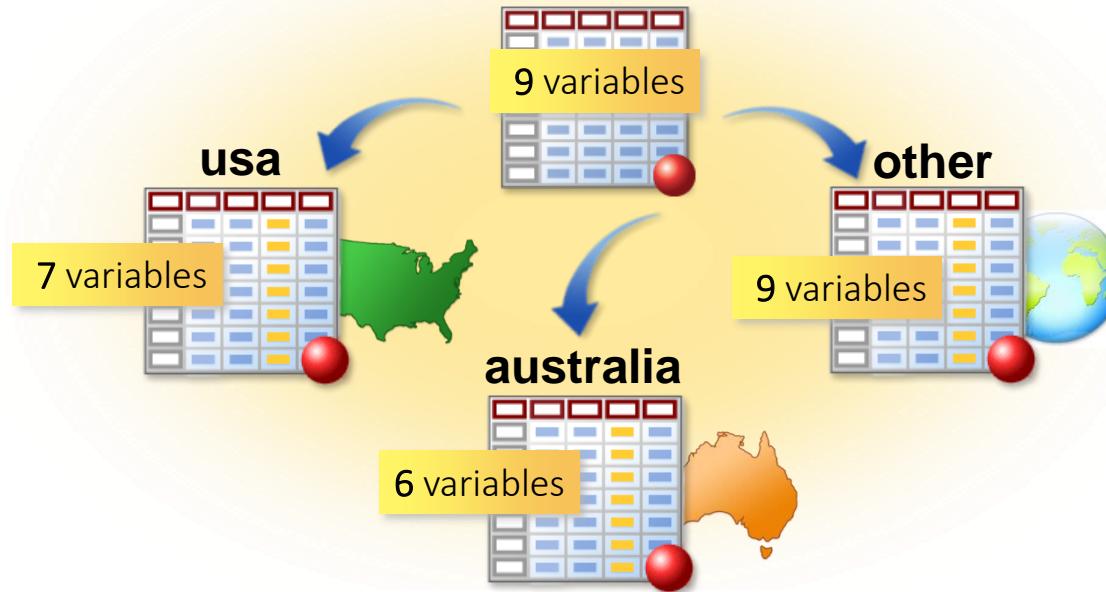
# Controlling Column Usage

Drop and Keep Data Set Options – Prep Guide Page 156

# Business Scenario

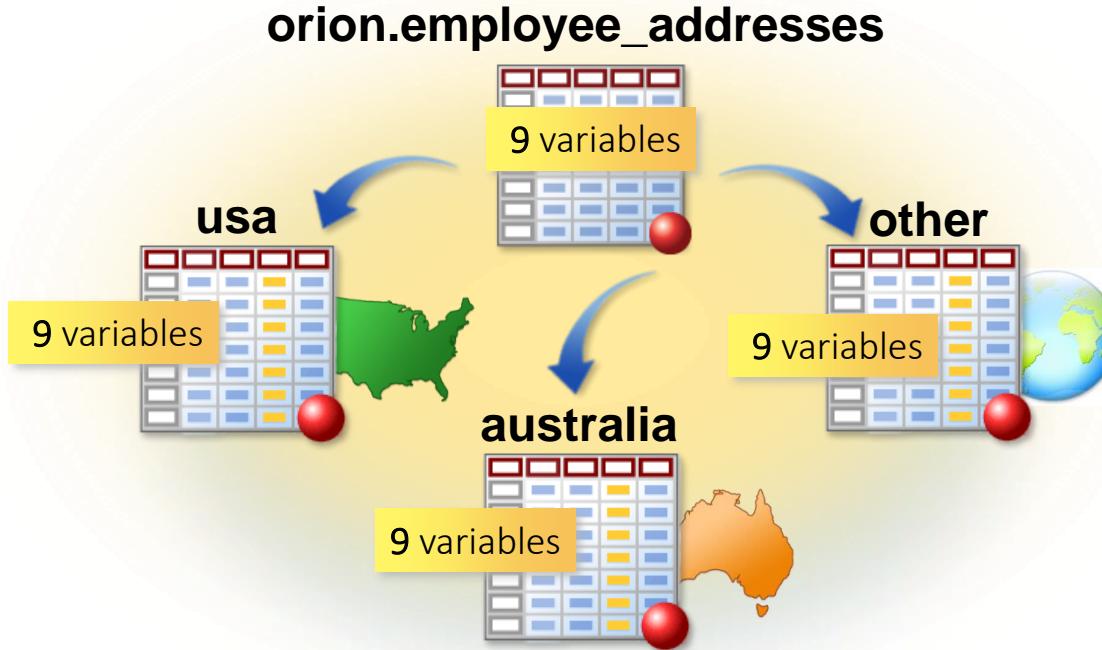
You have controlled the observations written to the three new data sets, and now you want to control the variables that are written to each.

## **orion.employee\_addresses**



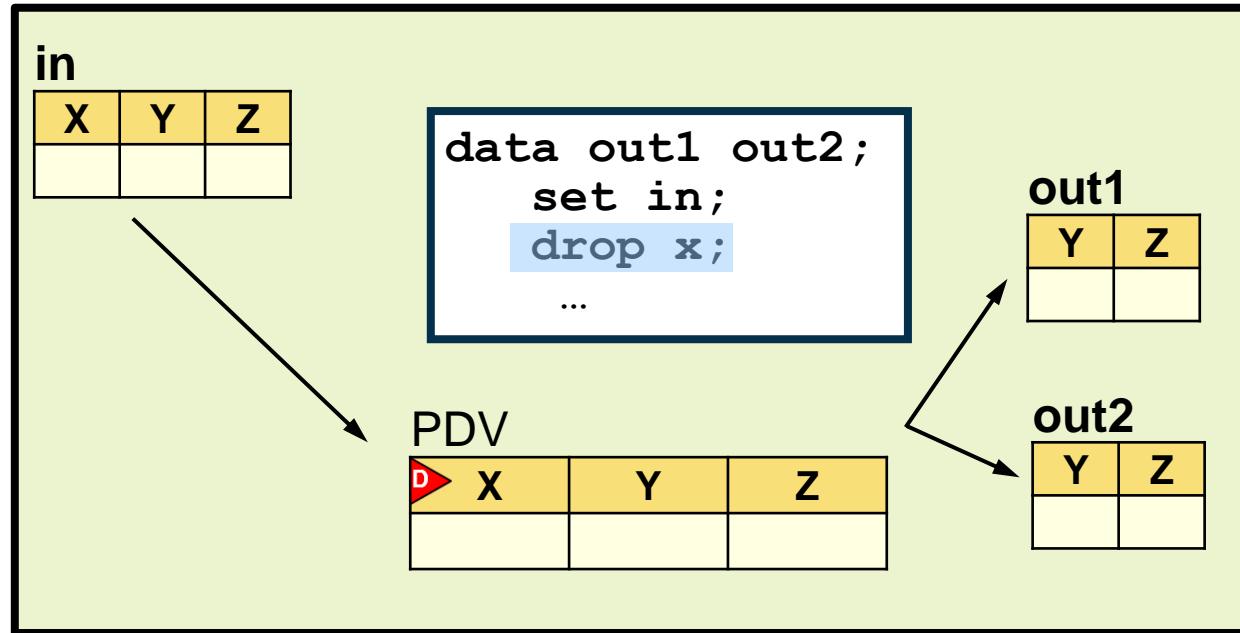
# Controlling Variable Output (Review)

By default, SAS writes all variables from the input data set to every output data set.



# Controlling Variable Output (Review)

In the DATA step, the DROP and KEEP statements can be used to control which variables are written to output data sets.



# DROP Statement

The *DROP statement* drops variables from every output data set.

```
data usa australia other;
  drop Street_ID;
  set orion.employee_addresses;
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 8 variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and 8
      variables.
NOTE: The data set WORK.OTHER has 8 observations and 8 variables.
```

# Controlling Variable Output

The task is to drop **Street\_ID** and **Country** from **usa**, drop **Street\_ID**, **Country**, and **State** from **australia**, and keep all variables in **other**.

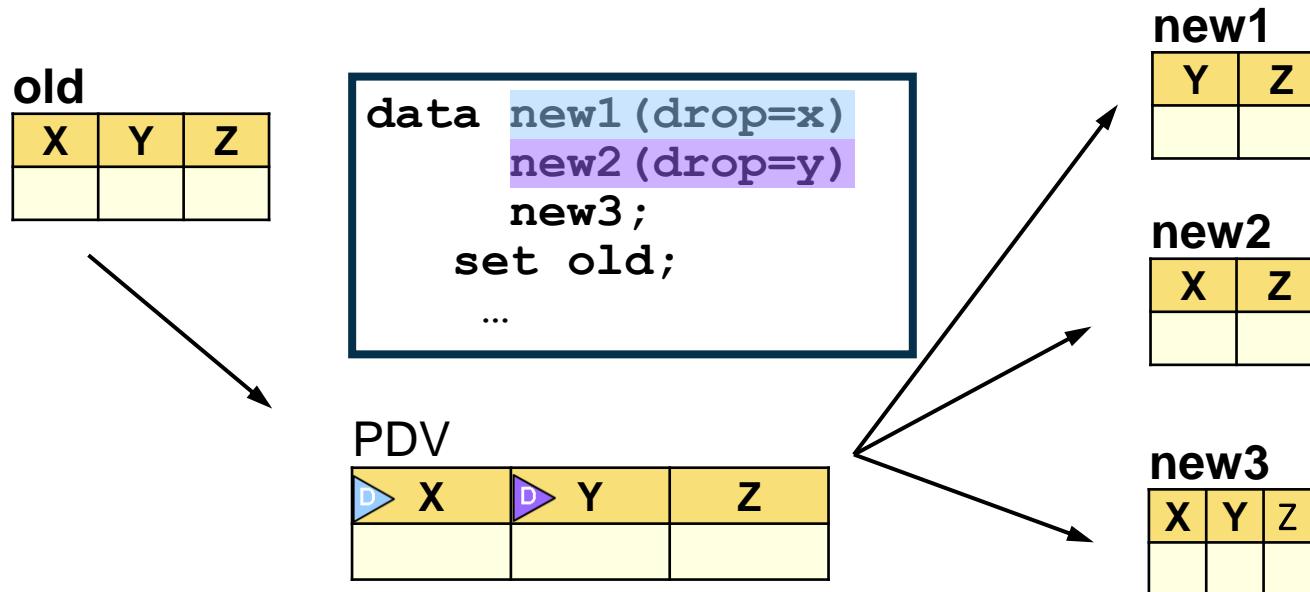
| Employee_ID   |                      | Employee_Name |                 | USA          |    |       |
|---------------|----------------------|---------------|-----------------|--------------|----|-------|
| Street_Number | Street_Name          | City          | State           | Postal_Code  |    |       |
| 121044        | Abbott, Ray          | 2267          | Edwards Mill Rd | Miami-Dade   | FL | 33135 |
| 120761        | Akinfolarin, Tameaka | 5             | Donnybrook Rd   | Philadelphia | PA | 19145 |
| 120656        | Amos, Salley         | 3524          | Calico Ct       | San Diego    | CA | 92116 |

| Employee_ID   |                 | Australia |                |           |      |
|---------------|-----------------|-----------|----------------|-----------|------|
| Street_Number | Street_Name     | City      | Postal_Code    |           |      |
| 120145        | Aisbitt, Sandy  | 30        | Bingera Street | Melbourne | 2001 |
| 120185        | Bahlman, Sharon | 24        | LaTrobe Street | Sydney    | 2165 |
| 120109        | Baker, Gabriele | 166       | Toorak Road    | Sydney    | 2119 |

| Other       |                  |            |               |               |              |       |             |         |
|-------------|------------------|------------|---------------|---------------|--------------|-------|-------------|---------|
| Employee_ID | Employee_Name    | Street_ID  | Street_Number | Street_Name   | City         | State | Postal_Code | Country |
| 121019      | Desanctis, Scott | 9260121087 | 765           | Greenhaven Ln | Philadelphia | PA    | 19102       | us      |
| 120997      | Donathan, Mary   | 9260121069 | 4923          | Gateridge Dr  | Philadelphia | PA    | 19152       | us      |
| 120747      | Farthing, Zashia | 9260123756 | 763           | Chatterson Dr | San Diego    | CA    | 92116       | us      |

# DROP= Option on an Output Data Set

SAS-data-set(**DROP=variable-1 <variable-2 ...variable-n>**)



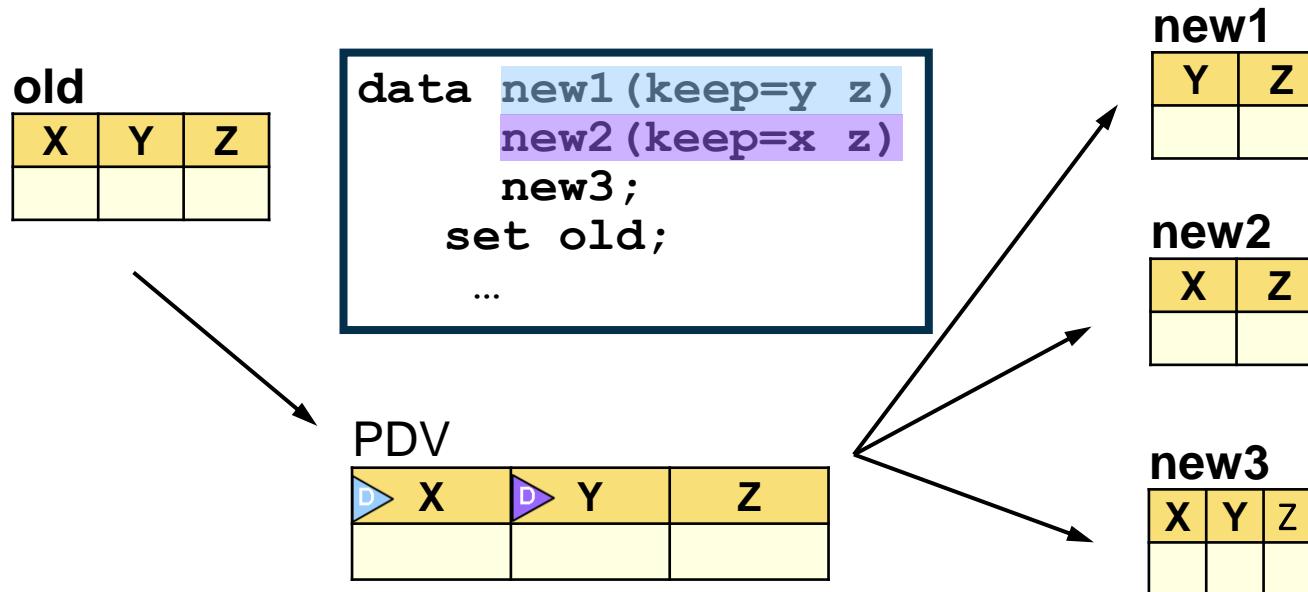
# Using the DROP= Data Set Option

```
data usa(drop=Street_ID Country)
       australia(drop=Street_ID State Country)
       other;
set orion.employee_addresses;
if Country='US' then output usa;
else if Country='AU' then
      output australia;
else output other;
run;
```

NOTE: There were 424 observations read from the data set  
ORION.EMPLOYEE\_ADDRESSES.  
NOTE: The data set WORK.USA has 311 observations  
and 7 variables.  
NOTE: The data set WORK.AUSTRALIA has 105 observations  
and 6 variables.  
NOTE: The data set WORK.OTHER has 8 observations  
and 9 variables.

# KEEP= Option on an Output Data Set

SAS-data-set(**KEEP=variable-1 <variable-2 ...variable-n>**)



# Using the DROP= and KEEP= Options

The DROP= and KEEP= options can both be used in a SAS program.

```
data usa(keep=Employee_Name City State)
       australia(drop=Street_ID State)
       other;
  set orion.employee_addresses;
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

# Short Answer Poll

The data set **orion.employee\_addresses** contains nine variables. How many variables are in the **usa**, **australia**, and **other** data sets?

```
data usa(keep=Employee_Name City State Country)
      australia(drop=Street_ID State Country)
      other;
  set orion.employee_addresses;
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

# Short Answer Poll – Correct Answer

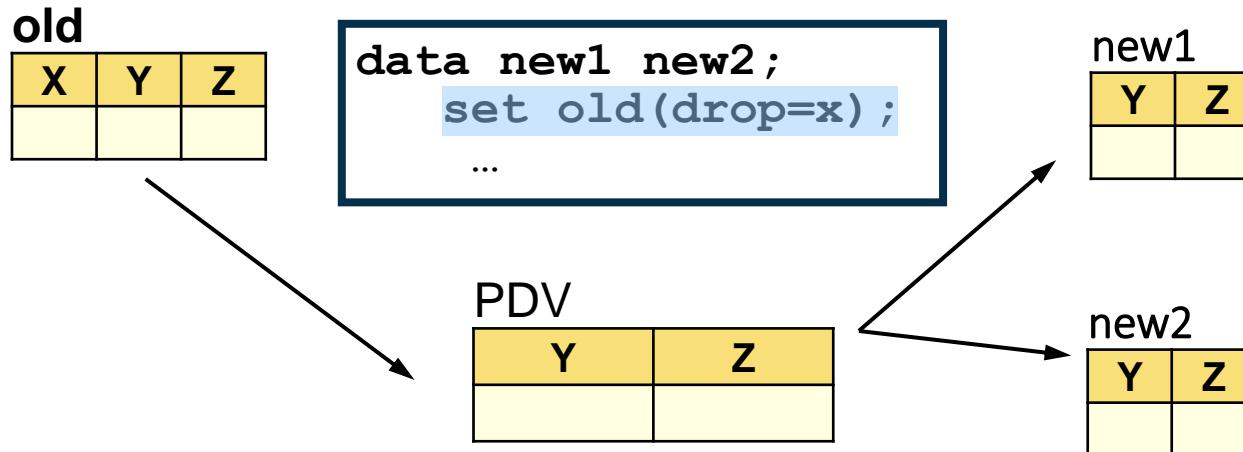
The data set `orion.employee_addresses` contains nine variables. How many variables are in the `usa`, `australia`, and `other` data sets? 4, 6, 9

```
data usa(keep=Employee_Name City State Country)
       australia(drop=Street_ID State Country)
       other;
set orion.employee_addresses;
if Country='US' then output usa;
else if Country='AU' then output australia;
else output other;
run;
```

Four variables are kept in `usa`, three are dropped from `australia`, and there is no DROP or KEEP for `other`.

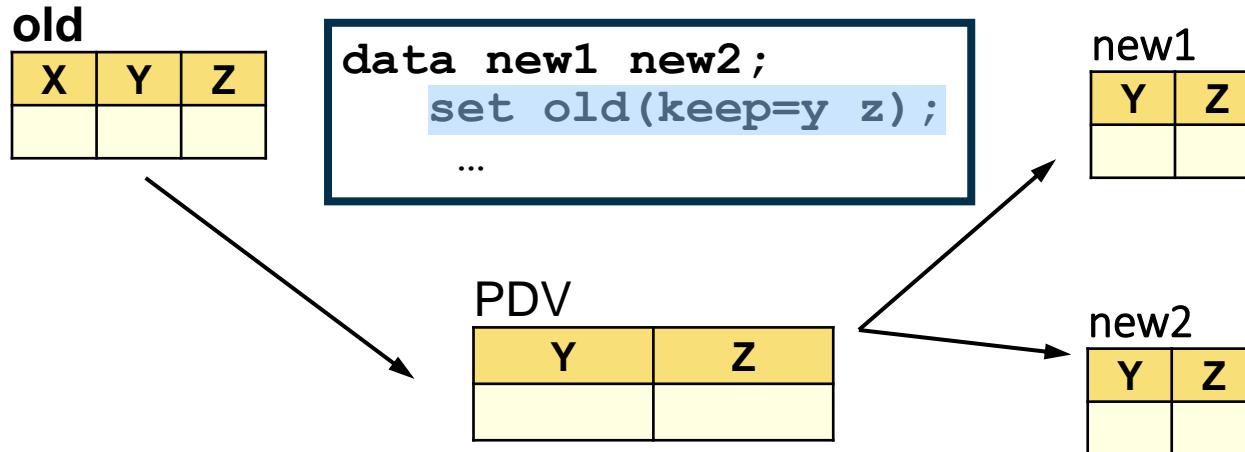
# Using DROP= Option on an Input Data Set

When a **DROP=** data set option is used on an input data set, the specified variables are not read into the PDV and therefore are *not* available for processing.



# Using KEEP= Option on an Input Data Set

When a **KEEP=** data set option is used on an input data set, only the specified variables are read into the PDV and therefore are available for processing.



# Short Answer Poll

Evaluate the program shown below. The intent is to drop Country, Street\_ID, and Employee\_ID from every data set, and to drop State from australia. What is wrong with the program?

```
data usa australia(drop=State) other;
  set orion.employee_addresses
    (drop=Country Street_ID Employee_ID);
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```



## 2.07 Short Answer Poll – Correct Answer

Country is dropped on input, and therefore it is not available for processing.  
Every observation is written to other.

```
data usa australia(drop=State) other;
  set orion.employee_addresses
    (drop=Country Street_ID Employee_ID);
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

PDV

Country is not included  
in the PDV.

| City | Country | Employee_Name | Postal_Code | D State | Street_Name | Street_Number |
|------|---------|---------------|-------------|---------|-------------|---------------|
|      | 🚫       |               |             |         |             |               |



# Improved Solution

Use a combination of the DROP= option and the DROP statement to achieve the desired results.

```
data usa australia(drop=State) other;
  set orion.employee_addresses
    (drop=Street_ID Employee_ID);
  drop Country;
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

State is dropped only from australia.

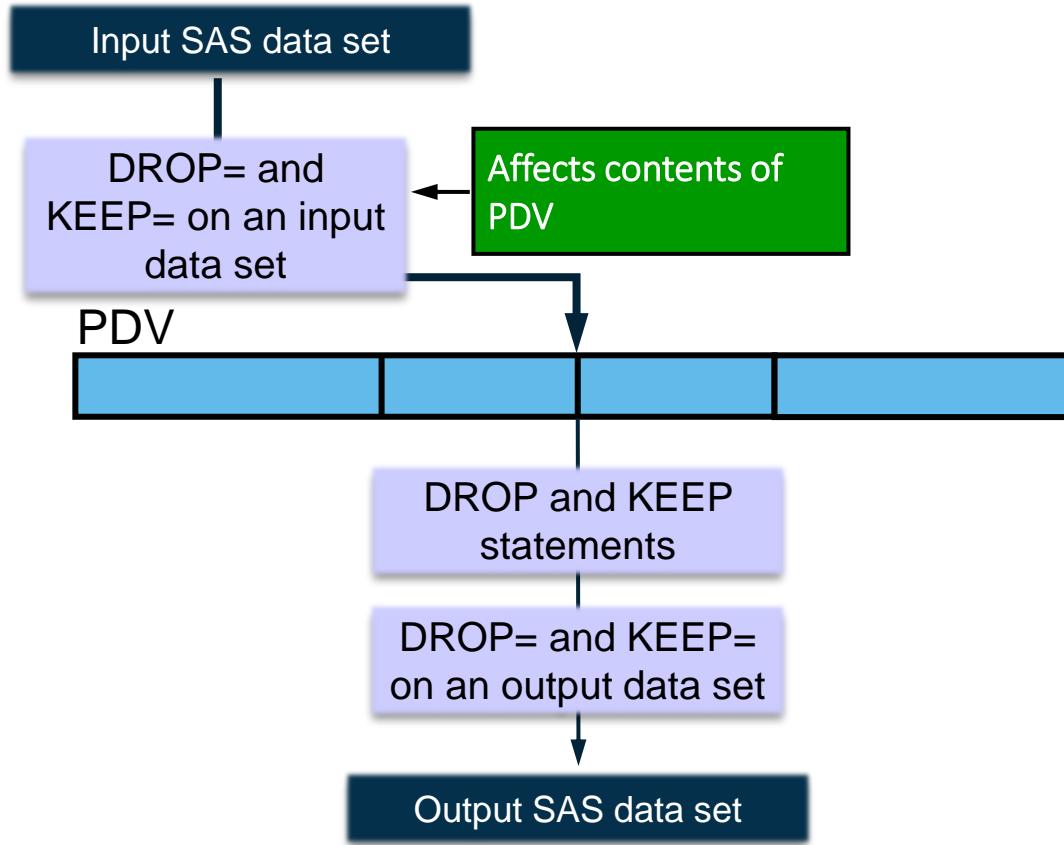
PDV

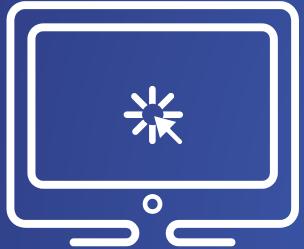
| City | D<br>Country | Employee_Name | Postal_Code | D<br>State | Street_Name | Street_Number |
|------|--------------|---------------|-------------|------------|-------------|---------------|
|      |              |               |             |            |             |               |

# Check the SAS Log

```
NOTE: There were 424 observations read from the data set  
      ORION.EMPLOYEE_ADDRESSES.  
NOTE: The data set WORK.USA has 311 observations  
      and 6 variables.  
NOTE: The data set WORK.AUSTRALIA has 105 observations  
      and 5 variables.  
NOTE: The data set WORK.OTHER has 8 observations  
      and 6 variables.
```

# Controlling Variable Input





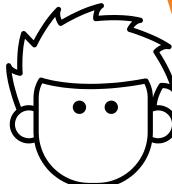
# Controlling Columns

This demonstration illustrates the effects of using a DROP option on the input data set.

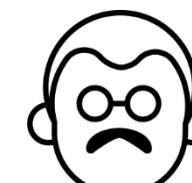
# Combining Data

Combining SAS Data Sets – Chapter 10

# Before Combining – Know Thy Data



How are the data sets related?

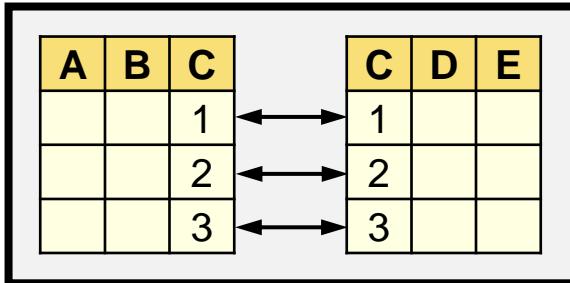


How are the rows ordered?



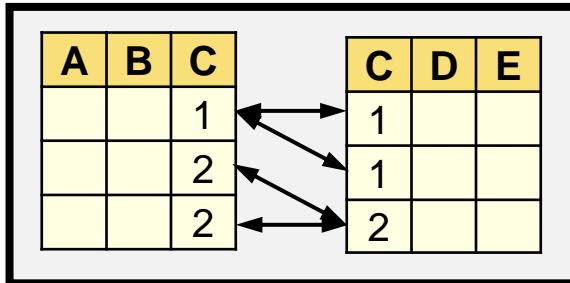
What columns do they have in common?

# Relationships in Data



## One-to-One

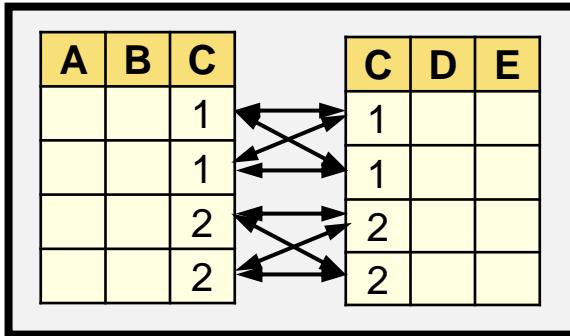
A single observation in one data set is related to exactly one observation in another data set based on the values of one or more selected variables.



## One-to-Many or Many-to-One

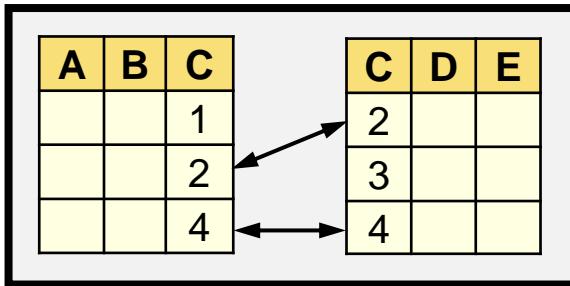
A single observation in one data set is related to more than one observation in another data set based on the values of one or more selected variables.

# Relationships in Data



## Many-to-Many

Multiple observations in one data set are related to more than one observation in another data set based on the values of one or more selected variables.



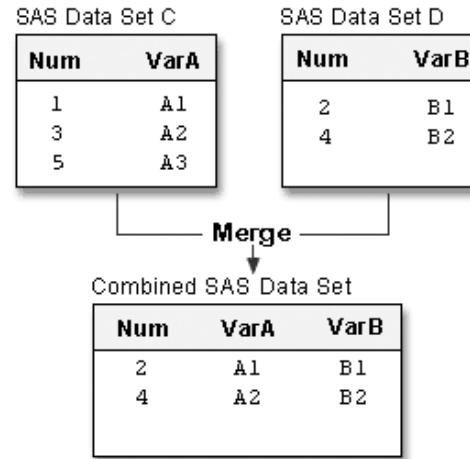
## Nonmatches

At least one observation in one data set is unrelated to any observation in another data set based on the values of one or more selected variables.



# One-to-One Reading

- Creates observations that contain all of the variables from each contributing data set.
- Variables with the same name overwrite with values of last data set.
- Combines observations based on their relative position in each data set.
- Using multiple SET statements - DATA step stops after it has read the last row from the smallest data set.
- Similar to MERGE with no BY, which reads all observations and populates extra rows with missing values.



| Num | VarA | VarB |
|-----|------|------|
| 2   | A1   | B1   |
| 4   | A2   | B2   |
| 5   | A3   |      |

# One-to-One Reading

In the following example, you have basic patient data in Cert.Patients that you want to combine with other patient data that is stored in Cert.Measure. The height and weight data is stored in the data set Cert.Measure. Both data sets are sorted by the variable ID.



Notice that Cert.Patients contains nine out of eleven observations in which the patient age is less than 60.

**Figure 10.2 Example: One-to-One Reading**

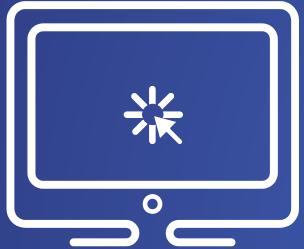
SAS Data Set Cert.Patients

|    | ID   | Sex | Age |
|----|------|-----|-----|
| 1  | 1129 | F   | 48  |
| 2  | 1387 | F   | 57  |
| 3  | 2304 | F   | 16  |
| 4  | 2406 | F   | 66  |
| 5  | 4755 | F   | 66  |
| 6  | 5438 | F   | 42  |
| 7  | 6488 | F   | 59  |
| 8  | 9012 | F   | 39  |
| 9  | 9125 | F   | 56  |
| 10 | 8045 | M   | 40  |
| 11 | 8125 | M   | 39  |

SAS Data Set Cert.Measure

|   | ID   | Height | Weight |
|---|------|--------|--------|
| 1 | 1129 | 61     | 137    |
| 2 | 1387 | 64     | 142    |
| 3 | 2304 | 61     | 102    |
| 4 | 5438 | 62     | 168    |
| 5 | 6488 | 64     | 154    |
| 6 | 8045 | 72     | 200    |
| 7 | 8125 | 70     | 176    |
| 8 | 9012 | 63     | 157    |
| 9 | 9125 | 65     | 148    |

```
data oneZone;  
  set cert.patients;  
  if age < 60 ;  
  set cert.measure;  
run;
```

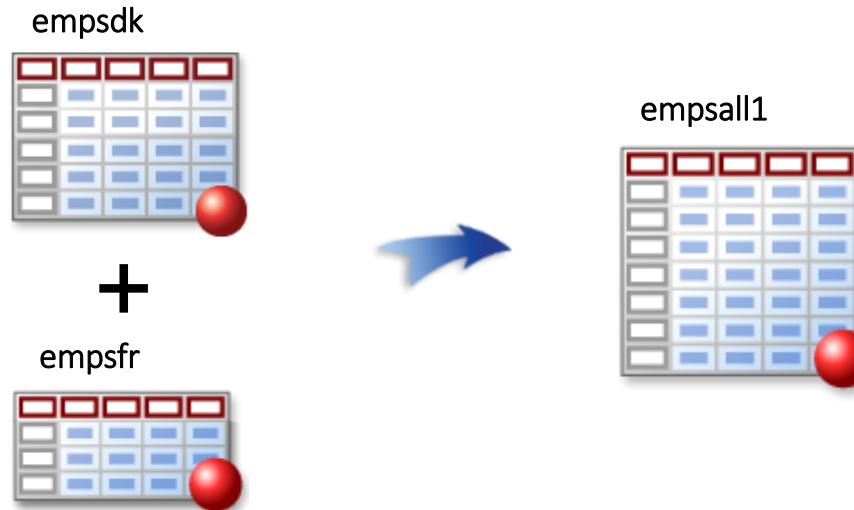


## One-to-One Reading

The demonstration illustrates what happens when data sets that are not sorted are combined using one-to-one reading.

# Business Scenario

You were asked to combine the data sets that contain information about Orion Star employees from Denmark and France into a new data set.



# Considerations

Concatenate like-structured data sets, **empsdk** and **empsfr**, to create a new data set named **empsall1**.

**empsdk**

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

**empsfr**

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |



**empsall1**

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |
| Sophie | F      | France  |

Both data sets contain the same variables.

# Using a DATA Step to Concatenate

Use a DATA step to concatenate the data sets.  
List the data sets in the SET statement.

```
data empsall1;
```

```
  set empsdk empsfr;
```

```
run;
```

```
SET SAS-data-set1 SAS-data-set2 . . .;
```

- The SET statement reads observations from each data set in the order in which they are listed.
- Any number of data sets can be included in the SET statement.

# Compilation

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

empsall1

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

Initialize PDV

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

empsall1

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
|-------|--------|---------|

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

Implicit OUTPUT;  
Implicit RETURN;

```
data empsall1;
  set empsdk empsfr;
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

Data set variables are not reinitialized.

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Kari  | F      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

Implicit OUTPUT;  
Implicit RETURN;

```
data empsall1;
  set empsdk empsfr;
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Kari  | F      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Kari  | F      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |

PDV is not reinitialized.

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Jonas | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

Implicit OUTPUT;  
Implicit RETURN;

```
data empsall1;
  set empsdk empsfr;
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Jonas | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Jonas | M      | Denmark |

PDV is not reinitialized.

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| EOF   | Jonas  | M       |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
| Jonas | M      | Denmark |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

Reinitialize PDV

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

PDV is reinitialized before  
processing the next data set.

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |

empsall1

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

Implicit OUTPUT;  
Implicit RETURN;

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |

PDV is not reinitialized.

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Sophie | F      | France  |

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

Implicit OUTPUT;  
Implicit RETURN;

```
data empsall1;
  set empsdk empsfr;
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Sophie | F      | France  |

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |
| Sophie | F      | France  |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| Sophie | F      | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Sophie | F      | France  |

PDV is not reinitialized.

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |
| Sophie | F      | France  |

# Execution

empsdk

| First | Gender | Country |
|-------|--------|---------|
| Lars  | M      | Denmark |
| Kari  | F      | Denmark |
| Jonas | M      | Denmark |

empsfr

| First  | Gender | Country |
|--------|--------|---------|
| Pierre | M      | France  |
| EOF    | phie   | France  |

```
data empsall1;  
  set empsdk empsfr;  
run;
```

PDV

| First  | Gender | Country |
|--------|--------|---------|
| Sophie | F      | France  |

empsall1

| First  | Gender | Country |
|--------|--------|---------|
| Lars   | M      | Denmark |
| Kari   | F      | Denmark |
| Jonas  | M      | Denmark |
| Pierre | M      | France  |
| Sophie | F      | France  |

# Viewing the Log

## Partial SAS Log

```
145 data empsall1;  
146     set empsdk empsfr;  
147 run;
```

```
NOTE: There were 3 observations read from the data set WORK.EMPSDK.  
NOTE: There were 2 observations read from the data set WORK.EMPSFR.  
NOTE: The data set WORK.EmpsAll1 has 5 observations and 3  
variables.
```

# Unlike-Structured Data Sets

Concatenate **empscn** and **empsjp** to create a new data set named **empsall2**.

**empscn**

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

**empsjp**

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

The data sets do not contain the same variables.

```
data empsall2;
  set empscn empsjp;
run;
```

# Short Answer Poll

How many variables will be in `empsall2` after concatenating `empscn` and `empsjp`?

`empscn`

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

`empsjp`

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall2;  
  set empscn empsjp;  
run;
```

# Short Answer Poll – Correct Answer

How many variables will be in `empsall2` after concatenating `empscn` and `empsjp`?

`empscn`

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

`empsjp`

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

Four variables: First, Gender, Country, and Region

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall12;
    set empscn empsjp;
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |



```
data empsall2;  
    set empscn empsjp;  
run;
```

PDV

| First | Gender | Country | Region |
|-------|--------|---------|--------|
|       |        |         |        |



# Final Results

empsall2

| First | Gender | Country | Region |
|-------|--------|---------|--------|
| Chang | M      | China   |        |
| Li    | M      | China   |        |
| Ming  | F      | China   |        |
| Cho   | F      |         | Japan  |
| Tomi  | M      |         | Japan  |

- **Region** has missing values due to PDV initialization.
- **Country** has missing values due to PDV reinitialization before processing the second data set.

# Business Scenario

Rename variables in one or more data sets to align columns.

**empscn**

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

**empsjp**

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

Rename **Region**  
to **Country**.



# RENAME= Data Set Option(Review)

The *RENAME=* *data set option* changes the name of a variable.

```
data empsall2;  
  set empscn empsjp(rename=(Region=Country));  
run;
```

SAS-data-set (RENAME=(*old-name-1=new-name-1*  
*old-name-2=new-name-2*  
...  
*old-name-n=new-name-n*))

- The RENAME= option must be specified in parentheses immediately after the appropriate SAS data set name.
- The name change affects the PDV and the output data set. It has no effect on the input data set.

# RENAME= Data Set Option

Multiples variables can be renamed in one or more data sets.

```
set empscn(rename=(Country=Region))  
      empsjp;
```

```
set empscn(rename=(First=Fname  
                     Country=Region))  
      empsjp(rename=(First=Fname));
```

```
set empscn  
      empsjp(rename=(Region=Country));
```

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall2;
  set empscn empsjp(rename=(Region=Country)) ;
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall2;  
  set empscn empsjp(rename=(Region=Country)) ;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall2;  
  set empscn empsjp(rename=(Region=Country)) ;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Compilation

empscn

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |

empsjp

| First | Gender | Region |
|-------|--------|--------|
| Cho   | F      | Japan  |
| Tomi  | M      | Japan  |

```
data empsall12;  
  set empscn empsjp(rename=(Region=Country)) ;  
run;
```

PDV

| First | Gender | Country |
|-------|--------|---------|
|       |        |         |

# Final Results

The **Region** values are stored in **Country**.

