The George Washington University
Department of Statistics

STAT 4197/6197 - Fall 2019

Week 5 – September 27, 2019

Major Topic: Controlling and Managing SAS Data Sets
(DATA/PROC Step)

Detailed Topics:

1. Controlling SAS Data Sets with Options/Statements
2. Filtering Observations
3. Sorting Data
4. Accessing Data Directly
5. Copying/Modifying SAS Data Sets
6. Removing Selected Attributes of the Variables
7. Downloading Zipped SAS Transport Files from the Web
8. Converting SAS Transport Files into SAS Data Sets
9. Restructuring Data

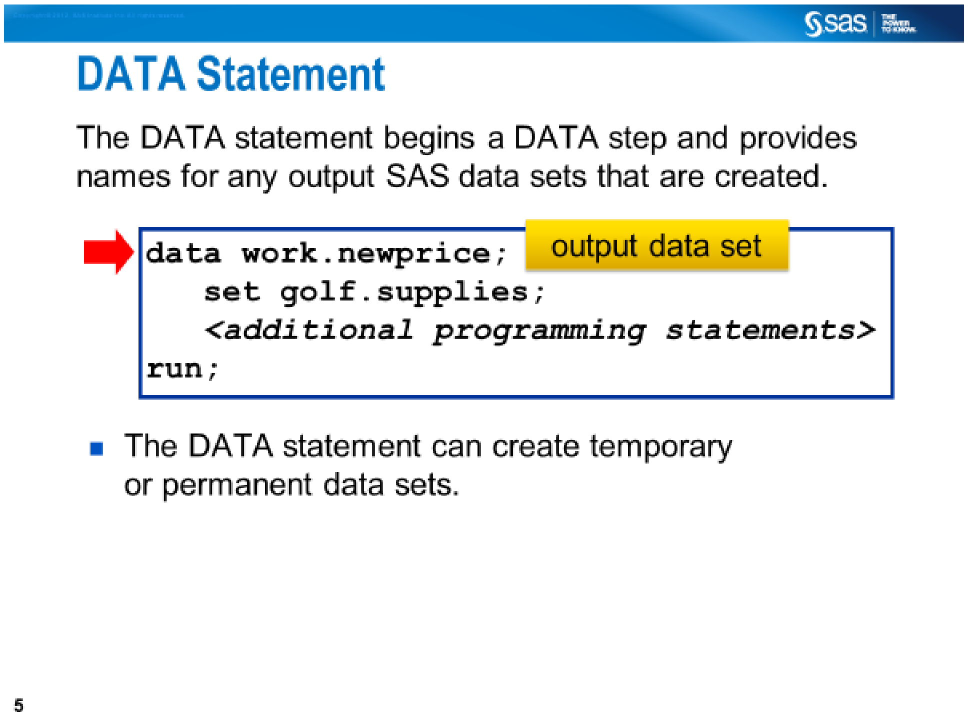
Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Readings:

1. Relevant Chapters/Sections - Delwiche L, and Slaughter S. *The Little SAS Book: A Primer*, Fifth Edition Paperback – November 7, 2012

2. Exercises from Relevant Chapters/Sections - Ottesen RA, Delwiche LD, and Slaughter SJ. *Exercises and Projects for The Little SAS Book*, Fifth Edition Paperback – July 1, 2015

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



DATA Statement

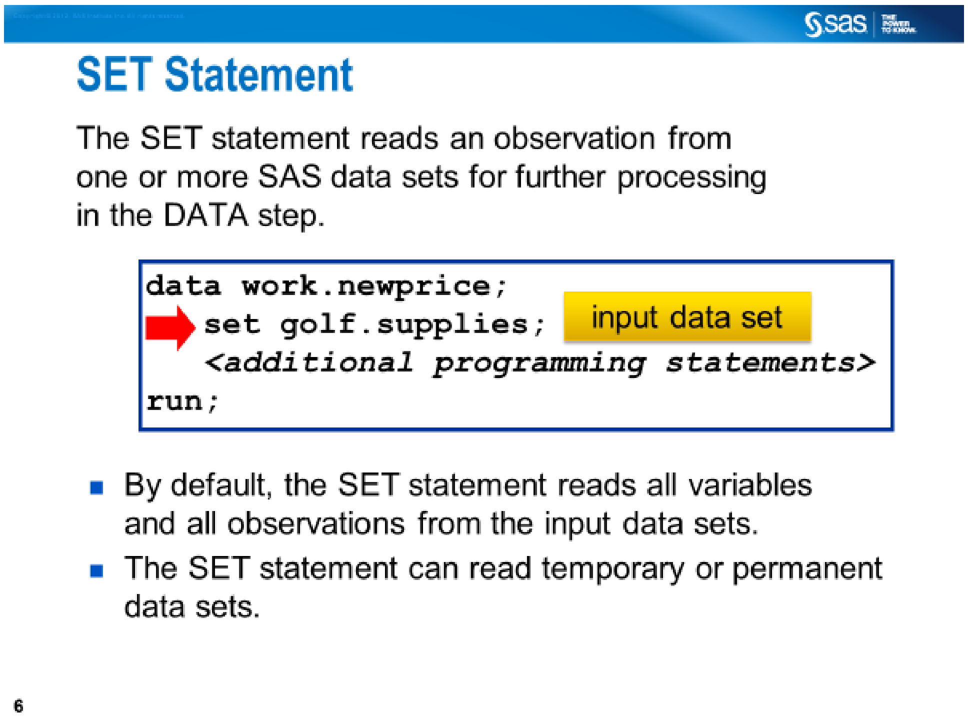
The DATA statement begins a DATA step and provides names for any output SAS data sets that are created.

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

output data set

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

5



SET Statement

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

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

input data set

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

6

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Additional Programming Statements

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

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

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

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

7

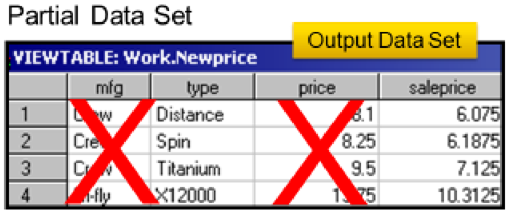
DROP and KEEP Statements

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

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

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

Partial Data Set

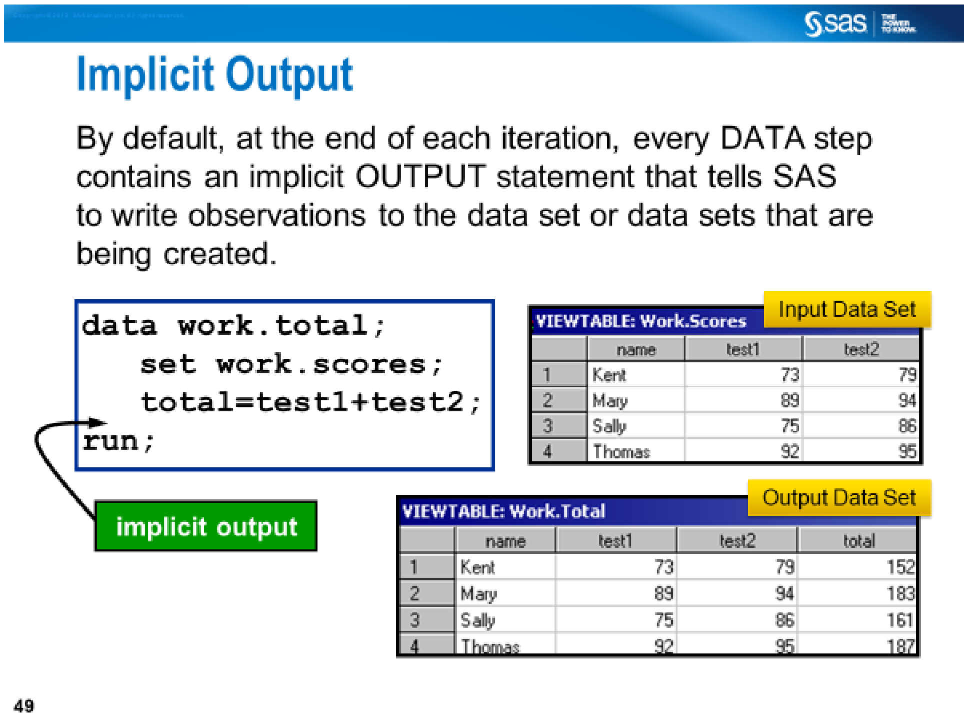


VIEWTABLE: Work.Newprice

	mfg	type	price	saleprice
1	[illegible]	Distance	8.1	6.075
2	[illegible]	Spin	8.25	6.1875
3	[illegible]	Titanium	9.5	7.125
4	[illegible]	X12000	[illegible]	10.3125

37

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Implicit Output

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

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

implicit output

Input Data Set

VIEWTABLE: Work.Scores

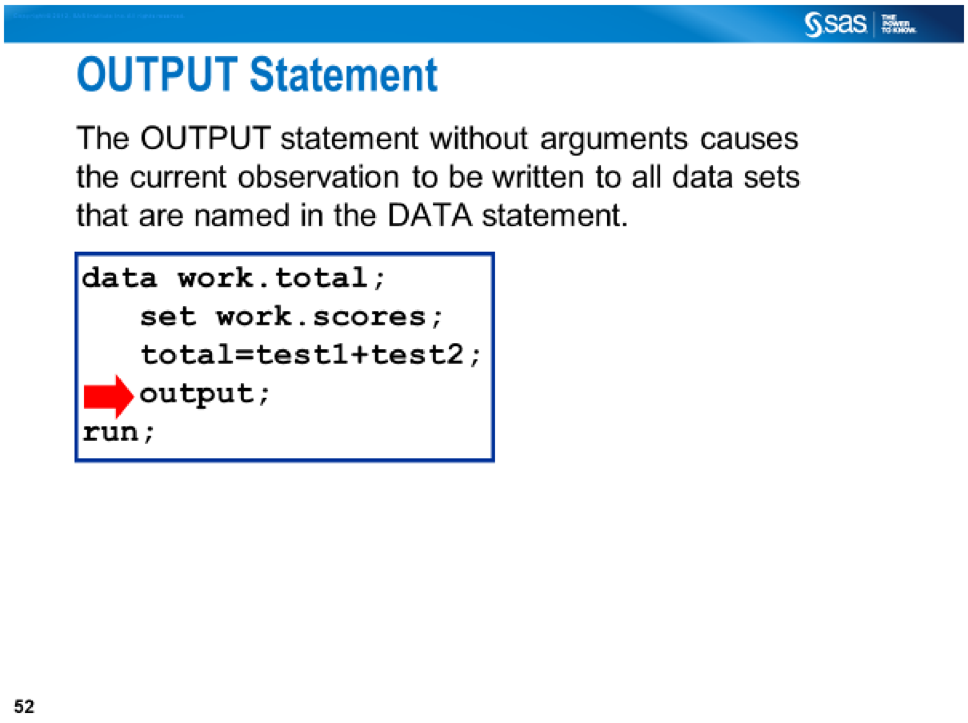
	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.Total

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

49



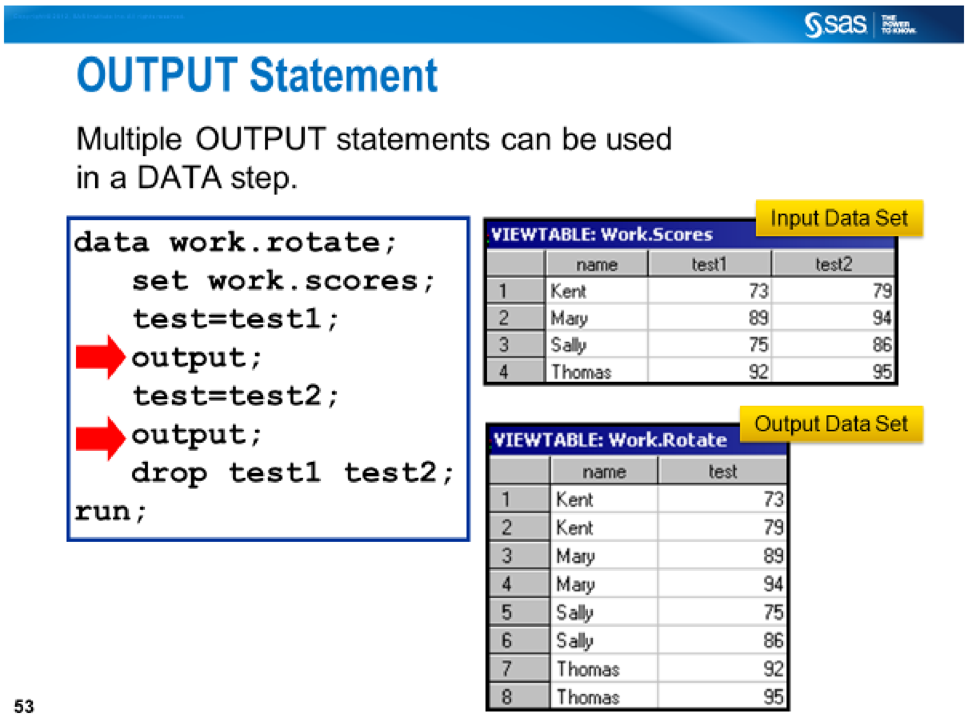
OUTPUT Statement

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

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

52

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



OUTPUT Statement

Multiple OUTPUT statements can be used in a DATA step.

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

Input Data Set

VIEWTABLE: Work.Scores

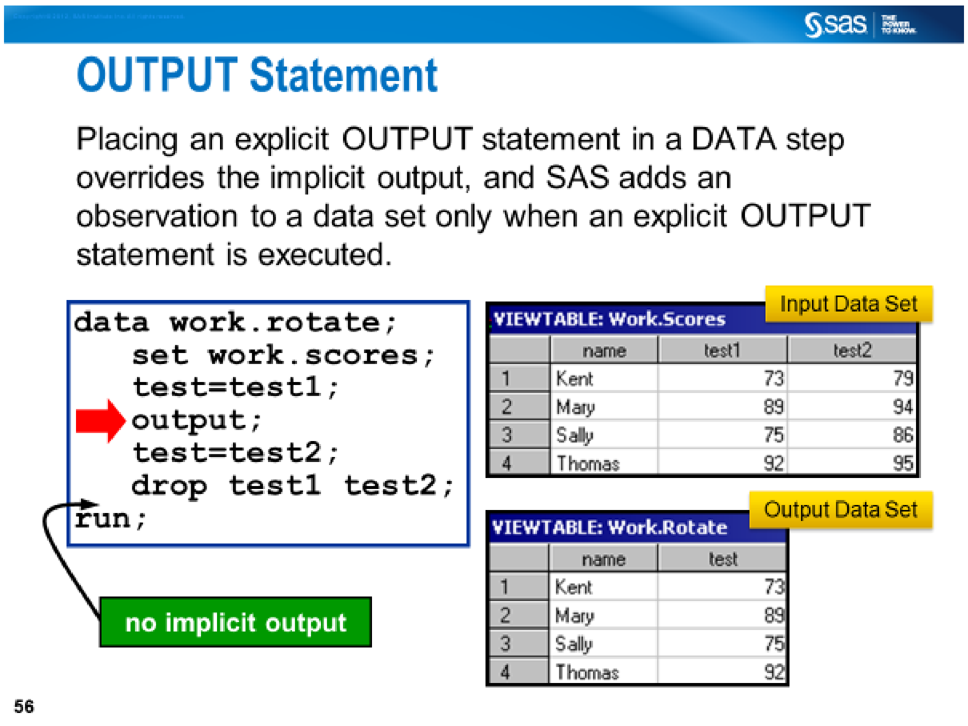
	name	test1	test2
1	Kent	73	79
2	Mary	89	94
3	Sally	75	86
4	Thomas	92	95