**empsall2**

| First | Gender | Country |
|-------|--------|---------|
| Chang | M      | China   |
| Li    | M      | China   |
| Ming  | F      | China   |
| Cho   | F      | Japan   |
| Tomi  | M      | Japan   |

# Combining Data Sets

Basic Match Merging

# Match-Merging: Sorting the Data Sets

```
PROC SORT DATA=input-table1 <OUT=sorted-output-table1>;  
  BY <DESCENDING> col-name(s);  
RUN;
```

Tables must have 1 or more variables with same name, type, and order

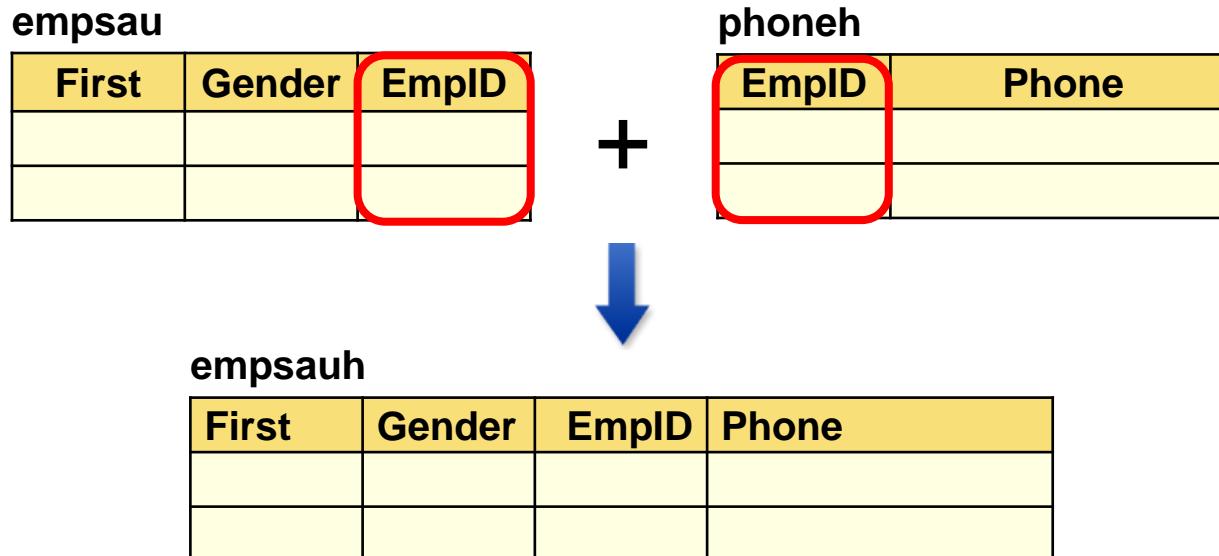
```
PROC SORT DATA=input-table2 <OUT=sorted-output-table2>;  
  BY <DESCENDING> col-name(s);  
RUN;
```

```
DATA combined-data-table;  
  MERGE sorted-output-table1 sorted-output-table2  
    BY <DESCENDING> col-name(s);  
RUN;
```

Uses MERGE instead of SET to combine tables

# Business Scenario

Merge the Australian employee data set with a phone data set to obtain each employee's home phone number. Store the results in a new data set.



# Match-Merging

The *MERGE statement* in a DATA step joins observations from two or more SAS data sets into single observations.

```
data empsauh;
```

```
  merge empsau phoneh;  
  by EmpID;
```

```
run;
```

**MERGE** SAS-data-set1 SAS-data-set2 . . . ;  
**BY** <DESCENDING> BY-variable(s);

A *BY statement* indicates a match-merge and lists the variable or variables to match.

# MERGE and BY Statements



Requirements for match-merging:

- Two or more data sets are listed in the MERGE statement.
- The variables in the BY statement must be common to all data sets.
- The data sets must be sorted by the variables listed in the BY statement.

# One-to-One Merge

One observation in **empsau** matches exactly one observation in **phoneh**.

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phoneh**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1793 |
| 121151 | +61(2)5555-1849 |
| 121152 | +61(2)5555-1665 |



The data sets are sorted by **EmpID**.

# Final Results

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phoneh

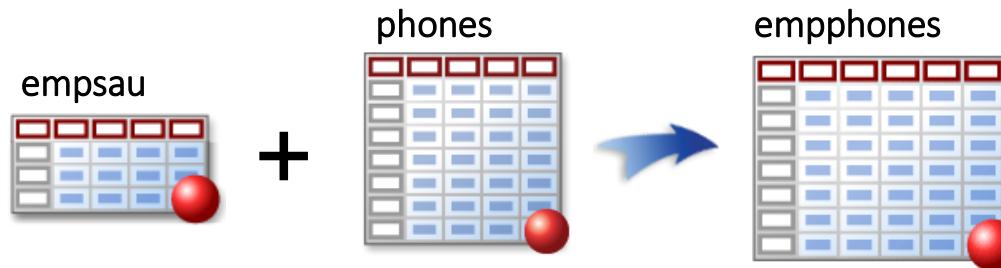
| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1793 |
| 121151 | +61(2)5555-1849 |
| 121152 | +61(2)5555-1665 |

empsauh

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1793 |
| Kylie | F      | 121151 | +61(2)5555-1849 |
| Birin | M      | 121152 | +61(2)5555-1665 |

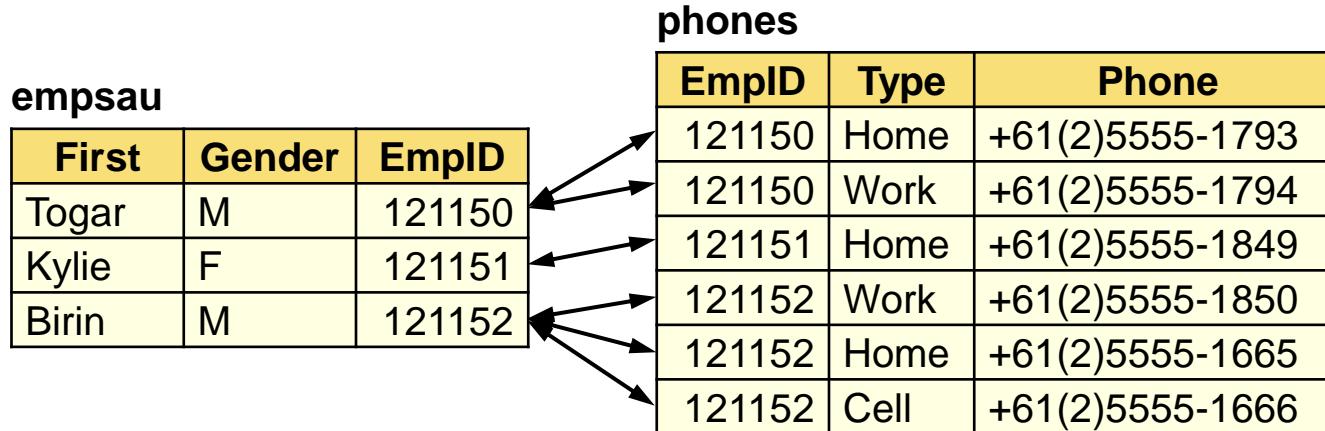
# Business Scenario

Merge the Australian employee information data set with the **phones** data set to obtain the phone numbers for each employee.



# Considerations

In this one-to-many merge, one observation in **empsau** matches one or more observations in **phones**.



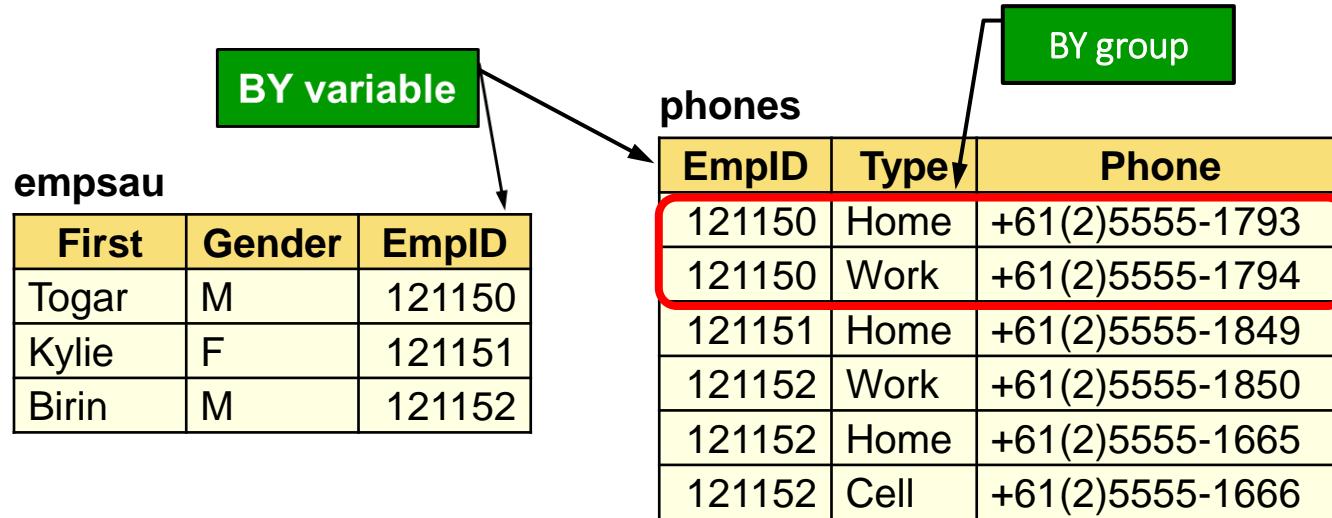
The data sets are sorted by **EmpID**.

# Match-Merging

Merge the two data sets by **EmpID** and create a new data set named **empphones**.

```
data empphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

# Match-Merging



- The common variable is called the *BY variable*.
- The value of the BY variable is the *BY value*.
- A group of observations with the same BY value is a *BY group*.

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphphones;  
merge empsau phones;  
by EmpID;  
run;
```

Initialize PDV

PDV

| First | Gender | EmplID | Type | Phone |
|-------|--------|--------|------|-------|
|       |        | .      |      |       |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

Do the EmplID values match?

Yes

PDV

| First | Gender | EmplID | Type | Phone |
|-------|--------|--------|------|-------|
|       |        | .      |      |       |

# Execution

**empsau**

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phones**

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;  
merge empsau phones;  
by EmpID;  
run;
```

Reads both observations  
into the PDV.

**PDV**

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data empphones;
  merge empsau phones;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data empphones;
  merge empsau phones;
  by EmpID;
run;
```

PDV

Data set variables are not reinitialized.

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplD values match?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Does either **EmplD** match PDV?

Yes

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

**empsau**

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phones**

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
  merge empsau phones;
  by EmpID;
run;
```

Read the matching observation  
into the PDV.

**PDV**

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
merge empsau phones;
by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplID values match?

Yes

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Did **EmplID** change?

Yes

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Read both observations into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data emphphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplD values match?

Yes

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
    merge empsau phones;
    by EmpID;
run;
```

Read both observations into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
merge empsau phones;
by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
    merge empsau phones;  
    by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Did EmplD change?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

phones

| EmpID  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Read the matching observation  
into the PDV.

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphones;  
merge empsau phones;  
by EmplID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphphones;  
    merge empsau phones;  
    by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Did EmplD change?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Reads the matching observation into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
merge empsau phones;  
by EmplID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphones;  
merge empsau phones;  
by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

EOF

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

# Final Results

## empphones

| First | Gender | EmpID  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |
| Togar | M      | 121150 | Work | +61(2)5555-1794 |
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |
| Birin | M      | 121152 | Work | +61(2)5555-1850 |
| Birin | M      | 121152 | Home | +61(2)5555-1665 |
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

NOTE: There were 3 observations read from the data set WORK.EMPSAU.

NOTE: There were 6 observations read from the data set WORK.PHONES.

NOTE: The data set WORK.EMPPHONES has 6 observations and 5 variables.

# Discussion

In a one-to-many merge, does it matter which data set is listed first in the MERGE statement?

- Reverse the order of the data sets on the MERGE statement and submit the data step.
- Observe the results. How are they different?

# Many-to-One Merge

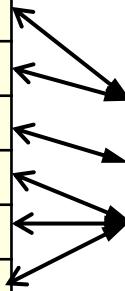
One or more rows in one data set match exactly one row in the other data set.

**phones**

| EmpID  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

**empsau**

| EmpID  | First | Gender |
|--------|-------|--------|
| 121150 | Togar | M      |
| 121151 | Kylie | F      |
| 121512 | Birin | M      |



```
data phones;
  merge phones empsau;
  by EmpID;
run;
```

# Viewing the Output

## PROC PRINT Output

| Obs | EmpID  | Type | Phone           | First | Gender |
|-----|--------|------|-----------------|-------|--------|
| 1   | 121150 | Home | +61(2)5555-1793 | Togar | M      |
| 2   | 121150 | Work | +61(2)5555-1794 | Togar | M      |
| 3   | 121151 | Home | +61(2)5555-1849 | Kylie | F      |
| 4   | 121152 | Work | +61(2)5555-1850 | Birin | M      |
| 5   | 121152 | Home | +61(2)5555-1665 | Birin | M      |
| 6   | 121152 | Cell | +61(2)5555-1666 | Birin | M      |

- The results are the same as the one-to-many merge.
- The order of variables is different.

# Lesson Quiz



10. What are the values for **First.City** and **Last.City** for the third row of the input table given the following information?

```
data StatePopulation;  
  set Population;  
  by State City;  
run;
```

| State | City       | Population |
|-------|------------|------------|
| NC    | Cary       | 162320     |
| NC    | Durham     | 263016     |
| NC    | Greenville | 91495      |
| SC    | Greenville | 67453      |
| SC    | Sumter     | 40723      |

- a. First.City=0 and Last.City=0
- b. First.City=1 and Last.City=0
- c. First.City=0 and Last.City=1
- d. First.City=1 and Last.City=1

10. What are the values for **First.City** and **Last.City** for the third row of the input table given the following information?

```
data StatePopulation;  
  set Population;  
  by State City;  
run;
```

| State | City       | Population |
|-------|------------|------------|
| NC    | Cary       | 162320     |
| NC    | Durham     | 263016     |
| NC    | Greenville | 91495      |
| SC    | Greenville | 67453      |
| SC    | Sumter     | 40723      |

- a. First.City=0 and Last.City=0
- b. First.City=1 and Last.City=0
- c. First.City=0 and Last.City=1
- d. First.City=1 and Last.City=1

# STAT604 SAS Lesson 13

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Combining Data Sets

Basic Match Merging

# Match-Merging: Sorting the Data Sets

```
PROC SORT DATA=input-table1 <OUT=sorted-output-table1>;  
  BY <DESCENDING> col-name(s);  
RUN;
```

Tables must have 1 or more variables with same name, type, and order

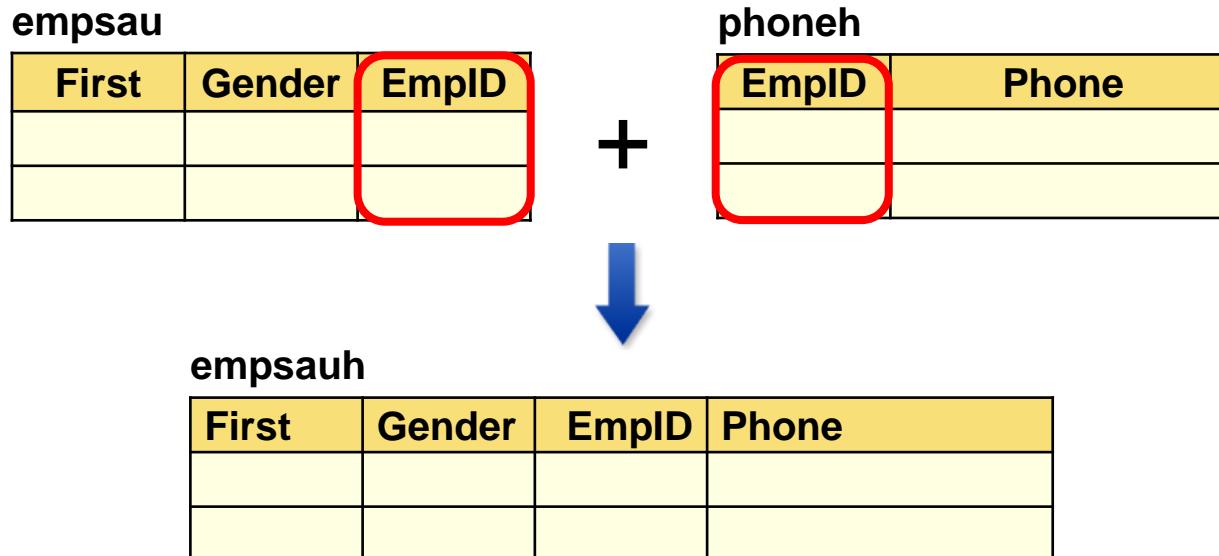
```
PROC SORT DATA=input-table2 <OUT=sorted-output-table2>;  
  BY <DESCENDING> col-name(s);  
RUN;
```

```
DATA combined-data-table;  
  MERGE sorted-output-table1 sorted-output-table2  
    BY <DESCENDING> col-name(s);  
RUN;
```

Uses MERGE instead of SET to combine tables

# Business Scenario

Merge the Australian employee data set with a phone data set to obtain each employee's home phone number. Store the results in a new data set.



# Match-Merging

The *MERGE statement* in a DATA step joins observations from two or more SAS data sets into single observations.

```
data empsauh;
```

```
  merge empsau phoneh;  
  by EmpID;
```

```
run;
```

**MERGE SAS-data-set1 SAS-data-set2 . . . ;  
BY <DESCENDING> BY-variable(s);**

↙↙ w | some , addr  
name type .  
← b. m data sets

A *BY statement* indicates a match-merge and lists the variable or variables to match.



# MERGE and BY Statements

Requirements for match-merging:

- Two or more data sets are listed in the MERGE statement.
- The variables in the BY statement must be common to all data sets.
- The data sets must be sorted by the variables listed in the BY statement.

# One-to-One Merge

One observation in **empsau** matches exactly one observation in **phoneh**.

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phoneh**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1793 |
| 121151 | +61(2)5555-1849 |
| 121152 | +61(2)5555-1665 |



The data sets are sorted by **EmpID**.

# Final Results

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phoneh

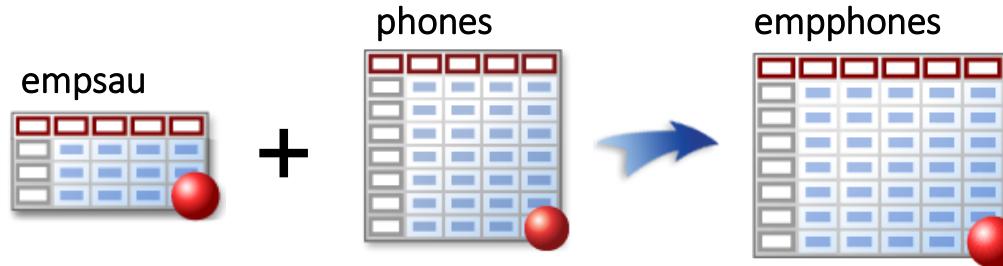
| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1793 |
| 121151 | +61(2)5555-1849 |
| 121152 | +61(2)5555-1665 |

empsauh

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1793 |
| Kylie | F      | 121151 | +61(2)5555-1849 |
| Birin | M      | 121152 | +61(2)5555-1665 |

# Business Scenario

Merge the Australian employee information data set with the **phones** data set to obtain the phone numbers for each employee.



# Considerations

In this **one-to-many merge**, one observation in **empsau** matches one or more observations in **phones**.

| empsau |        |        | phones |      |                 |
|--------|--------|--------|--------|------|-----------------|
| First  | Gender | EmpID  | EmpID  | Type | Phone           |
| Togar  | M      | 121150 | 121150 | Home | +61(2)5555-1793 |
| Kylie  | F      | 121151 | 121151 | Home | +61(2)5555-1849 |
| Birin  | M      | 121152 | 121152 | Work | +61(2)5555-1850 |
|        |        |        | 121152 | Home | +61(2)5555-1665 |
|        |        |        | 121152 | Cell | +61(2)5555-1666 |

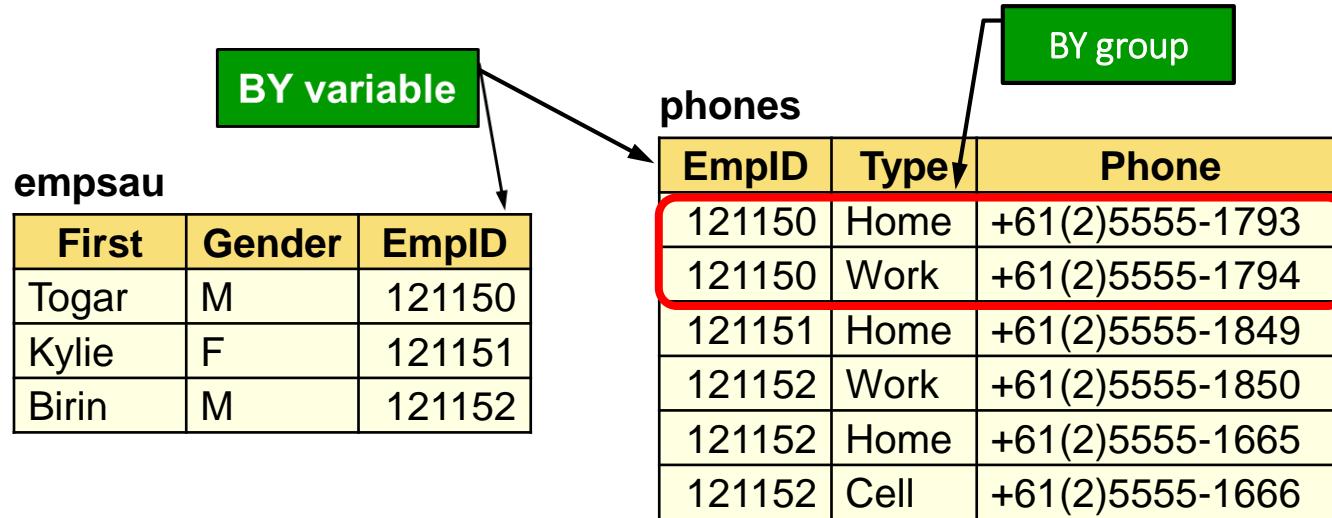
The data sets are sorted by **EmpID**.

# Match-Merging

Merge the two data sets by **EmpID** and create a new data set named **empphones**.

```
data empphones;  
  merge empsau phones;  
  by EmpID;  
run;
```

# Match-Merging



- The common variable is called the *BY variable*.
- The value of the BY variable is the *BY value*.
- A group of observations with the same BY value is a *BY group*.

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphphones;  
merge empsau phones;  
by EmpID;  
run;
```

Initialize PDV

PDV

| First | Gender | EmplD | Type | Phone |
|-------|--------|-------|------|-------|
|       |        | .     |      |       |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;  
merge empsau phones;  
by EmpID;  
run;
```

Do the EmplID values match?

Yes

PDV

| First | Gender | EmplID | Type | Phone |
|-------|--------|--------|------|-------|
|       |        | .      |      |       |

# Execution

**empsau**

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phones**

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;  
merge empsau phones;  
by EmpID;  
run;
```

Reads both observations  
into the PDV.

**PDV**

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data empphones;
  merge empsau phones;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data empphones;
  merge empsau phones;
  by EmpID;
run;
```

PDV

Data set variables are not reinitialized.

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplD values match?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
  merge empsau phones;  
  by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Does either **EmplID** match PDV?

Yes

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |

# Execution

**empsau**

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phones**

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
  merge empsau phones;
  by EmpID;
run;
```

Read the matching observation  
into the PDV.

**PDV**

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
  merge empsau phones;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplID values match?

Yes

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Did **EmplID** change?

Yes

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Work | +61(2)5555-1794 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Read both observations into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data emphphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

```
data empphones;  
merge empsau phones;  
by EmpID;  
  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

Do the EmplD values match?

Yes

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
    merge empsau phones;
    by EmpID;
run;
```

Read both observations into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

```
data emphones;
  merge empsau phones;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
    merge empsau phones;  
    by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Did **EmplD** change?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Work | +61(2)5555-1850 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
    merge empsau phones;  
    by EmpID;  
run;
```

phones

| EmpID  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Read the matching observation  
into the PDV.

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphones;  
merge empsau phones;  
by EmplID;
```

run;

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphphones;  
    merge empsau phones;  
    by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Did EmplD change?

No

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Home | +61(2)5555-1665 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
merge empsau phones;  
by EmpID;  
run;
```

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |



Reads the matching observation into the PDV.

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplID |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empphones;  
merge empsau phones;  
by EmplID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmplID | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

phones

| EmplID | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

# Execution

empsau

| First | Gender | EmplD  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data emphones;  
merge empsau phones;  
by EmplD;  
run;
```

phones

| EmplD  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

EOF

PDV

| First | Gender | EmplD  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

# Final Results

## empphones

| First | Gender | EmpID  | Type | Phone           |
|-------|--------|--------|------|-----------------|
| Togar | M      | 121150 | Home | +61(2)5555-1793 |
| Togar | M      | 121150 | Work | +61(2)5555-1794 |
| Kylie | F      | 121151 | Home | +61(2)5555-1849 |
| Birin | M      | 121152 | Work | +61(2)5555-1850 |
| Birin | M      | 121152 | Home | +61(2)5555-1665 |
| Birin | M      | 121152 | Cell | +61(2)5555-1666 |

NOTE: There were 3 observations read from the data set WORK.EMPSAU.

NOTE: There were 6 observations read from the data set WORK.PHONES.

NOTE: The data set WORK.EMPPHONES has 6 observations and 5 variables.

# Discussion

In a one-to-many merge, does it matter which data set is listed first in the MERGE statement?

- Reverse the order of the data sets on the MERGE statement and submit the data step.
- Observe the results. How are they different?

Only changes the order of  
the columns.

# Many-to-One Merge

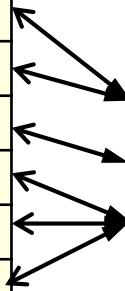
One or more rows in one data set match exactly one row in the other data set.

**phones**

| EmpID  | Type | Phone           |
|--------|------|-----------------|
| 121150 | Home | +61(2)5555-1793 |
| 121150 | Work | +61(2)5555-1794 |
| 121151 | Home | +61(2)5555-1849 |
| 121152 | Work | +61(2)5555-1850 |
| 121152 | Home | +61(2)5555-1665 |
| 121152 | Cell | +61(2)5555-1666 |

**empsau**

| EmpID  | First | Gender |
|--------|-------|--------|
| 121150 | Togar | M      |
| 121151 | Kylie | F      |
| 121512 | Birin | M      |



```
data phones;
  merge phones empsau;
  by EmpID;
run;
```

# Viewing the Output

## PROC PRINT Output

| Obs | EmpID  | Type | Phone           | First | Gender |
|-----|--------|------|-----------------|-------|--------|
| 1   | 121150 | Home | +61(2)5555-1793 | Togar | M      |
| 2   | 121150 | Work | +61(2)5555-1794 | Togar | M      |
| 3   | 121151 | Home | +61(2)5555-1849 | Kylie | F      |
| 4   | 121152 | Work | +61(2)5555-1850 | Birin | M      |
| 5   | 121152 | Home | +61(2)5555-1665 | Birin | M      |
| 6   | 121152 | Cell | +61(2)5555-1666 | Birin | M      |

- The results are the same as the one-to-many merge.
- The order of variables is different.

# Combining Data Sets

Merging Data Sets with Non-Matches

# Business Scenario

A manager in Australia requested  
an inventory of company phone numbers.



# Merge with Nonmatches

There are observations in **empsau** that do not have a match in **phonec**, and some in **phonec** that do not match any observations in **empsau**.

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phonec**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

The data sets are sorted by **EmpID**.

# Match-Merging

Merge **empsau** and **phonec** by **EmpID** to create a new data set named **empsauc**.

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Initialize PDV

PDV

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Do the EmpID values match?

Yes

PDV

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Reads both observations  
into the PDV.

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
  merge empsau phonec;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |

# Execution

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phonec**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Do the EmpID values match?

No

**PDV**

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Does either **EmpID** match  
PDV?

No

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reinitialize PDV

reinitialize  
blk grt dat  
match

PDV

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Which **EmpID** value sequentially comes first?

121151

PDV

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |



phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Reads that observation  
into the PDV.

PDV

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
  merge empsau phonec;  
  by EmpID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Do the EmpID values match?

Yes

PDV

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Does EmpID match PDV?

No

→ Not an issue already  
→ 'K' ~ all ~ matched

PDV

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

# Execution

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phonec**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reinitialize PDV

**PDV**

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

**phonec**

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

Reads both observations  
into the PDV.

**PDV**

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Birin | M      | 121152 | +61(2)5555-1667 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
  merge empsau phonec;
  by EmpID;
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Birin | M      | 121152 | +61(2)5555-1667 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

```
data empsauc;  
    merge empsau phonec;  
    by EmpID;  
run;
```

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

Does EmpID match PDV?

No

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Birin | M      | 121152 | +61(2)5555-1667 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

EOF

Reinitialize PDV

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

| First | Gender | EmpID | Phone |
|-------|--------|-------|-------|
|       |        | .     |       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

Reads the observation  
into the PDV.

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
|       |        | 121153 | +61(2)5555-1348 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;  
  merge empsau phonec;  
  by EmpID;  
run;
```

Implicit OUTPUT;  
Implicit RETURN;

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
|       |        | 121153 | +61(2)5555-1348 |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

phonec

| EmpID  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

EOF

```
data empsauc;
    merge empsau phonec;
    by EmpID;
run;
```

PDV

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
|       |        | 121153 | +61(2)5555-1348 |

# Final Results

**empsauc**

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |
| Kylie | F      | 121151 |                 |
| Birin | M      | 121152 | +61(2)5555-1667 |
|       |        | 121153 | +61(2)5555-1348 |

The final results include both matches and nonmatches.

# Short Answer Poll

Consider the data set **empsauc** created by the program in the previous example. Which input data sets contributed information to the last observation?

- a. empsau
- b. phonec
- c. both empsau and phonec
- d. There is insufficient information.

empsauc

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |
| Kylie | F      | 121151 |                 |
| Birin | M      | 121152 | +61(2)5555-1667 |
|       |        | 121153 | +61(2)5555-1348 |



# Short Answer Poll – Correct Answer

Consider the data set **empsauc** created by the program in the previous example. Which input data sets contributed information to the last observation?

- a. empsau
- b.** phonec
- c. both empsau and phonec
- d. There is insufficient information.

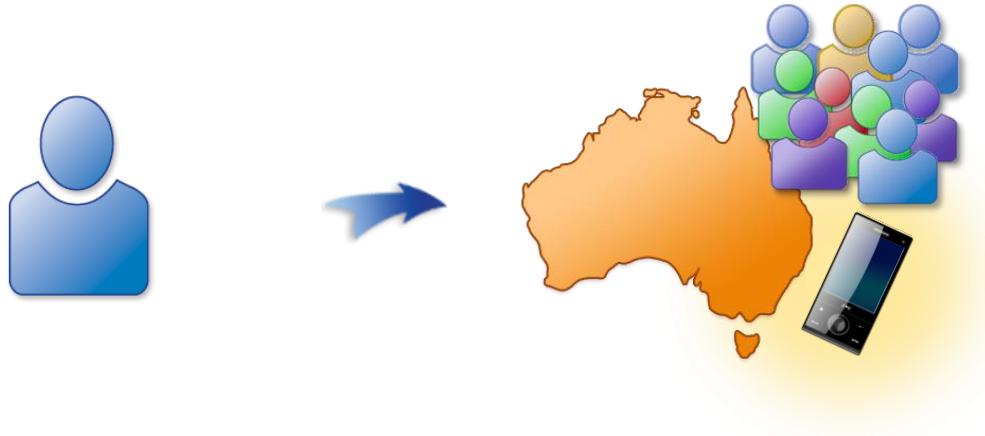
**empsauc**

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |
| Kylie | F      | 121151 |                 |
| Birin | M      | 121152 | +61(2)5555-1667 |
|       |        | 121153 | +61(2)5555-1348 |



# Business Scenario

The manager has now requested three phone inventory reports.



- employees with company phones
- employees without company phones
- phones with an invalid employee ID

# IN= Data Set Option

*note. after data set options were talked about this is fr.*

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

```
MERGE SAS-data-set (IN=variable) ...
```

- Keep/Drop
- Obs/firstobs
- Rename
- IN

*variable* is a temporary numeric variable that has two possible values:

|   |   |
|---|---|
| 0 | Indicates that the data set did <b>not</b> contribute to the current observation. |
| 1 | Indicates that the data set <b>did</b> contribute to the current observation.     |

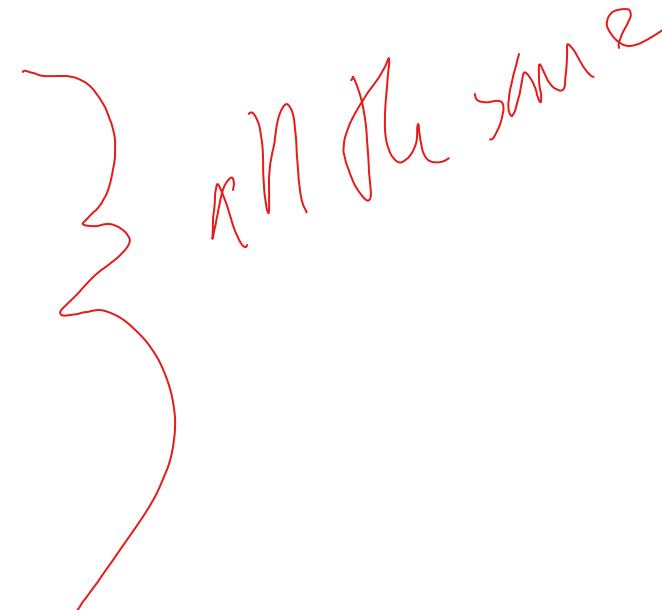
# IN= Data Set Option

MERGE statement examples:

```
merge empsau(in=Emps)  
      phonec(in=Cell);
```

```
merge empsau(in=E)  
      phonec(in=P);
```

```
merge empsau(in=AU)  
      phonec;
```



# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmplD  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau(in=Emps)
              phonec(in=Cell);
    by EmpID;
run;
```

match

PDV

| First | Gender | EmplD  | D>Emps | Phone           | D> Cell |
|-------|--------|--------|--------|-----------------|---------|
| Togar | M      | 121150 | 1      | +61(2)5555-1795 | 1       |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmplD  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau(in=Emps)
              phonec(in=Cell);
    by EmpID;
run;
```

nonmatch

PDV

| First | Gender | EmplD  | D→Emps | Phone | D→Cell |
|-------|--------|--------|--------|-------|--------|
| Kylie | F      | 121151 | 1      |       | 0      |

# Execution

empsau

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

phonec

| EmplD  | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau(in=Emps)
              phonec(in=Cell);
    by EmpID;
run;
```

match

PDV

| First | Gender | EmplD  | D>Emps | Phone           | D>Cell |
|-------|--------|--------|--------|-----------------|--------|
| Birin | M      | 121152 | 1      | +61(2)5555-1667 | 1      |

# Short Answer Poll

What are the values of **Emps** and **Cell**?

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

**phonec**

| EmplID | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau(in=Emps)
              phonec(in=Cell);
    by EmpID;
run;
```

**PDV**

| First | Gender | EmpID  | D→Emps | Phone           | D→ Cell |
|-------|--------|--------|--------|-----------------|---------|
|       |        | 121153 |        | +61(2)5555-1348 |         |

# Short Answer Poll – Correct Answer

What are the values of **Emps** and **Cell**?

**empsau**

| First | Gender | EmpID  |
|-------|--------|--------|
| Togar | M      | 121150 |
| Kylie | F      | 121151 |
| Birin | M      | 121152 |

EOF

**phonec**

| EmplID | Phone           |
|--------|-----------------|
| 121150 | +61(2)5555-1795 |
| 121152 | +61(2)5555-1667 |
| 121153 | +61(2)5555-1348 |

```
data empsauc;
    merge empsau(in=Emps)
              phonec(in=Cell);
    by EmpID;
run;
```

nonmatch

**PDV**

| First | Gender | EmplID | Emps | Phone           | Cell |
|-------|--------|--------|------|-----------------|------|
|       |        | 121153 | 0    | +61(2)5555-1348 | 1    |

# PDV Results

PDV

| First | Gender | EmplID | D<br>Emps | Phone           | D | Cell |
|-------|--------|--------|-----------|-----------------|---|------|
| Togar | M      | 121150 | 1         | +61(2)5555-1795 |   | 1    |
| Kylie | F      | 121151 | 1         |                 |   | 0    |
| Birin | M      | 121152 | 1         | +61(2)5555-1667 |   | 1    |
|       |        | 121153 | 0         | +61(2)5555-1348 |   | 1    |

The variables created with the IN= data set option are available only during DATA step execution.

- They are not written to the SAS data set.
- Their value can be tested using conditional logic.

# Matches Only

Add a subsetting IF statement to select the employees that have company phones.

```
data empsauc;
    merge empsau(in=Emps)
        phonec(in=Cell);
    by EmpID;
    if Emps=1 and Cell=1;
run;
```

empsauc

| First | Gender | EmplID | Phone           |
|-------|--------|--------|-----------------|
| Togar | M      | 121150 | +61(2)5555-1795 |
| Birin | M      | 121152 | +61(2)5555-1667 |

# Nonmatches from empsau

Select the employees that do not have company phones.

```
data empsauc;
  merge empsau(in=Emps)
        phonec(in=Cell);
  by EmpID;
  if Emps=1 and Cell=0;
run;
```

empsauc

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

# Nonmatches from phonec

Select the phones associated with an invalid employee ID.

```
data empsauc;
  merge empsau(in=Emps)
        phonec(in=Cell);
  by EmpID;
  if Emps=0 and Cell=1;
run;
```

empsauc

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
|       |        | 121153 | +61(2)5555-1348 |

# All Nonmatches

```
data empsauc;  
  merge empsau(in=Emps)  
        phonec(in=Cell);  
  by EmpID;  
  if Emps=0 or Cell=0;  
run;
```

empsauc

| First | Gender | EmpID  | Phone           |
|-------|--------|--------|-----------------|
| Kylie | F      | 121151 |                 |
|       |        | 121153 | +61(2)5555-1348 |



Use the OR operator, not the AND operator.



## Discussion

How could we make the creation of the three data sets more efficient?

# Outputting to Multiple Data Sets

The DATA statement can specify multiple output data sets.

```
data EmpsAUC EmpsOnly PhoneOnly;
  merge EmpsAU(in=Emps) PhoneC(in=Cell);
  by EmpID;
  if Emps=1 and Cell=1
    then output EmpsAUC;
  else if Emps=1 and Cell=0
    then output EmpsOnly;
  else if Emps=0 and Cell=1
    then output PhoneOnly;
run;
```

# Outputting to Multiple Data Sets

An OUTPUT statement can be used in a conditional statement to write the current observation to a specific data set that is listed in the DATA statement.

```
data EmpsAUC EmpsOnly PhoneOnly;
  merge EmpsAU(in=Emps) PhoneC(in=Cell);
  by EmpID;
  if Emps=1 and Cell=1
    then output EmpsAUC;
  else if Emps=1 and Cell=0
    then output EmpsOnly;
  else if Emps=0 and Cell=1
    then output PhoneOnly;
run;
```

# Outputting to Multiple Data Sets

## EmpsAUC

| First | Gender | EmpID  | Phone             |
|-------|--------|--------|-------------------|
| Togar | M      | 121150 | +61 (2) 5555-1795 |
| Birin | M      | 121152 | +61 (2) 5555-1667 |

## EmpsOnly

| First | Gender | EmpID  | Phone |
|-------|--------|--------|-------|
| Kylie | F      | 121151 |       |

## PhoneOnly

| First | Gender | EmpID  | Phone             |
|-------|--------|--------|-------------------|
|       |        | 121153 | +61 (2) 5555-1348 |

# Alternate Syntax

When checking a variable for a value of *1* or *0* as in the previous scenario, you can use the following syntax:

| Instead of            | You can use              |
|-----------------------|--------------------------|
| if Emps=1 and Cell=1; | if Emps and Cell;        |
| if Emps=1 and Cell=0; | if Emps and not Cell;    |
| if Emps=0 and Cell=1; | if not Emps and Cell;    |
| if Emps=0 or Cell=0;  | If not Emps or not Cell; |

# Alternate Syntax

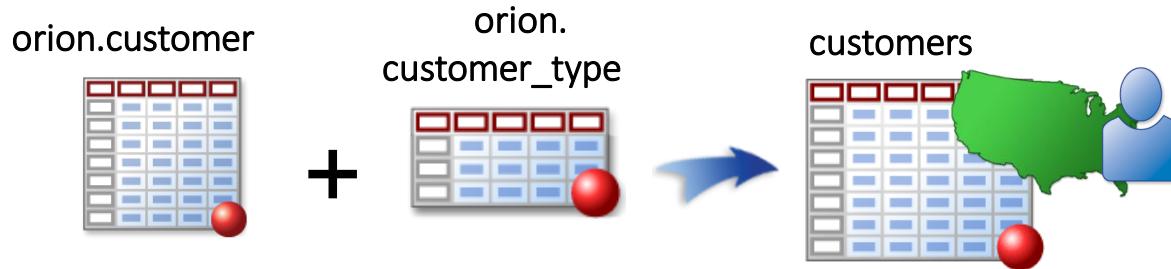
Both programs create a report of employees without cell phones.

```
data empsphone;
  merge empsact(in=inEmps)
        phoneact(in=inCell);
  by EmpID;
  if inEmps=1 and inCell=0;
run;
```

```
data empsphone;
  merge empsact(in=inEmps)
        phoneact(in=inCell);
  by EmpID;
  if inEmps and not inCell;
run;
```

# Business Scenario

Merge customer information with customer type data to obtain a customer description. The new data set should include only US customers.



# Considerations

The **orion.customer** data set is not sorted by **Customer\_Type\_ID**, the common variable. The subsetting variable, **Country**, is defined in only one data set.

**orion.customer**

| Customer_ID | Country | Customer_Name | ... | Birth_Date | Customer_Type_ID |
|-------------|---------|---------------|-----|------------|------------------|
|             |         |               |     |            |                  |

**orion.customer\_type**

| Customer_Group | Customer_Group_ID | Customer_Type | Customer_Type_ID |
|----------------|-------------------|---------------|------------------|
|                |                   |               |                  |

# Short Answer Poll

What change is needed to correct the error?

```
proc sort data=orion.customer
           out=cust_by_type;
   by Customer_Type_ID;
run;

data customers;
   merge cust_by_type orion.customer_type;
   by Customer_Type_ID;
   where Country='US';
run;
```

# Short Answer Poll – Correct Answer

What change is needed to correct the error?

```
397 proc sort data=orion.customer
398         out=cust_by_type;
399     by Customer_Type_ID;
400 run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUST_BY_TYPE has 77 observations and 12 variables.

401
402 data customers;
403     merge cust_by_type orion.customer_type;
404     by Customer_Type_ID;
405     where Country='US';
ERROR: Variable Country is not on file ORION.CUSTOMER_TYPE.
406 run;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.CUSTOMERS may be incomplete. When this step
was stopped there were 0 observations and 15 variables.
```

Country is not defined in both data sets. Replace the WHERE statement with a subsetting IF statement.

# Subsetting IF

Use a subsetting IF statement when the subsetting variable is not in all data sets that are named in the MERGE statement.

```
proc sort data=orion.customer
           out=cust_by_type;
   by Customer_Type_ID;
run;

data customers;
  merge cust_by_type  orion.customer_type;
  by Customer_Type_ID;
  if Country='US';
run;
```

# Viewing the Output

## Partial SAS Log

```
407 proc sort data=orion.customer
408         out=cust_by_type;
409     by Customer_Type_ID;
410 run;
NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CUST_BY_TYPE has 77 observations and 12 variables.

411
412 data customers;
413 merge cust_by_type orion.customer_type;
414     by Customer_Type_ID;
415     if Country='US';
416 run;

NOTE: There were 77 observations read from the data set WORK.CUST_BY_TYPE.
NOTE: There were 8 observations read from the data set ORION.CUSTOMER_TYPE.
NOTE: The data set WORK.CUSTOMERS has 28 observations and 15 variables.
```

# WHERE versus Subsetting IF Statement



| Step and Usage                           | WHERE | IF  |
|--|-------|-----|
| <b>PROC step</b>                         | Yes   | No  |
| <b>DATA step (source of variable)</b>    |       |     |
| SET statement                            | Yes   | Yes |
| assignment statement                     | No    | Yes |
| INPUT statement                          | No    | Yes |
| SET/MERGE statement (multiple data sets) |       |     |
| Variable in ALL data sets                | Yes   | Yes |
| Variable not in ALL data sets            | No    | Yes |

# Combining Data Sets

Merging Data Sets with Many-to-Many Matches

# Many-to-Many Merge

Merge **EmpsAUUS** and **PhoneO** by **Country** to create a new data set named **EmpsOfc**.

**EmpsAUUS**

| First  | Gender | Country |
|--------|--------|---------|
| Togar  | M      | AU      |
| Kylie  | F      | AU      |
| Stacey | F      | US      |
| Gloria | F      | US      |
| James  | M      | US      |

**PhoneO**

| Country | Phone             |
|---------|-------------------|
| AU      | +61 (2) 5555-1500 |
| AU      | +61 (2) 5555-1600 |
| AU      | +61 (2) 5555-1700 |
| US      | +1 (305) 555-1500 |
| US      | +1 (305) 555-1600 |

```
data EmpsOfc;
  merge EmpsAUUS PhoneO;
  by Country;
run;
```

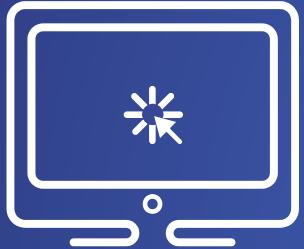
The data sets are sorted by  
**Country**.

# Many-to-Many Merge

DATA Step Results:

**EmpsOfc**

| First  | Gender | Country | Phone             |
|--------|--------|---------|-------------------|
| Togar  | M      | AU      | +61 (2) 5555-1500 |
| Kylie  | F      | AU      | +61 (2) 5555-1600 |
| Kylie  | F      | AU      | +61 (2) 5555-1700 |
| Stacey | F      | US      | +1 (305) 555-1500 |
| Gloria | F      | US      | +1 (305) 555-1600 |
| James  | M      | US      | +1 (305) 555-1600 |



# Merging Many-to-Many

This demonstration compares a many-to-many merge with a SQL join.

# Many-to-Many Merge

The SQL procedure creates different results than the DATA step for a many-to-many merge.

**EmpsAUUS**

| First  | Gender | Country |
|--------|--------|---------|
| Togar  | M      | AU      |
| Kylie  | F      | AU      |
| Stacey | F      | US      |
| Gloria | F      | US      |
| James  | M      | US      |

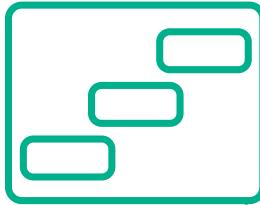
**PhoneO**

| Country | Phone             |
|---------|-------------------|
| AU      | +61 (2) 5555-1500 |
| AU      | +61 (2) 5555-1600 |
| AU      | +61 (2) 5555-1700 |
| US      | +1 (305) 555-1500 |
| US      | +1 (305) 555-1600 |

```
proc sql;
  create table EmpsOfc as
    select First, Gender, PhoneO.Country, Phone
    from EmpsAUUS, PhoneO
   where EmpsAUUS.Country=PhoneO.Country;
```

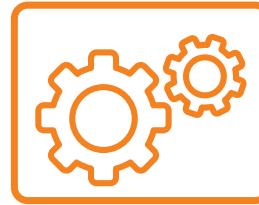
# DATA Step Merge and PROC SQL Join

## DATA step merge



- requires sorted input data
- efficient, sequential processing
- can create multiple tables for matches and nonmatches in one step
- provides additional complex data processing syntax

## PROC SQL join



- does not require sorted data
- matching columns do not need the same name
- easy to define complex matching criteria between multiple tables in a single query
- can be used to create a Cartesian product for many-to-many joins



# Combining Data Sets

Using Data Manipulation Techniques with Match-Merging

# Multiple Data Sets without a Common Variable

The following report needs to be created using data from three data sets.

Partial PROC PRINT Output

| Customer_Name       | Quantity | Total_Retail_Price | Product_Name                      | Supplier |
|---------------------|----------|--------------------|-----------------------------------|----------|
| Kyndal Hooks        | 2        | \$69.40            | Kids Sweat Round Neck, Large Logo | US 3298  |
| Kyndal Hooks        | 1        | \$14.30            | Fleece Cuff Pant Kid's            | US 1303  |
| Dericka Pockran     | 3        | \$37.80            | Children's Mitten                 | US 772   |
| Wendell Summersby   | 1        | \$39.40            | Bozeman Rain & Storm Set          | US 772   |
| Sandrina Stephano   | 1        | \$52.50            | Teen Profleece w/Zipper           | US 772   |
| Wendell Summersby   | 1        | \$50.40            | Butch T-Shirt with V-Neck         | ES 4742  |
| Karen Ballinger     | 2        | \$134.00           | Children's Knit Sweater           | ES 4742  |
| Wendell Summersby   | 2        | \$134.00           | Children's Knit Sweater           | ES 4742  |
| Patricia Bertolozzi | 1        | \$23.50            | Strap Pants BBQ                   | ES 798   |
| Kyndal Hooks        | 4        | \$56.80            | Osprey France Nylon Shorts        | US 3664  |
| Karen Ballinger     | 3        | \$60.90            | Osprey Girl's Tights              | US 3664  |
| Karen Ballinger     | 2        | \$60.60            | Logo Coord. Children's Sweatshirt | US 2963  |
| David Black         | 1        | \$117.60           | Big Guy Men's Clima Fit Jacket    | US 1303  |

orion.customer

work.order\_fact

orion.product\_dim

# Quiz

Any number of data sets can be merged in a single DATA step. However, the data sets must have a common variable and be sorted by that variable.

What is the common variable in the following data sets?

`orion.customer`

|                                 |
|---------------------------------|
| <code>Customer_ID</code>        |
| <code>Country</code>            |
| <code>Gender</code>             |
| <code>Personal_ID</code>        |
| <code>Customer_Name</code>      |
| <code>Customer_FirstName</code> |
| <code>Customer_LastName</code>  |
| <code>Birth_Date</code>         |
| <code>Customer_Address</code>   |
| ...                             |

`work.order_fact`

|                            |
|----------------------------|
| <code>Customer_ID</code>   |
| <code>Employee_ID</code>   |
| <code>Street_ID</code>     |
| <code>Order_Date</code>    |
| <code>Delivery_Date</code> |
| <code>Order_ID</code>      |
| <code>Order_Type</code>    |
| <code>Product_ID</code>    |
| <code>Quantity</code>      |
| ...                        |

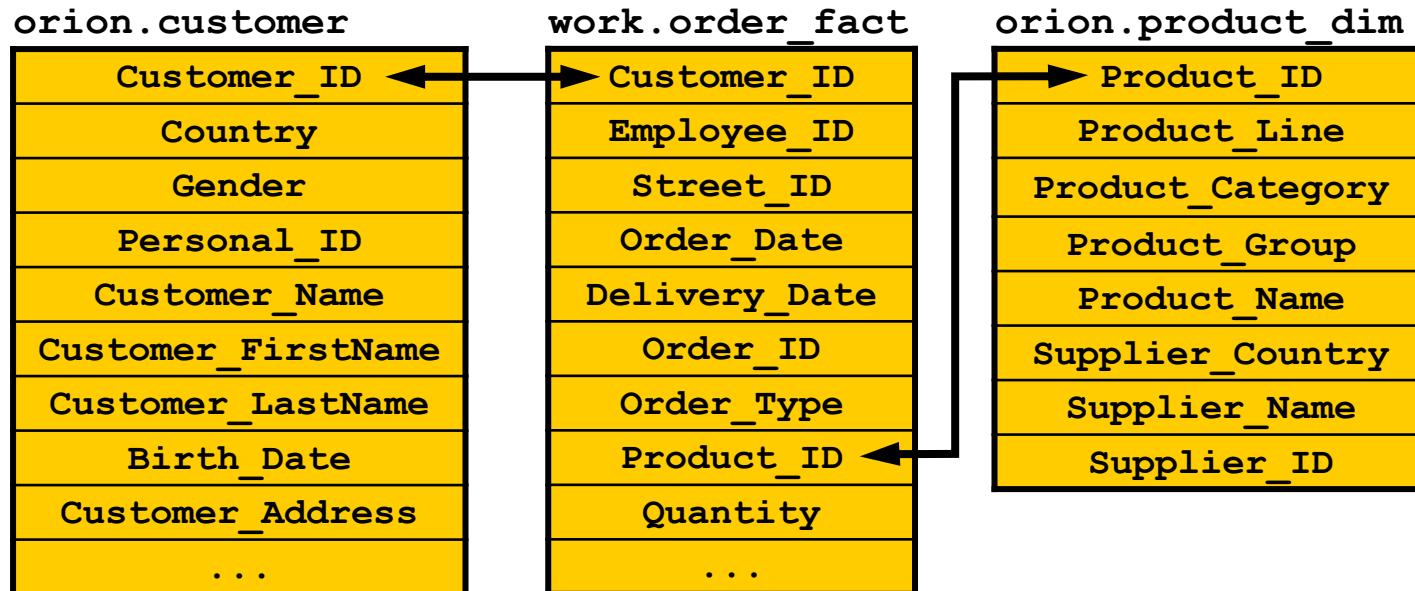
`orion.product_dim`

|                               |
|-------------------------------|
| <code>Product_ID</code>       |
| <code>Product_Line</code>     |
| <code>Product_Category</code> |
| <code>Product_Group</code>    |
| <code>Product_Name</code>     |
| <code>Supplier_Country</code> |
| <code>Supplier_Name</code>    |
| <code>Supplier_ID</code>      |

# Quiz – Correct Answer

What is the common variable in the following data sets?

 None. These data sets do not share one common variable. Therefore, they cannot be combined in a single DATA step.



# Match-Merge without a Common Variable

If data sets do not share a common variable, combine them by using a series of merges in separate DATA steps. As usual, the data sets must be sorted by the appropriate BY variable.



Step 1: Merge **orion.customer** and  
**work.order\_fact** by **Customer\_ID**.

Step 2: Merge the results of Step1 and  
**orion.product\_dim** by  
**Product\_ID**.

# Without a Common Variable – Step 1

Merge `orion.customer` and  
`work.order_fact` by `Customer_ID`.

```
proc sort data=orion.order_fact
           out=work.order_fact;
  by Customer_ID;
  where year(Order_Date)=2007;
run;

data CustOrd;
  merge orion.customer(in=cust)
        work.order_fact(in=order);
  by Customer_ID;
  if cust=1 and order=1;
  keep Customer_ID Customer_Name Quantity
        Total_Retail_Price Product_ID;
run;
```

orion.customer is in  
order by Customer\_ID

# Without a Common Variable – Step 2

Merge the results of Step 1, **CustOrd**, with **orion.product\_dim** by **Product\_ID**.

```
proc sort data=CustOrd;
  by Product_ID;
run;

data CustOrdProd;
  merge CustOrd(in=ord)
        orion.product_dim(in=prod) ;
  by Product_ID;
  if ord=1 and prod=1;
  Supplier=catx(' ',Supplier_Country,Supplier_ID);
  keep Customer_Name Quantity
      Total_Retail_Price Product_Name Supplier;
run;
```

Product\_dim is in  
order by Product\_ID

# Altering Variable Names

With match-merging, two situations might require altering variable names:

- The BY variables have different names in the input data sets being merged.
- The data sets being merged have identically named variables that must both be kept in the merged output.



In both cases, the RENAME= data set option can be used to alter the variable names to get the desired results.

# Business Scenario – Create Gift List

The Excel workbook **BonusGift.xlsx** contains a list of suppliers that want to send gifts to customers who purchased more than a specified minimum quantity of a product.

Use **work.CustOrdProd** and **BonusGift.xlsx** to determine the customers that will be sent gifts.

**work.CustOrdProd**

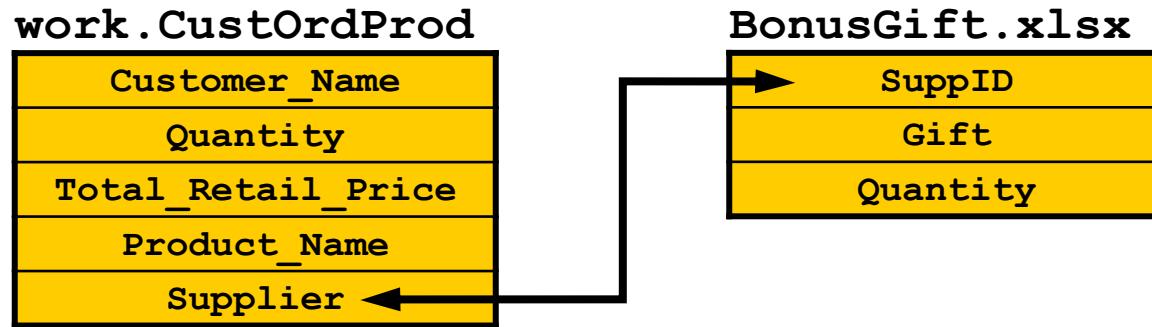
|                    |
|--------------------|
| Customer_Name      |
| Quantity           |
| Total_Retail_Price |
| Product_Name       |
| Supplier           |

**BonusGift.xlsx**

|          |
|----------|
| SuppID   |
| Gift     |
| Quantity |

# Business Scenario – Details

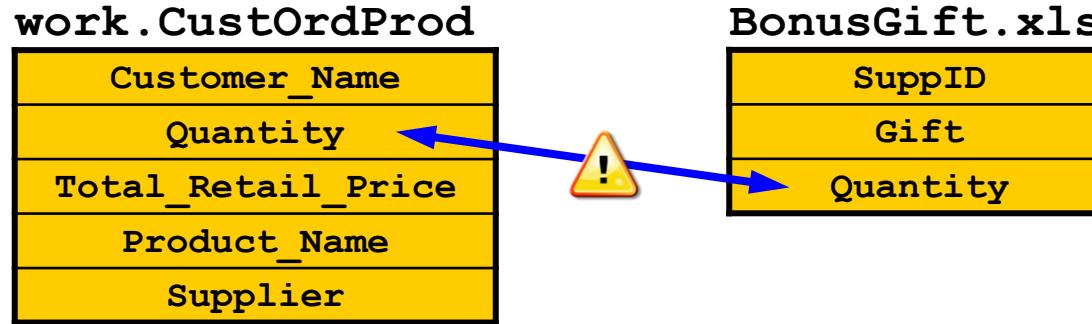
The data sets **work.CustOrdProd** and **BonusGift.xlsx** must be merged on values that are in two differently named variables.



The variables must have the same name for the match-merge to work correctly.

# Business Scenario – Details

You want to keep merged observations where the value of **Quantity** in **work.CustOrdProd** is more than the value of **Quantity** in **BonusGift.xls**.



The variables must have different names so that you can use a subsetting IF statement to compare them.

# Create Gift List – Solution

Use the IN= option and a condition in the subsetting IF statement to keep only the matches.

```
libname bonus xlsx 'C:\Sample Data\BonusGift.xlsx';

data CustOrdProdGift;
  merge CustOrdProd(in=c)
        bonus.Supplier(in=s
                         rename=(SuppID=Supplier
                                 Quantity=Minimum));
  by Supplier;
  if c=1 and s=1 and Quantity > Minimum;
run;

libname bonus clear;
```

# Create Gift List – Solution

Use the RENAME= data set option to ensure that the BY variable has the same name to use for merging.

```
libname bonus xlsx 'C:\Sample Data\BonusGift.xlsx';

data CustOrdProdGift;
  merge CustOrdProd(in=c)
        bonus.Supplier(in=s
                         rename=(SuppID=Supplier
                                 Quantity=Minimum));
  by Supplier;
  if c=1 and s=1 and Quantity > Minimum;
run;

libname bonus clear;
```

# Create Gift List – Solution

Change the name of the **Quantity** variable from the Excel worksheet so that it can be used in a subsetting IF.

```
libname bonus xlsx 'C:\Sample Data\BonusGift.xlsx';

data CustOrdProdGift;
    merge CustOrdProd(in=c)
          bonus.Supplier(in=s
                           rename=(SuppID=Supplier
                                   Quantity=Minimum));
    by Supplier;
    if c=1 and s=1 and Quantity > Minimum;
run;

libname bo
```

Quantity value from the CustOrdProd data set

Renamed Quantity value from the Supplier data set

# Create Gift List – Solution

Fifty-two gifts will be sent to customers.

Partial SAS log

```
208  
209  data CustOrdProdGift;  
210    merge CustOrdProd(in=c)  
211      bonus.Supplier(in=s  
212        rename=(SuppID=Supplier  
213          Quantity=Minimum));  
214    by Supplier;  
215    if c=1 and s=1 and Quantity > Minimum;  
216  run;  
  
NOTE: There were 148 observations read from the data set WORK.CUSTORDPROD.  
NOTE: There were 18 observations read from the data set BONUS.Supplier.  
NOTE: The data set WORK.CUSTORDPRODGIFT has 52 observations and 7 variables.  
NOTE: DATA statement used (Total process time):  
      real time          0.04 seconds  
      cpu time          0.03 seconds
```

# Create Gift List – Output

Sort the data set by customer name prior to printing the list of customers and the gifts that they should receive.

```
proc sort data=CustOrdProdGift;  
  by Customer_Name;  
run;  
  
proc print data=CustOrdProdGift;  
  var Customer_Name Gift;  
run;
```

# Create Gift List – Output

The output below shows the list of customers and gifts.

Partial PROC PRINT output

| Customer_Name    | Gift       |
|------------------|------------|
| Alvan Goheen     | Travel Mug |
| Angel Borwick    | Belt Pouch |
| Cynthia Martinez | Travel Set |
| Cynthia Martinez | Gift Card  |
| Cynthia Martinez | Travel Mug |
| Cynthia Mccluney | Tote Bag   |
| Cynthia Mccluney | Tote Bag   |
| Cynthia Mccluney | Gift Card  |
| David Black      | Backpack   |
| Dericka Pockran  | Coupon     |
| Dericka Pockran  | Travel Mug |
| Dericka Pockran  | Travel Mug |

# Merging Multiple Data Sets

The DATA statement can merge multiple input data sets as long as they all have a common variable.

**payroll06**

| Obs | Employee_ID | Employee_Gender | Salary | Birth_Date | Employee_Hire_Date | Employee_Term_Date | Marital_Status | Dependents |
|-----|-------------|-----------------|--------|------------|--------------------|--------------------|----------------|------------|
| 1   | 120101      | M               | 163040 | 6074       | 01JUL2003          | .                  | S              | 0          |
| 2   | 120102      | M               | 108255 | 3510       | 01JUN1989          | .                  | O              | 2          |
| 3   | 120103      | M               | 87975  | -3996      | 01JAN1974          | .                  | M              | 1          |
| 4   | 120104      | F               | 46230  | -2061      | 01JAN1981          | .                  | M              | 1          |
| 5   | 120105      | F               | 27110  | 5468       | 01MAY1999          | .                  | S              | 0          |

**payroll07**

| Obs | Employee_ID | Employee_Gender | Salary | Birth_Date | Employee_Hire_Date | Employee_Term_Date | Marital_Status | Dependents |
|-----|-------------|-----------------|--------|------------|--------------------|--------------------|----------------|------------|
| 1   | 120101      | M               | 167931 | 6074       | 01JUL2003          | .                  | S              | 0          |
| 2   | 120102      | M               | 111503 | 3510       | 01JUN1989          | .                  | O              | 2          |
| 3   | 120103      | M               | 90614  | -3996      | 01JAN1974          | .                  | M              | 1          |
| 4   | 120104      | F               | 47617  | -2061      | 01JAN1981          | .                  | M              | 1          |
| 5   | 120105      | F               | 27923  | 5468       | 01MAY1999          | .                  | S              | 0          |

**payroll08**

| Obs | Employee_ID | Employee_Gender | Salary | Birth_Date | Employee_Hire_Date | Employee_Term_Date | Marital_Status | Dependents |
|-----|-------------|-----------------|--------|------------|--------------------|--------------------|----------------|------------|
| 1   | 120101      | M               | 172969 | 6074       | 01JUL2003          | .                  | .              | .          |
| 2   | 120102      | M               | 114848 | 3510       | 01JUN1989          | .                  | .              | .          |
| 3   | 120103      | M               | 93332  | -3996      | 01JAN1974          | .                  | .              | .          |
| 4   | 120104      | F               | 49046  | -2061      | 01JAN1981          | .                  | .              | .          |
| 5   | 120105      | F               | 28761  | 5468       | 01MAY1999          | .                  | S              | 0          |

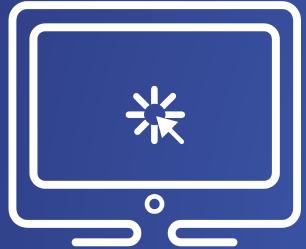
# Merging Multiple Data Sets

Note: The SAS log is extremely important in troubleshooting merges.

```
data payroll_hist;
  merge mylib.payroll06
        mylib.payroll07
        mylib.payroll08(drop=marital_status) ;
  by employee_id;
run;
```



- Common variables are overwritten from the right.
- Use UPDATE instead of MERGE to prevent missing data from overwriting existing data.



# Merging Multiple Data Sets

This demonstration shows how data can be overwritten in a merge and how the log can provide useful troubleshooting information.

# Lesson Quiz



1. Which of the following statements is true about merging SAS data sets by using the DATA step?
  - a. Merging combines observations from two or more data sets into a single observation in a new data set.
  - b. SAS can merge data sets based on the position of observations in the original data set or by the values of one or more common variables.
  - c. Match-merging is merging by values of one or more common variables.
  - d. To match-merge data sets, all input data sets must be sorted or indexed on the BY variable or variables.
  - e. all of the above

1. Which of the following statements is true about merging SAS data sets by using the DATA step?
  - a. Merging combines observations from two or more data sets into a single observation in a new data set.
  - b. SAS can merge data sets based on the position of observations in the original data set or by the values of one or more common variables.
  - c. Match-merging is merging by values of one or more common variables.
  - d. To match-merge data sets, all input data sets must be sorted or indexed on the BY variable or variables.
  - e. all of the above

2. Which of the following programs concatenates the data sets **sales** and **products**, in that order?

a.

```
data newsales;
  set products sales;
run;
```

b.

```
data newsales;
  set sales products;
run;
```

c.

```
data newsales;
  set sales;
  set products;
run;
```

2. Which of the following programs concatenates the data sets **sales** and **products**, in that order?

a.

```
data newsales;
  set products sales;
run;
```

b.

```
data newsales;
  set sales products;
run;
```

c.

```
data newsales;
  set sales;
  set products;
run;
```

3. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
   merge managers staff;
   by EmpID;
run;
```

- a. all of the observations from **managers**, and only those observations from **staff** with matching values for **EmpID**
- b. all of the observations from **staff**, and only those observations from **managers** with matching values for **EmpID**
- c. all observations from **staff** and all observations from **managers**, whether or not they have matching values
- d. only those observations from **staff** and **managers** with matching values for **EmpID**

3. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;  
  merge managers staff;  
  by EmpID;  
run;
```

- a. all of the observations from **managers**, and only those observations from **staff** with matching values for **EmpID**
- b. all of the observations from **staff**, and only those observations from **managers** with matching values for **EmpID**
- c. all observations from **staff** and all observations from **managers**, whether or not they have matching values
- d. only those observations from **staff** and **managers** with matching values for **EmpID**

4. If you concatenate the data sets below in the order shown, what is the value of **Sale** in observation 2 of the new data set?

| Reps |                 |
|------|-----------------|
| ID   | Name            |
| 1    | Nay Rong        |
| 2    | Kelly Windsor   |
| 3    | Julio Meraz     |
| 4    | Richard Krabill |

| Close |          |
|-------|----------|
| ID    | Sale     |
| 1     | \$28,000 |
| 2     | \$30,000 |
| 2     | \$40,000 |
| 3     | \$15,000 |
| 3     | \$20,000 |
| 3     | \$25,000 |
| 4     | \$35,000 |

- a. missing
- b. \$30,000
- c. \$40,000
- d. You cannot concatenate these data sets.

4. If you concatenate the data sets below in the order shown, what is the value of **Sale** in observation 2 of the new data set?

**Reps**

| ID | Name            |
|----|-----------------|
| 1  | Nay Rong        |
| 2  | Kelly Windsor   |
| 3  | Julio Meraz     |
| 4  | Richard Krabill |

**Close**

| ID | Sale     |
|----|----------|
| 1  | \$28,000 |
| 2  | \$30,000 |
| 2  | \$40,000 |
| 3  | \$15,000 |
| 3  | \$20,000 |
| 3  | \$25,000 |
| 4  | \$35,000 |

- a. missing
- b. \$30,000
- c. \$40,000
- d. You cannot concatenate these data sets.

5. What happens if you submit the following program to merge **donors1** and **donors2**, shown below?

```
data merged;
  merge donors1 donors2;
  by ID;
run;
```

**donors1**

| ID   | Type | Units |
|------|------|-------|
| 2304 | O    | 16    |
| 1129 | A    | 48    |
| 1129 | A    | 50    |
| 1129 | A    | 57    |
| 2486 | B    | 63    |

- a. The **merged** data set contains some missing values because not all observations have matching observations in the other data set.
- b. The **merged** data set contains eight observations.
- c. The DATA step produces errors.

**donors2**

| ID   | Code | Units |
|------|------|-------|
| 6488 | 65   | 27    |
| 1129 | 63   | 32    |
| 5438 | 62   | 39    |
| 2304 | 61   | 45    |
| 1387 | 64   | 67    |

5. What happens if you submit the following program to merge **donors1** and **donors2**, shown below?

```
data merged;
  merge donors1 donors2;
  by ID;
run;
```

**donors1**

| ID   | Type | Units |
|------|------|-------|
| 2304 | O    | 16    |
| 1129 | A    | 48    |
| 1129 | A    | 50    |
| 1129 | A    | 57    |
| 2486 | B    | 63    |

- a. The **merged** data set contains some missing values because not all observations have matching observations in the other data set.
- b. The **merged** data set contains eight observations.
- c. The DATA step produces errors.

**donors2**

| ID   | Code | Units |
|------|------|-------|
| 6488 | 65   | 27    |
| 1129 | 63   | 32    |
| 5438 | 62   | 39    |
| 2304 | 61   | 45    |
| 1387 | 64   | 67    |

6. Suppose you want to concatenate these data sets. Which DATA step creates an output data set that combines the values of **Color** and **Hue** in the single variable **Color**?

a.

```
data widgets_all;
  set widget1(rename=(Color=Hue))
    widget2;
run;
```

b.

```
data widgets_all;
  set widget1
    widget2(rename=(Hue=Color));
run;
```

c.

```
data widgets_all;
  set widget1
    widget2(Hue=Color);
run;
```

Widget1

| Tag   | Color  | Model |
|-------|--------|-------|
| 77904 | blue   | AB42  |
| 56012 | red    | BA25  |
| 35499 | orange | FC36  |

Widget2

| Tag   | Hue    | Model |
|-------|--------|-------|
| 89325 | red    | SP17  |
| 65888 | yellow | BA12  |
| 00167 | green  | PG20  |

6. Suppose you want to concatenate these data sets. Which DATA step creates an output data set that combines the values of **Color** and **Hue** in the single variable **Color**?

a.

```
data widgets_all;
  set widget1(rename=(Color=Hue))
    widget2;
run;
```

b.

```
data widgets_all;
  set widget1
    widget2(rename=(Hue=Color));
run;
```

c.

```
data widgets_all;
  set widget1
    widget2(Hue=Color);
run;
```

Widget1

| Tag   | Color  | Model |
|-------|--------|-------|
| 77904 | blue   | AB42  |
| 56012 | red    | BA25  |
| 35499 | orange | FC36  |

Widget2

| Tag   | Hue    | Model |
|-------|--------|-------|
| 89325 | red    | SP17  |
| 65888 | yellow | BA12  |
| 00167 | green  | PG20  |

7. What is the syntax error in this DATA step?

```
data returns_qtr1;
  set returns_jan(rename=(ID=CustID)
                    (Return=Item) )
    returns_feb(rename=(Dt=Date) )
    returns_mar;
run;
```

- a. You cannot specify more than two data sets in the SET statement.
- b. There are too many sets of parentheses in the RENAME= option.
- c. You cannot specify multiple variables in the RENAME= option.
- d. The BY statement is missing.

7. What is the syntax error in this DATA step?

```
data returns_qtr1;
  set returns_jan(rename=(ID=CustID)
                    (Return=Item) )
    returns_feb(rename=(Dt=Date) )
    returns_mar;
run;
```

- a. You cannot specify more than two data sets in the SET statement.
- b. There are too many sets of parentheses in the RENAME= option.
- c. You cannot specify multiple variables in the RENAME= option.
- d. The BY statement is missing.

8. In the second iteration of this DATA step, after the data is merged, what are the values of C and A?

```
data client_amount;  
  merge clients(in=C)  
        amounts(in=A);  
  by Name;  
run;
```

Clients

| Name     | EmplID |
|----------|--------|
| Ankerton | 11123  |
| Davis    | 22298  |
| Masters  | 33351  |
| Wolmer   | 44483  |

- a. C=1, A=0
- b. C=0, A=1
- c. C=1, A=1
- d. missing
- e. unknown

Amounts

| Name     | Date    | Amt |
|----------|---------|-----|
| Ankerton | 08OCT96 | 92  |
| Ankerton | 15OCT96 | 43  |
| Davis    | 04OCT96 | 16  |
| Masters  | .       | 27  |
| Thomas   | 21OCT96 | 15  |

8. In the second iteration of this DATA step,  
after the data is merged, what are the values  
of C and A?

```
data client_amount;
  merge clients(in=C)
        amounts(in=A);
  by Name;
run;
```

Clients

| Name     | EmplID |
|----------|--------|
| Ankerton | 11123  |
| Davis    | 22298  |
| Masters  | 33351  |
| Wolmer   | 44483  |

- a. C=1, A=0
- b. C=0, A=1
- c. C=1, A=1
- d. missing
- e. unknown

Amounts

| Name     | Date    | Amt |
|----------|---------|-----|
| Ankerton | 08OCT96 | 92  |
| Ankerton | 15OCT96 | 43  |
| Davis    | 04OCT96 | 16  |
| Masters  | .       | 27  |
| Thomas   | 21OCT96 | 15  |

9. If you run this DATA step, what observations does the data set **bonuses** contain?

- a. only the observations from **staff** that have no match in **managers**
- b. only the observations from **managers** that have no match in **staff**
- c. all observations from both **managers** and **staff**, whether or not they match
- d. no observations

```
data bonuses;  
  merge managers (in=M)  
        staff (in=S);  
  by EmpID;  
  if M=0 and S=1;  
run;
```

9. If you run this DATA step, what observations does the data set **bonuses** contain?