Output Data Set

VIEWTABLE: Work.Rotate

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

53



OUTPUT Statement

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

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

no implicit output

Input Data Set

VIEWTABLE: Work.Scores

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

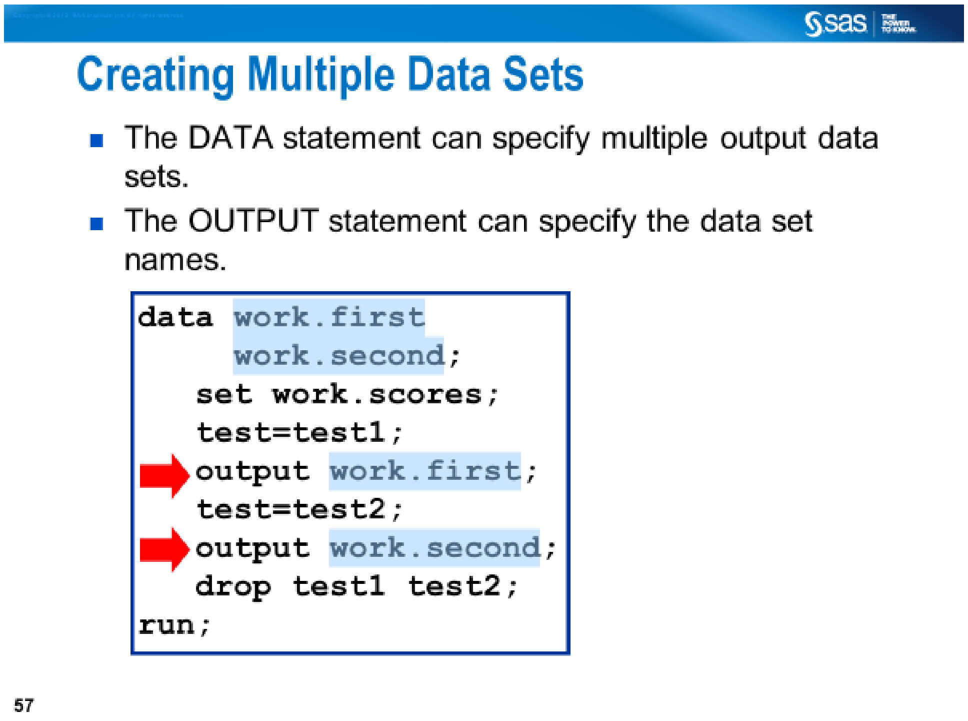
Output Data Set

VIEWTABLE: Work.Rotate

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

56

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

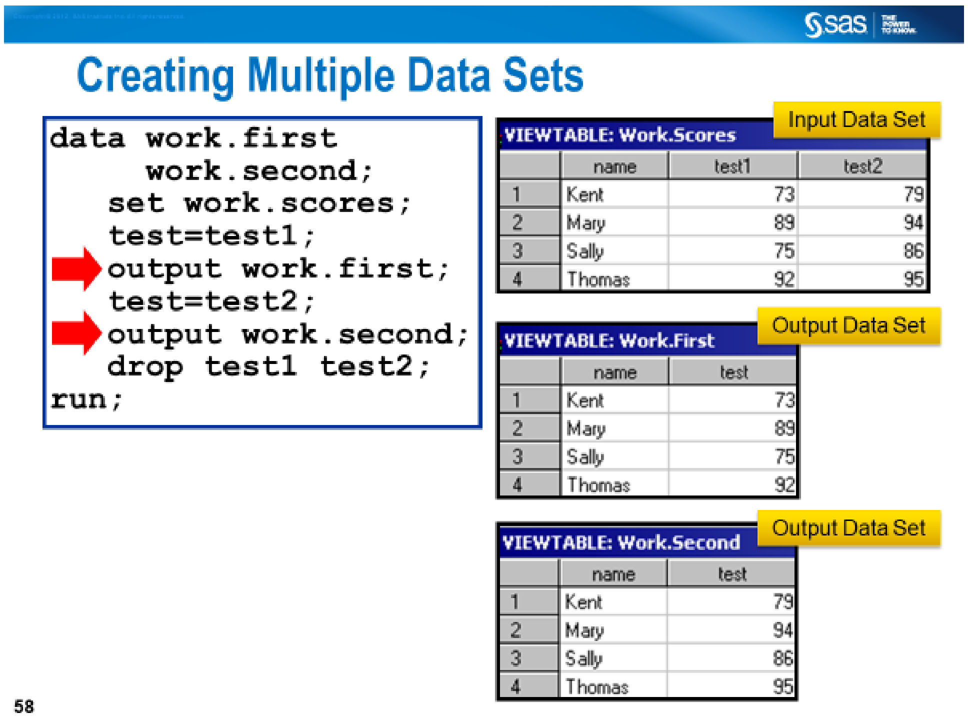


Creating Multiple Data Sets

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

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

57



Creating Multiple Data Sets

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

Input Data Set

VIEWTABLE: Work.Scores

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

Output Data Set

VIEWTABLE: Work.First

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

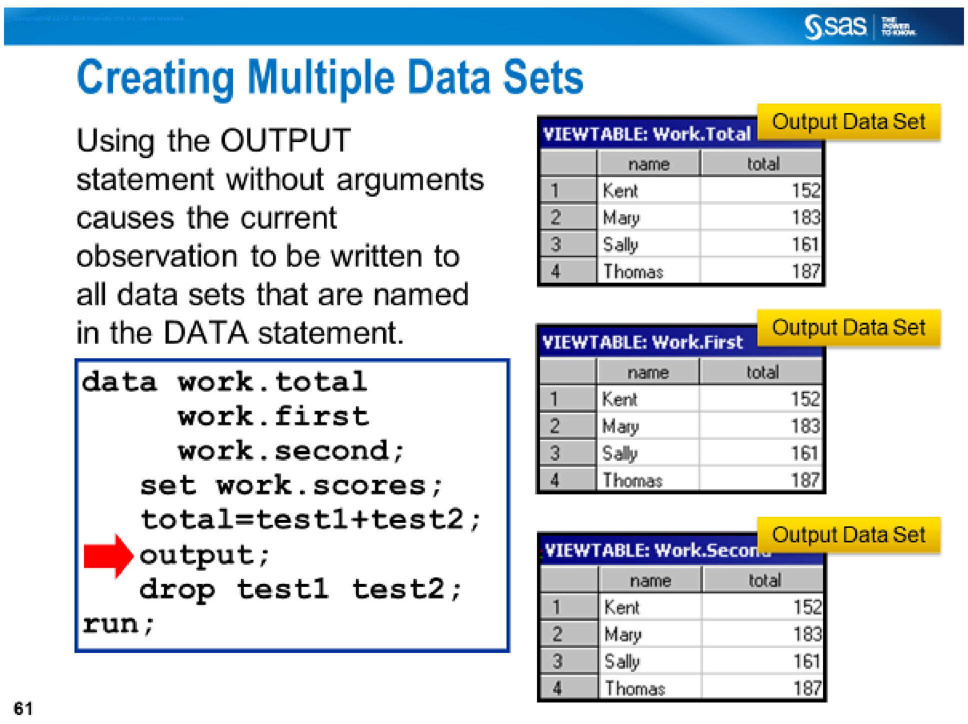
Output Data Set

VIEWTABLE: Work.Second

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

58

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Creating Multiple Data Sets

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

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

Output Data Set

VIEWTABLE: Work.Total

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

Output Data Set

VIEWTABLE: Work.First

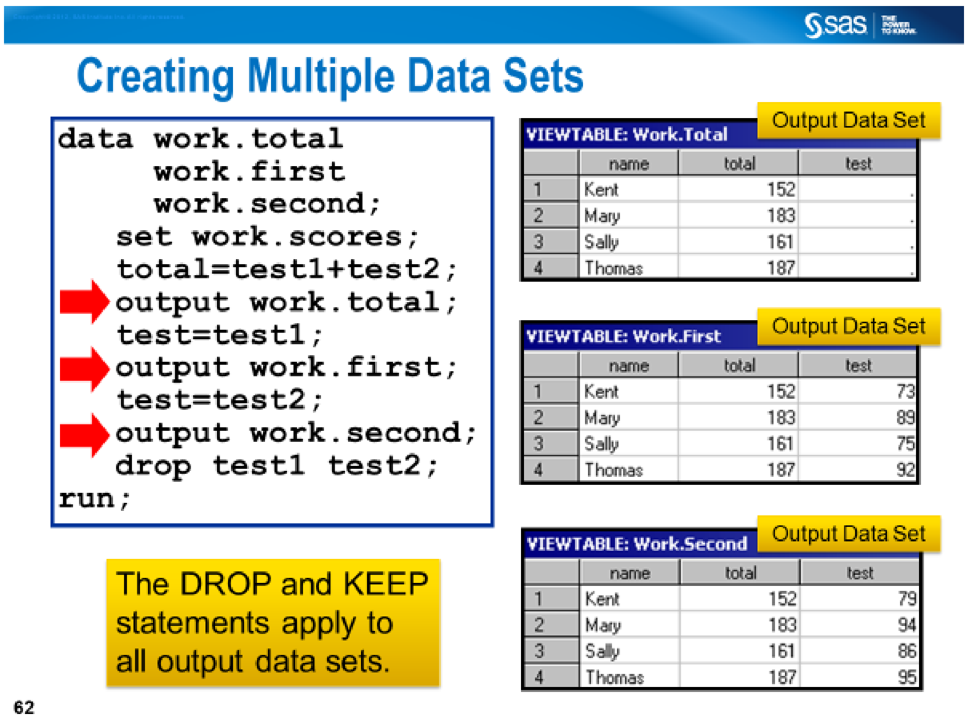
	name	total
1	Kent	152
2	Mary	183
3	Sally	161
4	Thomas	187

Output Data Set

VIEWTABLE: Work.Second

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

61



Creating Multiple Data Sets

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

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

Output Data Set

VIEWTABLE: Work.Total

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

Output Data Set

VIEWTABLE: Work.First

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

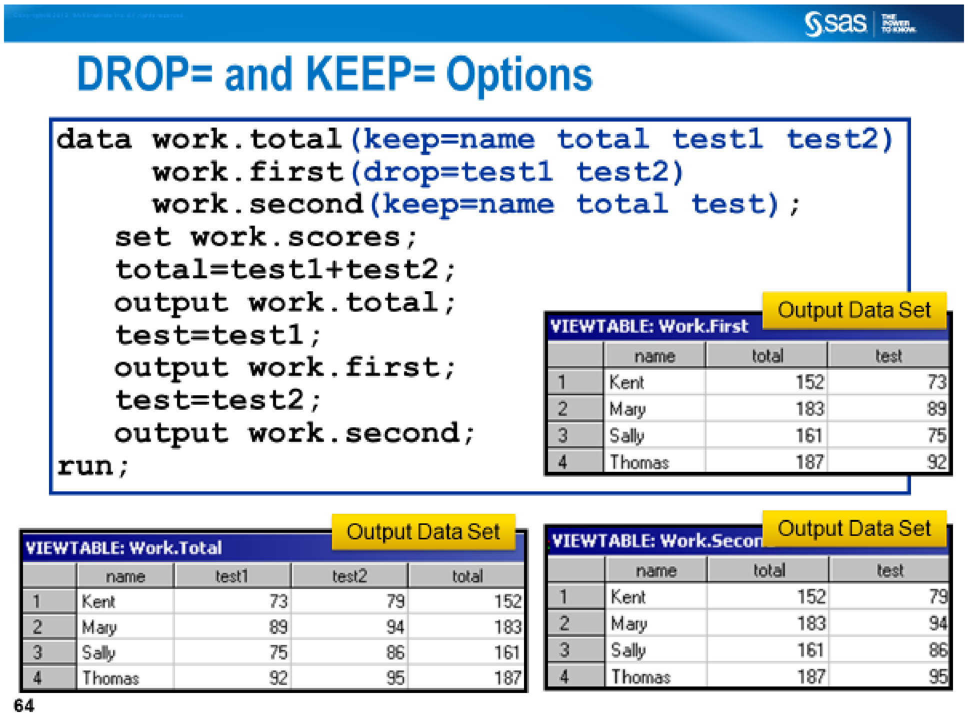
Output Data Set

VIEWTABLE: Work.Second

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

62

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



DROP= and KEEP= Options

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

Output Data Set

VIEWTABLE: Work.First

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

Output Data Set

VIEWTABLE: Work.Total

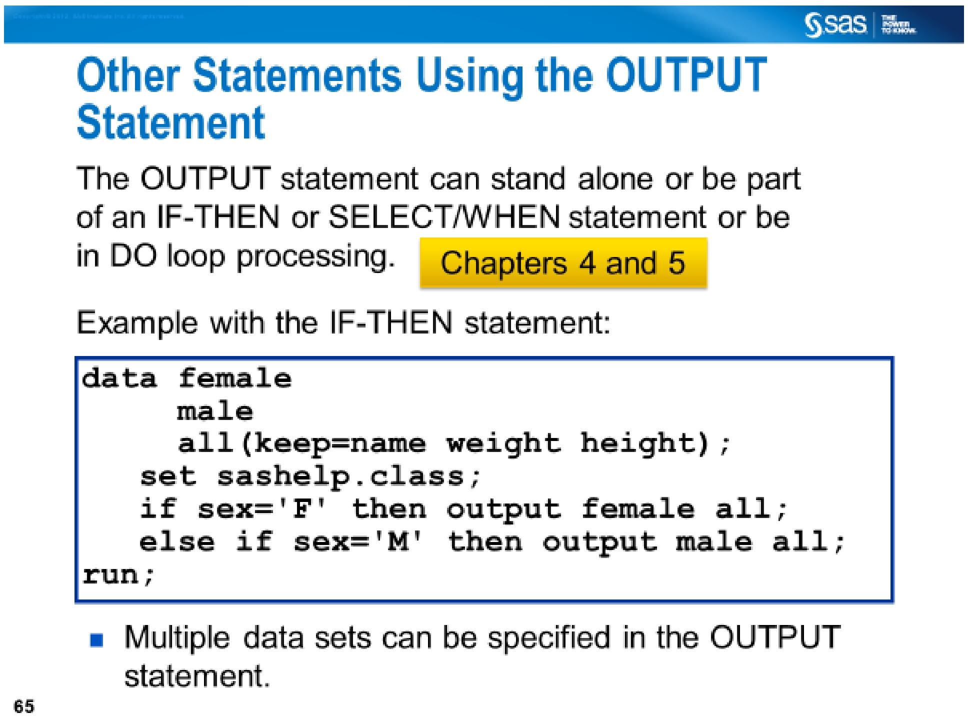
	name	test1	test2	total
1	Kent	73	79	152
2	Mary	89	94	183
3	Sally	75	86	161
4	Thomas	92	95	187

Output Data Set

VIEWTABLE: Work.Secon

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

64



Other Statements Using the OUTPUT Statement

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

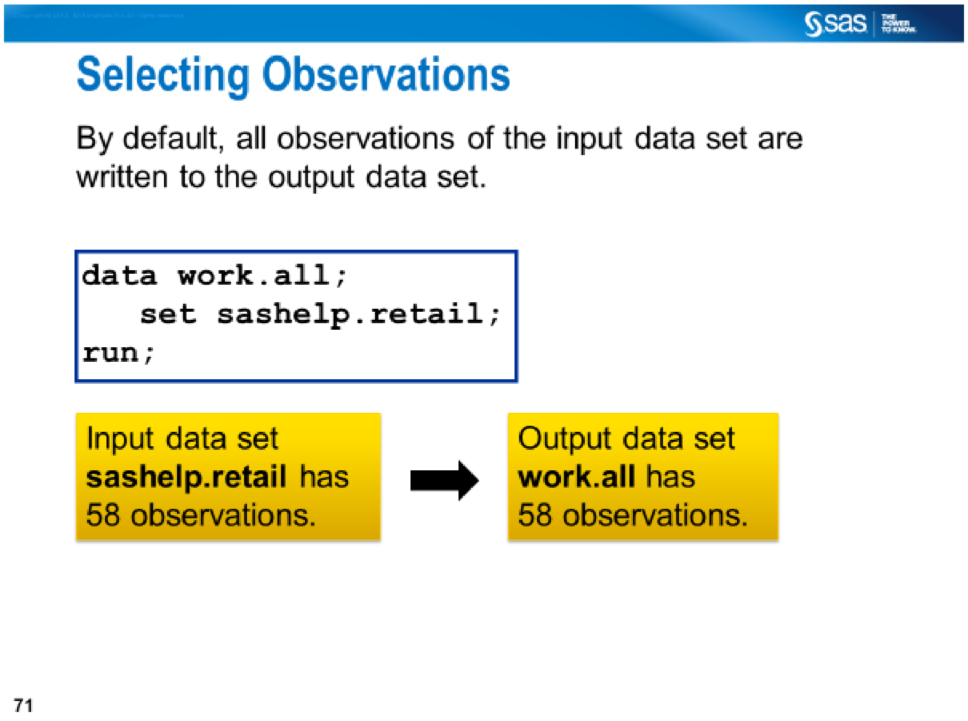
Example with the IF-THEN statement:

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

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

65

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



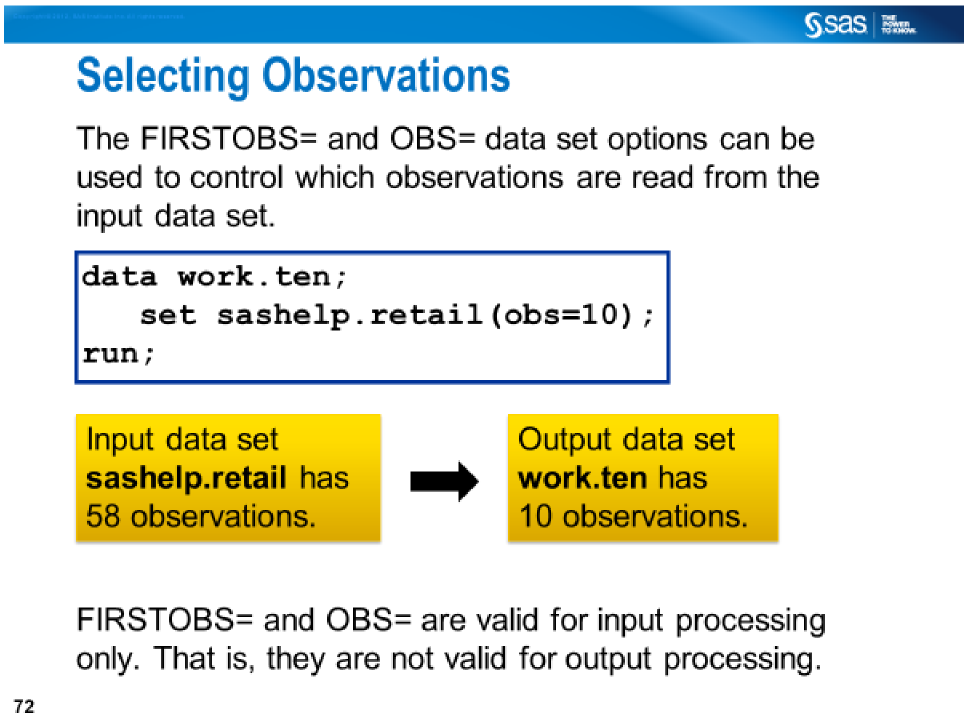
Selecting Observations

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

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

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

71



Selecting Observations

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

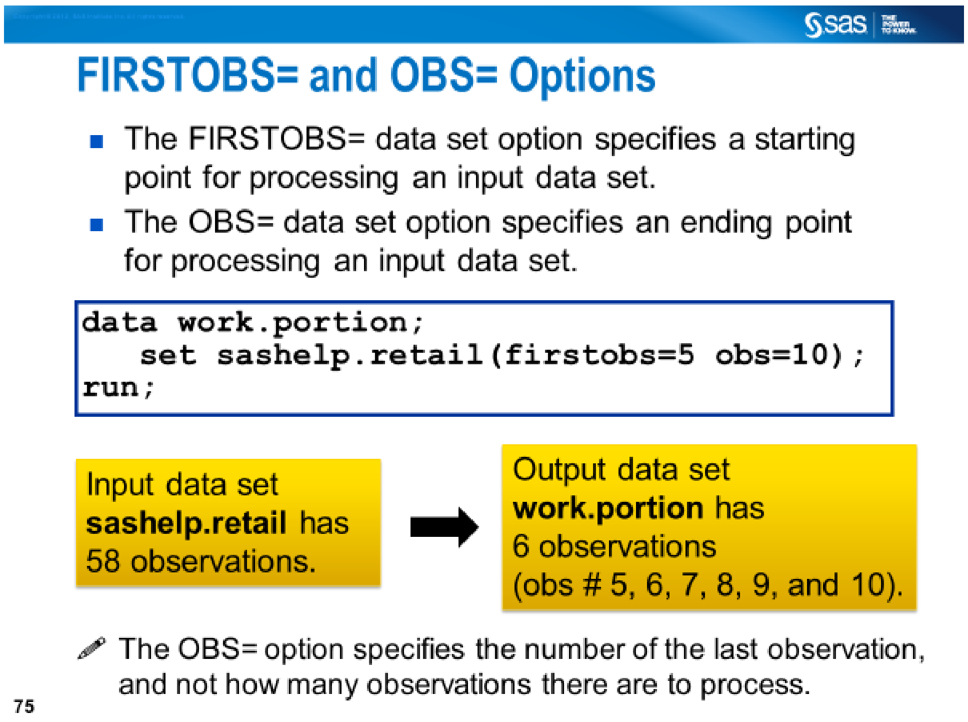
```
data work.ten;
   set sashelp.retail(obs=10);
run;
```

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

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

72

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



FIRSTOBS= and OBS= Options

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

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

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

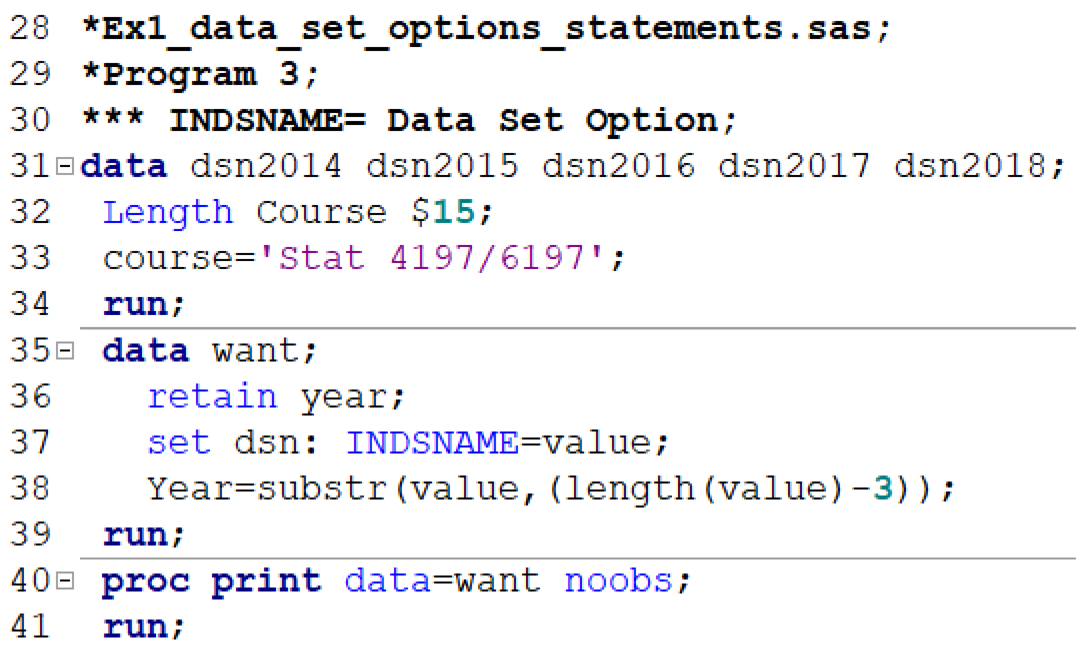
✎ The OBS= option specifies the number of the last observation, and not how many observations there are to process.

75

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Ex1_data_set_options_statements.sas

INDSNAME= Option with the SET Statement

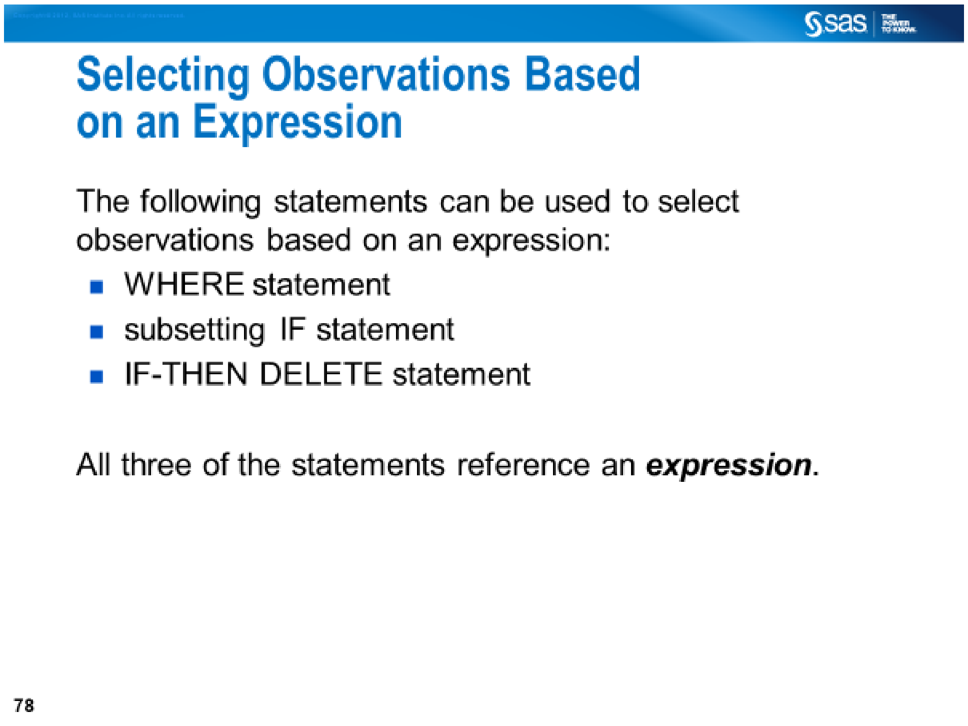


```
*Ex1_data_set_options_statements.sas;
*Program 3;
*** INDSNAME= Data Set Option;
data dsn2014 dsn2015 dsn2016 dsn2017 dsn2018;
 Length Course $15;
 course='Stat 4197/6197';
 run;
 data want;
   retain year;
   set dsn: INDSNAME=value;
   Year=substr(value,(length(value)-3));
 run;
 proc print data=want noobs;
 run;
```

Line 8: The INDSNAME= retrieves the name of the data set from which the current observation is read

Line 9: The LENGTH function returns the length of a non-blank character string to determine the position of the last character so that the SUBSTR function can extract it.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



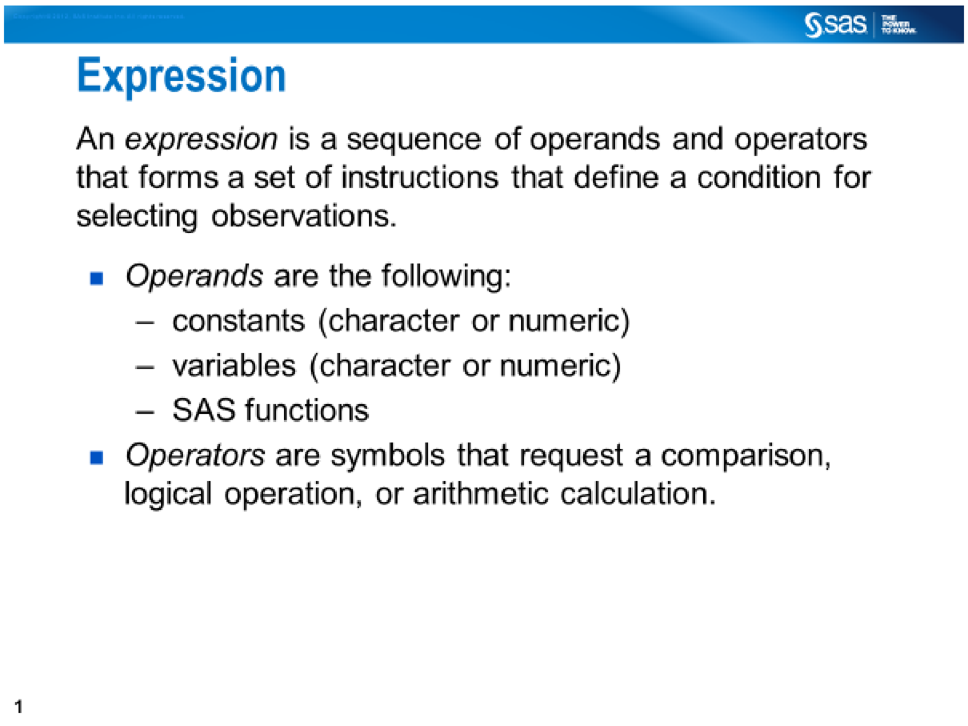
Selecting Observations Based on an Expression

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

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

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

78



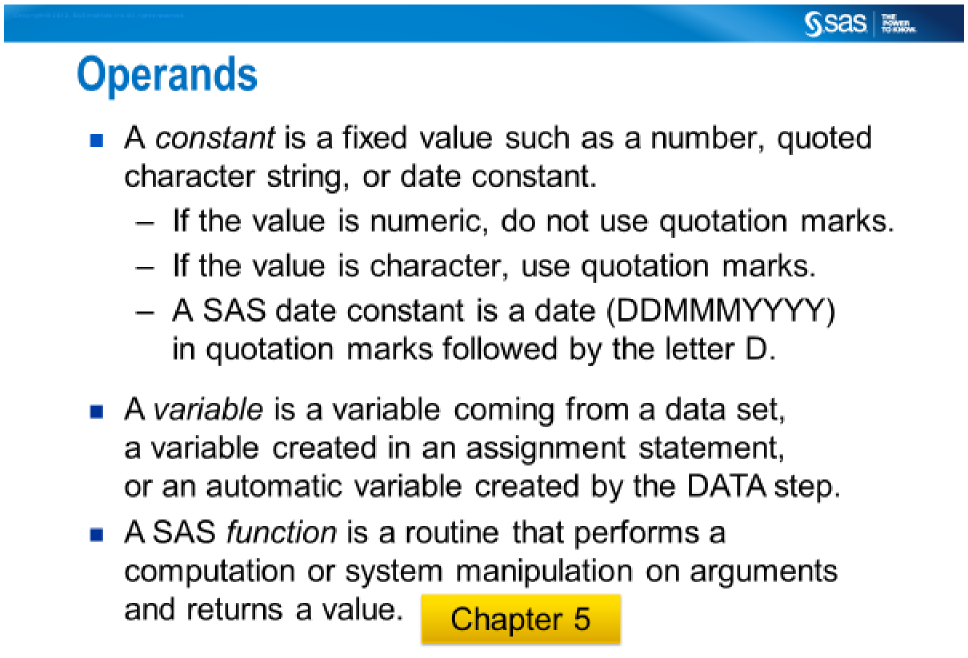
Expression

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

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

1

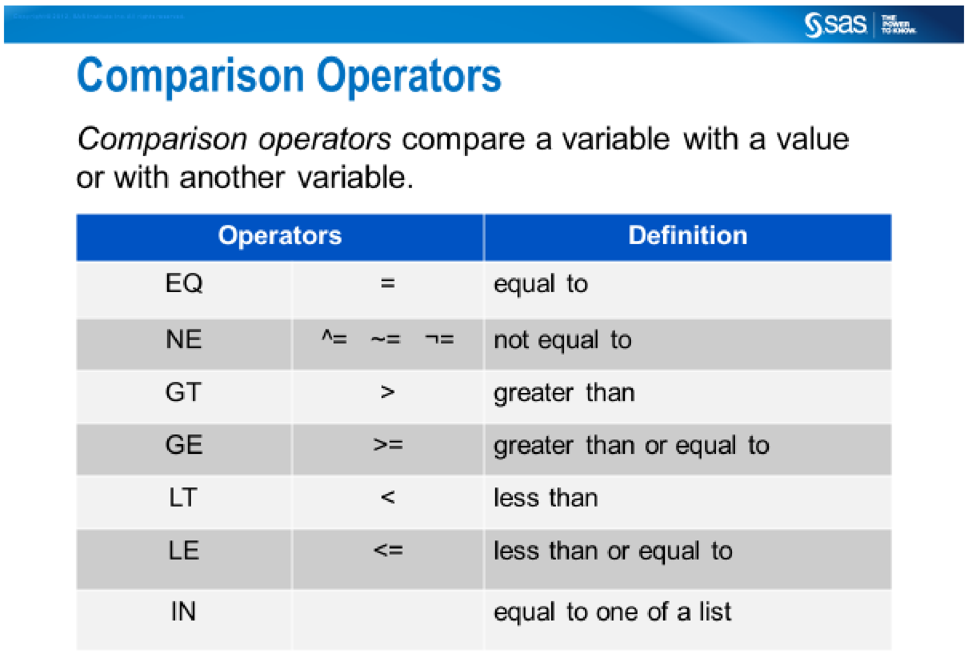
Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Operands