```
data bonuses;
  merge managers (in=M)
        staff (in=S);
  by EmpID;
  if M=0 and S=1;
run;
```

- a. only the observations from **staff** that have no match in **managers**
- b. only the observations from **managers** that have no match in **staff**
- c. all observations from both **managers** and **staff**, whether or not they match
- d. no observations

10. What is the relationship of the data set **first** to the data set **second** when merged by the variable **ID**?

- a. one-to-one
- b. one-to-many
- c. many-to-one
- d. many-to-many
- e. nonmatching

| first  |        |     | second |          |
|--------|--------|-----|--------|----------|
| Name   | ID     | Age | ID     | Date     |
| Togar  | 121150 | 39  | 121150 | 02/15/05 |
| Kylie  | 121152 | 34  | 121152 | 05/22/07 |
| Birin  | 121153 | 32  | 121153 | 03/04/06 |
| Gloria | 121154 | 12  | 121154 | 11/22/05 |
| James  | 121155 | 36  | 121155 | 07/08/06 |
| Gene   | 121156 | 28  | 121156 | 12/15/06 |
| Tom    | 121157 | 27  | 121157 | 04/30/07 |

10. What is the relationship of the data set **first** to the data set **second** when merged by the variable **ID**?

- a. one-to-one
- b. one-to-many
- c. many-to-one
- d. many-to-many
- e. nonmatching

| first  |        |     | second |          |
|--------|--------|-----|--------|----------|
| Name   | ID     | Age | ID     | Date     |
| Togar  | 121150 | 39  | 121150 | 02/15/05 |
| Kylie  | 121152 | 34  | 121152 | 05/22/07 |
| Birin  | 121153 | 32  | 121153 | 03/04/06 |
| Gloria | 121154 | 12  | 121154 | 11/22/05 |
| James  | 121155 | 36  | 121155 | 07/08/06 |
| Gene   | 121156 | 28  | 121156 | 12/15/06 |
| Tom    | 121157 | 27  | 121157 | 04/30/07 |

# STAT604 SAS Lesson 14

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Processing Repetitive Code

Using Iterative DO Loops – Prep Guide Chapter 11

# Processing Repetitive Code

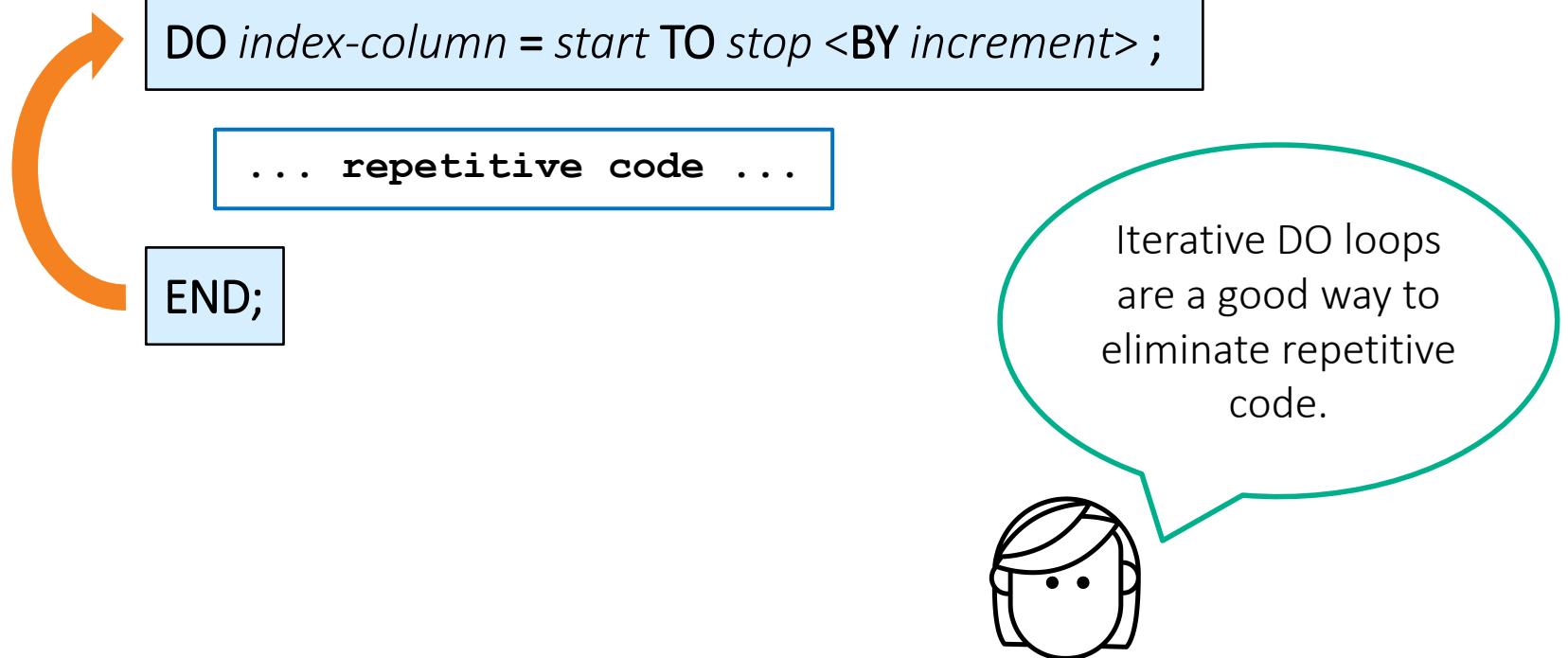
DATA step  
loop

```
data forecast;
  set sashelp.shoes(rename=(Sales=ProjectedSales));
  Year=1;
  ProjectedSales=ProjectedSales*1.05;
  output;
  Year=2;
  ProjectedSales=ProjectedSales*1.05;
  output;
  Year=3;
  ProjectedSales=ProjectedSales*1.05;
  output;
  keep Region Product Subsidiary Year ProjectedSales;
  format ProjectedSales dollar10.;

run;
```

NOTE: There were 395 observations read from the data set SASHELP.SHOES.  
NOTE: The data set WORK.FORECAST has 1185 observations and 5 variables

# Iterative DO Loop



# Iterative DO Statement

```
DO index-column = start TO stop <BY increment>;
```

column whose value  
controls DO loop execution

```
do Year =
```

# Iterative DO Statement

```
DO index-column = start TO stop <BY increment>;
```

the initial value of the index column

the value that the index column must exceed to stop execution of the DO loop

```
do Year = 1 to 3;
```

# Iterative DO Statement

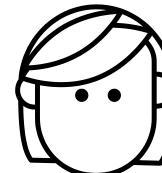
```
DO index-column = start TO stop <BY increment>;
```

a positive or negative number for incrementing the value of the index column

```
do Year = 1 to 3;
```

```
do Year = 1 to 3 by 1;
```

If you don't specify an increment, the default is 1.



# Iterative DO Statement

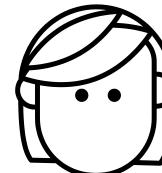
**DO *index-column* = *list*;**

A list of numeric or character values can also be used as index values

**do Year = 1, 2, 3;**

**do Mon = 'Jan' , 'Feb' ;**

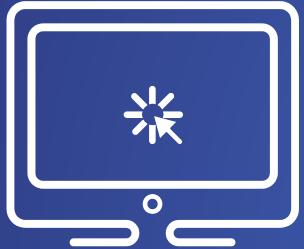
The loop will stop when it has used all members in the list.



# Executing an Iterative DO Loop



```
data forecast;
  set sashelp.shoes(rename=(Sales=ProjectedSales));
  do Year = 1 to 3;
    ProjectedSales=ProjectedSales*1.05;
    output;
  end;
  keep Region Product Subsidiary Year ProjectedSales;
  format ProjectedSales dollar10. ;
run;
```



# Executing an Iterative DO Loop

This demonstration illustrates using the DATA step debugger in PC SAS to see the execution of an iterative DO loop.

19-loops

sas

# Executing an Iterative DO Loop

Year is incremented by 1 at the bottom of the DO loop.

index = index + increment

The DO loop terminates when Year exceeds the stop value of 3.

```
1 data forecast / ldebug;
2   set sashelp.shoes(rename=(Sales=ProjectedSales));
3   do Year = 1 to 3;
4     ProjectedSales=ProjectedSales*1.05;
5     output;
6   end;    ← index = index + increment
7   keep Region Product Subsidiary Year
8   ProjectedSales;
9   format ProjectedSales dollar10.;
run;
```

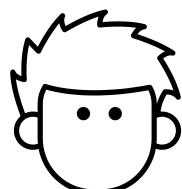
| Variable       | Value       | Watch                    |
|----------------|-------------|--------------------------|
| Region         | Africa      | <input type="checkbox"/> |
| Product        | Boot        | <input type="checkbox"/> |
| Subsidiary     | Addis Ababa | <input type="checkbox"/> |
| Stores         | 12          | <input type="checkbox"/> |
| ProjectedSales | \$34,452    | <input type="checkbox"/> |
| Inventory      | \$191,821   | <input type="checkbox"/> |
| Returns        | \$769       | <input type="checkbox"/> |
| Year           | 4           | <input type="checkbox"/> |
| _ERROR_        | 0           | <input type="checkbox"/> |
| N              | 1           | <input type="checkbox"/> |

# Scenario

```
data YearlySavings;  
  Amount=200;  
  do Month=1 to 12;  
    Savings+Amount;  
    output;  
  end;  
  format Savings 12.2;  
run;
```



How much money  
is in savings each  
month if we save  
\$200 per month  
for a year?



| Amount | Month | Savings |
|--------|-------|---------|
| 200    | 1     | 200.00  |
| 200    | 2     | 400.00  |
| 200    | 3     | 600.00  |
| 200    | 4     | 800.00  |
| 200    | 5     | 1000.00 |
| 200    | 6     | 1200.00 |
| 200    | 7     | 1400.00 |
| 200    | 8     | 1600.00 |
| 200    | 9     | 1800.00 |
| 200    | 10    | 2000.00 |
| 200    | 11    | 2200.00 |
| 200    | 12    | 2400.00 |

# Activity

1. Run the program on the previous slide. How much is in savings at month 12?
2. Delete the OUTPUT statement and run the program again.
3. How many rows are created?
4. What is the value of Month?
5. What is the value of Savings?

## Activity – Correct Answer

- How much is in savings at month 12? 2426.16

| (12) | Amount | (12) | Month | (12) | Savings |
|------|--------|------|-------|------|---------|
| 8    | 200    |      | 8     |      | 1612.05 |
| 9    | 200    |      | 9     |      | 1815.07 |
| 10   | 200    |      | 10    |      | 2018.43 |
| 11   | 200    |      | 11    |      | 2222.12 |
| 12   | 200    |      | 12    |      | 2426.16 |

- How many rows are created? one
- What is the value of Month? 13
- What is the value of Savings? 2426.16

| (12) | Amount | (12) | Month | (12) | Savings |
|------|--------|------|-------|------|---------|
|      | 200    |      | 13    |      | 2426.16 |

# Output inside the DO Loop

```
data YearlySavings;  
  
do Month=1 to 12;  
  
    output;  
end;  
  
run;
```

| Amount | Month | Savings |
|--------|-------|---------|
| 200    | 1     | 200.33  |
| 200    | 2     | 401.00  |
| 200    | 3     | 602.00  |
| 200    | 4     | 803.34  |
| 200    | 5     | 1005.01 |
| 200    | 6     | 1207.02 |
| 200    | 7     | 1409.36 |
| 200    | 8     | 1612.05 |
| 200    | 9     | 1815.07 |
| 200    | 10    | 2018.43 |
| 200    | 11    | 2222.12 |
| 200    | 12    | 2426.16 |

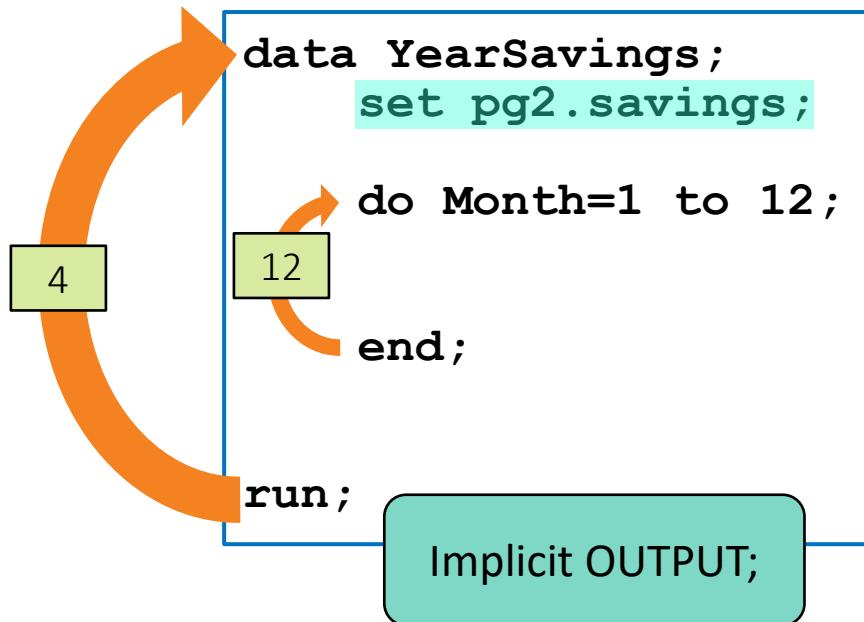
# Output outside the DO Loop

```
data YearlySavings;  
  do Month=1 to 12;  
    end;  
run;
```

Implicit OUTPUT;

| Amount | Month | Savings |
|--------|-------|---------|
| 200    | 13    | 2426.16 |

# DO Loop with an Input Table



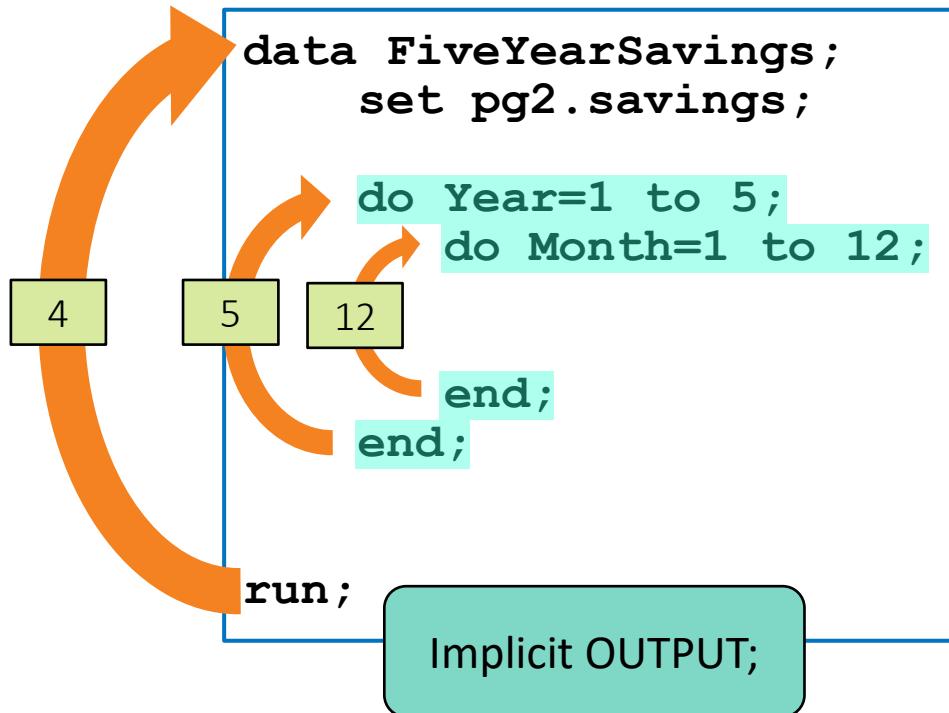
pg2.savings

| Name   | Amount |
|--------|--------|
| James  | 250    |
| Linda  | 300    |
| Mary   | 275    |
| Robert | 350    |

work.YearSavings

| Name   | Amount | Savings  |
|--------|--------|----------|
| James  | 250    | 3,032.70 |
| Linda  | 300    | 3,639.24 |
| Mary   | 275    | 3,335.97 |
| Robert | 350    | 4,245.78 |

# Nested DO Loops



pg2.savings

| Name   | Amount |
|--------|--------|
| James  | 250    |
| Linda  | 300    |
| Mary   | 275    |
| Robert | 350    |

work.FiveYearSavings

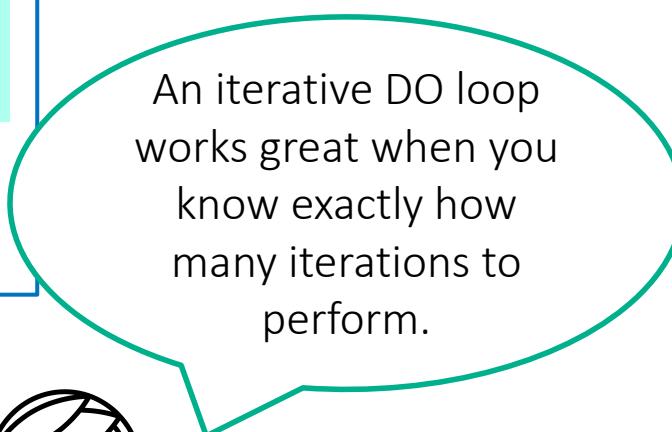
| Name   | Amount | Savings   |
|--------|--------|-----------|
| James  | 250    | 15,788.11 |
| Linda  | 300    | 18,945.73 |
| Mary   | 275    | 17,366.92 |
| Robert | 350    | 22,103.35 |

# Processing Repetitive Code

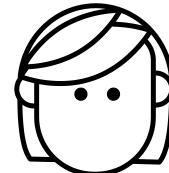
Using Conditional DO Loops

# Iterative DO Loop

```
data YearSavings;  
  set pg2.savings;  
  Savings=0;  
  do Month=1 to 12;  
    Savings+Amount;  
    Savings+(Savings*0.02/12);  
  end;  
  drop Month;  
  format Savings comma12.2;  
run;
```



An iterative DO loop works great when you know exactly how many iterations to perform.



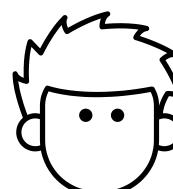
# Conditional DO Loop

```
data Savings3K;  
  set pg2.savings;  
  Month=0;  
  Savings=0;  
  do until (Savings>3000);  
    Month+1;  
    Savings+Amount;  
    Savings+(Savings*0.02/12);  
  end;  
  format Savings comma12.2;  
run;
```

?

Will save for 7 years  
will actually vestified  
and save for 7 years  
and will vestified  
at save for 7 years

The condition determines how many times the loop is executed.



# Conditional DO Loops

Is stuff left?

Tell condition  
loop

executes repetitively  
*until* a condition is true

DO UNTIL (*expression*);

... repetitive code ...

END;

executes repetitively  
*while* a condition is true

DO WHILE (*expression*);

... repetitive code ...

END;

# Conditional DO Loops

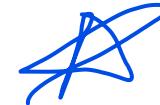
executes repetitively  
*until* a condition is true

```
do until (Savings>3000);  
  Month+1;  
  Savings+Amount;  
  Savings+(Savings*0.02/12);  
end;
```

executes repetitively  
*while* a condition is true

```
do while (Savings<=3000);  
  Month+1;  
  Savings+Amount;  
  Savings+(Savings*0.02/12);  
end;
```

# Checking the Condition



```
DO UNTIL (expression);
```

... repetitive code ...

```
END;
```

checks the condition at  
the bottom of the loop

```
DO WHILE (expression);
```

... repetitive code ...

```
END;
```

checks the condition at  
the top of the loop

# Checking the Condition

pg2.savings2

| Name   | Amount | Savings |
|--------|--------|---------|
| James  | 250    | 1250    |
| Linda  | 300    | 3600    |
| Mary   | 275    | 2200    |
| Robert | 350    | 1750    |

do until (Savings>3000) ;

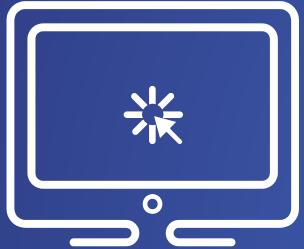
| Name   | Amount | Savings  | Month |
|--------|--------|----------|-------|
| James  | 250    | 3,026.36 | 7     |
| Linda  | 300    | 3,906.50 | 1     |
| Mary   | 275    | 3,038.77 | 3     |
| Robert | 350    | 3,167.54 | 4     |

DO UNTIL always  
executes once.

do while (Savings<=3000) ;

| Name   | Amount | Savings  | Month |
|--------|--------|----------|-------|
| James  | 250    | 3,026.36 | 7     |
| Linda  | 300    | 3,600.00 | 0     |
| Mary   | 275    | 3,038.77 | 3     |
| Robert | 350    | 3,167.54 | 4     |

DO WHILE executes only if  
the condition is true.



# Using Conditional DO Loops

This demonstration compares the operation of the UNTIL and WHILE loops.

# Combining Iterative and Conditional DO Loops

```
DO index-column = start TO stop <BY increment> UNTIL | WHILE (expression) ;
```

iterative

conditional

The DO loop stops executing when the stop value is exceeded or the condition is met, whichever is first.



# Iterative and Conditional DO Loop

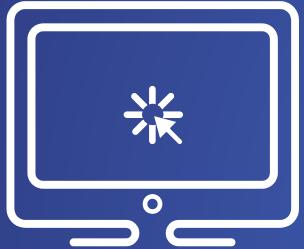
```
do Month=1 to 12 until (Savings>5000) ;  
  Savings+Amount;  
  Savings+(Savings*0.02/12);  
end;
```

At the bottom of loop, the condition is checked  
*before* the index column is incremented.

```
do Month=1 to 12 while (Savings<=5000) ;  
  Savings+Amount;  
  Savings+(Savings*0.02/12);  
end;
```

At the bottom of loop, the  
index column is incremented.

At the top of loop, the  
condition is checked.



# Combining Iterative and Conditional DO Loops

This demonstration illustrates modifying an existing DATA step with variations of the iterative and conditional DO loop.

# Processing Repetitive Code

SAS Array Processing

# Array Processing

You can use arrays to simplify programs that do the following:

- perform repetitive calculations
- create many variables with the same attributes
- read data
- compare variables
- perform a table lookup

# Business Scenario

The **orion.employee\_donations** data set contains quarterly contribution data for each employee. Orion management is considering a 25 percent matching program. Calculate each employee's quarterly contribution, including the proposed company supplement.

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
| 120265      | .    | .    | .    | 25   |
| 120267      | 15   | 15   | 15   | 15   |
| 120269      | 20   | 20   | 20   | 20   |
| 120270      | 20   | 10   | 5    | .    |
| 120271      | 20   | 20   | 20   | 20   |
| 120272      | 10   | 10   | 10   | 10   |

# Performing Repetitive Calculations

```
data charity;
  set orion.employee_donations;
  keep employee_id qtr1-qtr4;
  Qtr1=Qtr1*1.25;
  Qtr2=Qtr2*1.25;
  Qtr3=Qtr3*1.25;
  Qtr4=Qtr4*1.25;
run;
proc print data=charity noobs;
run;
```

Partial PROC PRINT Output

| Employee_ID | Qtr1  | Qtr2  | Qtr3  | Qtr4  |
|-------------|-------|-------|-------|-------|
| 120265      | .     | .     | .     | 31.25 |
| 120267      | 18.75 | 18.75 | 18.75 | 18.75 |
| 120269      | 25.00 | 25.00 | 25.00 | 25.00 |
| 120270      | 25.00 | 12.50 | 6.25  | .     |

# Performing Repetitive Calculations

The four calculations cannot be replaced by a single calculation inside a DO loop because they are not identical.

```
data charity;
  set orion.employee_donations;
  keep employee_id qtr1-qtr4;
  Qtr1=Qtr1*1.25;
  Qtr2=Qtr2*1.25;
  Qtr3=Qtr3*1.25;
  Qtr4=Qtr4*1.25;
run;
proc print data=charity noobs;
run;
```

do i=1 to 4;  
?

A SAS array can be used to simplify this code.

# Use Arrays to Simplify Repetitive Calculations

An array provides an alternate way to access values in the PDV, which simplifies repetitive calculations.

```
data charity;  
  set orion.employee_donations;  
  keep employee_id qtr1-qtr4;  
  Qtr1=Qtr1*1.25;  
  Qtr2=Qtr2*1.25;  
  Qtr3=Qtr3*1.25;  
  Qtr4=Qtr4*1.25;  
run;
```

An array can be used to access Qtr1-Qtr4.

PDV

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
|             |      |      |      |      |

# What Is a SAS Array?



A SAS *array*

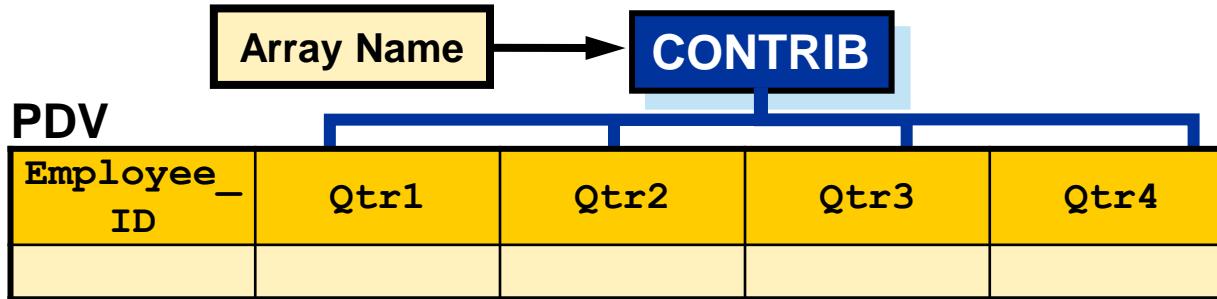
- is a temporary grouping of SAS variables that are arranged in a particular order
- is identified by an *array name*
- must contain all numeric or all character variables
- exists only for the duration of the current DATA step
- is **not** a variable.

must all be of  
the same type -



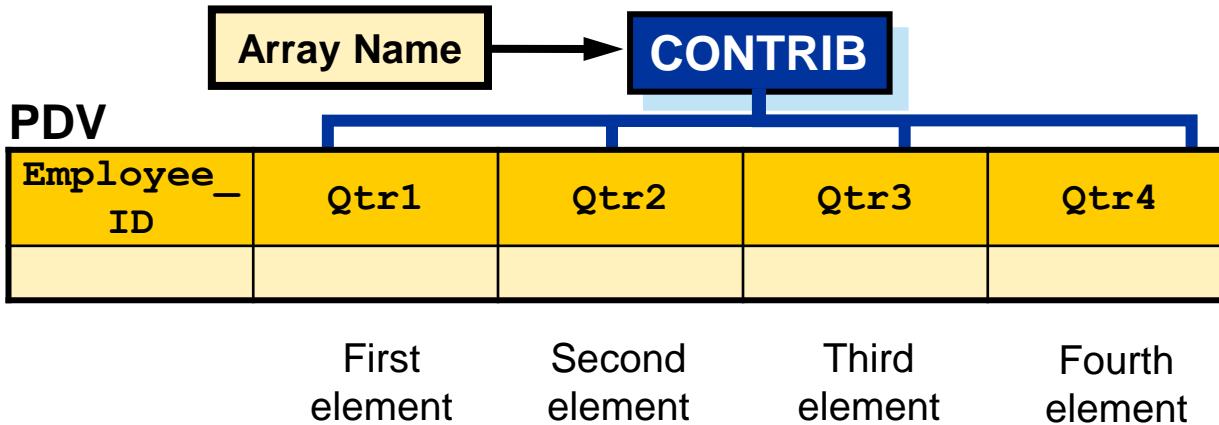
# Why Use a SAS Array?

Create an array named **Contrib** and use it to access the four numeric variables, **Qtr1 – Qtr4**.



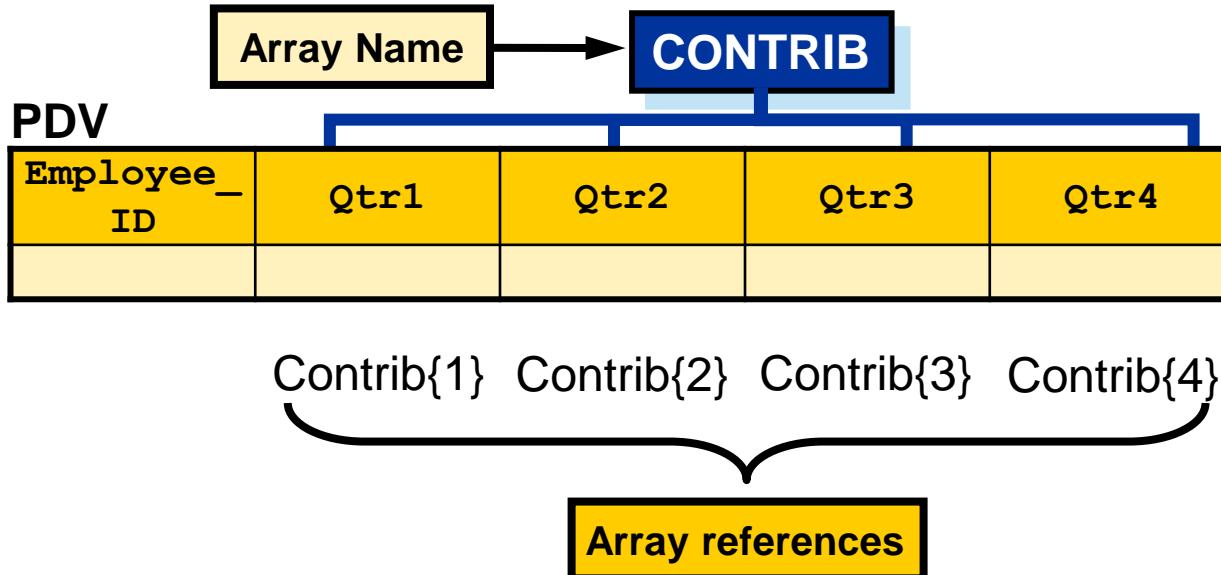
# Array Elements

Each value in an array is called an *element*.



# Referencing Array Elements

Each element is identified by a *subscript* that represents its position in the array. When you use an *array reference*, the corresponding value is substituted for the reference.



# The ARRAY Statement

The ARRAY statement is a compile-time statement that defines the elements in an array. The elements are created if they do not already exist in the PDV.

```
ARRAY array-name {subscript} <$> <length>  
      <array-elements>;
```

{subscript}

*reserved for  
character*

the number of elements

\$

indicates character elements

length

the length of elements

array-elements

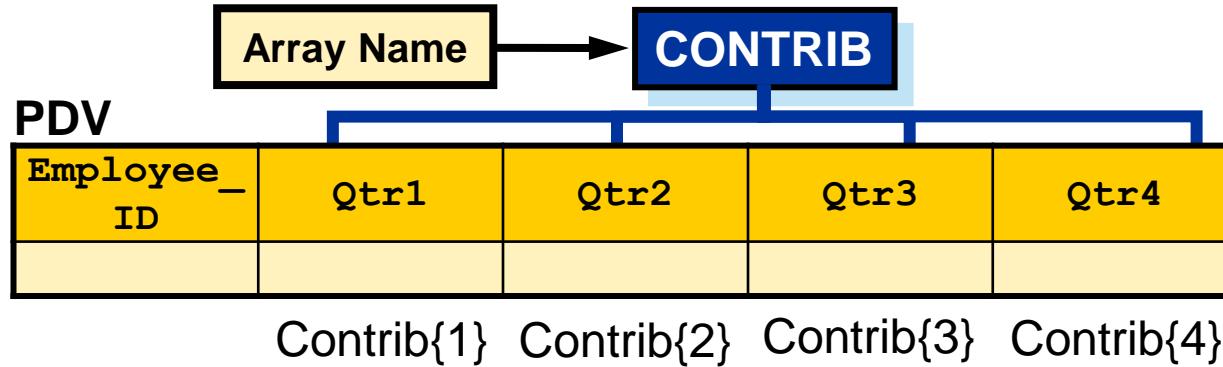
the names of elements

v/c done  
at compilation  
but do variable  
substitution or  
calculations w/  
array

# Defining an Array

The following ARRAY statement defines an array, **Contrib**, to access the four quarterly contribution variables.

```
array Contrib{4} qtr1 qtr2 qtr3 qtr4;
```



# Defining an Array

An alternate syntax uses an asterisk instead of a subscript. SAS determines the subscript by counting the variables in the element-list. The element-list must be included.

```
array Contrib{*} qtr1 qtr2 qtr3 qtr4;
```

Subscript is 4

element-list

The alternate syntax is often used when the array elements are defined with a SAS variable list.

```
array Contrib{*} qtr:;
```

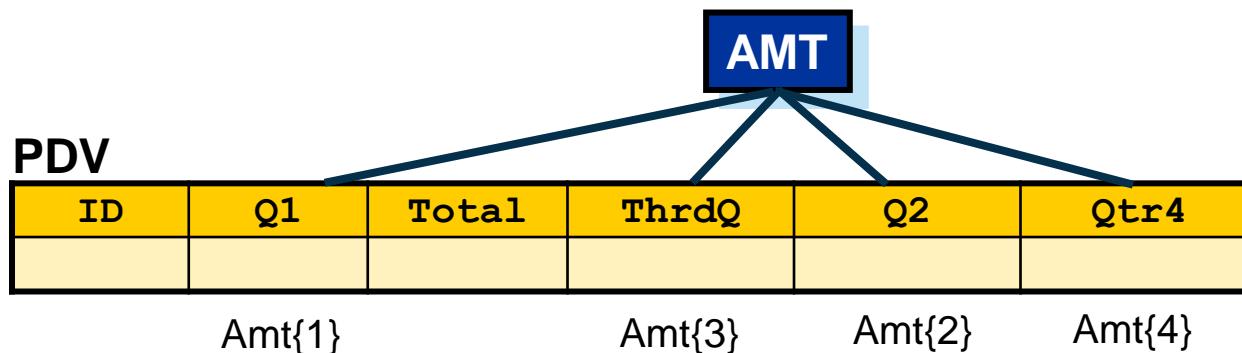
*is very flexible*

# Defining an Array

Variables that are elements of an array do not need the following:

- to have similar, related, or numbered names
- to be stored sequentially
- to be adjacent

```
array Amt{*} Q1 Q2 ThrdQ Qtr4;
```



# Quiz

What do you think would cause an error if you executed the code below?

```
data charity(keep=employee_id qtr1-qtr4);  
  set orion.employee_donations;  
  array Contrib1{3} qtr1-qtr4; — Too many vars →  
  array Contrib2{5} qtr:; — *END ← ↴  
  /* additional SAS statements */  
run;
```

# Quiz – Correct Answer

The subscript and the number of elements in the list do not agree.

```
data charity(keep=employee_id qtr1-qtr4);  
  set orion.employee_donations;  
  array Contrib1{3} qtr1-qtr4;  
  array Contrib2{5} qtr:;  
  /* additional SAS statements */  
run;
```

The subscript and  
element-list must agree.

Partial SAS Log

```
177   array Contrib1{3} qtr1-qtr4;  
ERROR: Too many variables defined for the dimension(s) specified  
for the array Contrib1.  
178   array Contrib2{5} qtr:;  
ERROR: Too few variables defined for the dimension(s) specified  
for the array Contrib2.
```

# Using a DO Loop to Process an Array

Array processing often occurs within an iterative DO loop in the following form:

```
DO index-variable=1 TO number-of-elements-in-array;  
  <additional SAS statements>  
END;
```

To reference an element, the *index-variable* is often used as a subscript:

```
array-name{index-variable}
```

# Using a DO Loop to Process an Array

```
data charity;
  set orion.employee_donations;
  keep employee_id qtr1-qtr4;
  array Contrib{4} qtr1-qtr4;
  do i=1 to 4;
    Contrib{i}=Contrib{i}*1.25;
  end;
run;
```

The index variable, **i**, is not written to the output data set because it is not listed in the KEEP statement.

# First Iteration of the DO Loop

```
data charity;  
  set orion.employee_donations;  
  keep employee_id qtr1-qtr4;  
  array Contrib{4} qtr1-qtr4;  
  do i=1 to 4;  
    Contrib{i}=Contrib{i}*1.25;  
  end;  
run;
```

when  $i=1$



$\text{Contrib}\{1\}=\text{Contrib}\{1\}*1.25;$



$\text{Qtr1}=\text{Qtr1}*1.25;$

## Second Iteration of the DO Loop

```
data charity;  
  set orion.employee_donations;  
  keep employee_id qtr1-qtr4;  
  array Contrib{4} qtr1-qtr4;  
  do i=1 to 4;  
    Contrib{i}=Contrib{i}*1.25;  
  end;  
run;
```

when  $i=2$



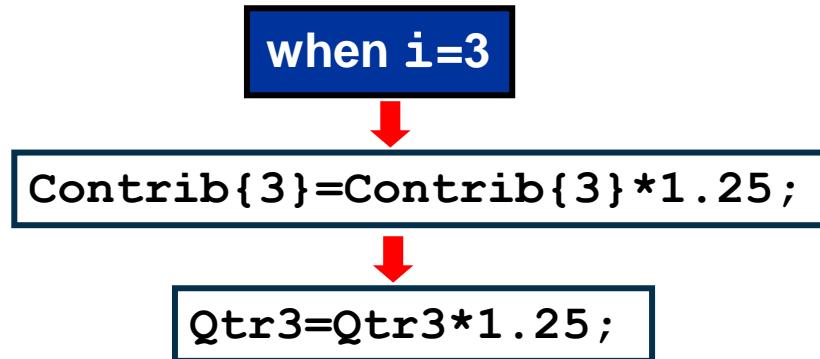
$\text{Contrib}\{2\}=\text{Contrib}\{2\}*1.25;$



$\text{Qtr2}=\text{Qtr2}*1.25;$

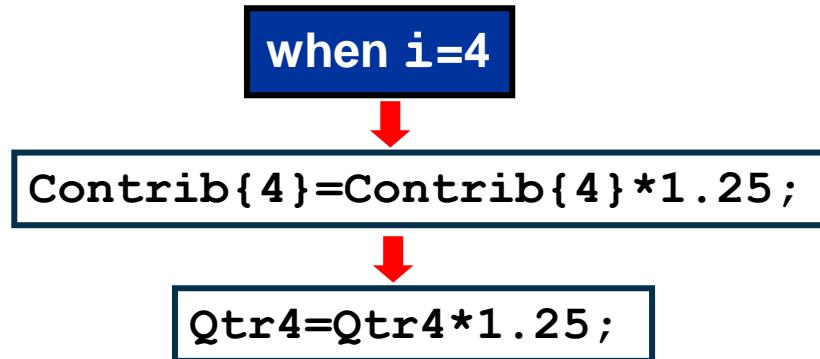
# Third Iteration of the DO Loop

```
data charity;  
  set orion.employee_donations;  
  keep employee_id qtr1-qtr4;  
  array Contrib{4} qtr1-qtr4;  
  do i=1 to 4;  
    Contrib{i}=Contrib{i}*1.25;  
  end;  
run;
```



# Fourth Iteration of the DO Loop

```
data charity;  
  set orion.employee_donations;  
  keep employee_id qtr1-qtr4;  
  array Contrib{4} qtr1-qtr4;  
  do i=1 to 4;  
    Contrib{i}=Contrib{i}*1.25;  
  end;  
run;
```



# Output: Using a Do Loop to Process an Array

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

Partial PROC PRINT Output

| Employee_ID | Qtr1  | Qtr2  | Qtr3  | Qtr4  |
|-------------|-------|-------|-------|-------|
| 120265      | .     | .     | .     | 31.25 |
| 120267      | 18.75 | 18.75 | 18.75 | 18.75 |
| 120269      | 25.00 | 25.00 | 25.00 | 25.00 |
| 120270      | 25.00 | 12.50 | 6.25  | .     |
| 120271      | 25.00 | 25.00 | 25.00 | 25.00 |
| 120272      | 12.50 | 12.50 | 12.50 | 12.50 |
| 120275      | 18.75 | 18.75 | 18.75 | 18.75 |
| 120660      | 31.25 | 31.25 | 31.25 | 31.25 |
| 120662      | 12.50 | .     | 6.25  | 6.25  |

# Processing Repetitive Code

Using SAS Arrays

# Using an Array as a Function Argument

The program below passes an array to the SUM function.

```
data test;
  set orion.employee_donations;
  array val{4} qtr1-qtr4;
  Tot1=sum(of qtr1-qtr4);
  Tot2=sum(of val{*});
run;
proc print data=test;
  var employee_id tot1 tot2;
run;
```

The array is passed as  
if it were a variable list.

Partial PROC PRINT Output

| Obs | Employee_ID | Tot1 | Tot2 |
|-----|-------------|------|------|
| 1   | 120265      | 25   | 25   |
| 2   | 120267      | 60   | 60   |
| 3   | 120269      | 80   | 80   |

# The DIM Function ~~X~~

The DIM function returns the number of elements in an array. This value is often used as the stop value in a DO loop.

General form of the DIM function:

DIM(*array\_name*)

```
array Contrib{*} qtr:;  
num_elements=dim(Contrib);  
  
do i=1 to num_elements;  
    Contrib{i}=Contrib{i}*1.25;  
end;  
run;
```

# The DIM Function

A call to the DIM function can be used in place of the stop value in the DO loop.

```
data charity;
  set orion.employee_donations;
  keep employee_id qtr1-qtr4;
  array Contrib{*} qtr:;
  do i=1 to dim(Contrib);
    Contrib{i}=Contrib{i}*1.25;
  end;
run;
```

# Using an Array to Create Numeric Variables

An ARRAY statement can be used to create new variables in the program data vector.

```
array discount{4} discount1-discount4;
```

If **discount1-discount4** do not exist in the PDV, they are created.

```
array Pct{4};
```

Four new variables are created:

**PDV**

| Pct1<br>N 8 | Pct2<br>N 8 | Pct3<br>N 8 | Pct4<br>N 8 |
|-------------|-------------|-------------|-------------|
|             |             |             |             |

# Using an Array to Create Character Variables

Define an array named **Month** to create six variables to hold character values with a length of 10.