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

80



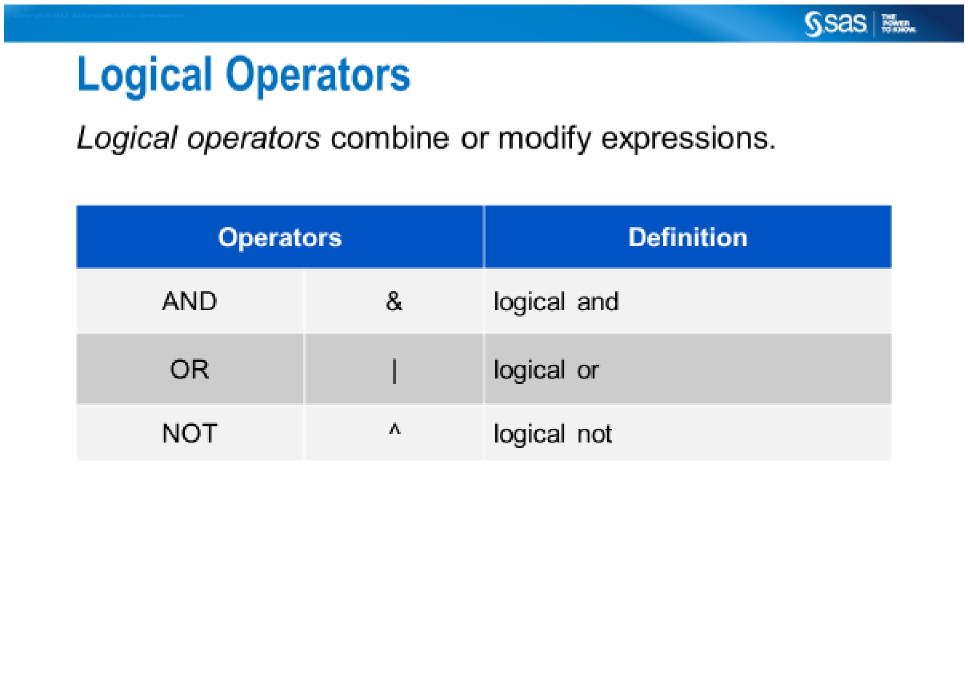
Comparison Operators

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

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

81

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

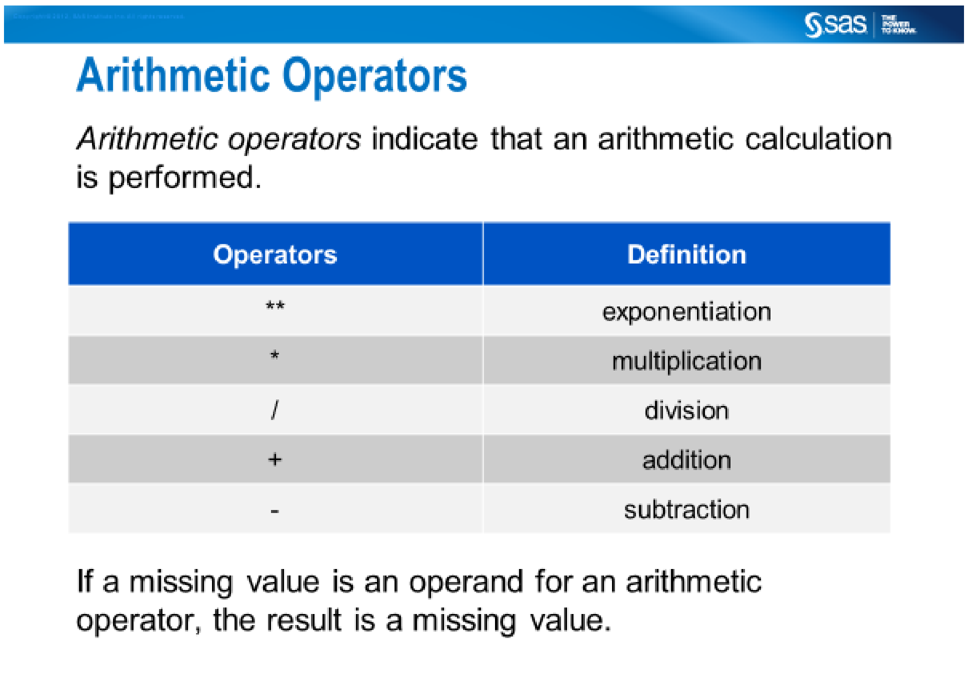


Logical Operators

Logical operators combine or modify expressions.

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

84



Arithmetic Operators

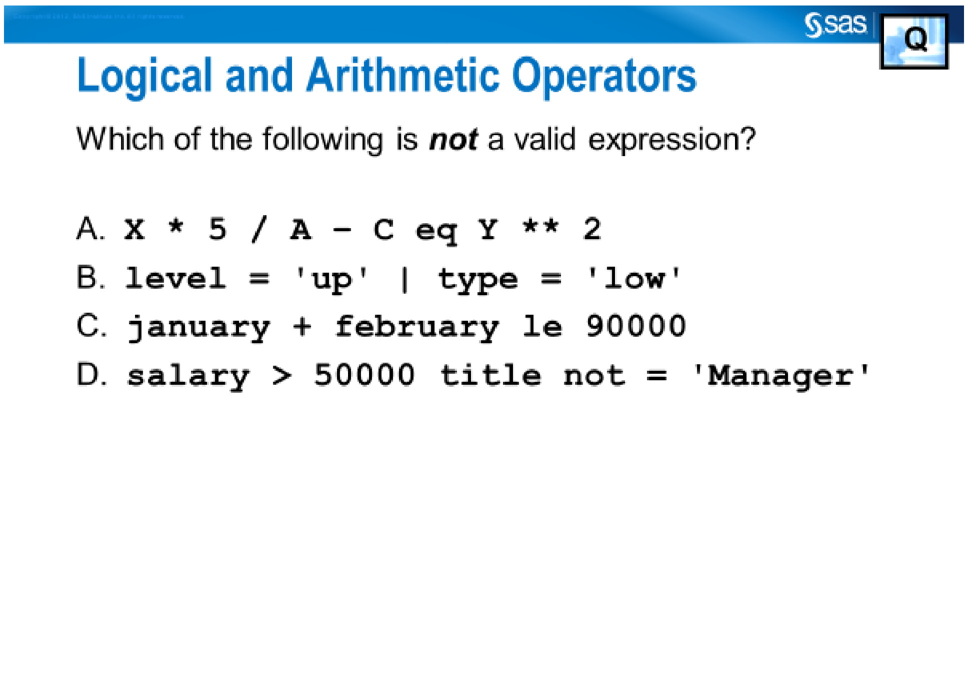
Arithmetic operators indicate that an arithmetic calculation is performed.

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

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

85

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

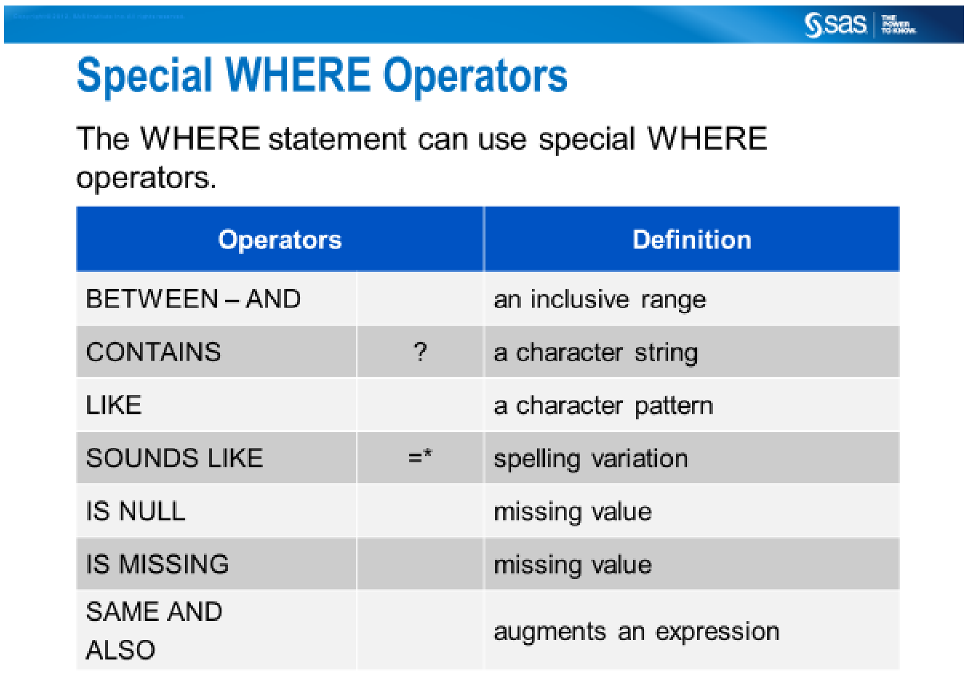


Logical and Arithmetic Operators

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

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

86



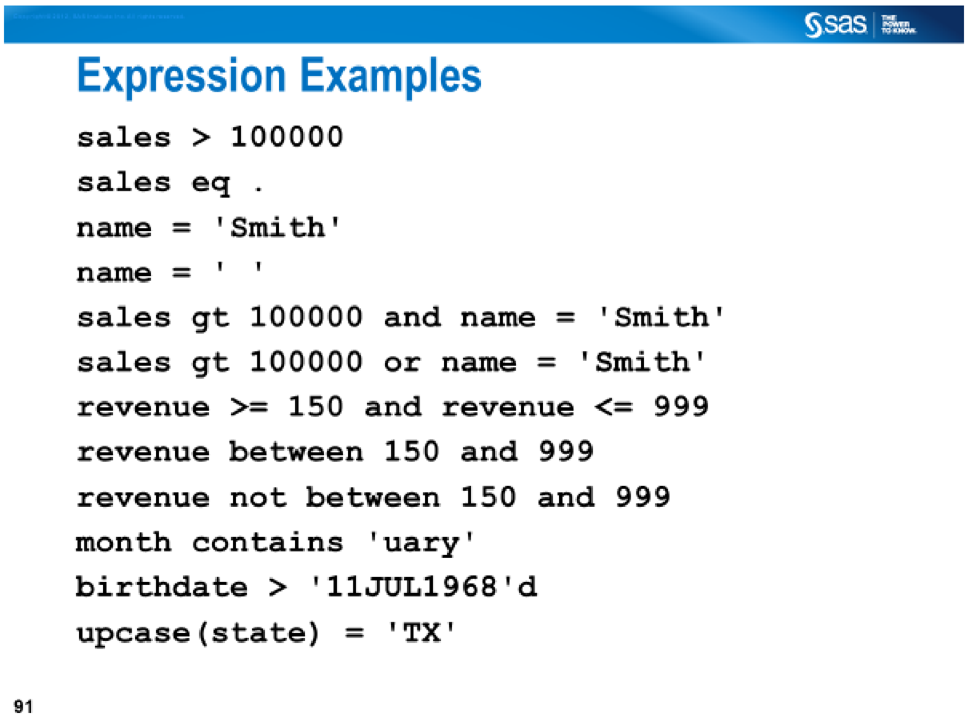
Special WHERE Operators

The WHERE statement can use special WHERE operators.

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

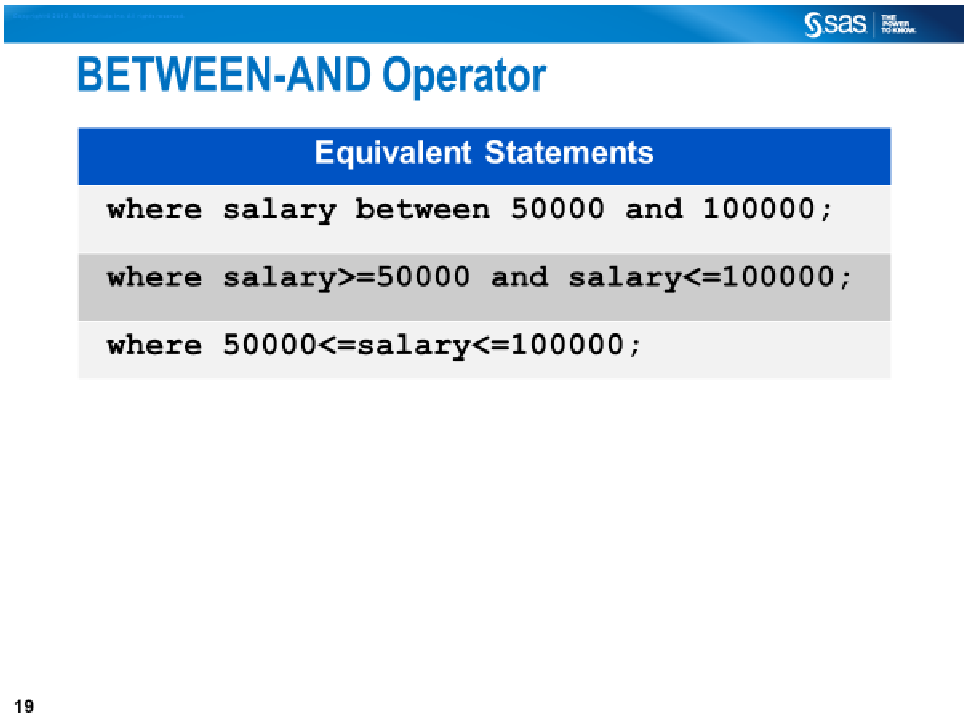
88

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Expression Examples

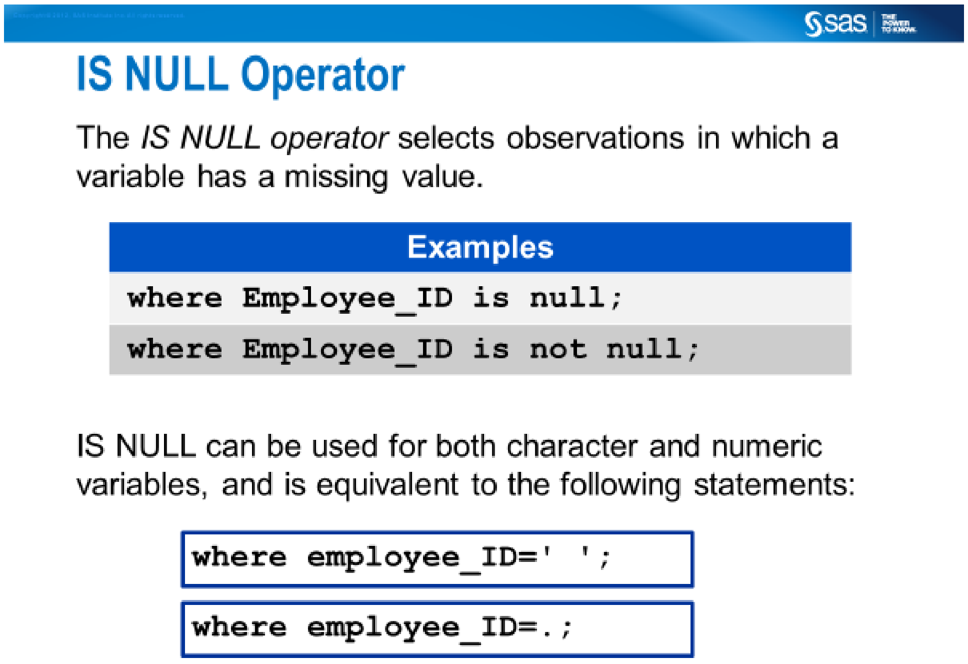
```
sales > 100000
sales eq .
name = 'Smith'
name = ' '
sales gt 100000 and name = 'Smith'
sales gt 100000 or name = 'Smith'
revenue >= 150 and revenue <= 999
revenue between 150 and 999
revenue not between 150 and 999
month contains 'uary'
birthdate > '11JUL1968'd
upcase(state) = 'TX'
```



BETWEEN-AND Operator

Equivalent Statements
`where salary between 50000 and 100000;`
`where salary>=50000 and salary<=100000;`
`where 50000<=salary<=100000;`

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



IS NULL Operator

The *IS NULL operator* selects observations in which a variable has a missing value.