```
array Month{6} $ 10;
```

## PDV

| Month1<br>\$ 10 | Month2<br>\$ 10 | Month3<br>\$ 10 | Month4<br>\$ 10 | Month5<br>\$ 10 | Month6<br>\$ 10 |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                 |                 |                 |                 |                 |                 |

# Business Scenario

Using **orion.employee\_donations** as input, calculate the percentage that each quarterly contribution represents of the employee's total annual contribution. Create four new variables to hold the percentages.

Partial Listing of **orion.employee\_donations**

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
| 120265      | .    | .    | .    | 25   |
| 120267      | 15   | 15   | 15   | 15   |
| 120269      | 20   | 20   | 20   | 20   |
| 120270      | 20   | 10   | 5    | .    |
| 120271      | 20   | 20   | 20   | 20   |
| 120272      | 10   | 10   | 10   | 10   |

# Creating Variables with Arrays

```
data percent(drop=i) ;
  set orion.employee_donations;
  array Contrib{4} qtr1-qtr4;
  array Percent{4};
  Total=sum(of contrib{*});
  do i=1 to 4;
    percent{i}=contrib{i}/total;
  end;
run;
```

The second ARRAY statement creates four numeric variables: **Percent1**, **Percent2**, **Percent3**, and **Percent4**.

# Output: Creating Variables with Arrays

```
proc print data=percent noobs;  
  var Employee_ID percent1-percent4;  
  format percent1-percent4 percent6.;  
run;
```

Partial PROC PRINT Output

| Employee_ID | Percent1 | Percent2 | Percent3 | Percent4 |
|-------------|----------|----------|----------|----------|
| 120265      | .        | .        | .        | 100%     |
| 120267      | 25%      | 25%      | 25%      | 25%      |
| 120269      | 25%      | 25%      | 25%      | 25%      |
| 120270      | 57%      | 29%      | 14%      | .        |
| 120271      | 25%      | 25%      | 25%      | 25%      |
| 120272      | 25%      | 25%      | 25%      | 25%      |
| 120275      | 25%      | 25%      | 25%      | 25%      |
| 120660      | 25%      | 25%      | 25%      | 25%      |
| 120662      | 50%      | .        | 25%      | 25%      |
| 120663      | .        | .        | 100%     | .        |
| 120668      | 25%      | 25%      | 25%      | 25%      |

# Business Scenario

Using **orion.employee\_donations** as input, calculate the difference in each employee's contribution from one quarter to the next.

Partial Listing of **orion.employee\_donations**

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
| 120265      | .    | .    | .    | 25   |
| 120267      | 15   | 15   | 15   | 15   |
| 120269      | 20   | 20   | 20   | 20   |
| 120270      | 20   | 10   | 5    | .    |
| 120271      | 20   | 20   | 20   | 20   |
| 120272      | 10   | 10   | 10   | 10   |

First difference: Qtr2 – Qtr1

Second difference: Qtr3 – Qtr2

Third difference: Qtr4 – Qtr3

# Quiz

How many ARRAY statements would you use to calculate the difference in each employee's contribution from one quarter to the next?

Partial Listing of `orion.employee_donations`

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
| 120265      | .    | .    | .    | 25   |
| 120267      | 15   | 15   | 15   | 15   |
| 120269      | 20   | 20   | 20   | 20   |

First difference: Qtr2 – Qtr1  
Second difference: Qtr3 – Qtr2  
Third difference: Qtr4 – Qtr3

# Quiz – Correct Answer

How many ARRAY statements would you use to calculate the difference in each employee's contribution from one quarter to the next? Answers can vary, but one solution is to use two arrays.

Partial Listing of `orion.employee_donations`

| Employee_ID | Qtr1 | Qtr2 | Qtr3 | Qtr4 |
|-------------|------|------|------|------|
| 120265      | .    | .    | .    | 25   |
| 120267      | 15   | 15   | 15   | 15   |
| 120269      | 20   | 20   |      |      |

First difference: Qtr2 – Qtr1  
Second difference: Qtr3 – Qtr2  
Third difference: Qtr4 – Qtr3

Use one array to refer to the existing variables and a second array to create the three **Difference** variables.

# Creating Variables with Arrays

```
data change;
  set orion.employee_donations;
  drop i;
  array Contrib{4} Qtr1-Qtr4;
  array Diff{3};
  do i=1 to 3;
    Diff{i}=Contrib{i+1}-Contrib{i};
  end;
run;
```

The **Contrib** array refers to existing variables. The **Diff** array creates three variables: **Diff1**, **Diff2**, and **Diff3**.

# Creating Variables with Arrays

```
data change;  
  set orion.employee_donations;  
  drop i;  
  array Contrib{4} Qtr1-Qtr4;  
  array Diff{3};  
  do i=1 to 3;  
    Diff{i}=Contrib{i+1}-Contrib{i};  
  end;  
run;
```

when  $i=1$



$Diff_1 = Qtr2 - Qtr1;$



$Diff1 = Qtr2 - Qtr1;$

# Creating Variables with Arrays

```
data change;  
  set orion.employee_donations;  
  drop i;  
  array Contrib{4} Qtr1-Qtr4;  
  array Diff{3};  
  do i=1 to 3;  
    Diff{i}=Contrib{i+1}-Contrib{i};  
  end;  
run;
```

when  $i=2$



$Diff\{2\}=Contrib\{3\}-Contrib\{2\};$



$Diff2=Qtr3-Qtr2;$

# Creating Variables with Arrays

```
data change;  
  set orion.employee_donations;  
  drop i;  
  array Contrib{4} Qtr1-Qtr4;  
  array Diff{3};  
  do i=1 to 3;  
    Diff{i}=Contrib{i+1}-Contrib{i};  
  end;  
run;
```

when  $i=3$



$Diff\{3\}=Contrib\{4\}-Contrib\{3\};$



$Diff3=Qtr4-Qtr3;$

# Creating Variables with Arrays

```
proc print data=change noobs;  
  var Employee_ID Diff1-Diff3;  
run;
```

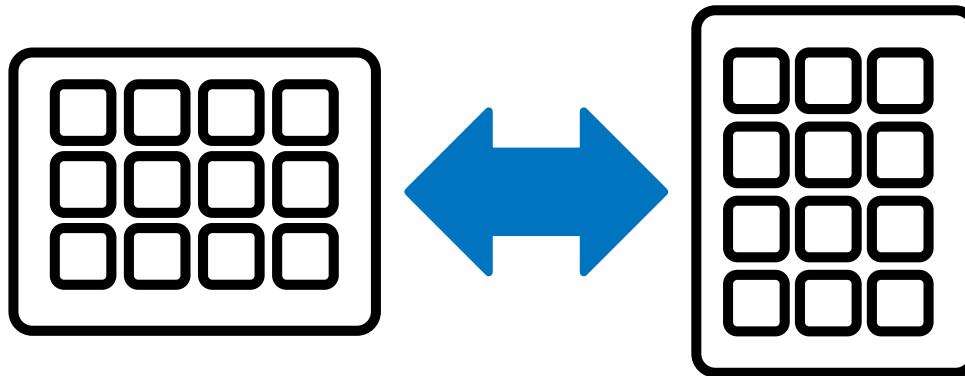
Partial PROC PRINT Output

| Employee_ID | Diff1 | Diff2 | Diff3 |
|-------------|-------|-------|-------|
| 120265      | .     | .     | .     |
| 120267      | 0     | 0     | 0     |
| 120269      | 0     | 0     | 0     |
| 120270      | -10   | -5    | .     |
| 120271      | 0     | 0     | 0     |
| 120272      | 0     | 0     | 0     |
| 120275      | 0     | 0     | 0     |
| 120660      | 0     | 0     | 0     |
| 120662      | .     | .     | 0     |

# Restructuring Tables

Restructuring Data with the DATA Step

# Restructuring Tables



# Table Structure

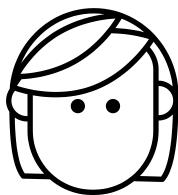
class\_test\_wide

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

class\_test\_narrow

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

Both tables include  
the same information,  
but they are  
structured differently.



# Multiple Choice Question

Which table and column (or columns) could you use with PROC MEANS to calculate an average for all test scores combined?

- a. class\_test\_wide, Math and Reading
- b. class\_test\_narrow, TestScore

```
proc means data=???;  
  var ???;  
run;
```

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

class\_test\_wide

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

class\_test\_narrow

# Multiple Choice Question – Correct Answer

Which table and column (or columns) could you use with PROC MEANS to calculate an average for all test scores combined?

- a. class\_test\_wide, Math and Reading
- b. class\_test\_narrow, TestScore

```
proc means data=pg2.class_test_narrow  
           maxdec=1;  
   var TestScore;  
run;
```

| The MEANS Procedure           |      |         |         |         |
|-------------------------------|------|---------|---------|---------|
| Analysis Variable : TestScore |      |         |         |         |
| N                             | Mean | Std Dev | Minimum | Maximum |
| 38                            | 77.7 | 11.3    | 55.0    | 99.0    |

# Restructuring Data with the DATA Step

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide



|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

narrow

You can use the  
DATA step to read  
one row and write  
multiple rows.



# Creating a Narrow Table with the DATA Step

```
data class_test_narrow(keep=Name Subject Score) ;  
    set pg2.class_test_wide;  
    length Subject $ 7;  
    Subject="Math";  
    Score=Math;  
    output;  
    Subject="Reading";  
    Score=Reading;  
    output;  
  
run;
```

How could this be more efficient or programmer friendly?

# Restructuring Data with the DATA Step

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

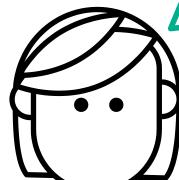
narrow



|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide

You can use the DATA step to read multiple rows before writing one row to the output table.



# Restructuring Data with the DATA Step

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

narrow

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide

```
if TestSubject="Math" then Math=TestScore;  
else if TestSubject="Reading" then Reading=TestScore;
```

# Activity

1. Examine the last DATA step code in **19-loops.sas** and run the program. What statement is necessary to carry the data from the first iteration over to the second?
2. Add a statement to include only the last row per student in the output table. Run the program.
3. What must be true of the input table for the DATA step to work?

## Activity – Correct Answer

1. Examine the DATA step code and run the program. Add the RETAIN statement and run the program again. Why is the RETAIN statement necessary?

The RETAIN statement hold values in the PDV across multiple iterations of the DATA step. The last row for each student includes both test scores.

| Name   | Math | Reading |
|--------|------|---------|
| Alfred | 82   | .       |
| Alfred | .    | 79      |
| Alice  | 71   | .       |
| Alice  | .    | 67      |

without RETAIN

| Name   | Math | Reading |
|--------|------|---------|
| Alfred | 82   | .       |
| Alfred | 82   | 79      |
| Alice  | 71   | 79      |
| Alice  | 71   | 67      |

with RETAIN

## Activity – Correct Answer

2. Add a subsetting IF statement to include only the last row per student in the output table.

```
data class_wide;
  set pg2.class_test_narrow;
  by name;
  retain Name Math Reading;
  keep Name Math Reading;
  if TestSubject="Reading" then Reading=TestScore;
  else if TestSubject="Math" then Math=TestScore;
  if last.name=1 then output;
run;
```

3. What must be true of the input table for the DATA step to work?  
The data must be sorted by Name.

# Iterative Processing - Lesson Quiz



1. Which output table is produced from the following step?

```
data Earnings(keep=Qtr Earned) ;  
    Amount=1000; Rate=.075/4;  
    do Qtr=1 to 4;  
        Earned+ (Amount+Earned)*Rate;  
    end;  
run;
```

- a.
- | (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 4   |      | 77.135865784 |
- b.
- | (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 5   |      | 77.135865784 |
- c.
- | (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 1   |      | 18.75        |
|      | 2   |      | 37.8515625   |
|      | 3   |      | 57.311279297 |
|      | 4   |      | 77.135865784 |
- d.
- | (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 1   |      | 18.75        |
|      | 2   |      | 37.8515625   |
|      | 3   |      | 57.311279297 |
|      | 4   |      | 77.135865784 |
|      | 5   |      | 77.135865784 |

1. Which output table is produced from the following step?

```
data Earnings(keep=Qtr Earned);  
  Amount=1000; Rate=.075/4;  
  do Qtr=1 to 4;  
    Earned+(Amount+Earned)*Rate;  
  end;  
run;
```

a.

| (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 4   |      | 77.135865784 |

b.

| (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 5   |      | 77.135865784 |

c.

| (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 1   |      | 18.75        |
|      | 2   |      | 37.8515625   |
|      | 3   |      | 57.311279297 |
|      | 4   |      | 77.135865784 |

d.

| (12) | Qtr | (12) | Earned       |
|------|-----|------|--------------|
|      | 1   |      | 18.75        |
|      | 2   |      | 37.8515625   |
|      | 3   |      | 57.311279297 |
|      | 4   |      | 77.135865784 |
|      | 5   |      | 77.135865784 |

2. Which statement is true regarding the iterative DO loop?

```
DO index-column = start TO stop <BY increment>;
```

- a. The start and stop values can be character or numeric values.
- b. If an increment value is not specified, the default increment is 0.
- c. The index column is incremented at the bottom of each DO loop.
- d. The index column is not in the final table unless specifically kept.

2. Which statement is true regarding the iterative DO loop?

**DO** *index-column* = *start* **TO** *stop* <**BY** *increment*> ;

- a. The start and stop values can be character or numeric values.
- b. If an increment value is not specified, the default increment is 0.
- c. The index column is incremented at the bottom of each DO loop.
- d. The index column is not in the final table unless specifically kept.

3. How many rows are in the output table given the following?

pg2.savings

| Name   | Amount |
|--------|--------|
| James  | 250    |
| Linda  | 300    |
| Mary   | 275    |
| Robert | 350    |

- a. 1
- b. 4
- c. 5
- d. 20

```
data work.savings;
  set pg2.savings;
  Savings=0;
  do Year=1 to 5;
    do qtr=1 to 4;
      Savings+Amount;
      Savings+(Savings*0.02/12);
    end;
  end;
run;
```

3. How many rows are in the output table given the following?

pg2.savings

| Name   | Amount |
|--------|--------|
| James  | 250    |
| Linda  | 300    |
| Mary   | 275    |
| Robert | 350    |

- a. 1
- b. 4
- c. 5
- d. 20

```
data work.savings;
  set pg2.savings;
  Savings=0;
  do Year=1 to 5;
    do qtr=1 to 4;
      Savings+Amount;
      Savings+(Savings*0.02/12);
    end;
  end;
run;
```

4. What is the final value of Year given the following step?

```
data invest;
    do Year=2010 to 2019;
        Capital+5000;
        Capital+(Capital*.03);
    end;
run;
```

- a. . (missing)
- b. 2010
- c. 2019
- d. 2020

4. What is the final value of Year given the following step?

```
data invest;
    do Year=2010 to 2019;
        Capital+5000;
        Capital+(Capital*.03);
    end;
run;
```

- a. . (missing)
- b. 2010
- c. 2019
- d. 2020

5. Which of the following statements contains valid syntax?

- a. `do 1 to 10 by 2;`
- b. `do while (Year>2025) ;`
- c. `do until Earnings<=100000;`
- d. `do date='01JAN2019' to '31JAN2019' ;`

5. Which of the following statements contains valid syntax?

- a. `do 1 to 10 by 2;`
- b. `do while (Year>2025) ;`
- c. `do until Earnings<=100000;`
- d. `do date='01JAN2019' to '31JAN2019' ;`

6. How many rows are in the output table given the following?

work.bikeinfo

| ⚠ | name   | ⌚ | bike |
|---|--------|---|------|
|   | Marco  | ⌚ | 12   |
|   | Angela | ⌚ | 10   |

- a. 2
- b. 3
- c. 6
- d. 12
- e. 24

```
data bikeinfo2;  
  set bikeinfo;  
  do month=1 to 3;  
    do week=1 to 4;  
      bike=bike+2;  
    end;  
    output;  
  end;  
run;
```

6. How many rows are in the output table given the following?

work.bikeinfo

| ⚠ | name   | ⌚ | bike |
|---|--------|---|------|
|   | Marco  | ⌚ | 12   |
|   | Angela | ⌚ | 10   |

- a. 2
- b. 3
- c. 6
- d. 12
- e. 24

```
data bikeinfo2;  
  set bikeinfo;  
  do month=1 to 3;  
    do week=1 to 4;  
      bike=bike+2;  
    end;  
    output;  
  end;  
run;
```

7. What is the value of x at the completion of the DATA step?

```
data test;  
  x=15;  
  do until(x>12);  
    x+1;  
  end;  
run;
```

- a. . (missing)
- b. 13
- c. 15
- d. 16

7. What is the value of x at the completion of the DATA step?

```
data test;  
  x=15;  
  do until(x>12);  
    x+1;  
  end;  
run;
```

- a. . (missing)
- b. 13
- c. 15
- d. 16

8. Which statement is false?
- a. The DO UNTIL loop executes until a condition is true.
  - b. The DO WHILE loop always executes at least one time.
  - c. The DO WHILE loop checks the condition at the top of the loop.
  - d. The DO UNTIL loop checks the condition at the bottom of the loop.

8. Which statement is false?
- a. The DO UNTIL loop executes until a condition is true.
  - b.** The DO WHILE loop always executes at least one time.
  - c. The DO WHILE loop checks the condition at the top of the loop.
  - d. The DO UNTIL loop checks the condition at the bottom of the loop.

9. Which of the following statements contains valid syntax?
- a. `do Age=10 to 14 and while (Weight<150) ;`
  - b. `do week=1 to 52 do until (Mileage ge 2750) ;`
  - c. `do Increase=5 to 10 while (temperature lt 102) ;`
  - d. `do Year=2018 to 2028 or until (Earnings<=100000) ;`

9. Which of the following statements contains valid syntax?
- a. do Age=10 to 14 and while (Weight<150) ;
  - b. do week=1 to 52 do until (Mileage ge 2750) ;
  - c. do Increase=5 to 10 while (temperature lt 102) ;
  - d. do Year=2018 to 2028 or until (Earnings<=100000) ;

10. Which output table is produced from the following step?

```
data test;  
  bike=10;  
  do day=1 to 7 while (bike lt 13);  
    bike=bike+2;  
  end;  
run;
```

- a. 

|     |      |     |     |
|-----|------|-----|-----|
| (2) | bike | (2) | day |
|     | 14   |     | 2   |
- b. 

|     |      |     |     |
|-----|------|-----|-----|
| (2) | bike | (2) | day |
|     | 14   |     | 3   |
- c. 

|     |      |     |     |
|-----|------|-----|-----|
| (2) | bike | (2) | day |
|     | 24   |     | 7   |
- d. 

|     |      |     |     |
|-----|------|-----|-----|
| (2) | bike | (2) | day |
|     | 24   |     | 8   |

10. Which output table is produced from the following step?

```
data test;  
  bike=10;  
  do day=1 to 7 while (bike lt 13);  
    bike=bike+2;  
  end;  
run;
```

- a. 
- b. 
- c. 
- d. 

# Restructuring Tables - Lesson Quiz



1. Which is the better description for the following table?

| Year   | Jan | Feb | Mar | Apr | May | Jun |
|--------|-----|-----|-----|-----|-----|-----|
| Yr1956 | 284 | 277 | 317 | 313 | 318 | 374 |
| Yr1957 | 315 | 301 | 356 | 348 | 355 | 422 |
| Yr1958 | 340 | 318 | 362 | 348 | 363 | 435 |

- a. wide table
- b. narrow table

1. Which is the better description for the following table?

| Year   | Jan | Feb | Mar | Apr | May | Jun |
|--------|-----|-----|-----|-----|-----|-----|
| Yr1956 | 284 | 277 | 317 | 313 | 318 | 374 |
| Yr1957 | 315 | 301 | 356 | 348 | 355 | 422 |
| Yr1958 | 340 | 318 | 362 | 348 | 363 | 435 |

- a. wide table
- b. narrow table

2. Which statement is needed for creating multiple rows from a single row when using the DATA step to go from a wide to a narrow table?
- a. WIDE
  - b. NARROW
  - c. RETAIN
  - d. OUTPUT

2. Which statement is needed for creating multiple rows from a single row when using the DATA step to go from a wide to a narrow table?
- a. WIDE
  - b. NARROW
  - c. RETAIN
  - d. OUTPUT

3. How many rows will be in the final table if `work.airwide` contains three rows?

- a. 3
- b. 6
- c. 9
- d. 12

```
data work.airnarrow;
    set work.airwide;
    Month='Jan';
    Air=Jan;
    output;
    Month='Feb';
    Air=Feb;
    output;
    Month='Mar';
    Air=Mar;
    output;
    keep Year Month Air;
run;
```

3. How many rows will be in the final table if `work.airwide` contains three rows?

- a. 3
- b. 6
- c. 9
- d. 12

```
data work.airnarrow;
    set work.airwide;
    Month='Jan';
    Air=Jan;
    output;
    Month='Feb';
    Air=Feb;
    output;
    Month='Mar';
    Air=Mar;
    output;
    keep Year Month Air;
run;
```

4. When using the DATA step to go from a narrow table to a wide table, the KEEP statement is needed to hold values in the PDV across multiple iterations of the DATA step.
- a. True
- b. False

4. When using the DATA step to go from a narrow table to a wide table, the KEEP statement is needed to hold values in the PDV across multiple iterations of the DATA step.
- a. True
- b. False

5. Which statement needs to be added to the DATA step to include only the last row per **Year** in the output table?

```
data work.airwide2(keep=Year Jan Feb Mar);  
  set work.airnarrow;  
  by Year;  
  retain Jan Feb Mar;  
  if Month='Jan' then Jan=Air;  
  else if Month='Feb' then Feb=Air;  
  else if Month='Mar' then Mar=Air;  
  ... insert statement here ...  
run;
```

- a. **output;**
- b. **if Last then output;**
- c. **if Last.Year=1 then output;**
- d. **if Last.Year=0 then output;**

5. Which statement needs to be added to the DATA step to include only the last row per **Year** in the output table?

```
data work.airwide2(keep=Year Jan Feb Mar);  
  set work.airnarrow;  
  by Year;  
  retain Jan Feb Mar;  
  if Month='Jan' then Jan=Air;  
  else if Month='Feb' then Feb=Air;  
  else if Month='Mar' then Mar=Air;  
  ... insert statement here ...  
run;
```

- a. **output;**
- b. **if Last then output;**
- c. **if Last.Year=1 then output;**
- d. **if Last.Year=0 then output;**

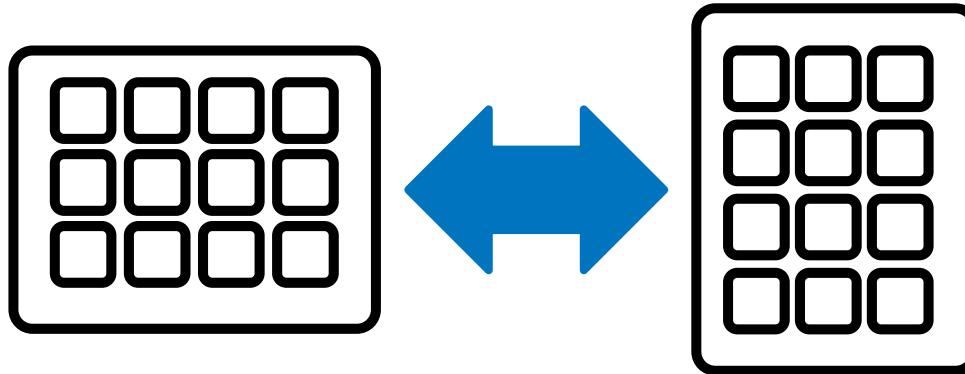
# STAT604 SAS Lesson 15

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Restructuring Tables

Restructuring Data with the DATA Step

# Restructuring Tables



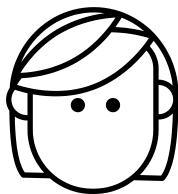
wide / flat  
table

# Table Structure

class\_test\_wide

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

Both tables include  
the same information,  
but they are  
structured differently.



Narrow  
table

class\_test\_narrow

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

# Multiple Choice Question

Which table and column (or columns) could you use with PROC MEANS to calculate an average for all test scores combined?

- a. class\_test\_wide, Math and Reading
- b. class\_test\_narrow, TestScore

```
proc means data=???;  
  var ???;  
run;
```

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

class\_test\_wide

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

class\_test\_narrow

# Multiple Choice Question – Correct Answer

Which table and column (or columns) could you use with PROC MEANS to calculate an average for all test scores combined?

- a. class\_test\_wide, Math and Reading
- b. class\_test\_narrow, TestScore

```
proc means data=pg2.class_test_narrow  
           maxdec=1;  
   var TestScore;  
run;
```

| The MEANS Procedure           |      |         |         |         |
|-------------------------------|------|---------|---------|---------|
| Analysis Variable : TestScore |      |         |         |         |
| N                             | Mean | Std Dev | Minimum | Maximum |
| 38                            | 77.7 | 11.3    | 55.0    | 99.0    |

# Restructuring Data with the DATA Step

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide



|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

narrow

You can use the  
DATA step to read  
one row and write  
multiple rows.



# Creating a Narrow Table with the DATA Step

```
data class_test_narrow(keep=Name Subject Score) ;  
    set pg2.class_test_wide;  
    length Subject $ 7;  
    Subject="Math";  
    Score=Math;  
    output;  
    Subject="Reading";  
    Score=Reading;  
    output;  
  
run;
```

How could this be more efficient or programmer friendly?

# Restructuring Data with the DATA Step

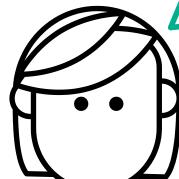
|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

narrow

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide

You can use the DATA step to read multiple rows before writing one row to the output table.



# Restructuring Data with the DATA Step

|   | Name    | TestSubject | TestScore |
|---|---------|-------------|-----------|
| 1 | Alfred  | Math        | 82        |
| 2 | Alfred  | Reading     | 79        |
| 3 | Alice   | Math        | 71        |
| 4 | Alice   | Reading     | 67        |
| 5 | Barbara | Math        | 96        |
| 6 | Barbara | Reading     | 86        |

narrow

|   | Name    | Math | Reading |
|---|---------|------|---------|
| 1 | Alfred  | 82   | 79      |
| 2 | Alice   | 71   | 67      |
| 3 | Barbara | 96   | 86      |

wide

```
if TestSubject="Math" then Math=TestScore;  
else if TestSubject="Reading" then Reading=TestScore;
```

# Activity

1. Examine the last DATA step code in **19-loops.sas** and run the program. What statement is necessary to carry the data from the first iteration over to the second?
2. Add a statement to include only the last row per student in the output table. Run the program.
3. What must be true of the input table for the DATA step to work?

## Activity – Correct Answer

1. Examine the DATA step code and run the program. Add the RETAIN statement and run the program again. Why is the RETAIN statement necessary?

The RETAIN statement hold values in the PDV across multiple iterations of the DATA step. The last row for each student includes both test scores.

| Name   | Math | Reading |
|--------|------|---------|
| Alfred | 82   | .       |
| Alfred | .    | 79      |
| Alice  | 71   | .       |
| Alice  | .    | 67      |

without RETAIN

| Name   | Math | Reading |
|--------|------|---------|
| Alfred | 82   | .       |
| Alfred | 82   | 79      |
| Alice  | 71   | 79      |
| Alice  | 71   | 67      |

with RETAIN

## Activity – Correct Answer

2. Add a subsetting IF statement to include only the last row per student in the output table.

```
data class_wide;
  set pg2.class_test_narrow;
  by name;
  retain Name Math Reading;
  keep Name Math Reading;
  if TestSubject="Reading" then Reading=TestScore;
  else if TestSubject="Math" then Math=TestScore;
  if last.name=1 then output;
run;
```

3. What must be true of the input table for the DATA step to work?  
The data must be sorted by Name.

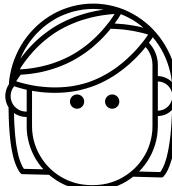
# Restructuring Tables

Restructuring Data with the TRANSPOSE Procedure

# Restructuring Data with PROC TRANSPOSE

```
PROC TRANSPOSE DATA=input-table <OUT=output-table>;  
  <ID col-name;>  
  <VAR col-name(s);>  
RUN;
```

PROC TRANSPOSE  
can restructure  
data with simple  
statements.



# Activity

Open the **20-transpose.sas** program and perform the following tasks:

1. Highlight the PROC PRINT step and run the selection. Note how many rows are in the **sashelp.class** table.
2. Highlight the PROC TRANSPOSE step and run the selection. Answer the following questions:

Which columns from the input table are transposed into rows?

What does each column in the output table represent?

What is the name of the output table?

Keep this  
program open for  
the next activity.

## Activity – Correct Answer

Which columns from the input table are transposed into rows?

Only the numeric columns are transposed (Age, Height, and Weight).

Each column must be all the same data type; by definition only numeric cols are transposed

What does each column in the output table represent?

Each column corresponds to a student (row) from the input table.

| ⚠_NAME_ | COL1  | COL2 | COL3 | COL4  | COL5  | COL6 | COL7 |
|---------|-------|------|------|-------|-------|------|------|
| Age     | 14    | 13   | 13   | 14    | 14    | 12   | 12   |
| Height  | 69    | 56.5 | 65.3 | 62.8  | 63.5  | 57.3 | 59.8 |
| Weight  | 112.5 | 84   | 98   | 102.5 | 102.5 | 83   | 84.5 |

What is the name of the output table?

work.data1

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.DATA1 has 3 observations and 20 variables.

Keep this program open for the next activity.

# Activity

Use the program from the previous activity to perform the following tasks:

1. Add the OUT= option in the PROC TRANSPOSE statement to create an output table named **class\_t**.
2. Add the following ID statement and run the step. What changes in the results?

```
id Name;
```

3. Add the following VAR statement and run the step. What changes in the results?

```
var Height Weight;
```

# Activity – Correct Answer

```
proc transpose data=sashelp.class out=class_t;  
  id Name;  
  var Height Weight;  
run;
```

The values of the ID column are assigned as column names.

|         |          |         |           |         |         |   |
|---------|----------|---------|-----------|---------|---------|---|
| ⚠_NAME_ | ⑪ Alfred | ⑪ Alice | ⑪ Barbara | ⑪ Carol | ⑪ Henry | ⑪ |
| Height  | 69       | 56.5    | 65.3      | 62.8    | 63.5    |   |
| Weight  | 112.5    | 84      | 98        | 102.5   | 102.5   |   |

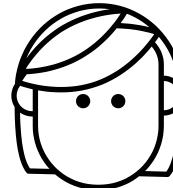
The VAR statement limits the columns that are transposed to rows.

# Transposing Values within Groups

```
PROC TRANSPOSE DATA=input-table <OUT=output-table>;  
  <VAR col-name(s)>;  
  <ID col-name>;  
  <BY col-name(s)>;  
RUN;
```

Use the BY statement  
to transpose data  
within groups.

The input table  
must be sorted by  
the same columns  
that you specify in  
the BY statement.



# Transposing Values within Groups

| Season | Basin | Name   | WindRank | WindMPH |
|--------|-------|--------|----------|---------|
| 1980   | EP    | AGATHA | 1        | 100     |
| 1980   | EP    | AGATHA | 2        | 95      |
| 1980   | EP    | AGATHA | 3        | 90      |
| 1980   | EP    | AGATHA | 4        | 85      |
| 1980   | EP    | BLAS   | 1        | 50      |
| 1980   | EP    | BLAS   | 2        | 50      |
| 1980   | EP    | BLAS   | 3        | 50      |
| 1980   | EP    | BLAS   | 4        | 45      |
| 1980   | EP    | CELIA  | 1        | 65      |
| 1980   | EP    | CELIA  | 2        | 65      |

narrow

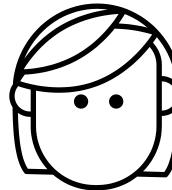
by Season Basin Name;

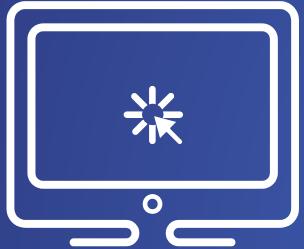
| Season | Basin | Name   | Wind1 | Wind2 | Wind3 | Wind4 |
|--------|-------|--------|-------|-------|-------|-------|
| 1980   | EP    | AGATHA | 100   | 95    | 90    | 85    |
| 1980   | EP    | BLAS   | 50    | 50    | 50    | 45    |
| 1980   | EP    | CELIA  | 65    | 65    | 65    | 65    |

wide



Each unique combination of BY values creates one row in the output table.





# Creating a Wide Table with PROC TRANSPOSE

This demonstration illustrates using PROC TRANSPOSE to transpose data values within groups into rows. The demo also uses options to customize the output table.

# Transposing Values into Groups

| Season | Basin | Name   | Wind1 | Wind2 | Wind3 | Wind4 |
|--------|-------|--------|-------|-------|-------|-------|
| 1980   | EP    | AGATHA | 100   | 95    | 90    | 85    |
| 1980   | EP    | BLAS   | 50    | 50    | 50    | 45    |
| 1980   | EP    | CELIA  | 65    | 65    | 65    | 65    |

wide



| Season | Basin | Name   | WindRank | WindMPH |
|--------|-------|--------|----------|---------|
| 1980   | EP    | AGATHA | 1        | 100     |
| 1980   | EP    | AGATHA | 2        | 95      |
| 1980   | EP    | AGATHA | 3        | 90      |
| 1980   | EP    | AGATHA | 4        | 85      |
| 1980   | EP    | BLAS   | 1        | 50      |
| 1980   | EP    | BLAS   | 2        | 50      |
| 1980   | EP    | BLAS   | 3        | 50      |
| 1980   | EP    | BLAS   | 4        | 45      |
| 1980   | EP    | CELIA  | 1        | 65      |
| 1980   | EP    | CELIA  | 2        | 65      |

narrow

You can also use  
PROC TRANSPOSE  
to convert one row  
into multiple rows.



# Activity

Go back to **20-transpose.sas** and perform the following tasks:

1. Run the next program. Notice that, by default, PROC TRANSPOSE transposes all the numeric columns, **Wind1-Wind4**.
2. Add a VAR statement in PROC TRANSPOSE to transpose only the **Wind1** and **Wind2** columns. Run the program.
3. What are the names of the columns that contain the column names and values that have been transposed?

## Activity – Correct Answer

2. Add a VAR statement in PROC TRANSPOSE to transpose only the Wind1 and Wind2 columns. Run the program.

```
var Wind1 Wind2;
```

3. What are the names of the columns that contain the column names and values that have been transposed? \_NAME\_ and **COL1**

| Storm Narrow |        |       |        |        |      |
|--------------|--------|-------|--------|--------|------|
| Obs          | Season | Basin | Name   | _NAME_ | COL1 |
| 1            | 1980   | EP    | AGATHA | Wind1  | 100  |
| 2            | 1980   | EP    | AGATHA | Wind2  | 95   |
| 3            | 1980   | EP    | BLAS   | Wind1  | 50   |
| 4            | 1980   | EP    | BLAS   | Wind2  | 50   |
| 5            | 1980   | EP    | CELIA  | Wind1  | 65   |
| 6            | 1980   | EP    | CELIA  | Wind2  | 65   |

# Changing Column Names

```
PROC TRANSPOSE DATA=input-table <OUT=output-table>  
    <NAME=column> <PREFIX=column>;
```

```
proc transpose data=pg2.storm_top4_wide name=WindRank  
    prefix=WindMPH;  
    by Season Basin Name;  
    var wind1-wind4;  
run;
```

| ② Season | ⚠ Basin | ⚠ Name | ⚠ WindRank | ② WindMPH1 |
|----------|---------|--------|------------|------------|
| 1980     | EP      | AGATHA | Wind1      | 100        |
| 1980     | EP      | AGATHA | Wind2      | 95         |
| 1980     | EP      | AGATHA | Wind3      | 90         |
| 1980     | EP      | AGATHA | Wind4      | 85         |
| 1980     | EP      | BLAS   | Wind1      | 50         |
| 1980     | EP      | BLAS   | Wind2      | 50         |

# Changing Column Names

```
proc transpose data=pg2.storm_top4_wide name=WindRank  
               prefix=WindMPH;  
   by Season Basin Name;  
   var wind1-wind4;  
run;
```

How could you  
change the name of  
the column in the  
output table to  
exclude the number  
1?

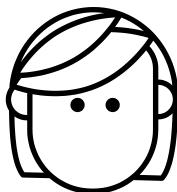


| Season | Basin | Name   | WindRank | WindMPH1 |
|--------|-------|--------|----------|----------|
| 1980   | EP    | AGATHA | Wind1    | 100      |
| 1980   | EP    | AGATHA | Wind2    | 95       |
| 1980   | EP    | AGATHA | Wind3    | 90       |
| 1980   | EP    | AGATHA | Wind4    | 85       |
| 1980   | EP    | BLAS   | Wind1    | 50       |
| 1980   | EP    | BLAS   | Wind2    | 50       |

# Changing Column Names

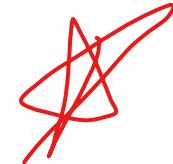
```
proc transpose data=pg2.storm_top4_wide name=WindRank  
               out=storm_rotate(rename=(col1=WindMPH));  
   by Season Basin Name;  
   var wind1-wind4;  
run;
```

Create an output table and use the RENAME= data set option.



| (2) | Season | (2) Basin | (2) Name | (2) WindRank | (2) WindMPH |
|-----|--------|-----------|----------|--------------|-------------|
|     | 1980   | EP        | AGATHA   | Wind1        | 100         |
|     | 1980   | EP        | AGATHA   | Wind2        | 95          |
|     | 1980   | EP        | AGATHA   | Wind3        | 90          |
|     | 1980   | EP        | AGATHA   | Wind4        | 85          |
|     | 1980   | EP        | BLAS     | Wind1        | 50          |
|     | 1980   | EP        | BLAS     | Wind2        | 50          |

# Recap PROC TRANSPOSE Options



```
PROC TRANSPOSE DATA=input-table <OUT=output-table>  
      <NAME=column> <PREFIX=column>;
```

- OUT – Controls name and library of output table
- NAME – Renames the \_NAME\_ column
- PREFIX – Changes COL1... to something meaningful

# Recap PROC TRANSPOSE Statements

A

```
PROC TRANSPOSE DATA=input-table <OUT=output-table>;  
  <VAR col-name(s);*>  
  <ID col-name;*>  
  <BY col-name(s);*>  
RUN;
```

- VAR – Specifies columns to be transposed (all numeric by default)
- ID – Values of the ID column are assigned as column names or suffix
- BY – Specifies grouping of transposed data



## Discussion

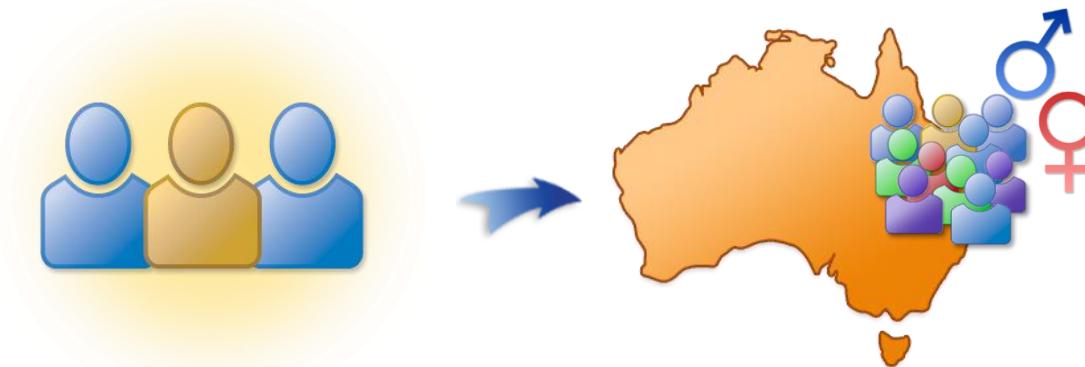
When might you prefer to use the DATA step instead of PROC TRANSPOSE to restructure data and vice versa?

# Producing Descriptive Statistics

The FREQ Procedure – Prep Guide Chapter 15

# Business Scenario

Orion Star management wants to know the number of male and female sales employees in Australia.



# Considerations

Use the FREQ procedure to analyze the **Gender** variable in a subset of **orion.sales**.

## The FREQ Procedure

| Gender | Frequency | Percent |
|--------|-----------|---------|
| F      | XX        | XX.XX   |
| M      | XX        | XX.XX   |

# FREQ Procedure

The FREQ procedure produces a one-way frequency table for each variable named in the TABLES statement.

```
proc freq data=orion.sales;
  tables Gender;
  where Country='AU';
run;
```

```
PROC FREQ DATA=SAS-data-set;
  <TABLES variable(s) </ options>>;
RUN;
```



If the TABLES statement is omitted, a one-way frequency table is produced for **every** variable in the data set. This can produce a large amount of output and is seldom preferred.

# Viewing the Output

A one-way frequency table was created for **Gender**. It lists the discrete values found in the data set and the number of observations in which the variable has that value.

| The FREQ Procedure |           |         |                      |                    |
|--------------------|-----------|---------|----------------------|--------------------|
| Gender             | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| F                  | 27        | 42.86   | 27                   | 42.86              |
| M                  | 36        | 57.14   | 63                   | 100.00             |

The default output includes frequency and percentage values, including cumulative statistics.

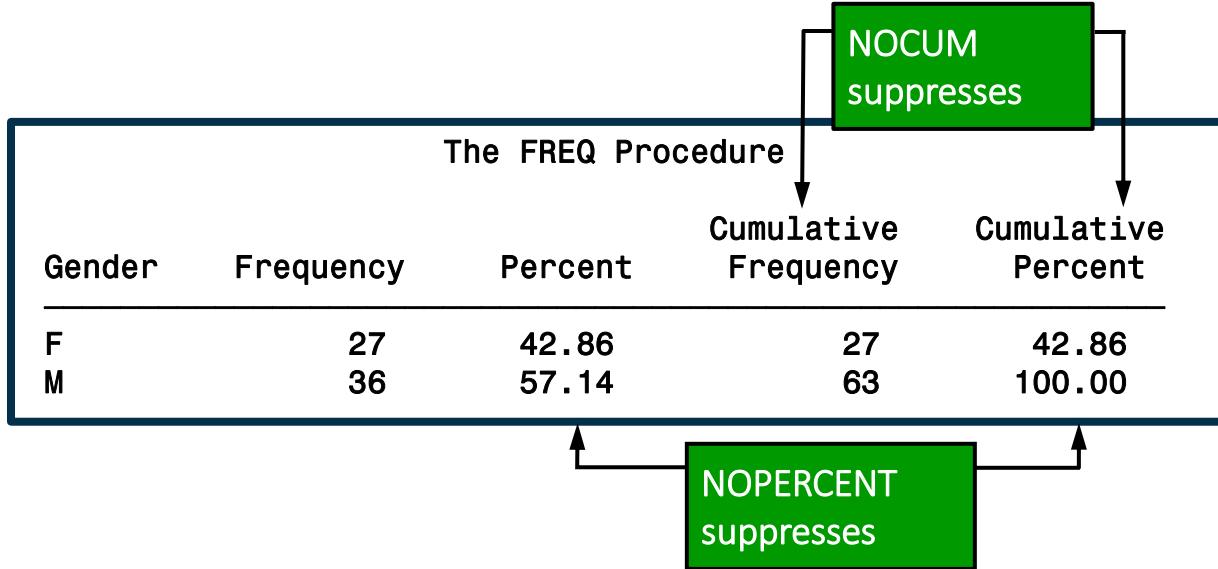
# Options to Suppress Statistics

Use options in the TABLES statement to suppress the display of selected default statistics.

```
TABLES variable(s) / options ;
```

| Option    | Description                           |
|-----------|---------------------------------------|
| NOCUM     | Suppresses the cumulative statistics. |
| NOPERCENT | Suppresses the percentage display.    |

# Options to Suppress Statistics



# Short Answer Poll

What change was needed to correct this program?

```
proc freq data=orion.sales;
  tables country nocum nopercnt;
run;
```

# Short Answer Poll – Correct Answer

What change was needed? A slash is required before the options in the TABLES statement.

```
31 proc freq data=orion.sales;
32      tables country nocum nopercent;
ERROR: Variable NOCUM not found.
ERROR: Variable NOPERCENT not found.
33 run;
```

```
proc freq data=orion.sales;
      tables country / nocum nopercent;
run;
```

The FREQ Procedure  
Country      Frequency

|    |     |
|----|-----|
| AU | 63  |
| US | 102 |

# Idea Exchange

This step creates a table for every variable in the data set:

```
proc freq data=orion.sales;  
run;
```

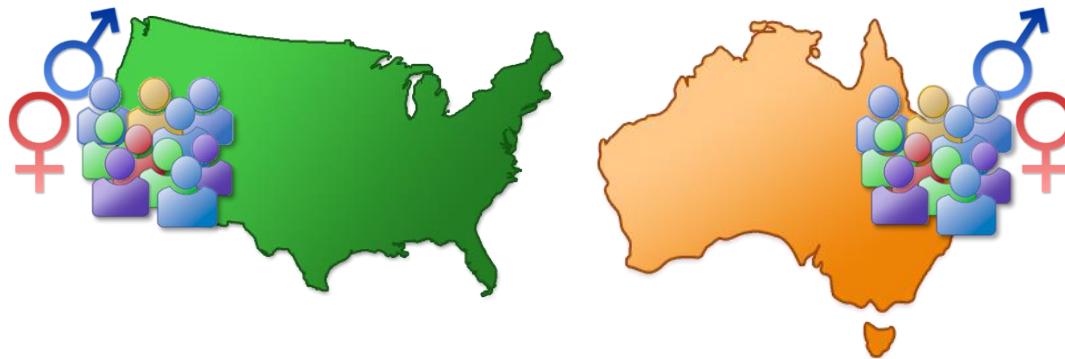
- Employee\_ID
- First\_Name
- Last\_Name
- Gender
- Salary
- Job\_Title
- Country
- Birth\_Date
- Hire\_Date

Which variables are most appropriate for a frequency analysis? Why?



# Business Scenario

Orion Star management wants to know how many sales employees are in each country, as well as the count of males and females.



# TABLES Statement

You can list multiple variables in a TABLES statement.  
A separate table is produced for each variable.

```
proc freq data=orion.sales;
  tables Gender Country;
run;
```

The FREQ Procedure

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

| Country | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|---------|-----------|---------|----------------------|--------------------|
| AU      | 63        | 38.18   | 63                   | 38.18              |
| US      | 102       | 61.82   | 165                  | 100.00             |

# BY Statement

The BY statement is used to request separate analyses for each BY group.

```
proc sort data=orion.sales out=sorted;
  by Country;
run;

proc freq data=sorted;
  tables Gender;
  by Country;
run;
```

The data set must be sorted or indexed by the variable (or variables) named in the BY statement.

# Viewing the Output

Each group appears on a separate page with a BY line.

| Country=AU |           |         |                      |                    |
|------------|-----------|---------|----------------------|--------------------|
| Gender     | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| F          | 27        | 42.86   | 27                   | 42.86              |
| M          | 36        | 57.14   | 63                   | 100.00             |

| Country=US |           |         |                      |                    |
|------------|-----------|---------|----------------------|--------------------|
| Gender     | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| F          | 41        | 40.20   | 41                   | 40.20              |
| M          | 61        | 59.80   | 102                  | 100.00             |

# Crosstabulation Table

An asterisk between two variables generates a two-way frequency table, or *crosstabulation table*.

```
proc freq data=orion.sales;
  tables Gender*Country;
run;
```

rows

columns

A two-way frequency table generates a single table with statistics for each distinct combination of values of the selected variables.

# Viewing the Output

## PROC FREQ Output

| The FREQ Procedure |                            |         |       |        |
|--------------------|----------------------------|---------|-------|--------|
|                    | Table of Gender by Country |         |       |        |
|                    | Gender                     | Country |       |        |
| Frequency          |                            | AU      | US    | Total  |
| F                  | 27                         | 41      | 68    |        |
|                    | 16.36                      | 24.85   | 41.21 |        |
|                    | 39.71                      | 60.29   |       |        |
|                    | 42.86                      | 40.20   |       |        |
| Percent            | M                          | 36      | 61    | 97     |
|                    |                            | 21.82   | 36.97 | 58.79  |
|                    |                            | 37.11   | 62.89 |        |
|                    |                            | 57.14   | 59.80 |        |
| Row Pct            | Total                      | 63      | 102   | 165    |
|                    |                            | 38.18   | 61.82 | 100.00 |

# Options to Suppress Statistics

Use options in the TABLES statement to suppress the display of selected default statistics.

```
TABLES variable(s) / options ;
```

| Option    | Description                                      |
|-----------|--|
| NOROW     | Suppresses the display of the row percentage.    |
| NOCOL     | Suppresses the display of the column percentage. |
| NOPERCENT | Suppresses the percentage display.               |
| NOFREQ    | Suppresses the frequency display.                |

# Options to Suppress Statistics

The FREQ Procedure

|       |       | Table of Gender by Country |       |        |
|-------|-------|----------------------------|-------|--------|
|       |       | Country                    |       |        |
|       |       | AU                         | US    | Total  |
| F     | 27    | 41                         | 68    |        |
|       | 16.36 | 24.85                      | 41.21 |        |
|       | 39.71 | 60.29                      |       |        |
|       | 42.86 | 40.20                      |       |        |
| M     | 36    | 61                         | 97    |        |
|       | 21.82 | 36.97                      | 58.79 |        |
|       | 37.11 | 62.89                      |       |        |
|       | 57.14 | 59.80                      |       |        |
| Total |       | 63                         | 102   | 165    |
|       |       | 38.18                      | 61.82 | 100.00 |

**NOFREQ**  
suppresses

**NOPERCENT**  
suppresses

# Options to Suppress Statistics

The FREQ Procedure

| Frequency |       | Table of Gender by Country |       |        |
|-----------|-------|----------------------------|-------|--------|
|           |       | Country                    |       |        |
| Gender    |       | AU                         | US    | Total  |
| F         | 27    | 41                         | 68    |        |
|           | 16.36 | 24.85                      | 41.21 |        |
|           | 39.71 | 60.29                      |       |        |
|           | 42.86 | 40.20                      |       |        |
|           |       |                            |       |        |
| M         | 36    | 61                         | 97    |        |
|           | 21.82 | 36.97                      | 58.79 |        |
|           | 37.11 | 62.89                      |       |        |
|           | 57.14 | 59.80                      |       |        |
| Total     |       | 63                         | 102   | 165    |
|           |       | 38.18                      | 61.82 | 100.00 |

NOROW suppresses

NOCOL suppresses

# Creating N-Way Tables

The FREQ procedure can create a series of two-way tables with a table for each level of the other tables.

```
proc freq data=orion.sales;
  tables country*gender*job_title;
run;
```

- You can include up to 50 variables in a single request.

# Creating N-Way Tables

Partial output:

| Frequency<br>Percent<br>Row Pct<br>Col Pct | Table 1 of Gender by Job_Title |                             |                               |                              |                               |                              |                            |                          |                       |
|--|--------------------------------|-----------------------------|-------------------------------|------------------------------|-------------------------------|------------------------------|----------------------------|--------------------------|-----------------------|
|  | Controlling for Country=AU     |                             |                               |                              |                               |                              |                            |                          |                       |
|  | Gender                         | Job_Title                   |                               |                              |                               |                              |                            |                          |                       |
|  |                                | Chief Sales Officer         | Sales Manager                 | Sales Rep. I                 | Sales Rep. II                 | Sales Rep. III               | Sales Rep. IV              | Senior Sales Manager     | Total                 |
|  | F                              | 0<br>0.00<br>0.00<br>. .    | 0<br>0.00<br>0.00<br>0.00     | 8<br>12.70<br>29.63<br>38.10 | 10<br>15.87<br>37.04<br>55.56 | 7<br>11.11<br>25.93<br>41.18 | 2<br>3.17<br>7.41<br>40.00 | 0<br>0.00<br>0.00<br>. . | 27<br>42.86<br>.<br>. |
| M  | 0<br>0.00<br>0.00<br>. .       | 2<br>3.17<br>5.56<br>100.00 | 13<br>20.63<br>36.11<br>61.90 | 8<br>12.70<br>22.22<br>44.44 | 10<br>15.87<br>27.78<br>58.82 | 3<br>4.76<br>8.33<br>60.00   | 0<br>0.00<br>0.00<br>. .   | 36<br>57.14<br>.<br>.    |                       |
| Total                                      | 0<br>0.00                      | 2<br>3.17                   | 21<br>33.33                   | 18<br>28.57                  | 17<br>26.98                   | 5<br>7.94                    | 0<br>0.00                  | 63<br>100.00             |                       |

# LIST and CROSSLIST Options

You can use the LIST and CROSSLIST options in the TABLES statement to “flatten” the output.

```
proc freq data=orion.sales;
  tables Gender*Country /list;
run;
```

| Gender | Country | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|--------|---------|-----------|---------|----------------------|--------------------|
| F      | AU      | 27        | 16.36   | 27                   | 16.36              |
| F      | US      | 41        | 24.85   | 68                   | 41.21              |
| M      | AU      | 36        | 21.82   | 104                  | 63.03              |
| M      | US      | 61        | 36.97   | 165                  | 100.00             |

# LIST and CROSSLIST Options

You can use the LIST and CROSSLIST options in the TABLES statement to “flatten” the output.

```
proc freq data=orion.sales;
  tables Gender*Country /crosstlist;
run;
```

| Table of Gender by Country |         |           |         |             |                |  |
|----------------------------|---------|-----------|---------|-------------|----------------|--|
| Gender                     | Country | Frequency | Percent | Row Percent | Column Percent |  |
| F                          | AU      | 27        | 16.36   | 39.71       | 42.86          |  |
|                            | US      | 41        | 24.85   | 60.29       | 40.20          |  |
|                            | Total   | 68        | 41.21   | 100.00      |                |  |
| M                          | AU      | 36        | 21.82   | 37.11       | 57.14          |  |
|                            | US      | 61        | 36.97   | 62.89       | 59.80          |  |
|                            | Total   | 97        | 58.79   | 100.00      |                |  |
| Total                      | AU      | 63        | 38.18   |             | 100.00         |  |
|                            | US      | 102       | 61.82   |             | 100.00         |  |
|                            | Total   | 165       | 100.00  |             |                |  |

# Business Scenario

A new data set, `orion.nonsales2`, must be validated. It contains information about non-sales employees and might include invalid and missing values.

Partial `orion.nonsales2`

| Employee_ID | First   | Last       | Gender | Salary | Job_Title         | Country |
|-------------|---------|------------|--------|--------|-------------------|---------|
| 120101      | Patrick | Lu         | M      | 163040 | Director          | AU      |
| 120104      | Kareen  | Billington | F      | 46230  | Admin Mgr         | au      |
| 120105      | Liz     | Povey      | F      | 27110  | Secretary I       | AU      |
| 120106      | John    | Hornsey    | M      | .      | Office Asst II    | AU      |
| 120107      | Sherie  | Sheedy     | F      | 30475  | Office Asst II    | AU      |
| 120108      | Gladys  | Gromek     | F      | 27660  | Warehouse Asst II | AU      |

# Considerations

Use the FREQ procedure to screen for invalid, missing, and duplicate data values.

Requirements of non-sales employee data:

- **Employee\_ID** values must be unique and not missing.
- **Gender** must be *F* or *M*.
- **Job\_Title** must not be missing.
- **Country** must have a value of *AU* or *US*.
- **Salary** values must be in the numeric range of 24000 to 500000.

# Short Answer Poll

What problems exist with the data in this partial data set?

| Employee_ID | First     | Last       | Gender | Salary | Job_Title               | Country |
|-------------|-----------|------------|--------|--------|-------------------------|---------|
| 120101      | Patrick   | Lu         | M      | 163040 | Director                | AU      |
| 120104      | Kareen    | Billington | F      | 46230  | Administration Manager  | au      |
| 120105      | Liz       | Povey      | F      | 27110  | Secretary I             | AU      |
| 120106      | John      | Hornsey    | M      |        | Office Assistant II     | AU      |
| 120107      | Sherie    | Sheedy     | F      | 30475  | Office Assistant III    | AU      |
| 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II  | AU      |
| 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I   | AU      |
| 120110      | Dennis    | Entwistle  | M      | 28615  | Warehouse Assistant III | AU      |
| 120111      | Ubaldo    | Spillane   | M      | 26895  | Security Guard II       | AU      |
| 120112      | Ellis     | Glattback  | F      | 26550  |                         | AU      |
| 120113      | Riu       | Horsey     | F      | 26870  | Security Guard II       | AU      |
| 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager        | AU      |
| 120115      | Hugh      | Nichollas  | M      | 2650   | Service Assistant I     | AU      |
|             | Austen    | Ralston    | M      | 29250  | Service Assistant II    | AU      |
| 120117      | Bill      | McCleary   | M      | 31670  | Cabinet Maker III       | AU      |
| 120118      | Darshi    | Hartshorn  | M      | 28090  | Cabinet Maker II        | AU      |

Hint: There are seven data problems.

# Short Answer Poll – Correct Answer

What problems exist with the data in this partial data set?

| Employee_ID | First     | Last       | Gender | Salary | Job_Title               | Country |
|-------------|-----------|------------|--------|--------|-------------------------|---------|
| 120101      | Patrick   | Lu         | M      | 163040 | Director                | AU      |
| 120104      | Kareen    | Billington | F      | 46230  | Administration Manager  | au      |
| 120105      | Liz       | Povey      | F      | 27110  | Secretary I             | AU      |
| 120106      | John      | Hornsey    | M      |        | Office Assistant II     | AU      |
| 120107      | Sherie    | Sheedy     | F      | 30475  | Office Assistant III    | AU      |
| 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II  | AU      |
| 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I   | AU      |
| 120110      | Dennis    | Entwistle  | M      | 28615  | Warehouse Assistant III | AU      |
| 120111      | Ubaldo    | Spillane   | M      | 26895  | Security Guard II       | AU      |
| 120112      | Ellis     | Glattback  | F      | 26550  |                         | AU      |
| 120113      | Riu       | Horsey     | F      | 26870  | Security Guard II       | AU      |
| 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager        | AU      |
| 120115      | Hugh      | Nichollas  | M      | 2650   | Service Assistant I     | AU      |
|             | Austen    | Ralston    | M      | 29250  | Service Assistant II    | AU      |
| 120117      | Bill      | McCleary   | M      | 31670  | Cabinet Maker III       | AU      |
| 120118      | Darshi    | Hartshorn  | M      | 28090  | Cabinet Maker II        | AU      |

Hint: There are seven data problems.

# FREQ Procedure for Data Validation

The FREQ procedure lists all discrete values for a variable and reports missing values.

```
proc freq data=orion.nonsales2;
  tables Gender Country / nocum nopercent;
run;
```

# Viewing the Output

## PROC FREQ Output

| The FREQ Procedure    |           |
|-----------------------|-----------|
| Gender                | Frequency |
| F                     | 110       |
| G                     | 1         |
| M                     | 123       |
| Frequency Missing = 1 |           |
| Country Frequency     |           |
| Country               | Frequency |
| AU                    | 33        |
| US                    | 196       |
| au                    | 3         |
| us                    | 3         |

# NLEVELS Option

The *NLEVELS option* displays a table that provides the number of distinct values for each analysis variable.

```
proc freq data=orion.nonsales2 nlevels;
  tables Gender Country / nocum nopercent;
run;
```

```
PROC FREQ DATA=SAS-data-set NLEVELS;
  TABLES variable(s);
RUN;
```

# Viewing the Output

PROC FREQ Output

| Number of Variable Levels |        |                |                   |
|---------------------------|--------|----------------|-------------------|
| Variable                  | Levels | Missing Levels | Nonmissing Levels |
| Gender                    | 4      | 1              | 3                 |
| Country                   | 4      | 0              | 4                 |

| Gender                           | Frequency |
|----------------------------------|-----------|
| F                                | 110       |
| G                                | 1         |
| M                                | 123       |
| <b>Frequency Missing<br/>= 1</b> |           |

| Country | Frequency |
|---------|-----------|
| AU      | 33        |
| US      | 196       |
| au      | 3         |
| us      | 3         |

# Check for Uniqueness

The values of **Employee\_ID** must be unique and not missing. PROC FREQ can be used to check for duplicate or missing values.

```
proc freq data=orion.nonsales2 order=freq;
  tables Employee_ID / nocum nopercent;
run;
```

The ORDER=FREQ option displays the results in descending frequency order.

# Viewing the Output

## Partial PROC FREQ Output

| The FREQ Procedure |           |
|--------------------|-----------|
| Employee_ID        | Frequency |
| 120108             | 2         |
| 120101             | 1         |
| 120104             | 1         |
| 120105             | 1         |
| 120106             | 1         |

|        |   |
|--------|---|
| 121134 | 1 |
| 121141 | 1 |
| 121142 | 1 |
| 121146 | 1 |
| 121147 | 1 |
| 121148 | 1 |

Frequency Missing = 1

# NLEVELS Option

NLEVELS can also be used to identify duplicates, when the number of distinct values is known.

```
proc freq data=orion.nonsales2 nlevels;
  tables Employee_ID / noprint;
run;
```

This example uses the NOPRINT option to suppress the frequency table. Only the Number of Variable Levels table is displayed.

# Viewing the Output

Partial PROC FREQ Output

| The FREQ Procedure        |        |                |                   |
|---------------------------|--------|----------------|-------------------|
| Number of Variable Levels |        |                |                   |
| Variable                  | Levels | Missing Levels | Nonmissing Levels |
| Employee_ID               | 234    | 1              | 233               |

There are 235 employees, but there are only 234 distinct Employee\_ID values. Therefore, there is one duplicate value and one missing value for Employee\_ID.

# NLEVELS Option

The `_ALL_` keyword with the NOPRINT option displays the number of levels for all variables without displaying frequency counts.

```
proc freq data=orion.nonsales2 nlevels;
  tables _all_ / noprint;
run;
```

# Viewing the Output

## PROC FREQ Output

| The FREQ Procedure<br>Number of Variable Levels |        |                   |                      |
|---|--------|-------------------|----------------------|
| Variable  | Levels | Missing<br>Levels | Nonmissing<br>Levels |
| Employee_ID                                     | 234    | 1                 | 233                  |
| First   | 204    | 0                 | 204                  |
| Last  | 228    | 0                 | 228                  |
| Gender  | 4      | 1                 | 3                    |
| Salary  | 230    | 1                 | 229                  |
| Job_Title                                       | 125    | 1                 | 124                  |
| Country   | 4      | 0                 | 4                    |

No frequency tables were displayed.

# Identifying Observations with Invalid Data

PROC FREQ uncovered the existence of invalid data values for **Gender**, **Country**, and **Employee\_ID**. Use PROC PRINT to display the observations with invalid values.

```
proc print data=orion.nonsales2;
  where Gender not in ('F','M') or
        Country not in ('AU','US') or
        Job_Title is null or
        Employee_ID is missing or
        Employee_ID=120108;
run;
```

# Viewing the Output

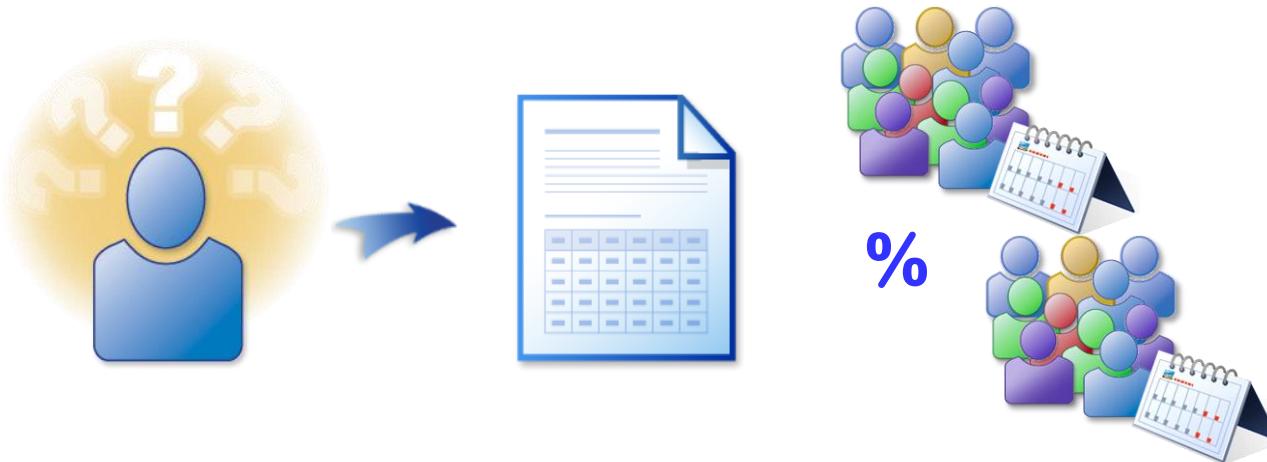
## PROC PRINT Output

| Obs | Employee_ID | First     | Last       | Gender | Salary | Job_Title                 | Country |
|-----|-------------|-----------|------------|--------|--------|---------------------------|---------|
| 2   | 120104      | Kareen    | Billington | F      | 46230  | Administration Manager    | au      |
| 6   | 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II    | AU      |
| 7   | 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I     | AU      |
| 10  | 120112      | Ellis     | Glattback  | F      | 26550  |                           | AU      |
| 12  | 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager          | AU      |
| 14  | .           | Austen    | Ralston    | M      | 29250  | Service Assistant II      | AU      |
| 84  | 120695      | Trent     | Moffat     | M      | 28180  | Warehouse Assistant II    | au      |
| 87  | 120698      | Geoff     | Kistanna   | M      | 26160  | Warehouse Assistant I     | au      |
| 101 | 120723      | Deanna    | Olsen      |        | 33950  | Corp. Comm. Specialist II | US      |
| 125 | 120747      | Zashia    | Farthing   | F      | 43590  | Financial Controller I    | us      |
| 197 | 120994      | Danelle   | Sergeant   | F      | 31645  | Office Administrator I    | us      |
| 200 | 120997      | Mary      | Donathan   | F      | 27420  | Shipping Administrator I  | us      |

original  
observation  
numbers

# Business Scenario

The manager of Human Resources requested a report that shows the number and percent of sales employees who are hired each year.



# Using Formats in PROC FREQ

A FORMAT statement can be used in PROC FREQ to format data values.

```
proc freq data=orion.sales;
  tables Hire_Date / nocum;
  format Hire_Date date9.;
run;
```

Partial PROC FREQ Output

| The FREQ Procedure |           |         |
|--------------------|-----------|---------|
| Hire_Date          | Frequency | Percent |
| 01JAN1978          | 17        | 10.30   |
| 01FEB1978          | 2         | 1.21    |
| 01APR1978          | 1         | 0.61    |
| 01JUL1978          | 1         | 0.61    |
| 01AUG1978          | 1         | 0.61    |

many discrete values, and  
not what the manager  
requested

# Using Formats in PROC FREQ

A FORMAT statement can also be used in PROC FREQ to group the data.

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

Partial PROC FREQ Output

| The FREQ Procedure |           |         |
|--------------------|-----------|---------|
| Hire_Date          | Frequency | Percent |
| 1978               | 23        | 13.94   |
| 1979               | 2         | 1.21    |
| 1980               | 4         | 2.42    |
| 1981               | 3         | 1.82    |
| 1982               | 7         | 4.24    |

fewer discrete values

# Short Answer Poll

Can user-defined formats be used to group data?

# Short Answer Poll – Correct Answer

Can user-defined formats be used to group data? **yes**

| The FREQ Procedure |           |         |                      |                    |
|--------------------|-----------|---------|----------------------|--------------------|
| Salary             | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| Tier1              | 1         | 0.61    | 1                    | 0.61               |
| Tier2              | 158       | 95.76   | 159                  | 96.36              |
| Tier3              | 4         | 2.42    | 163                  | 98.79              |
| Tier4              | 2         | 1.21    | 165                  | 100.00             |

# Lesson Quiz



6. Which statement is false concerning the TRANSPOSE procedure?
- a. Columns are transposed into rows.
  - b. By default, numeric columns are transposed.
  - c. Use a BY statement to sort the data while transposing.
  - d. Use a VAR statement to specifically specify the character and numeric columns to transpose.

6. Which statement is false concerning the TRANSPOSE procedure?

- a. Columns are transposed into rows.
- b. By default, numeric columns are transposed.
- c. Use a BY statement to sort the data while transposing.
- d. Use a VAR statement to specifically specify the character and numeric columns to transpose.

7. Which statements are needed in a PROC TRANSPOSE step for the following example (narrow  $\Rightarrow$  wide)?

| Day      | Meal      | Food     |
|----------|-----------|----------|
| Saturday | Breakfast | Yogurt   |
| Saturday | Lunch     | Sandwich |
| Saturday | Dinner    | Steak    |
| Sunday   | Breakfast | Pancakes |
| Sunday   | Lunch     | Salad    |
| Sunday   | Dinner    | Lasagna  |



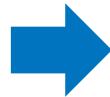
| Day      | Breakfast | Lunch    | Dinner  |
|----------|-----------|----------|---------|
| Saturday | Yogurt    | Sandwich | Steak   |
| Sunday   | Pancakes  | Salad    | Lasagna |

- a. **by Day;  
var Meal Food;**
- c. **by Day;  
id Food;  
var Meal;**

- b. **id Day;  
var Food Meal;**
- d. **by Day;  
id Meal;  
var Food;**

7. Which statements are needed in a PROC TRANSPOSE step for the following example (narrow  $\Rightarrow$  wide)?

| Day      | Meal      | Food     |
|----------|-----------|----------|
| Saturday | Breakfast | Yogurt   |
| Saturday | Lunch     | Sandwich |
| Saturday | Dinner    | Steak    |
| Sunday   | Breakfast | Pancakes |
| Sunday   | Lunch     | Salad    |
| Sunday   | Dinner    | Lasagna  |



| Day      | Breakfast | Lunch    | Dinner  |
|----------|-----------|----------|---------|
| Saturday | Yogurt    | Sandwich | Steak   |
| Sunday   | Pancakes  | Salad    | Lasagna |

- a. **by Day;  
var Meal Food;**
- c. **by Day;  
id Food;  
var Meal;**

- b. **id Day;  
var Food Meal;**
- d. **by Day;  
id Meal;  
var Food;**

8. Which statement or statements are needed in a PROC TRANSPOSE step for the following example (wide  $\Rightarrow$  narrow)?

| ⚠ Day    | ⚠ Breakfast | ⚠ Lunch  | ⚠ Dinner |
|----------|-------------|----------|----------|
| Saturday | Yogurt      | Sandwich | Steak    |
| Sunday   | Pancakes    | Salad    | Lasagna  |



| ⚠ _NAME_  | ⚠ COL1   | ⚠ COL2   |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. **by Day;**
- b. **var Breakfast Lunch Dinner;**
- c. **id Day;**
- d. **id Day;  
var Breakfast Lunch Dinner;**

8. Which statement or statements are needed in a PROC TRANSPOSE step for the following example (wide  $\Rightarrow$  narrow)?

| ⚠ Day    | ⚠ Breakfast | ⚠ Lunch  | ⚠ Dinner |
|----------|-------------|----------|----------|
| Saturday | Yogurt      | Sandwich | Steak    |
| Sunday   | Pancakes    | Salad    | Lasagna  |



| ⚠ _NAME_  | ⚠ COL1   | ⚠ COL2   |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. **by Day;**
- b. **var Breakfast Lunch Dinner;**
- c. **id Day;**
- d. **id Day;  
var Breakfast Lunch Dinner;**

9. Which option is needed in the PROC TRANSPOSE statement to rename the \_NAME\_ column?

| _NAME_    | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |



| Meal      | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. **\_name\_=Meal**
- b. **name=Meal**
- c. **prefix=Meal**
- d. **rename=Meal**

9. Which option is needed in the PROC TRANSPOSE statement to rename the \_NAME\_ column?

| _NAME_    | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |



| Meal      | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. **\_name\_=Meal**
- b. **name=Meal**
- c. **prefix=Meal**
- d. **rename=Meal**

10. Which option is needed in the PROC TRANSPOSE statement to rename the COL columns?

| Meal      | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |



| Meal      | Day7     | Day1     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. `out=meals2 (COL1=Day_7 COL2=Day_1)`
- b. `out=meals2 (name= (COL1=Day_7 COL2=Day_1) )`
- c. `out=meals2 (rename= (COL1=Day_7 COL2=Day_1) )`
- d. `out=meals2 (prefix= (COL1=Day_7 COL2=Day_1) )`

10. Which option is needed in the PROC TRANSPOSE statement to rename the COL columns?



| Meal      | COL1     | COL2     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

| Meal      | Day7     | Day1     |
|-----------|----------|----------|
| Breakfast | Yogurt   | Pancakes |
| Lunch     | Sandwich | Salad    |
| Dinner    | Steak    | Lasagna  |

- a. `out=meals2 (COL1=Day_7 COL2=Day_1)`
- b. `out=meals2 (name= (COL1=Day_7 COL2=Day_1) )`
- c. `out=meals2 (rename= (COL1=Day_7 COL2=Day_1) )`
- d. `out=meals2 (prefix= (COL1=Day_7 COL2=Day_1) )`

1. Which of these procedures produces output that is most useful for detecting duplicate values?
  - a. PROC PRINT
  - b. PROC FREQ
  - c. PROC MEANS
  - d. PROC UNIVARIATE

1. Which of these procedures produces output that is most useful for detecting duplicate values?
  - a. PROC PRINT
  - b. PROC FREQ**
  - c. PROC MEANS
  - d. PROC UNIVARIATE

2. Which of these programs is most useful for determining the exact observation that contains a numeric variable with an extreme value?
- a. proc print data=sales.totals;  
    var ProdNum Sales Region;  
    run;
  
  - b. proc freq data=sales.totals;  
    tables ProdNum Sales Region;  
    run;
  
  - c. proc univariate data=sales.totals;  
    run;

2. Which of these programs is most useful for determining the exact observation that contains a numeric variable with an extreme value?
- a. proc print data=sales.totals;  
    var ProdNum Sales Region;  
    run;
  
  - b. proc freq data=sales.totals;  
    tables ProdNum Sales Region;  
    run;
  
  - c. proc univariate data=sales.totals;  
    run;

3. A PROC FREQ analysis identified invalid and missing values in a data set. Which of these procedures displays the observations that contain invalid or missing values?
- a. PROC PRINT
  - b. PROC FREQ
  - c. PROC MEANS
  - d. PROC UNIVARIATE

3. A PROC FREQ analysis identified invalid and missing values in a data set. Which of these procedures displays the observations that contain invalid or missing values?
- a. PROC PRINT
  - b. PROC FREQ
  - c. PROC MEANS
  - d. PROC UNIVARIATE

4. Which PROC FREQ step creates the output shown here?

- a. proc freq data=orion.qtr1\_2007;  
    tables Order\_Type;  
    run;
- b. proc freq data=orion.qtr1\_2007  
    nlevels;  
    tables Order\_Type / nocum;  
    run;
- c. proc freq data=orion.qtr1\_2007  
    nlevels;  
    tables Order\_Type / noprint;  
    run;
- d. proc freq data=otion.qtr1\_2007  
    nlevels;  
    tables Order\_Type nocum;  
    run;

| Number of Variable Levels |            |        |
|---------------------------|------------|--------|
| Variable                  | Label      | Levels |
| Order_Type                | Order Type | 3      |

| Order Type |           |         |
|------------|-----------|---------|
| Order_Type | Frequency | Percent |
| 1          | 13        | 59.09   |
| 2          | 2         | 9.09    |
| 3          | 7         | 31.82   |

4. Which PROC FREQ step creates the output shown here?

a. proc freq data=orion.qtr1\_2007;  
    tables Order\_Type;  
    run;

b. **proc freq data=orion.qtr1\_2007  
    nlevels;  
    tables Order\_Type / nocum;  
    run;**

c. proc freq data=orion.qtr1\_2007  
    nlevels;  
    tables Order\_Type / noprint;  
    run;

d. proc freq data=otion.qtr1\_2007  
    nlevels;  
    tables Order\_Type nocum;  
    run;

| Number of Variable Levels |            |        |
|---------------------------|------------|--------|
| Variable                  | Label      | Levels |
| Order_Type                | Order Type | 3      |

| Order Type |           |         |
|------------|-----------|---------|
| Order_Type | Frequency | Percent |
| 1          | 13        | 59.09   |
| 2          | 2         | 9.09    |
| 3          | 7         | 31.82   |

5. This PROC MEANS step creates all of the statistics listed below.

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

- minimum and maximum
  - the total number of observations that PROC MEANS processes for each subgroup (**N Obs**)
  - mean and standard deviation
  - the number of nonmissing values (**N**)
- True
- False

5. This PROC MEANS step creates all of the statistics listed below.

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

minimum and maximum

- the total number of observations that PROC MEANS processes for each subgroup (N Obs)
- mean and standard deviation
- the number of nonmissing values (N)

True

False

6. What must be added to the PROC MEANS statement to produce this output?

The MEANS Procedure

```
proc means data=orion.customer_dim  
           _____;  
var Customer_Age;  
class Customer_Gender;  
where Customer_Country ne 'US';  
run;
```

| Analysis Variable : Customer_Age<br>Customer Age |       |      |
|--|-------|------|
| Customer Gender                                  | Range | Mean |
| F  | 54.0  | 35.1 |
| M  | 54.0  | 47.0 |

- a. nonobs
- b. range mean
- c. range mean nonobs bestw.
- d. range mean nonobs maxdec=1

6. What must be added to the PROC MEANS statement to produce this output?

The MEANS Procedure

```
proc means data=orion.customer_dim  
           _____;  
var Customer_Age;  
class Customer_Gender;  
where Customer_Country ne 'US';  
run;
```

| Analysis Variable : Customer_Age<br>Customer Age |       |      |
|--|-------|------|
| Customer Gender                                  | Range | Mean |
| F  | 54.0  | 35.1 |
| M  | 54.0  | 47.0 |

- a. nonobs
- b. range mean
- c. range mean nonobs bestw.
- d. range mean nonobs maxdec=1

7. Which option enables you to specify the number of extreme observations that are displayed by PROC UNIVARIATE?
- a. NEXTROBS=
  - b. NLEVELS
  - c. NOPRINT
  - d. \_ALL\_

7. Which option enables you to specify the number of extreme observations that are displayed by PROC UNIVARIATE?
- a. NEXTROBS=
  - b. NLEVELS
  - c. NOPRINT
  - d. \_ALL\_

# STAT604 SAS Lesson 16

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Producing Descriptive Statistics

The FREQ Procedure – Prep Guide Chapter 15

# Business Scenario

A new data set, `orion.nonsales2`, must be validated. It contains information about non-sales employees and might include invalid and missing values.

Partial `orion.nonsales2`

| Employee_ID | First   | Last       | Gender | Salary | Job_Title         | Country |
|-------------|---------|------------|--------|--------|-------------------|---------|
| 120101      | Patrick | Lu         | M      | 163040 | Director          | AU      |
| 120104      | Kareen  | Billington | F      | 46230  | Admin Mgr         | au      |
| 120105      | Liz     | Povey      | F      | 27110  | Secretary I       | AU      |
| 120106      | John    | Hornsey    | M      | .      | Office Asst II    | AU      |
| 120107      | Sherie  | Sheedy     | F      | 30475  | Office Asst II    | AU      |
| 120108      | Gladys  | Gromek     | F      | 27660  | Warehouse Asst II | AU      |

# Considerations

Use the FREQ procedure to screen for invalid, missing, and duplicate data values.

Requirements of non-sales employee data:

- **Employee\_ID** values must be unique and not missing.
- **Gender** must be *F* or *M*.
- **Job\_Title** must not be missing.
- **Country** must have a value of *AU* or *US*.
- **Salary** values must be in the numeric range of 24000 to 500000.

# Short Answer Poll

What problems exist with the data in this partial data set?

| Employee_ID | First     | Last       | Gender | Salary | Job_Title               | Country |
|-------------|-----------|------------|--------|--------|-------------------------|---------|
| 120101      | Patrick   | Lu         | M      | 163040 | Director                | AU      |
| 120104      | Kareen    | Billington | F      | 46230  | Administration Manager  | au      |
| 120105      | Liz       | Povey      | F      | 27110  | Secretary I             | AU      |
| 120106      | John      | Hornsey    | M      |        | Office Assistant II     | AU      |
| 120107      | Sherie    | Sheedy     | F      | 30475  | Office Assistant III    | AU      |
| 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II  | AU      |
| 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I   | AU      |
| 120110      | Dennis    | Entwistle  | M      | 28615  | Warehouse Assistant III | AU      |
| 120111      | Ubaldo    | Spillane   | M      | 26895  | Security Guard II       | AU      |
| 120112      | Ellis     | Glattback  | F      | 26550  |                         | AU      |
| 120113      | Riu       | Horsey     | F      | 26870  | Security Guard II       | AU      |
| 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager        | AU      |
| 120115      | Hugh      | Nichollas  | M      | 2650   | Service Assistant I     | AU      |
|             | Austen    | Ralston    | M      | 29250  | Service Assistant II    | AU      |
| 120117      | Bill      | McCleary   | M      | 31670  | Cabinet Maker III       | AU      |
| 120118      | Darshi    | Hartshorn  | M      | 28090  | Cabinet Maker II        | AU      |

Hint: There are seven data problems.

# Short Answer Poll – Correct Answer

What problems exist with the data in this partial data set?

| Employee_ID | First     | Last       | Gender | Salary | Job_Title               | Country |
|-------------|-----------|------------|--------|--------|-------------------------|---------|
| 120101      | Patrick   | Lu         | M      | 163040 | Director                | AU      |
| 120104      | Kareen    | Billington | F      | 46230  | Administration Manager  | au      |
| 120105      | Liz       | Povey      | F      | 27110  | Secretary I             | AU      |
| 120106      | John      | Hornsey    | M      |        | Office Assistant II     | AU      |
| 120107      | Sherie    | Sheedy     | F      | 30475  | Office Assistant III    | AU      |
| 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II  | AU      |
| 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I   | AU      |
| 120110      | Dennis    | Entwistle  | M      | 28615  | Warehouse Assistant III | AU      |
| 120111      | Ubaldo    | Spillane   | M      | 26895  | Security Guard II       | AU      |
| 120112      | Ellis     | Glattback  | F      | 26550  |                         | AU      |
| 120113      | Riu       | Horsey     | F      | 26870  | Security Guard II       | AU      |
| 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager        | AU      |
| 120115      | Hugh      | Nichollas  | M      | 2650   | Service Assistant I     | AU      |
|             | Austen    | Ralston    | M      | 29250  | Service Assistant II    | AU      |
| 120117      | Bill      | McCleary   | M      | 31670  | Cabinet Maker III       | AU      |
| 120118      | Darshi    | Hartshorn  | M      | 28090  | Cabinet Maker II        | AU      |

Hint: There are seven data problems.

# FREQ Procedure for Data Validation

The FREQ procedure lists all discrete values for a variable and reports missing values.