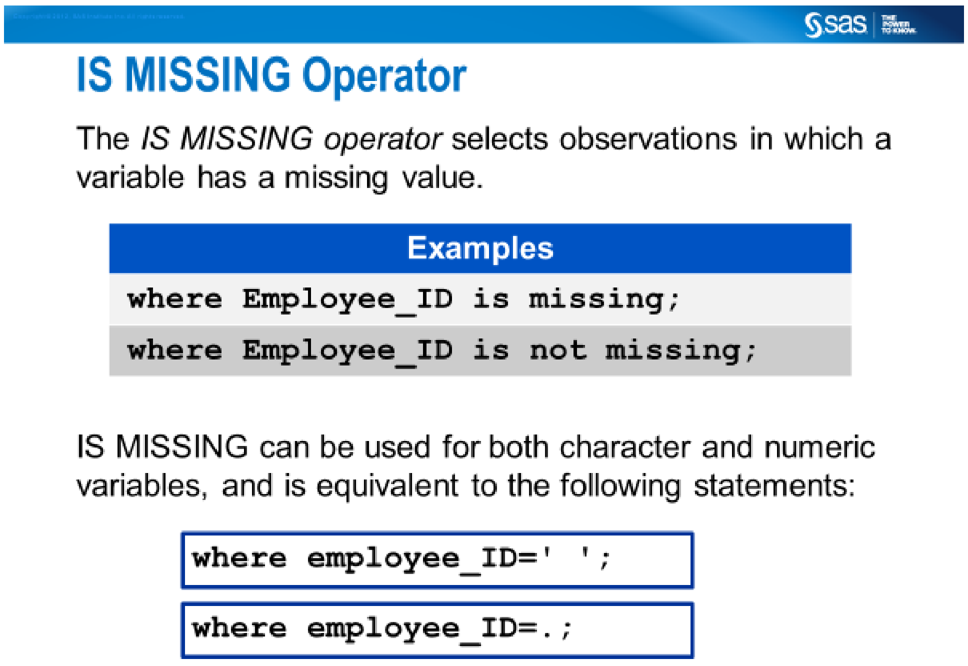
Examples
`where Employee_ID is null;`
`where Employee_ID is not null;`

IS NULL can be used for both character and numeric variables, and is equivalent to the following statements:

```
where employee_ID=' ';
```

```
where employee_ID=.;
```

22



IS MISSING Operator

The *IS MISSING operator* selects observations in which a variable has a missing value.

Examples
`where Employee_ID is missing;`
`where Employee_ID is not missing;`

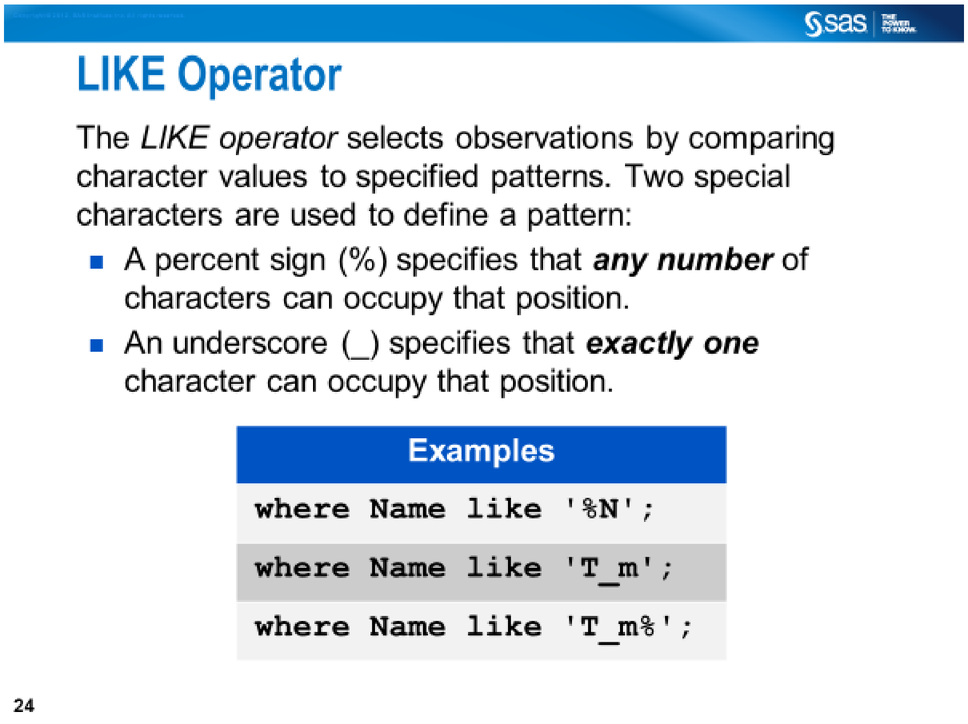
IS MISSING can be used for both character and numeric variables, and is equivalent to the following statements:

```
where employee_ID=' ';
```

```
where employee_ID=.;
```

23

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



LIKE Operator

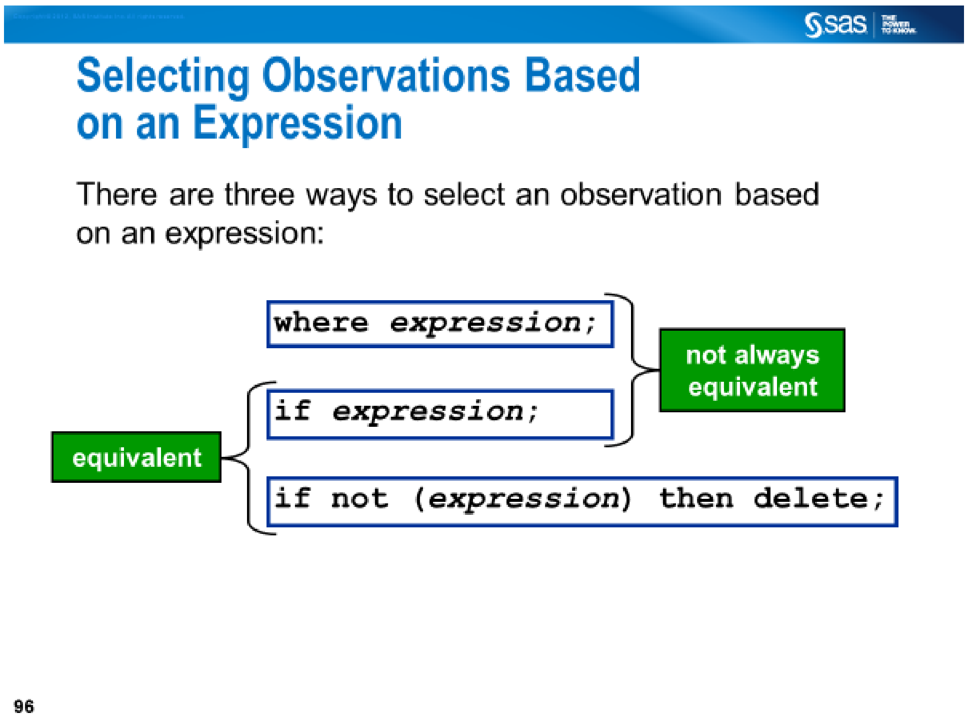
The *LIKE operator* selects observations by comparing character values to specified patterns. Two special characters are used to define a pattern:

- A percent sign (%) specifies that ***any number*** of characters can occupy that position.
- An underscore (_) specifies that ***exactly one*** character can occupy that position.

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

24

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



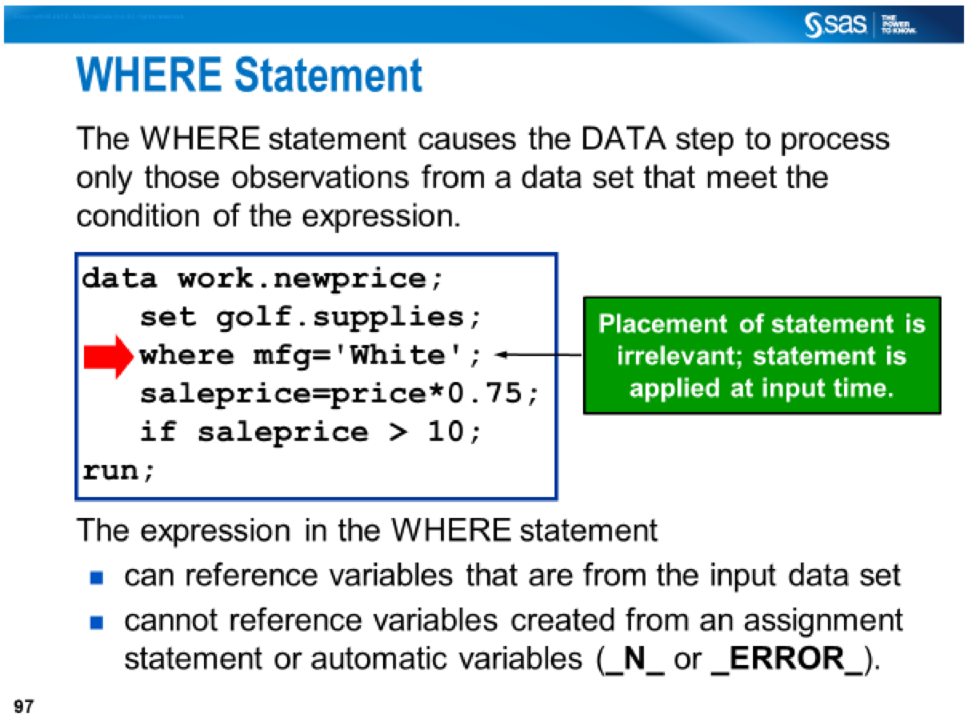
Selecting Observations Based on an Expression

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

```
where expression;
if expression;
if not (expression) then delete;
```

not always equivalent
equivalent

96



WHERE Statement

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

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

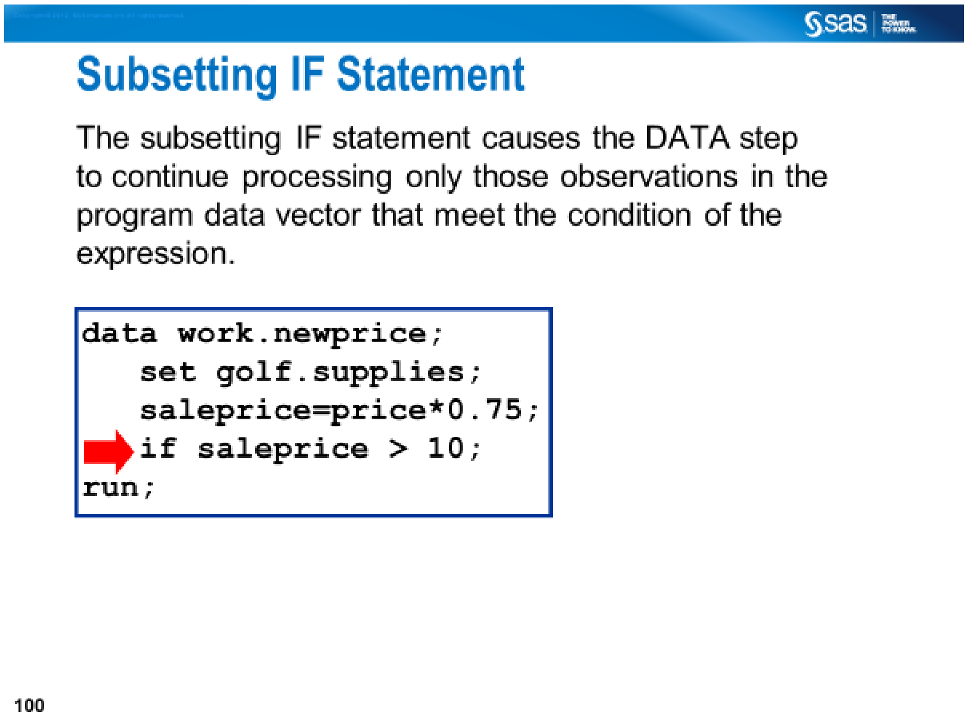
Placement of statement is irrelevant; statement is applied at input time.

The expression in the WHERE statement

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

97

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

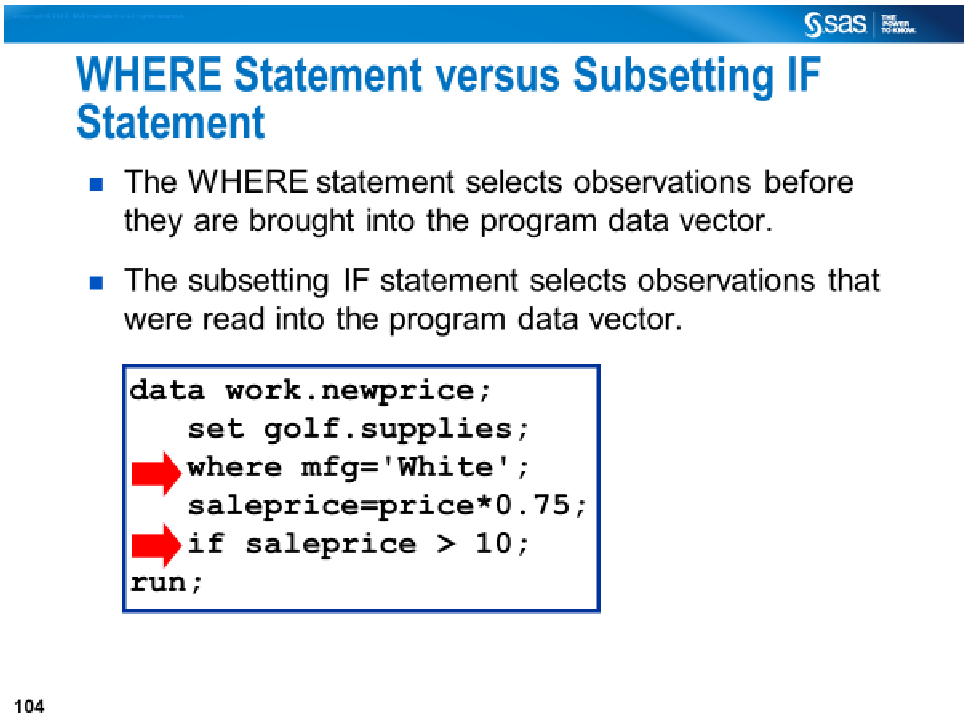


Subsetting IF Statement

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

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

100



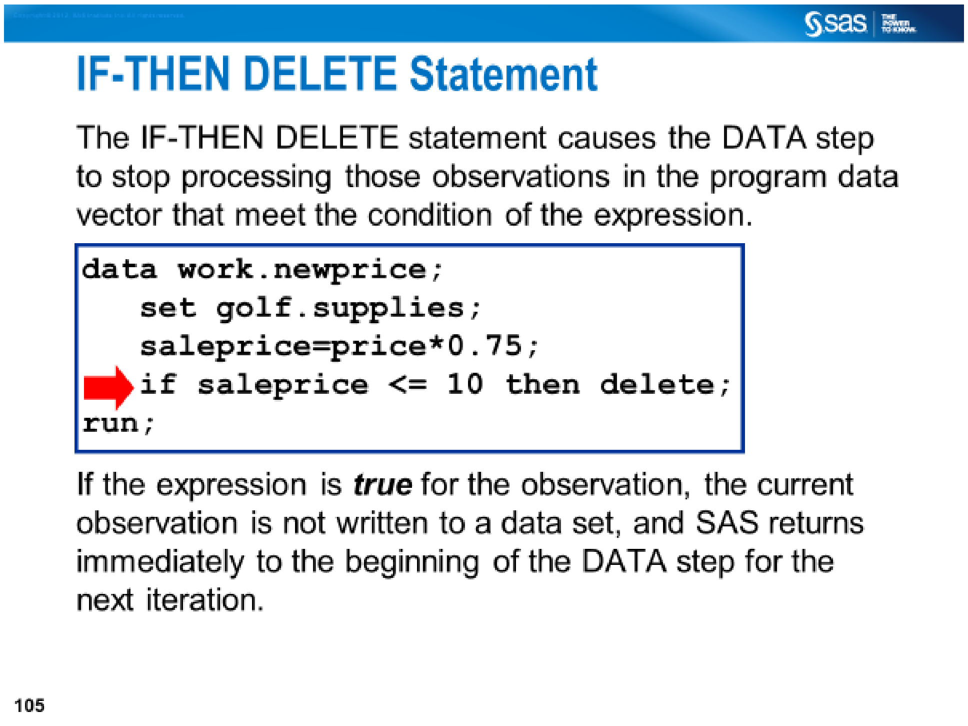
WHERE Statement versus Subsetting IF Statement

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

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

104

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



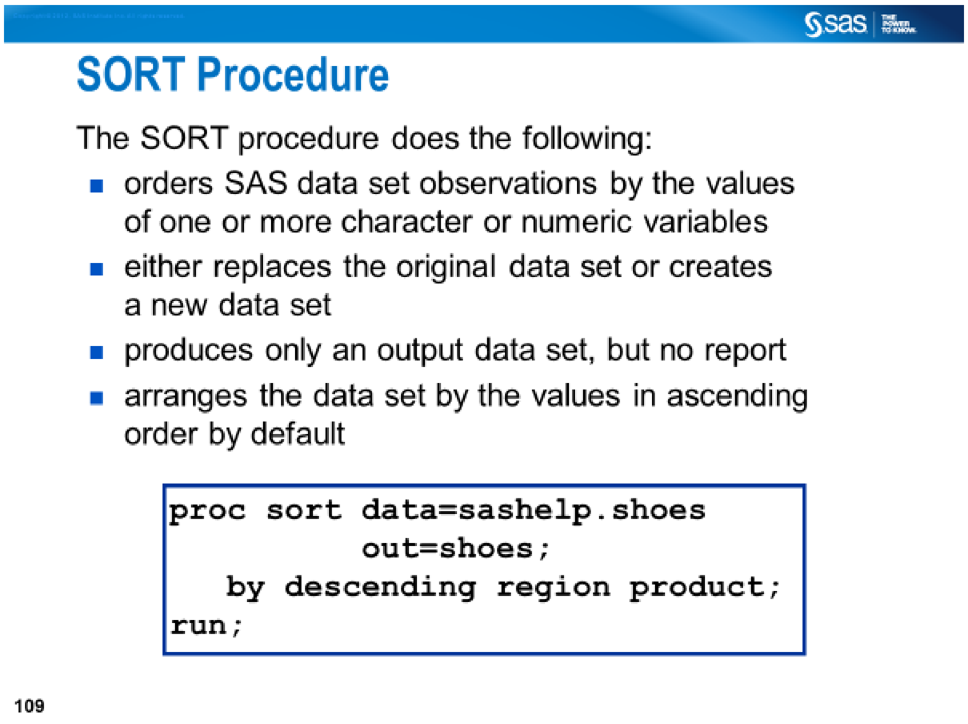
IF-THEN DELETE Statement

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

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

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

105



SORT Procedure

The SORT procedure does the following:

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

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

109

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

PROC SORT Statement

Examples:

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

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

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

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

112

BY Statement

The BY statement specifies the sorting variables.

Examples:

Ascending is the default order.

```
by region;
```

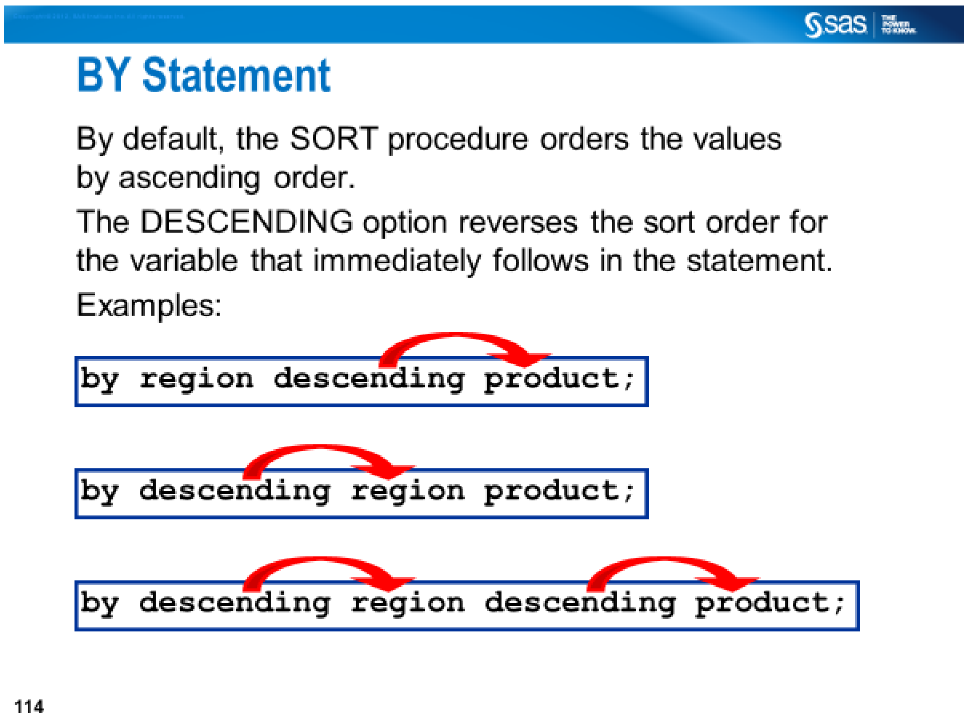
```
by region product;
```

```
by region subsidiary product;
```

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

113

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



BY Statement

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

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

Examples:

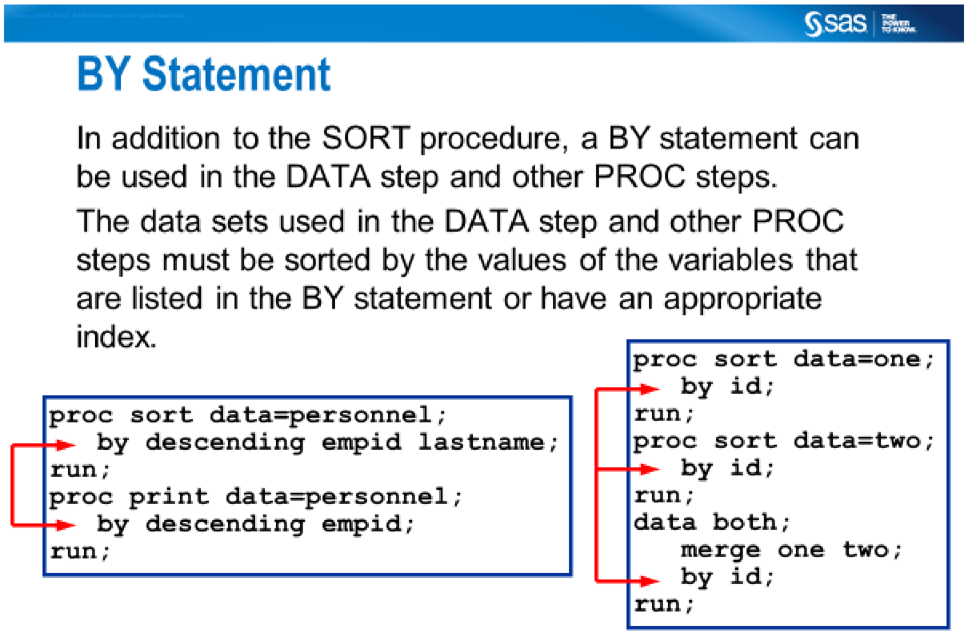
```
by region descending product;
```

```
by descending region product;
```

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

114

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



BY Statement

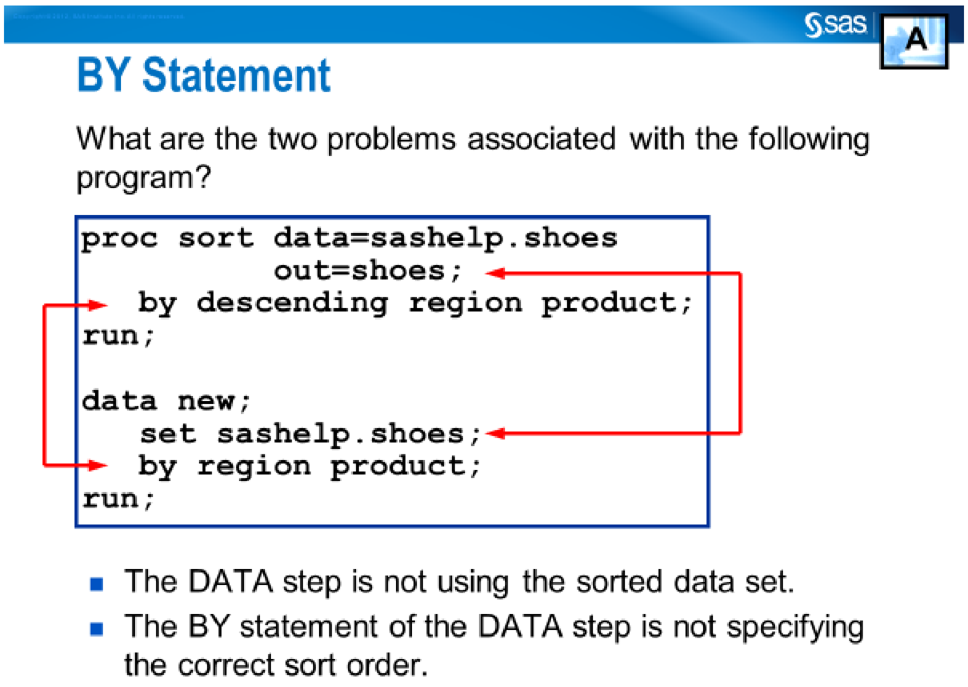
In addition to the SORT procedure, a BY statement can be used in the DATA step and other PROC steps.

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

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

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

117



BY Statement

What are the two problems associated with the following program?

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

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

- The DATA step is not using the sorted data set.
- The BY statement of the DATA step is not specifying the correct sort order.

119

Ex2B_SORT_various_Options.SAS

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Selected Options with PROC SORT

Example Data

```
 ID      visit_date          visit_type

 A01     01/15/2015     Emergency Room Visit
 A01     07/25/2015     Physician Office Visit
 A01     07/25/2015     Physician Office Visit
 A02     02/20/2015     Physician Office Visit
 A02     02/20/2015     Emergency Room Visit
 A05     01/12/2015     Outpatient Visit
```

`Ex2A_SORT_nodupkey_noduprecs.sas`

```
34 title2 "NODUPKEY Option with PROC SORT - One BY-variable (ID)";
35 proc sort data = work.HAVE nodupkey
36     out=work.nodupkey_1BY;
37 by ID ;
38 proc print data=work.nodupkey_1By noobs;
39 run;
```

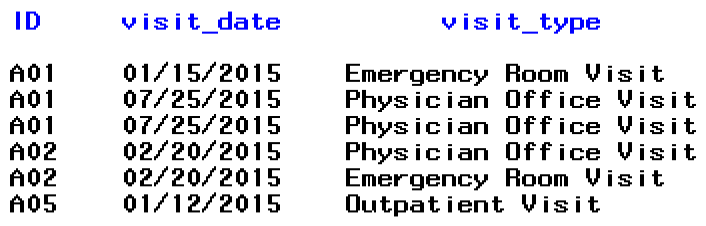
```
                 Ex2A_SORT_nodupkey_noduprecs.sas
 NODUPKEY Option with PROC SORT - One BY-variable (ID)

       ID      visit_date          visit_type

       A01     01/15/2015     Emergency Room Visit
       A02     02/20/2015     Physician Office Visit
       A05     01/12/2015     Outpatient Visit
```

Line 35: With the NODUPKEY option, PROC SORT keeps one observation and deletes all subsequent duplicates by comparing the **variable** (i.e. ID) specified in the BY statement.