```
proc freq data=orion.nonsales2;
  tables Gender Country / nocum nopercent;
run;
```

# Viewing the Output

## PROC FREQ Output

| The FREQ Procedure    |           |
|-----------------------|-----------|
| Gender                | Frequency |
| F                     | 110       |
| G                     | 1         |
| M                     | 123       |
| Frequency Missing = 1 |           |
| Country               |           |
| Frequency             |           |
| AU                    | 33        |
| US                    | 196       |
| au                    | 3         |
| us                    | 3         |

# NLEVELS Option

The *NLEVELS option* displays a table that provides the number of distinct values for each analysis variable.

```
proc freq data=orion.nonsales2 nlevels;
  tables Gender Country / nocum nopercent;
run;
```

```
PROC FREQ DATA=SAS-data-set NLEVELS;
  TABLES variable(s);
RUN;
```

# Viewing the Output

PROC FREQ Output

| Number of Variable Levels |        |                |                   |
|---------------------------|--------|----------------|-------------------|
| Variable                  | Levels | Missing Levels | Nonmissing Levels |
| Gender                    | 4      | 1              | 3                 |
| Country                   | 4      | 0              | 4                 |

| Gender                       | Frequency |
|------------------------------|-----------|
| F                            | 110       |
| G                            | 1         |
| M                            | 123       |
| <b>Frequency Missing = 1</b> |           |

| Country | Frequency |
|---------|-----------|
| AU      | 33        |
| US      | 196       |
| au      | 3         |
| us      | 3         |

# Check for Uniqueness

The values of **Employee\_ID** must be unique and not missing. PROC FREQ can be used to check for duplicate or missing values.

```
proc freq data=orion.nonsales2 order=freq;
  tables Employee_ID / nocum nopercent;
run;
```

The ORDER=FREQ option displays the results in descending frequency order.

# Viewing the Output

## Partial PROC FREQ Output

| The FREQ Procedure |           |
|--------------------|-----------|
| Employee_ID        | Frequency |
| 120108             | 2         |
| 120101             | 1         |
| 120104             | 1         |
| 120105             | 1         |
| 120106             | 1         |

|        |   |
|--------|---|
| 121134 | 1 |
| 121141 | 1 |
| 121142 | 1 |
| 121146 | 1 |
| 121147 | 1 |
| 121148 | 1 |

Frequency Missing = 1

# NLEVELS Option

NLEVELS can also be used to identify duplicates, when the number of distinct values is known.

```
proc freq data=orion.nonsales2 nlevels;
  tables Employee_ID / noprint;
run;
```

This example uses the NOPRINT option to suppress the frequency table. Only the Number of Variable Levels table is displayed.

# Viewing the Output

Partial PROC FREQ Output

Just gotta go back to the slide 10

| The FREQ Procedure        |        |                |                   |
|---------------------------|--------|----------------|-------------------|
| Number of Variable Levels |        |                |                   |
| Variable                  | Levels | Missing Levels | Nonmissing Levels |
| Employee_ID               | 234    | 1              | 233               |

There are 235 employees, but there are only 234 distinct Employee\_ID values. Therefore, there is one duplicate value and one missing value for Employee\_ID.

# NLEVELS Option

The `_ALL_` keyword with the NOPRINT option displays the number of levels for all variables without displaying frequency counts.

```
proc freq data=orion.nonsales2 nlevels;
  tables _all_ / noprint;
run;
```

# Viewing the Output

## PROC FREQ Output

| The FREQ Procedure<br>Number of Variable Levels |        |                   |                      |
|---|--------|-------------------|----------------------|
| Variable  | Levels | Missing<br>Levels | Nonmissing<br>Levels |
| Employee_ID                                     | 234    | 1                 | 233                  |
| First   | 204    | 0                 | 204                  |
| Last  | 228    | 0                 | 228                  |
| Gender  | 4      | 1                 | 3                    |
| Salary  | 230    | 1                 | 229                  |
| Job_Title                                       | 125    | 1                 | 124                  |
| Country   | 4      | 0                 | 4                    |

No frequency tables were displayed.

# Identifying Observations with Invalid Data

PROC FREQ uncovered the existence of invalid data values for **Gender**, **Country**, and **Employee\_ID**. Use PROC PRINT to display the observations with invalid values.

```
proc print data=orion.nonsales2;
  where Gender not in ('F','M') or
        Country not in ('AU','US') or
        Job_Title is null or
        Employee_ID is missing or
        Employee_ID=120108;
run;
```

# Viewing the Output

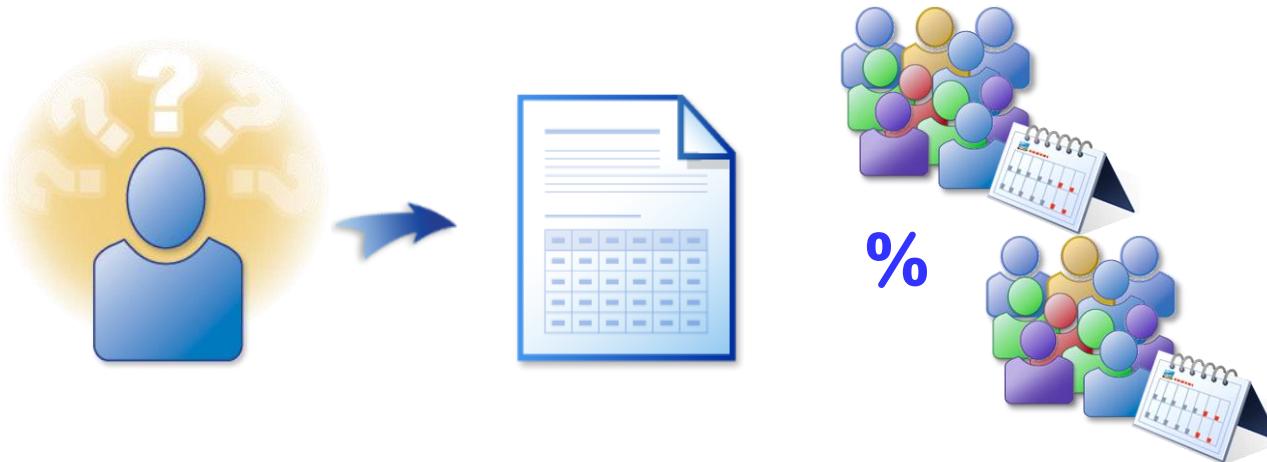
## PROC PRINT Output

| Obs | Employee_ID | First     | Last       | Gender | Salary | Job_Title                 | Country |
|-----|-------------|-----------|------------|--------|--------|---------------------------|---------|
| 2   | 120104      | Kareen    | Billington | F      | 46230  | Administration Manager    | au      |
| 6   | 120108      | Gladys    | Gromek     | F      | 27660  | Warehouse Assistant II    | AU      |
| 7   | 120108      | Gabriele  | Baker      | F      | 26495  | Warehouse Assistant I     | AU      |
| 10  | 120112      | Ellis     | Glattback  | F      | 26550  |                           | AU      |
| 12  | 120114      | Jeannette | Buddery    | G      | 31285  | Security Manager          | AU      |
| 14  | .           | Austen    | Ralston    | M      | 29250  | Service Assistant II      | AU      |
| 84  | 120695      | Trent     | Moffat     | M      | 28180  | Warehouse Assistant II    | au      |
| 87  | 120698      | Geoff     | Kistanna   | M      | 26160  | Warehouse Assistant I     | au      |
| 101 | 120723      | Deanna    | Olsen      |        | 33950  | Corp. Comm. Specialist II | US      |
| 125 | 120747      | Zashia    | Farthing   | F      | 43590  | Financial Controller I    | us      |
| 197 | 120994      | Danelle   | Sergeant   | F      | 31645  | Office Administrator I    | us      |
| 200 | 120997      | Mary      | Donathan   | F      | 27420  | Shipping Administrator I  | us      |

original  
observation  
numbers

# Business Scenario

The manager of Human Resources requested a report that shows the number and percent of sales employees who are hired each year.



# Using Formats in PROC FREQ

A FORMAT statement can be used in PROC FREQ to format data values.

```
proc freq data=orion.sales;
  tables Hire_Date / nocum;
  format Hire_Date date9.;
run;
```

Partial PROC FREQ Output

| The FREQ Procedure |           |         |
|--------------------|-----------|---------|
| Hire_Date          | Frequency | Percent |
| 01JAN1978          | 17        | 10.30   |
| 01FEB1978          | 2         | 1.21    |
| 01APR1978          | 1         | 0.61    |
| 01JUL1978          | 1         | 0.61    |
| 01AUG1978          | 1         | 0.61    |

many discrete values, and  
not what the manager  
requested

# Using Formats in PROC FREQ

A FORMAT statement can also be used in PROC FREQ to group the data.

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

Partial PROC FREQ Output

| The FREQ Procedure |           |         |
|--------------------|-----------|---------|
| Hire_Date          | Frequency | Percent |
| 1978               | 23        | 13.94   |
| 1979               | 2         | 1.21    |
| 1980               | 4         | 2.42    |
| 1981               | 3         | 1.82    |
| 1982               | 7         | 4.24    |

fewer discrete values

# Short Answer Poll

Can user-defined formats be used to group data?

\)(Y)

# Short Answer Poll – Correct Answer

Can user-defined formats be used to group data? **yes**

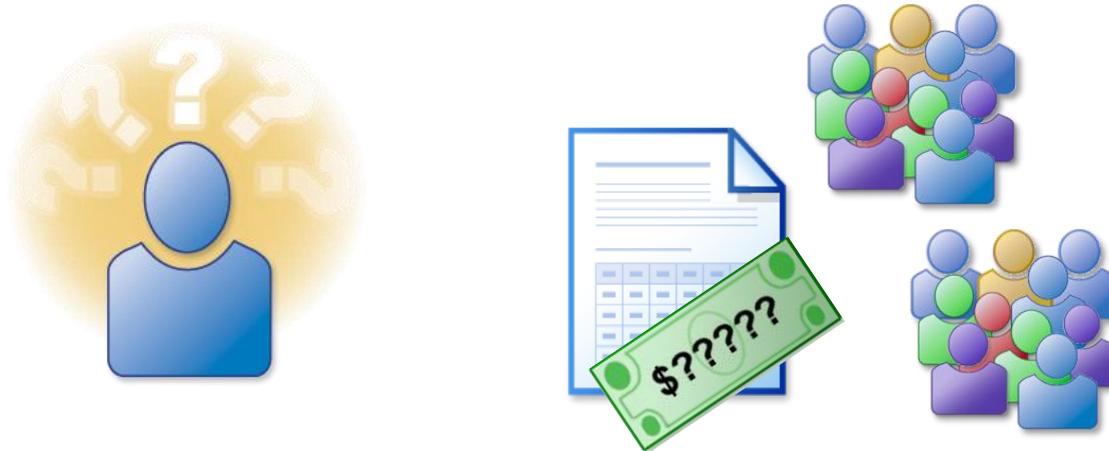
| The FREQ Procedure |           |         |                      |                    |
|--------------------|-----------|---------|----------------------|--------------------|
| Salary             | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| Tier1              | 1         | 0.61    | 1                    | 0.61               |
| Tier2              | 158       | 95.76   | 159                  | 96.36              |
| Tier3              | 4         | 2.42    | 163                  | 98.79              |
| Tier4              | 2         | 1.21    | 165                  | 100.00             |

# Producing Descriptive Statistics

The MEANS Procedure – Prep Guide Chapter 15

# Business Scenario

The payroll manager would like to see the average salary for all employees.



# MEANS Procedure

The MEANS procedure produces summary reports with descriptive statistics.

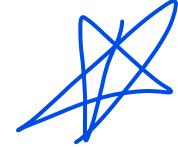
```
proc means data=orion.sales;  
run;
```

```
PROC MEANS DATA=input-data-set <options statistics>;  
    <VAR analysis-variable(s);>  
    <CLASS classification-variable(s);>  
RUN;
```

- *Analysis variables* are the **numeric** variables for which statistics are to be computed.
- *Classification variables* are variables whose values define subgroups for the analysis. They can be character or numeric.



# Viewing the Output



## PROC MEANS Output

| The MEANS Procedure |     |           |             |           |           |
|---------------------|-----|-----------|-------------|-----------|-----------|
| Variable            | N   | Mean      | Std Dev     | Minimum   | Maximum   |
| Employee_ID         | 165 | 120713.90 | 450.0866939 | 120102.00 | 121145.00 |
| Salary              | 165 | 31160.12  | 20082.67    | 22710.00  | 243190.00 |
| Birth_Date          | 165 | 3622.58   | 5456.29     | -5842.00  | 10490.00  |
| Hire_Date           | 165 | 12054.28  | 4619.94     | 5114.00   | 17167.00  |

Default statistics are displayed for all numeric variables.

# VAR Statement

The VAR statement identifies the analysis variable (or variables) and their order in the output.

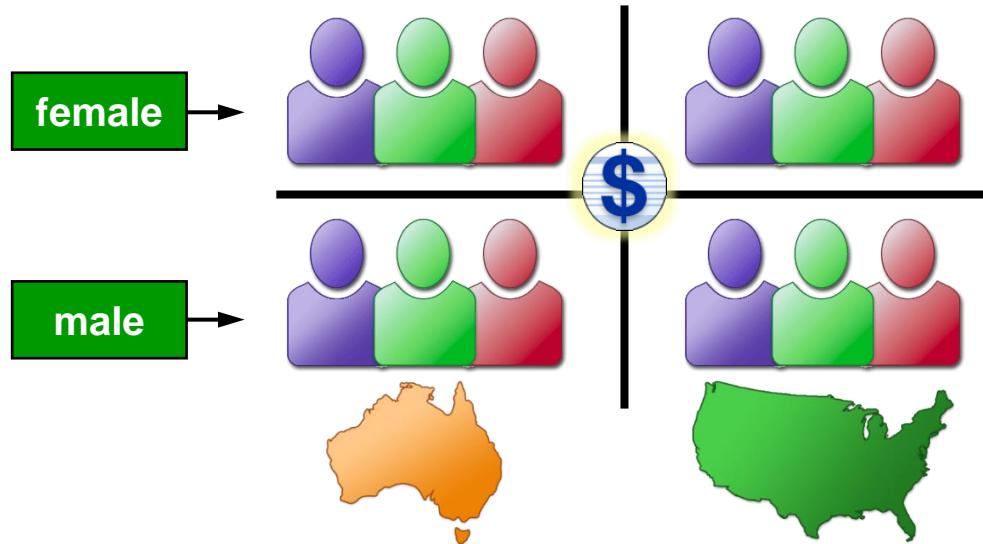
```
proc means data=orion.sales;
  var Salary;
run;
```

VAR *variable(s)*;

| The MEANS Procedure        |          |          |          |           |
|----------------------------|----------|----------|----------|-----------|
| Analysis Variable : Salary |          |          |          |           |
| N                          | Mean     | Std Dev  | Minimum  | Maximum   |
| 165                        | 31160.12 | 20082.67 | 22710.00 | 243190.00 |

# Business Scenario

Analyze Salary by Country within Gender.



# CLASS Statement

The *CLASS statement* identifies variables whose values define subgroups for the analysis.

```
proc means data=orion.sales;
  var Salary;
  class Gender Country;
run;
```

CLASS *classification-variable(s)*;

- Classification variables are character or numeric.
- They typically have few discrete values. ~~XX~~
- The data set does *not* need to be sorted or indexed by ~~XX~~  
the classification variables. ~~XX~~

# Viewing the Output

Statistics are produced for each combination of values of the classification variables.

| The MEANS Procedure        |         |       |    |          |          |          |           |  |
|----------------------------|---------|-------|----|----------|----------|----------|-----------|--|
| Analysis Variable : Salary |         |       |    |          |          |          |           |  |
| Gender                     | Country | N Obs | N  | Mean     | Std Dev  | Minimum  | Maximum   |  |
| F                          | AU      | 27    | 27 | 27702.41 | 1728.23  | 25185.00 | 30890.00  |  |
|                            | US      | 41    | 41 | 29460.98 | 8847.03  | 25390.00 | 83505.00  |  |
| M                          | AU      | 36    | 36 | 32001.39 | 16592.45 | 25745.00 | 108255.00 |  |
|                            | US      | 61    | 61 | 33336.15 | 29592.69 | 22710.00 | 243190.00 |  |

- *N Obs* – the number of observations with each unique combination of **class** variables
- *N* – the number of observations with nonmissing values of the analysis variable (or variables)

# Short Answer Poll

For a given data set, there are 63 observations with a **Country** value of AU. Of those 63 observations, only 61 observations have a value for **Salary**.

Which output is correct?

a.

| Analysis Variable : Salary |          |    |
|----------------------------|----------|----|
| Country                    | N<br>Obs | N  |
| AU                         | 63       | 61 |

b.

| Analysis Variable : Salary |          |    |
|----------------------------|----------|----|
| Country                    | N<br>Obs | N  |
| AU                         | 61       | 63 |

# Short Answer Poll – Correct Answer

For a given data set, there are 63 observations with a **Country** value of AU. Of those 63 observations, only 61 observations have a value for **Salary**.

Which output is correct?

a.

Analysis Variable : Salary

| Country | N<br>Obs | N  |
|---------|----------|----|
| AU      | 63       | 61 |

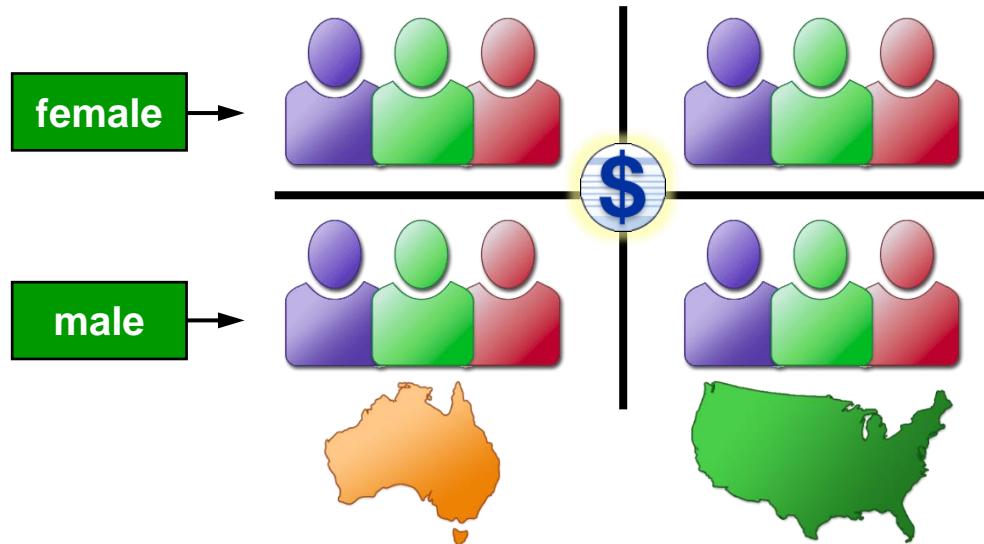
b.

Analysis Variable : Salary

| Country | N<br>Obs | N  |
|---------|----------|----|
| AU      | 61       | 63 |

# Business Scenario

Analyze Salary by Country within Gender. Generate a report that includes the number of missing Salary values, as well as the minimum, maximum, and sum of salaries.



# PROC MEANS Statistics

Use options in the PROC MEANS statement to request specific statistics.

```
proc means data=orion.sales nmiss min max sum;  
  var Salary;  
  class Gender Country;  
run;
```

oneline  
Specify! must specify  
all the over re word

The requested statistics override the default statistics.

# PROC MEANS Statistics

The statistics are displayed in the order in which they are requested.

| The MEANS Procedure        |         |       |        |          |           |            |
|----------------------------|---------|-------|--------|----------|-----------|------------|
| Analysis Variable : Salary |         |       |        |          |           |            |
| Gender                     | Country | N Obs | N Miss | Minimum  | Maximum   | Sum        |
| F                          | AU      | 27    | 0      | 25185.00 | 30890.00  | 747965.00  |
|                            | US      | 41    | 0      | 25390.00 | 83505.00  | 1207900.00 |
| M                          | AU      | 36    | 0      | 25745.00 | 108255.00 | 1152050.00 |
|                            | US      | 61    | 0      | 22710.00 | 243190.00 | 2033505.00 |

mode not available

in R.  
~~not~~

# Other PROC MEANS Statistics

| Descriptive Statistic Keywords |        |          |        |        |
|--------------------------------|--------|----------|--------|--------|
| CLM                            | CSS    | CV       | LCLM   | MAX    |
| MEAN                           | MIN    | MODE     | N      | NMISS  |
| KURTOSIS                       | RANGE  | SKEWNESS | STDDEV | STDERR |
| SUM                            | SUMWGT | UCLM     | USS    | VAR    |

| Quantile Statistic Keywords |     |     |     |          |
|-----------------------------|-----|-----|-----|----------|
| MEDIAN   P50                | P1  | P5  | P10 | Q1   P25 |
| Q3   P75                    | P90 | P95 | P99 | QRANGE   |

| Hypothesis Testing Keywords |   |  |  |  |
|-----------------------------|---|--|--|--|
| PROBT                       | T |  |  |  |

# PROC MEANS Statement Options

Options can also be placed in the PROC MEANS statement.

| Option  | Description  |
|---------|--|
| MAXDEC= | Specifies the number of decimal places to display. |
| NONOBS  | Suppresses the N Obs column.                       |

# MAXDEC= Option

The MEANS Procedure

MAXDEC=0

| Analysis Variable : Salary |       |     |       |         |         |         |
|----------------------------|-------|-----|-------|---------|---------|---------|
| Country                    | N Obs | N   | Mean  | Std Dev | Minimum | Maximum |
| AU                         | 63    | 63  | 30159 | 12699   | 25185   | 108255  |
| US                         | 102   | 102 | 31778 | 23556   | 22710   | 243190  |

The MEANS Procedure

MAXDEC=1

| Analysis Variable : Salary |       |     |         |         |         |          |
|----------------------------|-------|-----|---------|---------|---------|----------|
| Country                    | N Obs | N   | Mean    | Std Dev | Minimum | Maximum  |
| AU                         | 63    | 63  | 30159.0 | 12699.1 | 25185.0 | 108255.0 |
| US                         | 102   | 102 | 31778.5 | 23555.8 | 22710.0 | 243190.0 |

# NONOBS Option

N Obs included by default

| Analysis Variable : Salary |       |     |          |          |          |           |
|----------------------------|-------|-----|----------|----------|----------|-----------|
| Country                    | N Obs | N   | Mean     | Std Dev  | Minimum  | Maximum   |
| AU                         | 63    | 63  | 30158.97 | 12699.14 | 25185.00 | 108255.00 |
| US                         | 102   | 102 | 31778.48 | 23555.84 | 22710.00 | 243190.00 |

NONOBS option

| Analysis Variable : Salary |     |          |          |          |           |
|----------------------------|-----|----------|----------|----------|-----------|
| Country                    | N   | Mean     | Std Dev  | Minimum  | Maximum   |
| AU                         | 63  | 30158.97 | 12699.14 | 25185.00 | 108255.00 |
| US                         | 102 | 31778.48 | 23555.84 | 22710.00 | 243190.00 |

# BY Statement

The BY statement is used to request separate analyses (tables) for each BY group.

```
proc means data=sorted;
  var salary;
  class Gender;
  by Country;
run;
```

The data set must be sorted or indexed by the variable (or variables) named in the BY statement.

# BY Statement

Country=AU

| Analysis Variable : Salary |       |    |          |          |          |           |
|----------------------------|-------|----|----------|----------|----------|-----------|
| Gender                     | N Obs | N  | Mean     | Std Dev  | Minimum  | Maximum   |
| F                          | 27    | 27 | 27702.41 | 1728.23  | 25185.00 | 30890.00  |
| M                          | 36    | 36 | 32001.39 | 16592.45 | 25745.00 | 108255.00 |

Country=US

| Analysis Variable : Salary |       |    |          |          |          |           |
|----------------------------|-------|----|----------|----------|----------|-----------|
| Gender                     | N Obs | N  | Mean     | Std Dev  | Minimum  | Maximum   |
| F                          | 41    | 41 | 29460.98 | 8847.03  | 25390.00 | 83505.00  |
| M                          | 61    | 61 | 33336.15 | 29592.69 | 22710.00 | 243190.00 |

# Output Data Sets

PROC MEANS produces output data sets using the following method:

```
OUTPUT OUT=SASdataset <options>;
```

The output data set contains the following variables:

- BY variables
- class variables
- the automatic variables **\_TYPE\_** and **\_FREQ\_**
- the variables requested in the OUTPUT statement or **\_STAT\_** if you do not specify variables for statistics

# OUTPUT Statement OUT= Option

The statistics in the PROC statement impact only the MEANS report, not the data set.

```
proc means data=orion.sales sum mean range;
  var Salary;
  class Gender Country;
  output out=work.means1;
run;

proc print data=work.means1;
run;
```

# OUTPUT Statement OUT= Option

Partial PROC PRINT Output

| Obs | Gender | Country | _TYPE_ | _FREQ_ | _STAT_ | Salary        |
|-----|--------|---------|--------|--------|--------|---------------|
| 1   |        |         |        | 0      | 165    | N 165.00      |
| 2   |        |         |        | 0      | 165    | MIN 22710.00  |
| 3   |        |         |        | 0      | 165    | MAX 243190.00 |
| 4   |        |         |        | 0      | 165    | MEAN 31160.12 |
| 5   |        |         |        | 0      | 165    | STD 20082.67  |
| 6   |        | AU      |        | 1      | 63     | N 63.00       |
| 7   |        |         |        |        |        | MIN 25185.00  |
| 8   |        |         |        |        |        | MAX 108255.00 |
| 9   |        | AU      |        | 1      | 63     | MEAN 30158.97 |
| 10  |        | AU      |        | 1      | 63     | STD 12699.14  |
| 11  |        | US      |        | 1      | 102    | N 102.00      |
| 12  |        | US      |        | 1      | 102    | MIN 22710.00  |
| 13  |        | US      |        | 1      | 102    | MAX 243190.00 |
| 14  |        | US      |        | 1      | 102    | MEAN 31778.48 |
| 15  |        | US      |        | 1      | 102    | STD 23555.84  |
| 16  | F      |         |        | 2      | 68     | N 68.00       |
| 17  | F      |         |        | 2      | 68     | MIN 25185.00  |
| 18  | F      |         |        | 2      | 68     | MAX 83505.00  |
| 19  | F      |         |        | 2      | 68     | MEAN 28762.72 |
| 20  | F      |         |        | 2      | 68     | STD 6974.15   |

# OUTPUT Statement OUT= Option

The OUTPUT statement can also do the following:

- specify the statistics for the output data set
- select and name variables

```
proc means data=orion.sales noprint;
  var Salary;
  class Gender Country;
  output out=work.means2
    min=minSalary max=maxSalary
    sum=sumSalary mean=aveSalary;
run;

proc print data=work.means2;
run;
```

The NOPRINT option suppresses the display of all output.

# OUTPUT Statement OUT= Option

PROC PRINT Output

| Obs | Gender | Country | _TYPE_ | _FREQ_ | min<br>Salary | max<br>Salary | sum<br>Salary | ave<br>Salary |
|-----|--------|---------|--------|--------|---------------|---------------|---------------|---------------|
| 1   |        |         |        | 0      | 165           | 22710         | 243190        | 5141420       |
| 2   |        | AU      |        | 1      | 63            | 25185         | 108255        | 1900015       |
| 3   |        | US      |        | 1      | 102           | 22710         | 243190        | 3241405       |
| 4   | F      |         |        | 2      | 68            | 25185         | 83505         | 1955865       |
| 5   | M      |         |        | 2      | 97            | 22710         | 243190        | 3185555       |
| 6   | F      | AU      |        | 3      | 27            | 25185         | 30890         | 747965        |
| 7   | F      | US      |        | 3      | 41            | 25390         | 83505         | 1207900       |
| 8   | M      | AU      |        | 3      | 36            | 25745         | 108255        | 1152050       |
| 9   | M      | US      |        | 3      | 61            | 22710         | 243190        | 2033505       |
|     |        |         |        |        |               |               |               | 33336.15      |

# OUTPUT Statement OUT= Option

\_TYPE\_ is a numeric variable that shows which combination of class variables produced the summary statistics in that observation.

PROC PRINT Output

| Obs                    | Gender | Country | _TYPE_ | min | max   | sum     | ave      |
|------------------------|--------|---------|--------|-----|-------|---------|----------|
| <b>overall summary</b> |        |         |        |     |       |         |          |
| 1                      |        |         | 0      | 165 | 22710 | 5141400 | 21160.10 |
| 2                      |        | AU      | 1      | 162 | 22710 | 243190  | 8244400  |
| 3                      |        | US      | 1      | 162 | 22710 | 243190  | 81778.10 |
| 4                      | F      |         | 2      | 27  | 25185 | 30890   | 27702.41 |
| 5                      | M      |         | 2      | 27  | 25185 | 30890   | 27702.41 |
| 6                      | F      | AU      | 3      | 27  | 25185 | 30890   | 27702.41 |
| 7                      | F      | US      | 3      | 27  | 25185 | 30890   | 27702.41 |
| 8                      | M      | AU      | 3      | 61  | 22710 | 243190  | 2033505  |
| 9                      | M      | US      | 3      | 61  | 22710 | 243190  | 33336.15 |

# OUTPUT Statement OUT= Option

| Obs | Gender | Country | _TYPE_ | _FREQ_ | min Salary | max Salary | sum Salary | ave Salary |
|-----|--------|---------|--------|--------|------------|------------|------------|------------|
| 1   |        |         | 0      | 165    | 22710      | 243190     | 5141420    | 31160.12   |
| 2   |        | AU      | 1      | 63     | 25185      | 108255     | 1900015    | 30158.97   |
| 3   |        | US      | 1      | 102    | 22710      | 243190     | 3241405    | 31778.48   |
| 4   | F      |         | 2      | 68     | 25185      | 83505      | 1955865    | 28762.72   |
| 5   | M      |         | 2      | 97     | 22710      | 243190     | 3185555    | 32840.77   |
| 6   | F      | AU      | 3      | 27     | 25185      | 30890      | 747965     | 27702.41   |
| 7   | F      | US      | 3      | 41     | 25390      | 83505      | 1207900    | 29460.98   |
| 8   | M      | AU      | 3      | 36     | 25745      | 108255     | 1152050    | 32001.39   |
| 9   | M      | US      | 3      | 61     | 22710      | 243190     | 2033505    | 33336.15   |

| _TYPE_ | Type of Summary                             | _FREQ_                                      |
|--------|---|---|
| 0      | overall summary                             | 165   |
| 1      | summary by <b>Country</b> only              | 63 AU + 102 US = 165                        |
| 2      | summary by <b>Gender</b> only               | 68 F + 97 M = 165                           |
| 3      | summary by <b>Country</b> and <b>Gender</b> | 27 F AU + 41 F US + 36 M AU + 61 M US = 165 |

# Idea Exchange

Which PROC MEANS statistics would you request when you are validating numeric variables?

min / max

(only doesn't help w) validating.



# Producing Descriptive Statistics

The UNIVARIATE Procedure

↳ use for extreme values.

# Business Scenario

Validate salary data in `orion.nonsales2`. Salary must be in the numeric range of 24000 to 500000.

Partial `orion.nonsales2`

| Employee_ID | First   | Last       | Gender | Salary | Job_Title         | Country |
|-------------|---------|------------|--------|--------|-------------------|---------|
| 120101      | Patrick | Lu         | M      | 163040 | Director          | AU      |
| 120104      | Kareen  | Billington | F      | 46230  | Admin Mgr         | au      |
| 120105      | Liz     | Povey      | F      | 27110  | Secretary I       | AU      |
| 120106      | John    | Hornsey    | M      | .      | Office Asst II    | AU      |
| 120107      | Sherie  | Sheedy     | F      | 30475  | Office Asst II    | AU      |
| 120108      | Gladys  | Gromek     | F      | 27660  | Warehouse Asst II | AU      |

# UNIVARIATE Procedure

*PROC UNIVARIATE* displays extreme observations, missing values, and other statistics for the variables included in the VAR statement.

```
proc univariate data=orion.nonsales2;  
    var Salary;  
run;
```

```
PROC UNIVARIATE DATA=SAS-data-set;  
    <VAR variable(s);>  
RUN;
```

If the VAR statement is omitted, PROC UNIVARIATE analyzes all numeric variables in the data set.

# Viewing the Output: Extreme Observations

The *Extreme Observations* section includes the five lowest and five highest values for the analysis variable and the corresponding observation numbers.

Partial PROC UNIVARIATE Output

*observation #*

| Extreme Observations |     |                   |     |
|----------------------|-----|-------------------|-----|
| -----Lowest-----     |     | -----Highest----- |     |
| Value                | Obs | Value             | Obs |
| 2401                 | 20  | 163040            | 1   |
| 2650                 | 13  | 194885            | 231 |
| 24025                | 25  | 207885            | 28  |
| 24100                | 19  | 268455            | 29  |
| 24390                | 228 | 433800            | 27  |

-  Obs is the observation number, not the count of observations with that value.

# NEXTROBS= Option

The *NEXTROBS= option* specifies the number of extreme observations to be displayed.

```
proc univariate data=orion.nonsales2  
    nextrobs=3;  
    var Salary;  
run;
```

Partial PROC UNIVARIATE Output

| The UNIVARIATE Procedure |     |                   |     |
|--------------------------|-----|-------------------|-----|
| Variable: Salary         |     |                   |     |
| Extreme Observations     |     |                   |     |
| -----Lowest-----         |     | -----Highest----- |     |
| Value                    | Obs | Value             | Obs |
| 2401                     | 20  | 207885            | 28  |
| 2650                     | 13  | 268455            | 29  |
| 24025                    | 25  | 433800            | 27  |

# ID Statement

The *ID statement* displays the value of the identifying variable (or variables) in addition to the observation number.