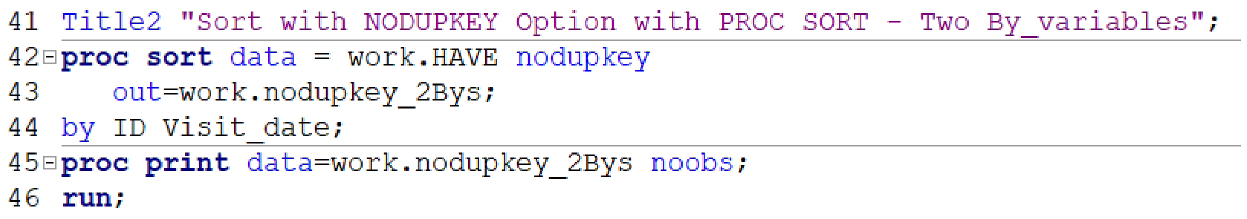
Line 36: The output data set has been named using the OUT= option so that the input data set is not overwritten.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

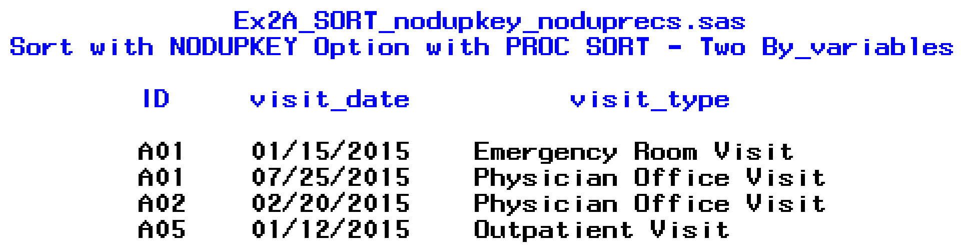
Example Data



```
 ID      visit_date           visit_type

A01     01/15/2015    Emergency Room Visit
A01     07/25/2015    Physician Office Visit
A01     07/25/2015    Physician Office Visit
A02     02/20/2015    Physician Office Visit
A02     02/20/2015    Emergency Room Visit
A05     01/12/2015    Outpatient Visit
```



```
Title2 "Sort with NODUPKEY Option with PROC SORT - Two By_variables";
proc sort data = work.HAVE nodupkey
   out=work.nodupkey_2Bys;
by ID Visit_date;
proc print data=work.nodupkey_2Bys noobs;
run;
```



```
                Ex2A_SORT_nodupkey_noduprecs.sas
Sort with NODUPKEY Option with PROC SORT - Two By_variables

         ID      visit_date           visit_type

         A01     01/15/2015    Emergency Room Visit
         A01     07/25/2015    Physician Office Visit
         A02     02/20/2015    Physician Office Visit
         A05     01/12/2015    Outpatient Visit
```

With the NODUPKEY option, PROC SORT keeps one observation and deletes all subsequent duplicates, if any, by comparing **two variables** specified in the BY statement.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Example Data

```
 ID     visit_date          visit_type

 A01    01/15/2015    Emergency Room Visit
 A01    07/25/2015    Physician Office Visit
 A01    07/25/2015    Physician Office Visit
 A02    02/20/2015    Physician Office Visit
 A02    02/20/2015    Emergency Room Visit
 A05    01/12/2015    Outpatient Visit
```

```
48 Title2 "NODUPRECS Option with PROC SORT  Only One BY-Variable (ID)";
49 proc sort data = work.HAVE noduprecs
50    out=work.noduprec_obs
51    DUPOUT=work.dupoutobs ;
52   BY ID ;
53 run;
54 proc print data=work.dupoutobs noobs;
55 run;
56 proc print data=work.noduprec_obs noobs;
57 run;
```

```
                Ex2A_SORT_nodupkey_noduprecs.sas
 NODUPRECS Option with PROC SORT  Only One BY-Variable (ID)

        ID     visit_date          visit_type

        A01    07/25/2015    Physician Office Visit
```

```
                Ex2A_SORT_nodupkey_noduprecs.sas
 NODUPRECS Option with PROC SORT  Only One BY-Variable (ID)

        ID     visit_date          visit_type

        A01    01/15/2015    Emergency Room Visit
        A01    07/25/2015    Physician Office Visit
        A02    02/20/2015    Physician Office Visit
        A02    02/20/2015    Emergency Room Visit
        A05    01/12/2015    Outpatient Visit
```

Line 49: With the NODUPRECS option, PROC SORT deletes the duplicate observation for all variables in the input data set, not writing that observation to the output data set. NODUPRECS checks only consecutive observations. NODUP is an alias for NODUPRECS.

Line 50: The OUT= option creates a SAS data set (i.e., work.noduprec_obs) that only includes nonduplicate observations.

Line 51: The DUPOUT= identifies a temporary data set (i.e., work.dupoutobs) that is created by the SORT procedure with duplicate records deleted by the NODUPREC option.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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.   Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

NOUNIQUEKEYS Option and OUT= Keyword with PROC SORT (Starting SAS® 9.3)

Example Data

```
 ID      visit_date            visit_type

A01     01/15/2015     Emergency Room Visit
A01     07/25/2015     Physician Office Visit
A01     07/25/2015     Physician Office Visit
A02     02/20/2015     Physician Office Visit
A02     02/20/2015     Emergency Room Visit
A05     01/12/2015     Outpatient Visit
```

```
60  ** New options with PROC SORT;
61 proc sort data = have nouniquekeys
62           out = duplicates
63           uniqueout = singles;
64  by ID Visit_date visit_type;
65  Title2 "NOUNIQUEKEYS and UNIQUEOUT OptionS with PROC SORT (Duplicates)";
66 proc print data=duplicates noobs; run;
67  Title2 "NOUNIQUEKEYS and UNIQUEOUT OptionS with PROC SORT (Singles)";
68 proc print data=singles noobs; run;
```

```
                 Ex2A_SORT_nodupkey_noduprecs.sas
NOUNIQUEKEYS and UNIQUEOUT OptionS with PROC SORT (Duplicates)

         ID      visit_date            visit_type

         A01     07/25/2015     Physician Office Visit
         A01     07/25/2015     Physician Office Visit
```

```
                Ex2A_SORT_nodupkey_noduprecs.sas
NOUNIQUEKEYS and UNIQUEOUT OptionS with PROC SORT (Singles

        ID      visit_date            visit_type

        A01     01/15/2015     Emergency Room Visit
        A02     02/20/2015     Emergency Room Visit
        A02     02/20/2015     Physician Office Visit
        A05     01/12/2015     Outpatient Visit
```

Line 61: The NOUNIQUEKEYS option deletes observations from the output SAS data set where the value of the BY-variable(s) is unique.

Line 62: The OUT= keyword stores observations with non-unique values of the BY-variables in an output SAS data set (i.e. duplicates). A BY-group is a group that is formed by one or more observations with the same value of the BY variables.

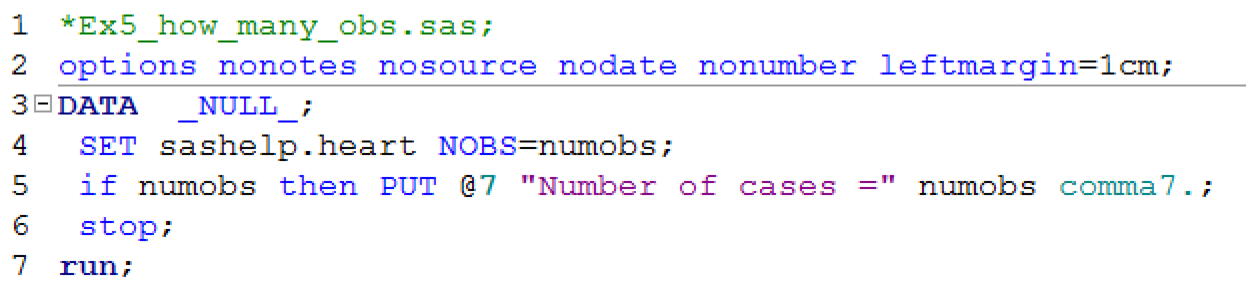
Line 63: The option UNIQUEOUT= specifies an output SAS data set (i.e., singles) containing the observations eliminated by the NOUNIQUEKEYS option.

[SAS® Documentation]

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Determining the Number of Observations in a SAS Data Set

Method 1



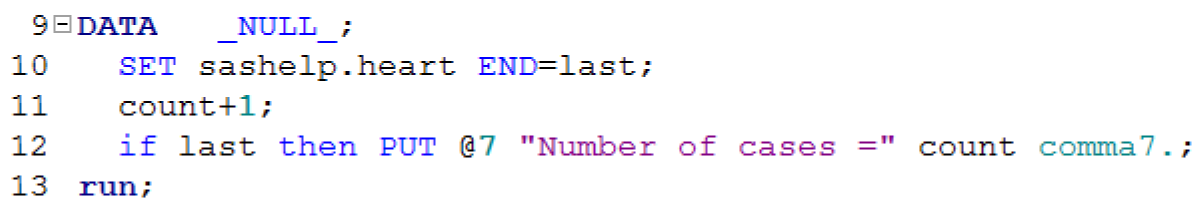
```
1  *Ex5_how_many_obs.sas;
2  options nonotes nosource nodate nonumber leftmargin=1cm;
3  DATA  _NULL_;
4   SET sashelp.heart NOBS=numobs;
5   if numobs then PUT @7 "Number of cases =" numobs comma7.;
6   stop;
7  run;
```

Line 4: The NOBS= option assigns the number of observations in the SAS data set to a temporary variable **numobs**.

(Partial SAS Log)

```
Number of cases =  5,209
```

Method 2



```
 9 DATA   _NULL_;
10    SET sashelp.heart END=last;
11    count+1;
12    if last then PUT @7 "Number of cases =" count comma7.;
13 run;
```

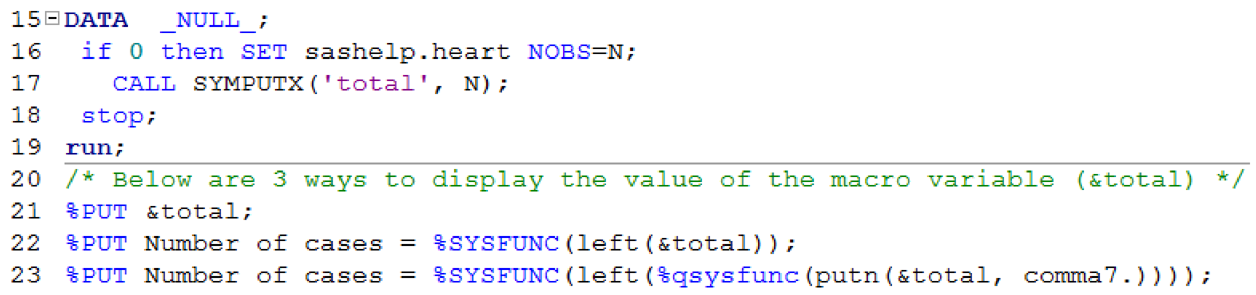
Line 10: With the END = option, SAS identifies the last observation processed by a SET statement. It creates a temporary variable **last** whose value is set to 0 for every observation except for the last observation in the data set. When the last observation is read, this temporary variable value is set to 1.

(Partial SAS Log)

```
Number of cases =  5,209
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Method 3

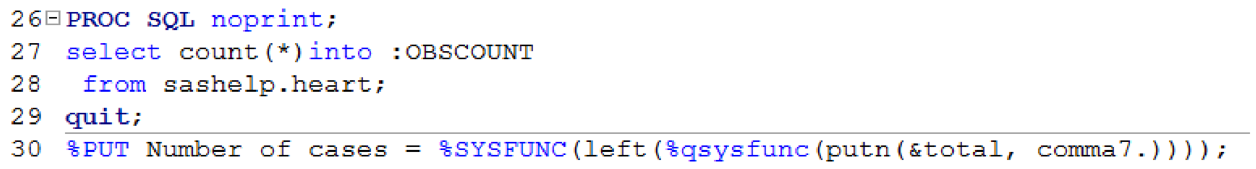


```
DATA  _NULL_;
 if 0 then SET sashelp.heart NOBS=N;
   CALL SYMPUTX('total', N);
 stop;
run;
/* Below are 3 ways to display the value of the macro variable (&total) */
%PUT &total;
%PUT Number of cases = %SYSFUNC(left(&total));
%PUT Number of cases = %SYSFUNC(left(%qsysfunc(putn(&total, comma7.))));
```

(Partial SAS Log)

```
5209
Number of cases = 5209
Number of cases = 5,209
```

Method 4



```
PROC SQL noprint;
select count(*)into :OBSCOUNT
 from sashelp.heart;
quit;
%PUT Number of cases = %SYSFUNC(left(%qsysfunc(putn(&total, comma7.))));
```

(Partial SAS Log)

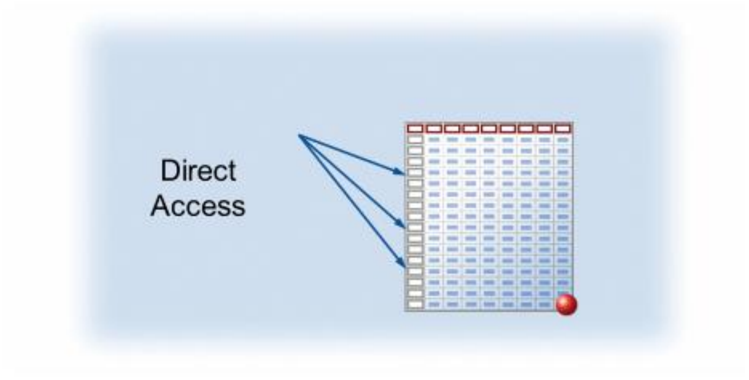
```
Number of cases = 5,209
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Accessing Observations

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

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



7

Using the DATA Step with the NOBS= Option

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

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

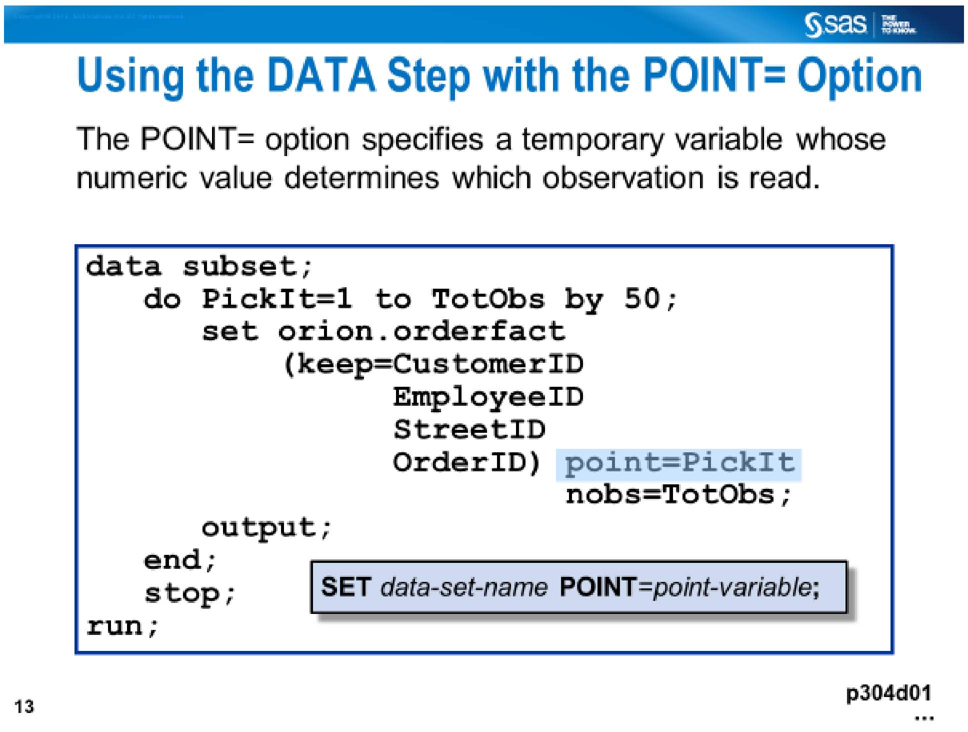
SET *data-set-name* **NOBS**=*observation-number*;

12

p304d01

...

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Using the DATA Step with the POINT= Option

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

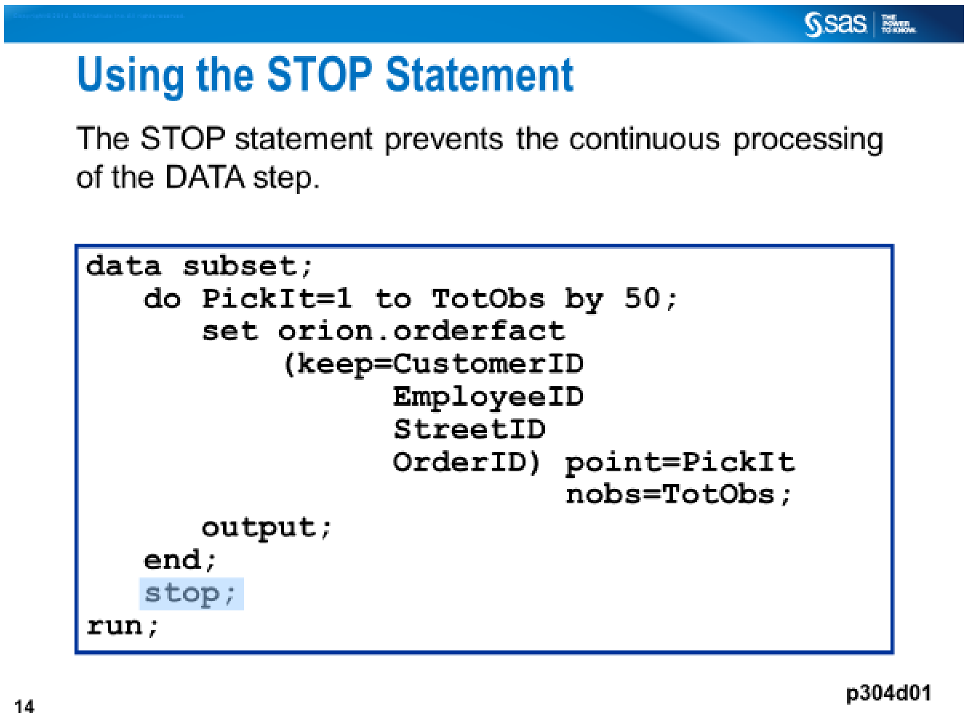
```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.orderfact
          (keep=CustomerID
                EmployeeID
                StreetID
                OrderID) point=PickIt
                         nobs=TotObs;
      output;
   end;
   stop;
run;
```

SET *data-set-name* **POINT=***point-variable***;**

13

p304d01

...