```
proc univariate data=orion.nonsales2;
  var Salary;
  id Employee_ID;
run;
```

ID variable(s);

# Viewing the Output

Partial PROC UNIVARIATE Output

| The UNIVARIATE Procedure |             |     |                   |             |     |
|--------------------------|-------------|-----|-------------------|-------------|-----|
| Variable: Salary         |             |     |                   |             |     |
| Extreme Observations     |             |     |                   |             |     |
| -----Lowest-----         |             |     | -----Highest----- |             |     |
| Value                    | Employee_ID | Obs | Value             | Employee_ID | Obs |
| 2401                     | 120191      | 20  | 163040            | 120101      | 1   |
| 2650                     | 120115      | 13  | 194885            | 121141      | 231 |
| 24025                    | 120196      | 25  | 207885            | 120260      | 28  |
| 24100                    | 120190      | 19  | 268455            | 120262      | 29  |
| 24390                    | 121132      | 228 | 433800            | 120259      | 27  |



# Viewing the Output: Missing Values Section

The *Missing Values* section displays the number and percentage of observations with missing values for the analysis variable.

Partial PROC UNIVARIATE Output

| Missing Values |       |                      |             |
|----------------|-------|----------------------|-------------|
| Missing Value  | Count | -----Percent Of----- |             |
|                |       | All Obs              | Missing Obs |
| .              | 1     | 0.43                 | 100.00      |

# Short Answer Poll

PROC UNIVARIATE identified two observations  
with **Salary** values less than 24,000.

What procedure can be used to display the observations that contain the invalid values?

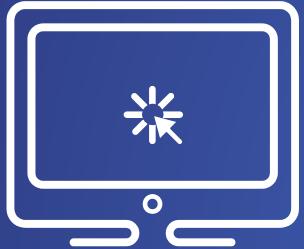
# Short Answer Poll – Correct Answer

PROC UNIVARIATE identified two observations with **Salary** values less than 24,000.

What procedure can be used to display the observations that contain the invalid values? **PROC PRINT**

```
proc print data=orion.nonsales2;
  where Salary<24000;
run;
```

| Obs | Employee_ID | First   | Last        | Gender | Salary | Job_Title           | Country |
|-----|-------------|---------|-------------|--------|--------|---------------------|---------|
| 4   | 120106      | John    | Hornsey     | M      | .      | Office Assistant II | AU      |
| 13  | 120115      | Hugh    | Nichollas   | M      | 2650   | Service Assistant I | AU      |
| 20  | 120191      | Jannene | Graham-Rowe | F      | 2401   | Trainee             | AU      |



# Analyzing data with PROC UNIVARIATE

This demonstration shows the full output produced by the UNIVARIATE procedure.

# Producing Descriptive Statistics

The TABULATE Procedure

# The TABULATE Procedure

The TABULATE procedure displays descriptive statistics in tabular format.

General form of the TABULATE procedure:

```
PROC TABULATE DATA=SASdataset <options>;
  CLASS classificationvariable(s);
  VAR analysis-variable(s);
  TABLE      pageexpression,
             rowexpression,
             columnexpression </ option(s)>;
RUN;
```

*None of these are optional*

# Dimensional Tables



The TABULATE procedure produces one-, two-, or three-dimensional tables.

|                   | page dimension | row dimension | column dimension |
|-------------------|----------------|---------------|------------------|
| one-dimensional   |                |               | ✓                |
| two-dimensional   |                | ✓             | ✓                |
| three-dimensional | ✓              | ✓             | ✓                |

# One-Dimensional Table

| Country |     |
|---------|-----|
| AU      | US  |
| N       | N   |
| 63      | 102 |

- **Country** is in the column dimension.

# Two-Dimensional Table

|        |   | Country |    |
|--------|---|---------|----|
|        |   | AU      | US |
|        |   | N       | N  |
| Gender |   |         |    |
|        | F | 27      | 41 |
| M      |   | 36      | 61 |

- **Country** is in the column dimension.
- **Gender** is in the row dimension.

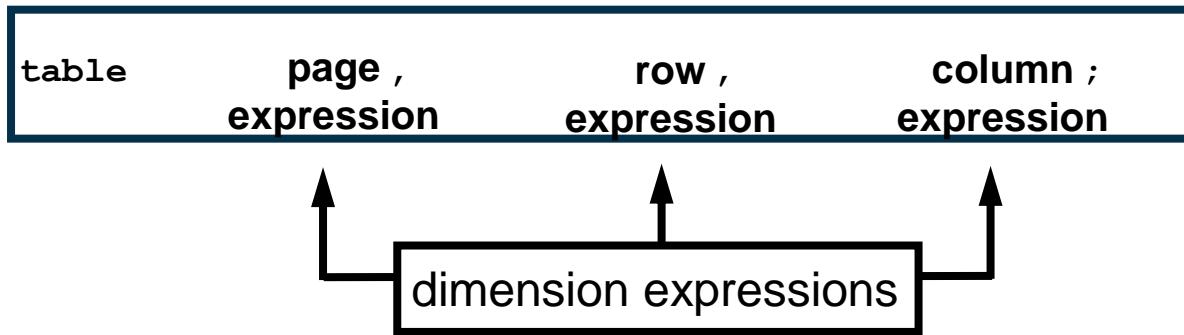
# Three-Dimensional Table

| Job_Title Sales Rep. I |   | Country |    |
|------------------------|---|---------|----|
|                        |   | AU      | US |
|                        |   | N       | N  |
| Gender                 |   |         |    |
|                        | F | 8       | 13 |
| M                      |   | 13      | 29 |

- **Country** is in the column dimension.
- **Gender** is in the row dimension.
- **Job\_Title** is in the page dimension.

# The TABLE Statement

The TABLE statement describes the structure of the table.



- Commas separate the dimension expressions.
- Every variable that is part of a dimension expression must be specified as a classification variable (CLASS statement) or an analysis variable (VAR statement).

# The TABLE Statement

PROC TABULATE

age        w        column

# The TABLE Statement

|       |                      |                     |                        |
|-------|----------------------|---------------------|------------------------|
| table | page ,<br>expression | row ,<br>expression | column ;<br>expression |
|-------|----------------------|---------------------|------------------------|

Examples:

```
table Country;
```

```
table Gender , Country;
```

```
table Job_Title , Gender , Country;
```

# The CLASS Statement

The CLASS statement identifies variables to be used as classification, or grouping, variables.

General form of the CLASS statement:

```
CLASS classificationvariable(s);
```

- N, the number of nonmissing values, is the default statistic for classification variables.
- Examples of classification variables:

**Job\_Title, Gender, and Country**

# The VAR Statement

The VAR statement identifies the numeric variables for which statistics are calculated.

General form of the VAR statement:

```
VAR analysisvariable(s);
```

- SUM is the default statistic for analysis variables.
- Examples of analysis variables:

**Salary** and **Bonus**

# One-Dimensional Table

```
proc tabulate data=orion.sales;
  class Country;
  table Country;
run;
```

| Country |     |
|---------|-----|
| AU      | US  |
| N       | N   |
| 63      | 102 |

# Two-Dimensional Table

```
proc tabulate data=orion.sales;
  class Gender Country;
  table Gender, Country;
run;
```

|        |  | Country |    |
|--------|--|---------|----|
|        |  | AU      | US |
|        |  | N       | N  |
| Gender |  |         |    |
| F      |  | 27      | 41 |
| M      |  | 36      | 61 |

# Three-Dimensional Table

```
proc tabulate data=orion.sales;
  class Job_Title Gender Country;
  table Job_Title, Gender, Country;
run;
```

| Job_Title Sales Rep. I |    |         |
|------------------------|----|---------|
|                        |    | Country |
|                        |    | AU US   |
|                        |    | N N     |
| Gender                 |    |         |
| F                      | 8  | 13      |
| M                      | 13 | 29      |

| Job_Title Sales Rep. II |    |         |
|-------------------------|----|---------|
|                         |    | Country |
|                         |    | AU US   |
|                         |    | N N     |
| Gender                  |    |         |
| F                       | 10 | 14      |
| M                       | 8  | 14      |

| Job_Title Sales Rep. III |    |         |
|--------------------------|----|---------|
|                          |    | Country |
|                          |    | AU US   |
|                          |    | N N     |
| Gender                   |    |         |
| F                        | 7  | 8       |
| M                        | 10 | 9       |

# Dimension Expression

Elements that can be used in a dimension expression:

- classification variables
- analysis variables
- the universal class variable ALL
- keywords for statistics

Operators that can be used in a dimension expression:

- blank, which concatenates table information
- asterisk \*, which crosses table information
- parentheses (), which group elements

# Dimension Expression

```
proc tabulate data=orion.sales;
  class Gender Country;
  var Salary;
  table Gender all, Country*Salary;
run;
```

|        |  | Country    |            |
|--------|--|------------|------------|
|        |  | AU         | US         |
|        |  | Salary     | Salary     |
|        |  | Sum        | Sum        |
| Gender |  |            |            |
| F      |  | 747965.00  | 1207900.00 |
| M      |  | 1152050.00 | 2033505.00 |
| All    |  | 1900015.00 | 3241405.00 |

# PROC TABULATE Statistics

| Descriptive Statistic Keywords |           |            |           |           |
|--------------------------------|-----------|------------|-----------|-----------|
|                                | CSS       | CV         | LCLM      | MAX       |
| MEAN                           | MIN       | MODE       | N         | NMISS     |
| KURTOSIS                       | RANGE     | SKEWNESS   | STDDEV    | STDERR    |
| SUM                            | SUMWGT    | UCLM       | USS       | VAR       |
| PCTN                           | REPPCTN   | PAGEPCTN   | ROWPCTN   | COLPCTN   |
| PCTSUM                         | REPPCTSUM | PAGEPCTSUM | ROWPCTSUM | COLPCTSUM |
| Quantile Statistic Keywords    |           |            |           |           |
| MEDIAN   P50                   | P1        | P5         | P10       | Q1   P25  |
| Q3   P75                       | P90       | P95        | P99       | QRANGE    |
| Hypothesis Testing Keywords    |           |            |           |           |
| PROBT                          | T         |            |           |           |

# PROC TABULATE Statistics

*FORMAT \*f=dollar12. ;*

```
proc tabulate data=orion.sales;
  class Gender Country;
  var Salary;
  table Gender all, Country*Salary*(min max)*f=dollar12. ;
run;
```

|        |  | Country  |           |          |           |
|--------|--|----------|-----------|----------|-----------|
|        |  | AU       |           | US       |           |
|        |  | Salary   |           | Salary   |           |
|        |  | Min      | Max       | Min      | Max       |
| Gender |  |          |           |          |           |
| F      |  | \$25,185 | \$30,890  | \$25,390 | \$83,505  |
| M      |  | \$25,745 | \$108,255 | \$22,710 | \$243,190 |
| All    |  | \$25,185 | \$108,255 | \$22,710 | \$243,190 |

# Additional SAS Statements

Additional statements can be added to enhance the report.

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

options nodate pageno=1;

proc tabulate data=orion.sales;
  class Gender Country;
  var Salary;
  table Gender all, Country*Salary*(min max) ;
  where Job_Title contains 'Rep';
  label Salary='Annual Salary';
  format Country $ctryfmt.;
  title 'Sales Rep Tabular Report';
run;
```

# Additional SAS Statements

## HTML Output

Sales Rep Tabular Report

| Gender | Country       |          |               |          |
|--------|---------------|----------|---------------|----------|
|        | Australia     |          | United States |          |
|        | Annual Salary |          | Annual Salary |          |
|        | Min           | Max      | Min           | Max      |
| F      | 25185.00      | 30890.00 | 25390.00      | 32985.00 |
| M      | 25745.00      | 36605.00 | 22710.00      | 35990.00 |
| All    | 25185.00      | 36605.00 | 22710.00      | 35990.00 |

# Output Data Sets

PROC TABULATE produces output data sets using the following method:

```
PROC TABULATE DATA=SAS-data-set  
OUT=SAS-data-set <options>;
```

The output data set contains the following variables:

- BY variables
- class variables
- the automatic variables **\_TYPE\_, \_PAGE\_, and  
\_TABLE\_**
- calculated statistics

# PROC Statement OUT= Option

```
proc tabulate data=orion.sales  
            out=work.tabulate;  
    where Job_Title contains 'Rep';  
    class Job_Title Gender Country;  
    table Country;  
    table Gender, Country;  
    table Job_Title, Gender, Country;  
run;  
  
proc print data=work.tabulate;  
run;
```

# PROC Statement OUT= Option

Partial PROC PRINT Output

| Obs | Job_Title     | Gender | Country | _TYPE_ | _PAGE_ | _TABLE_ | N  |
|-----|---------------|--------|---------|--------|--------|---------|----|
| 1   |               |        | AU      | 001    | 1      | 1       | 61 |
| 2   |               |        | US      | 001    | 1      | 1       | 98 |
| 3   |               | F      | AU      | 011    | 1      | 2       | 27 |
| 4   |               | F      | US      | 011    | 1      | 2       | 40 |
| 5   |               | M      | AU      | 011    | 1      | 2       | 34 |
| 6   |               | M      | US      | 011    | 1      | 2       | 58 |
| 7   | Sales Rep. I  | F      | AU      | 111    | 1      | 3       | 8  |
| 8   | Sales Rep. I  | F      | US      | 111    | 1      | 3       | 13 |
| 9   | Sales Rep. I  | M      | AU      | 111    | 1      | 3       | 13 |
| 10  | Sales Rep. I  | M      | US      | 111    | 1      | 3       | 29 |
| 11  | Sales Rep. II | F      | AU      | 111    | 2      | 3       | 10 |
| 12  | Sales Rep. II | F      | US      | 111    | 2      | 3       | 14 |
| 13  | Sales Rep. II | M      | AU      | 111    | 2      | 3       | 8  |
| 14  | Sales Rep. II | M      | US      | 111    | 2      | 3       | 14 |

# PROC Statement OUT= Option

\_TYPE\_ is a character variable that shows which combination of class variables produced the summary statistics in that observation.

Partial PROC PRINT Output

| Obs | Job_Title | Gender | Country | _TYPE_ | _PAGE_ | _TABLE_ | N  |
|-----|-----------|--------|---------|--------|--------|---------|----|
| 1   |           |        | AU      | 001    | 1      | 1       | 61 |
| 2   |           |        | US      | 001    | 1      | 1       | 98 |
| 3   |           | F      | AU      | 011    | 1      | 2       | 27 |
| 4   |           | F      | US      | 011    |        |         |    |
| 5   |           | M      | AU      | 011    |        |         |    |
| 6   |           | M      | US      | 011    |        |         |    |

0 for  
**Job\_Title**,  
1 for **Gender**, and  
1 for **Country**

# PROC Statement OUT= Option

PAGE is a numeric variable that shows the logical page number that contains that observation.

Partial PROC PRINT Output

|    |                |   |    |     |   |   |    |
|----|----------------|---|----|-----|---|---|----|
| 7  | Sales Rep. I   | F | AU | 111 | 1 | 3 | 8  |
| 8  | Sales Rep. I   | F | US | 111 | 1 | 3 | 29 |
| 9  | Sales Rep. I   | M | AU | 111 | 1 | 3 | 10 |
| 10 | Sales Rep. I   | M | US | 111 | 1 | 3 | 14 |
| 11 | Sales Rep. II  | F | AU | 111 | 2 | 3 | 7  |
| 12 | Sales Rep. II  | F | US | 111 | 2 | 3 | 9  |
| 13 | Sales Rep. II  | M | AU | 111 | 2 | 3 | 10 |
| 14 | Sales Rep. II  | M | US | 111 | 2 | 3 | 14 |
| 15 | Sales Rep. III | F | AU | 111 | 3 | 3 | 8  |
| 16 | Sales Rep. III | F | US | 111 | 3 | 3 | 29 |
| 17 | Sales Rep. III | M | AU | 111 | 3 | 3 | 10 |
| 18 | Sales Rep. III | M | US | 111 | 3 | 3 | 14 |

# PROC Statement OUT= Option

\_TABLE\_ is a numeric variable that shows the number of the TABLE statement that contains that observation.

Partial PROC PRINT Output

| Obs | Job_Title    | Gender | Country | _TYPE_ | _PAGE_ | _TABLE_ | N  |
|-----|--------------|--------|---------|--------|--------|---------|----|
| 1   |              |        |         |        |        | 1       | 61 |
| 2   |              |        |         |        |        | 1       | 98 |
| 3   |              | F      | AU      | 011    | 1      | 2       | 27 |
| 4   |              |        |         |        |        | 2       | 40 |
| 5   |              |        |         |        |        | 2       | 34 |
| 6   |              | M      | US      | 011    | 1      | 2       | 58 |
| 7   | Sales Rep. I | F      | AU      | 111    | 1      | 3       | 8  |
| 8   | Sales Rep.   |        |         |        |        | 3       | 13 |
| 9   | Sales Rep.   |        |         |        |        | 3       | 13 |
| 10  | Sales Rep. I | M      | US      | 111    | 1      | 3       | 29 |

# STAT604 SAS Lesson 17

Portions Copyright © 2018 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

# Graphics Techniques

The SGPlot and SGPanel Procedures – SAS Documents

# Introduction to SAS ODS Graphics

ODS Statistical Graphics (also known as ODS Graphics) is functionality for easily creating statistical graphics. It is available in a number of SAS products, including SAS/STAT, SAS/ETS, SAS/QC, and SAS/GRAF software.

Documentation (1882 pages):

## SAS® 9.4 ODS Graphics: Procedures Guide, Sixth Edition

[http://documentation.sas.com/api/docsets/grstatproc/9.4/content/grstatproc.pdf?locale=en  
#nameddest=n1e0ztxxbnqjgn182s1te817t3i](http://documentation.sas.com/api/docsets/grstatproc/9.4/content/grstatproc.pdf?locale=en#nameddest=n1e0ztxxbnqjgn182s1te817t3i)

# SAS ODS Graphics Procedures

There are seven ODS Graphics procedures. Each has a specific purpose:

- **SGPLOT**
  - creates single-cell plots with a variety of plot and chart types and overlays.
- **SGPANEL**
  - creates classification panels for one or more classification variables. Each graph cell in the panel can contain either a simple plot or multiple, overlaid plots.

*Note:* The SGPLOT and SGPANEL procedures largely support the same types of plots and charts and have an almost identical syntax. The main distinction between the two procedures is that the SGPANEL procedure produces a panel of graphs, one for each level of a classification variable.



# SAS ODS Graphics Procedures (Self-Study)

- SGPIE (Preproduction)
  - New in SAS 9.4M6. Creates pie and donut charts.
- SGMAP
  - Provides concise syntax for creating geographical maps and overlaying other plots onto the map.
- SGSCATTER
  - Creates scatter plot panels and scatter plot matrices with optional fits and ellipses.
- SGRENDER
  - Produces graphs from graph templates that are written in the Graph Template Language. You can also render a graph from a SAS ODS Graphics Editor (SGE) file.
- SGDESIGN
  - Creates graphical output based on a graph file that has been created by using the ODS Graphics Designer application.

# List of Plots and Charts

|              |                         |                 |
|--------------|-------------------------|-----------------|
| Band plot    | Heat map                | Pie chart       |
| Bar chart    | High-Low plot           | Regression plot |
| Block plot   | Histogram               | Scatter plot    |
| Box plot     | Line chart              | Series plot     |
| Bubble plot  | Line, drop              | Spline plot     |
| Density plot | Line, parameterized     | Step plot       |
| Donut chart  | Line, reference         | Text Inset      |
| Dot plot     | Loess plot              | Text plot       |
| Ellipse plot | Needle plot             | Vector plot     |
| Fringe plot  | Penalized B-Spline plot | Waterfall chart |

Available  
in SGPLOT  
Don't  
need to know  
any more  
than this

# Producing Charts with the SGLOT Procedure

General form of the PROC SGLOT procedure:

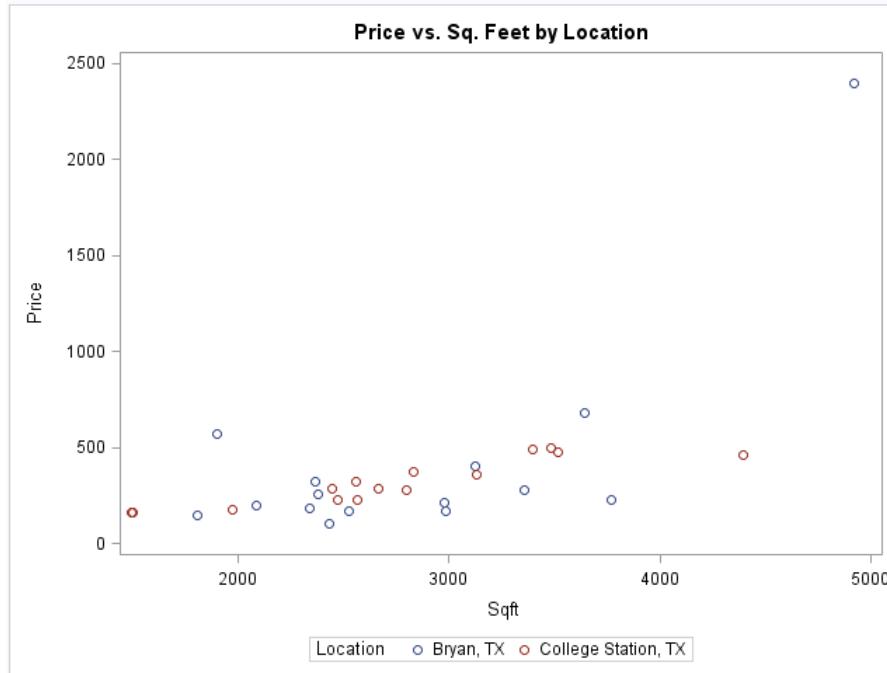
```
PROC SGLOT DATA=SAS-data-set;  
    TYPE1 chart-variable(s) . . . </ options>;  
    TYPE2 chart-variable(s) . . . </ options>;  
RUN;
```

# Producing Charts with the SGPLOT Procedure

```
title 'Price vs. Sq. Feet by Location';
proc sgplot data=bcs;
    scatter x=sqft y=price /group=location;
run;
```

# Producing Charts with the SGPLOT Procedure

Results:



# Fit Lines with the SGLOT Procedure

Example:

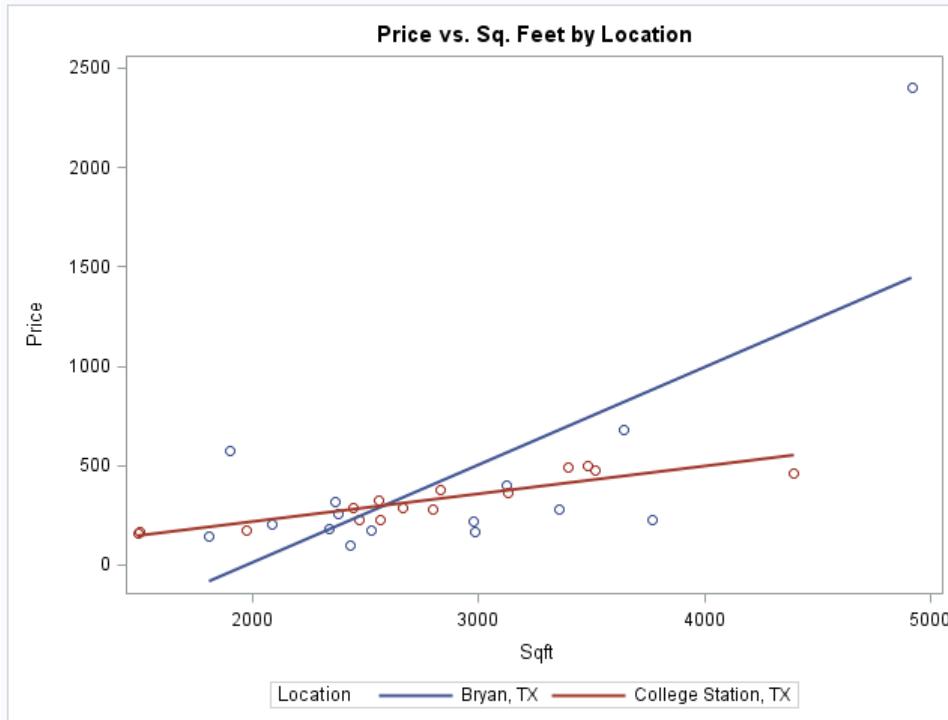
```
title 'Price vs. Sq. Feet by Location';
proc sgplot data=bcs;
    reg x=sqft y=price /group=location;
run;
```



Instead of  
SATTG, add SFIT  
one for each group

# Fit Lines with the SGPOINT Procedure

Results:



# Producing Lines with the SGLOT Procedure

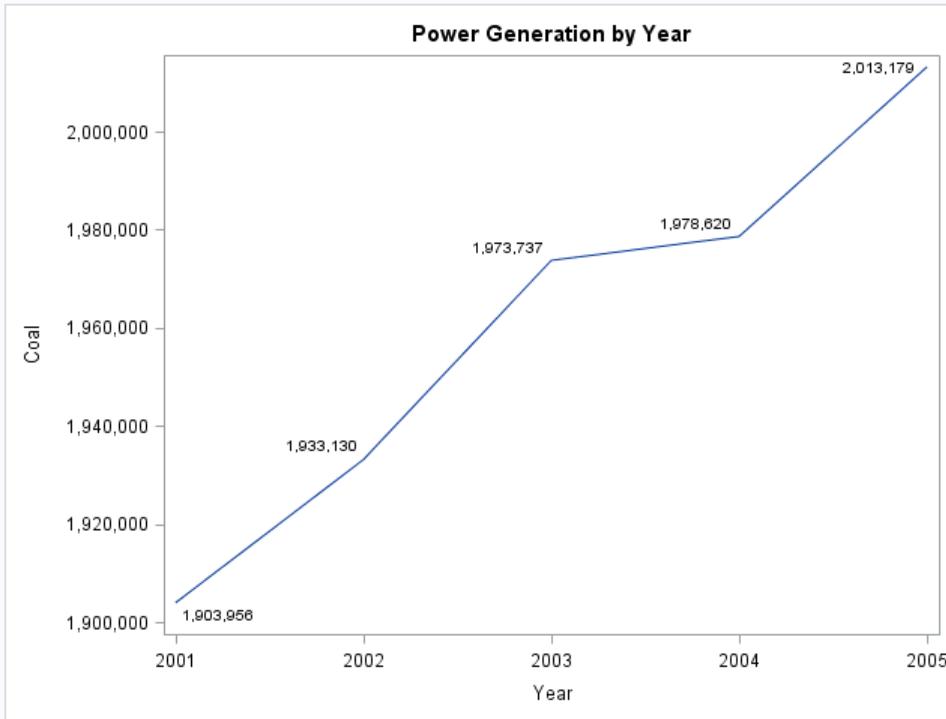
Example:

```
title 'Power Generation by Year';
proc sgplot data=sashelp.electric;
    where year >= 2001
        and customer="Residential";
    series x=year y=coal / datalabel;
run;
```

- can use where statement on proc sgplot
- series adds lines to the data plot.
- data label adds data values at point

# Producing Lines with the SGPOINT Procedure

Results:



# Bar Plots with the SGLOT Procedure

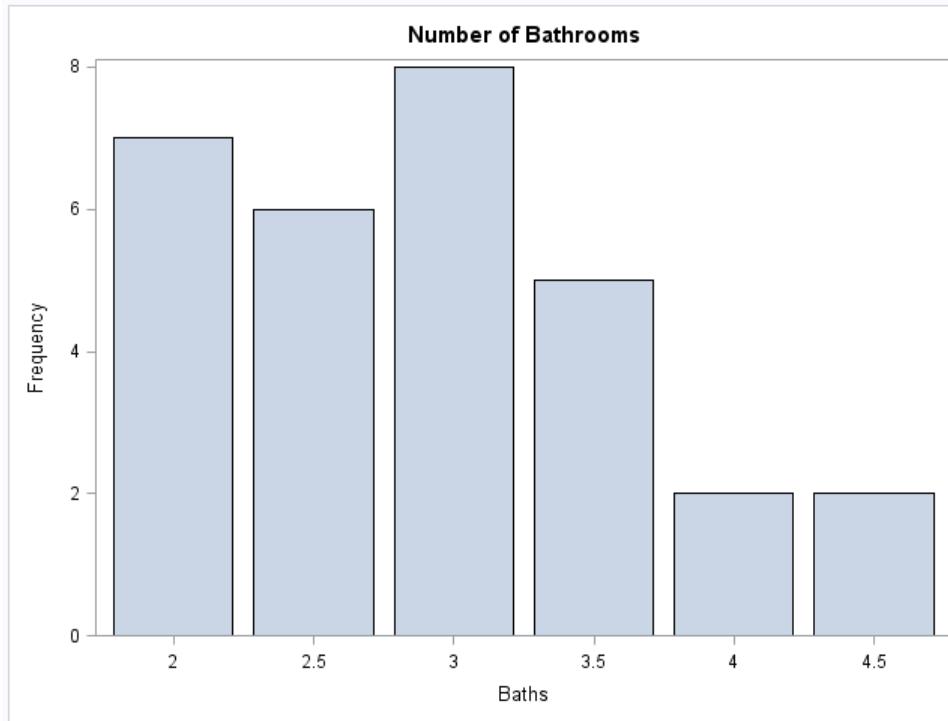
Example:

```
title 'Number of Bathrooms';  
proc sgplot data=bcs;  
    vbar baths;  
run;
```

your plot, where want  
your own.

# Bar Plots with the SGPOINT Procedure

Results:



# Histograms with the SGLOT Procedure

Example:

```
title 'Histogram of Sqft with Distribution';
proc sgplot data=bcs;
    histogram sqft / binwidth=500 ;
    density sqft;
run;
```

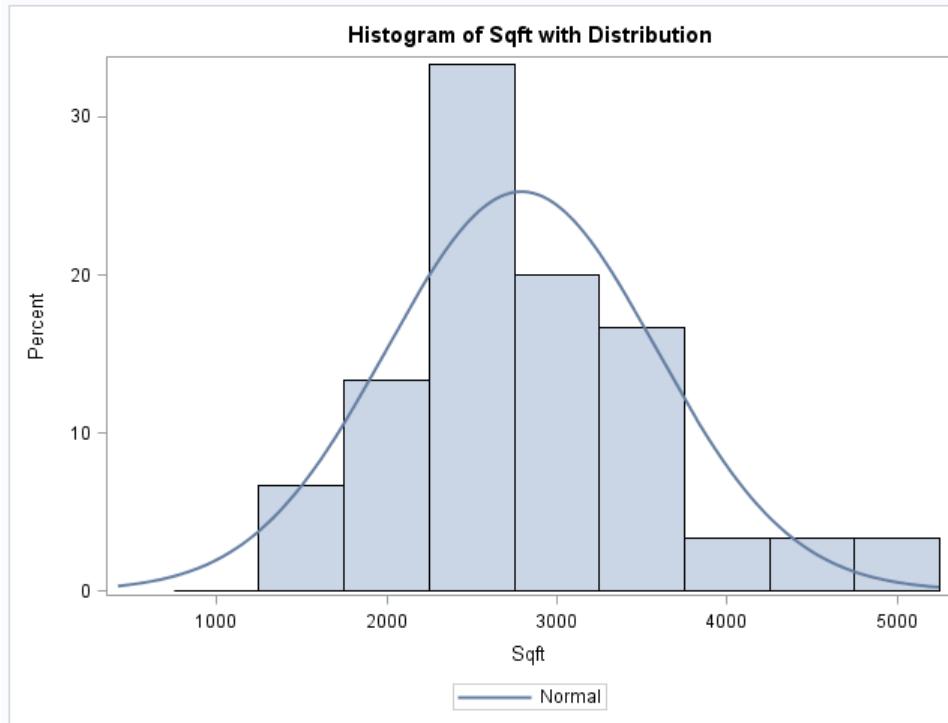
yes doing fine.

Sqft

binwidth.

# Histograms with the SGPOINT Procedure

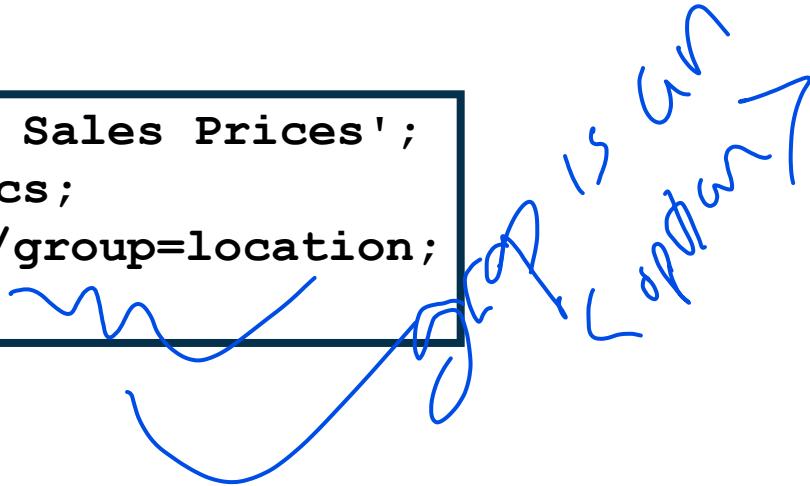
Results:



# Boxplots with the SGLOT Procedure

Example:

```
title 'Real Estate Sales Prices';  
proc sgplot data=bcs;  
    vbox price /group=location;  
run;
```



# Boxplots with the SGPOINT Procedure

Results:



# Producing Charts with the SGPROC Procedure

General form of the PROC SGPROC procedure:

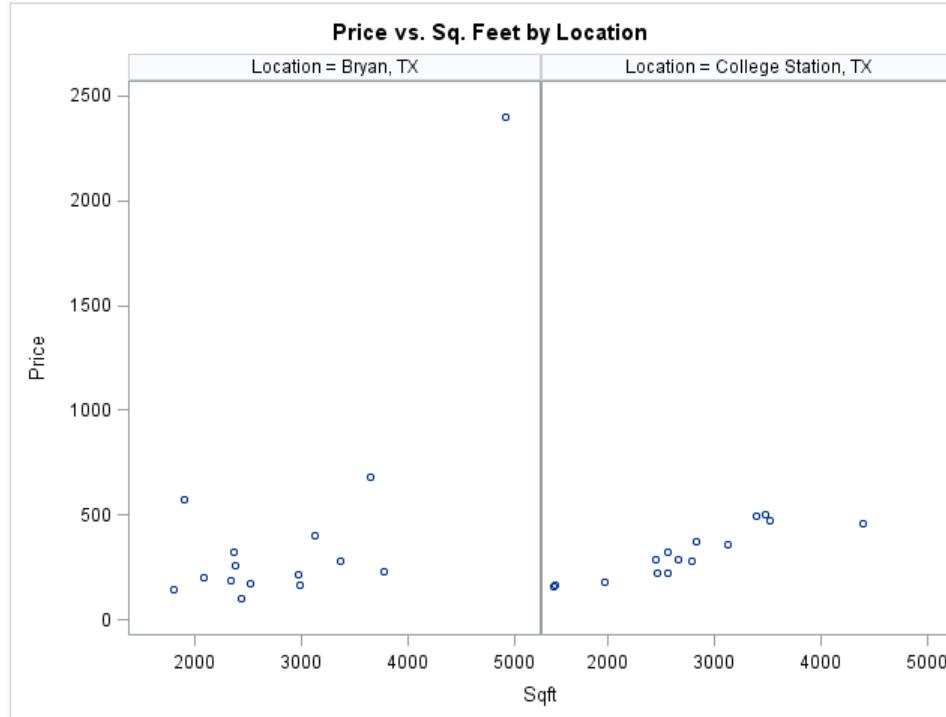
```
PROC SGPROC DATA=SAS-data-set;  
  PANELBY group-variable;  
    TYPE1 chart-variable(s) . . . </ options>;  
    TYPE2 chart-variable(s) . . . </ options>;  
RUN;
```

# Producing Charts with the SGpanel Procedure

```
title 'Price vs. Sq. Feet by Location';
proc sgpanel data=bcs;
    panelby location;
    scatter x=sqft y=price;
run;
```

# Producing Charts with the SGPANEL Procedure

Results:





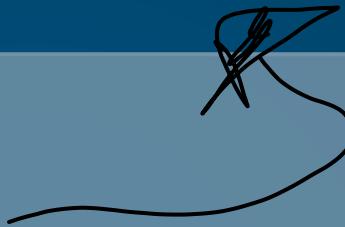
SAS®  
Graphics  
and  
SAS®  
Screens

## Visualizing data with graphics procedures

This demonstration explores a few of the options for the GPLOT and GPANEL procedures.

# Linear Models

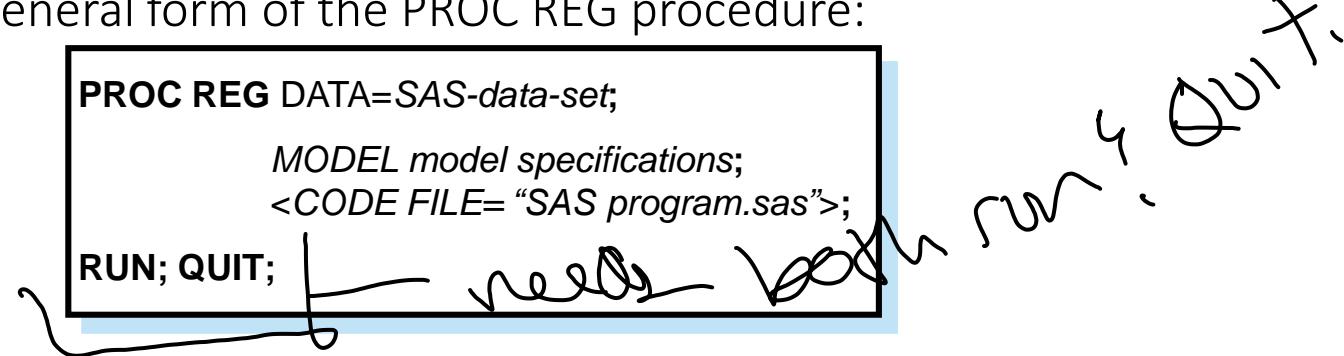
The REG Procedure



# Producing Models with the REG Procedure

General form of the PROC REG procedure:

```
PROC REG DATA=SAS-data-set;  
    MODEL model specifications;  
    <CODE FILE= "SAS program.sas">;  
RUN; QUIT;
```



- Requires SAS STAT license
- MODEL - specifies the dependent and independent variables, etc.
- CODE – writes DATA STEP code for predicting values according to the fitted model

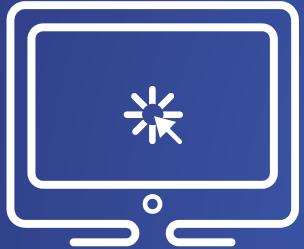
# Producing Models with the REG Procedure

Example:

```
title 'Model of Min Pressure/Max Wind';
PROC REG DATA=pg1.storm_final;
    MODEL maxwindmph=minpressure;
    CODE FILE = "&path.stormmodel.sas";
run;
```

Handwritten notes explaining the code:

- maxwindmph = dependent variable
- minpressure = independent variable
- &path = macro variable containing path
- macro as delimiter
- as delimiter
- macro as delimiter
- macro as delimiter



# Analyzing data with PROC REG

This demonstration shows the basic operation and output produced by the REG procedure

# STAT 604 Takeaways

1. Know thy data
2. Pay attention to detail
3. Error-free log does not mean results are correct.
4. Think!