Using the STOP Statement

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

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

14

p304d01

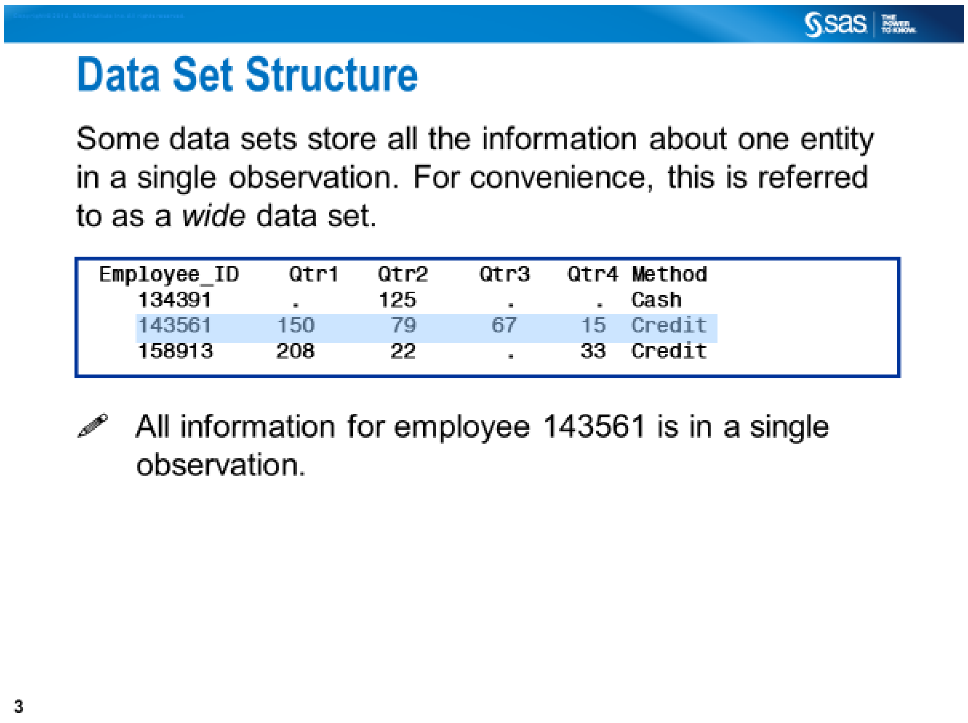
`Ex3_Direct_Access.sas`
`Ex4_sample_select.sas`

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Transposing Data

Transposing means converting rows (i.e., observations) to columns (variables) or vice versa. This can be accomplished using at least through

- Data Step (ARRAY statement and DO Loop)
- PROC TRANSPOSE



Data Set Structure

Some data sets store all the information about one entity in a single observation. For convenience, this is referred to as a *wide* data set.

```
Employee_ID     Qtr1     Qtr2      Qtr3     Qtr4 Method
   134391          .      125         .        . Cash
   143561        150       79        67       15 Credit
   158913        208       22         .       33 Credit
```

All information for employee 143561 is in a single observation.

3

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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.   Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Data Set Structure

Other data sets have multiple observations per entity. For convenience, this is referred to as a *narrow* data set.

```
Employee_ID  Period   Amount
     134391  Qtr2        125
     143561  Qtr1        150
     143561  Qtr2         79
     143561  Qtr3         67
     143561  Qtr4         15
     158913  Qtr1        208
     158913  Qtr2         22
```

✐ The information for employee 143561 is stored in four observations. Each observation represents a donation for a different quarter.

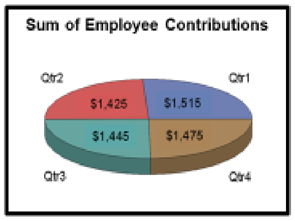
4

Business Scenario: Reports

The Orion Payroll Manager asked for a report showing the number of Orion Star employees who made charitable donations each quarter and a pie chart with quarterly sum of the contributions.

Sketch of the Desired Reports

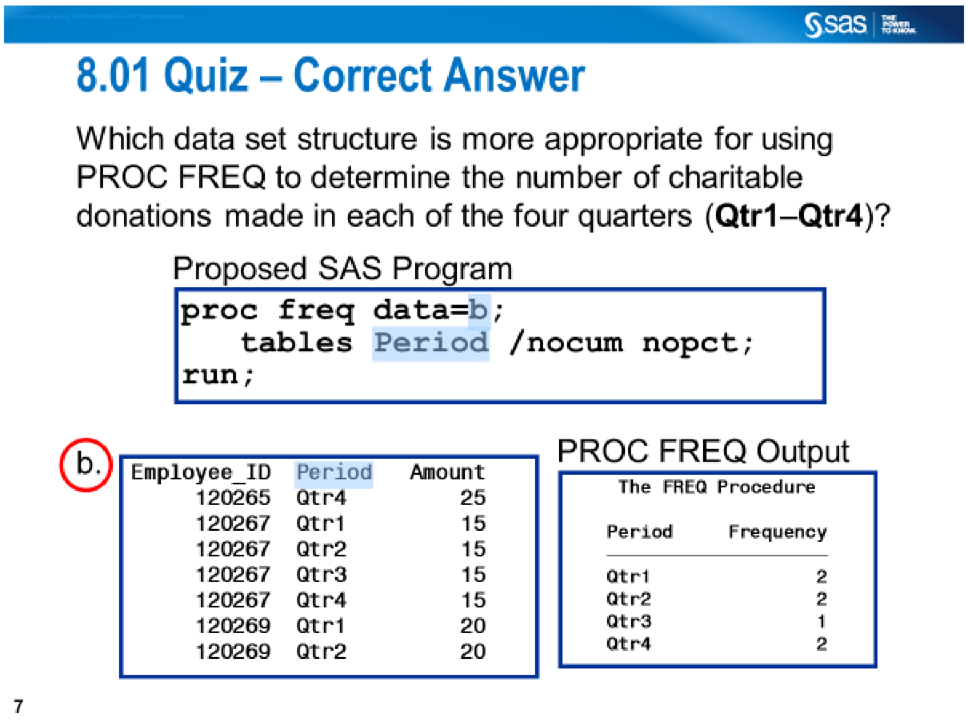
Period	Frequency
Qtr1	56
Qtr2	99
Qtr3	24
Qtr4	75



✐ The FREQ and GCHART procedures can be used to generate the desired reports.

5

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



8.01 Quiz – Correct Answer

Which data set structure is more appropriate for using PROC FREQ to determine the number of charitable donations made in each of the four quarters (**Qtr1–Qtr4**)?

Proposed SAS Program

```
proc freq data=b;
   tables Period /nocum nopct;
run;
```

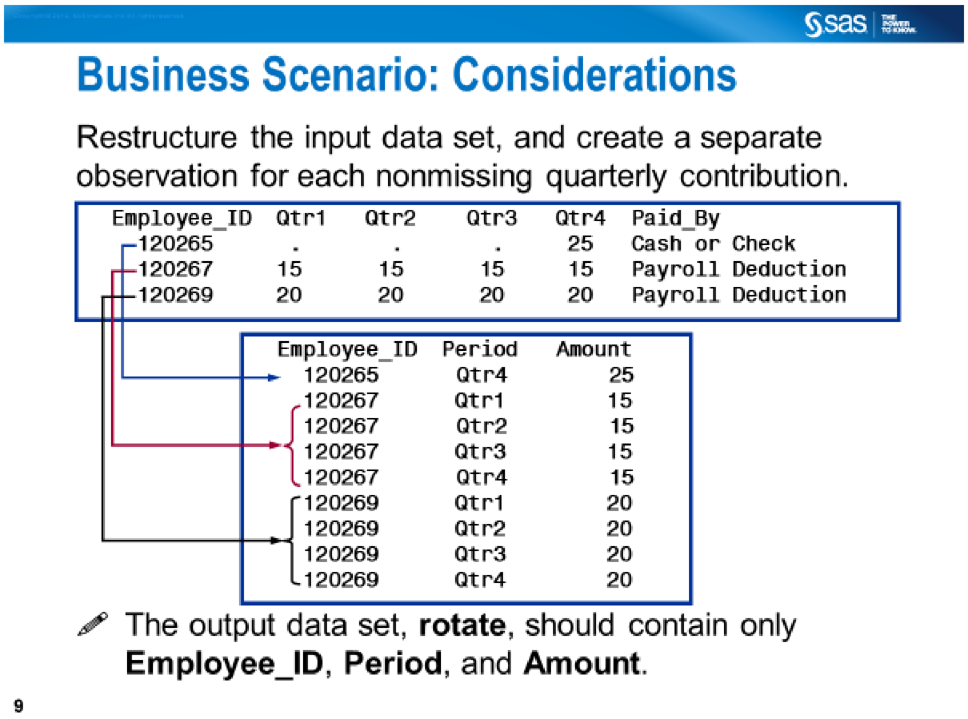
[✓ b.]

```
Employee_ID  Period    Amount
     120265  Qtr4          25
     120267  Qtr1          15
     120267  Qtr2          15
     120267  Qtr3          15
     120267  Qtr4          15
     120269  Qtr1          20
     120269  Qtr2          20
```

PROC FREQ Output

```
    The FREQ Procedure

  Period      Frequency
  ---------------------
  Qtr1                2
  Qtr2                2
  Qtr3                1
  Qtr4                2
```

7



Business Scenario: Considerations

Restructure the input data set, and create a separate observation for each nonmissing quarterly contribution.

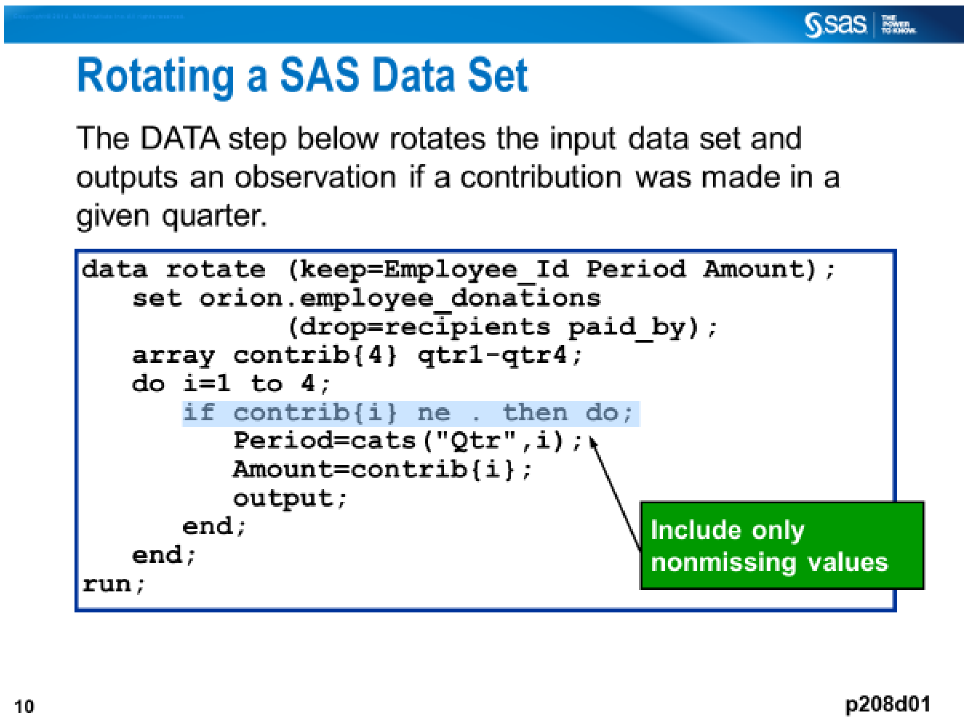
```
Employee_ID  Qtr1   Qtr2    Qtr3   Qtr4  Paid_By
   120265      .      .       .     25   Cash or Check
   120267     15     15      15     15   Payroll Deduction
   120269     20     20      20     20   Payroll Deduction
```

```
Employee_ID  Period    Amount
   120265     Qtr4        25
   120267     Qtr1        15
   120267     Qtr2        15
   120267     Qtr3        15
   120267     Qtr4        15
   120269     Qtr1        20
   120269     Qtr2        20
   120269     Qtr3        20
   120269     Qtr4        20
```

The output data set, **rotate**, should contain only **Employee_ID**, **Period**, and **Amount**.

9

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Rotating a SAS Data Set

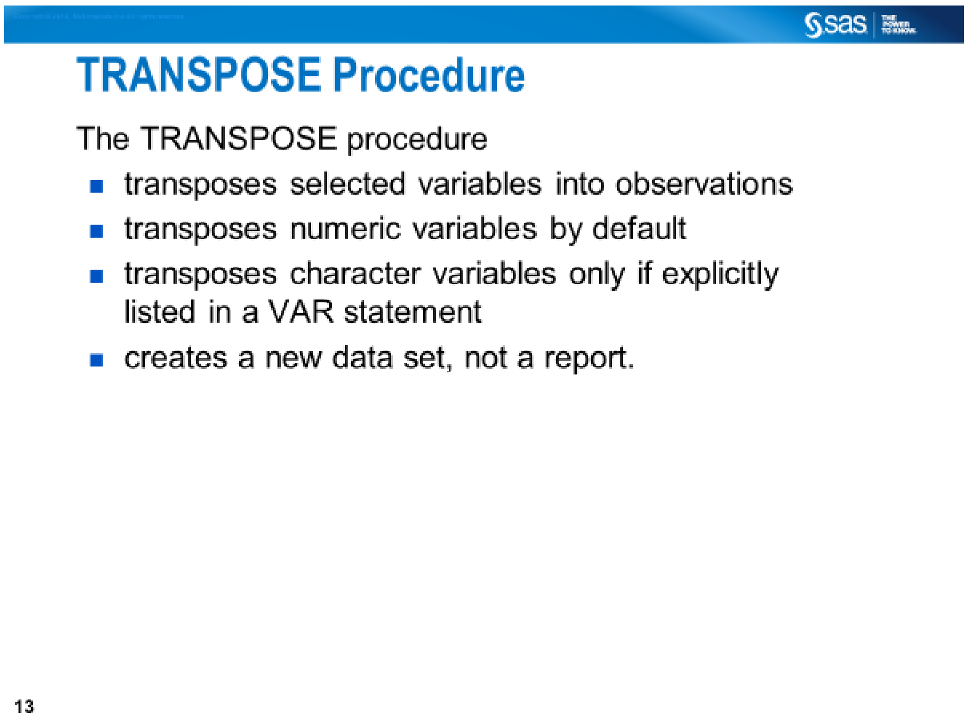
The DATA step below rotates the input data set and outputs an observation if a contribution was made in a given quarter.

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
      if contrib{i} ne . then do;
         Period=cats("Qtr",i);
         Amount=contrib{i};
         output;
      end;
   end;
run;
```

Include only nonmissing values

10 p208d01

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

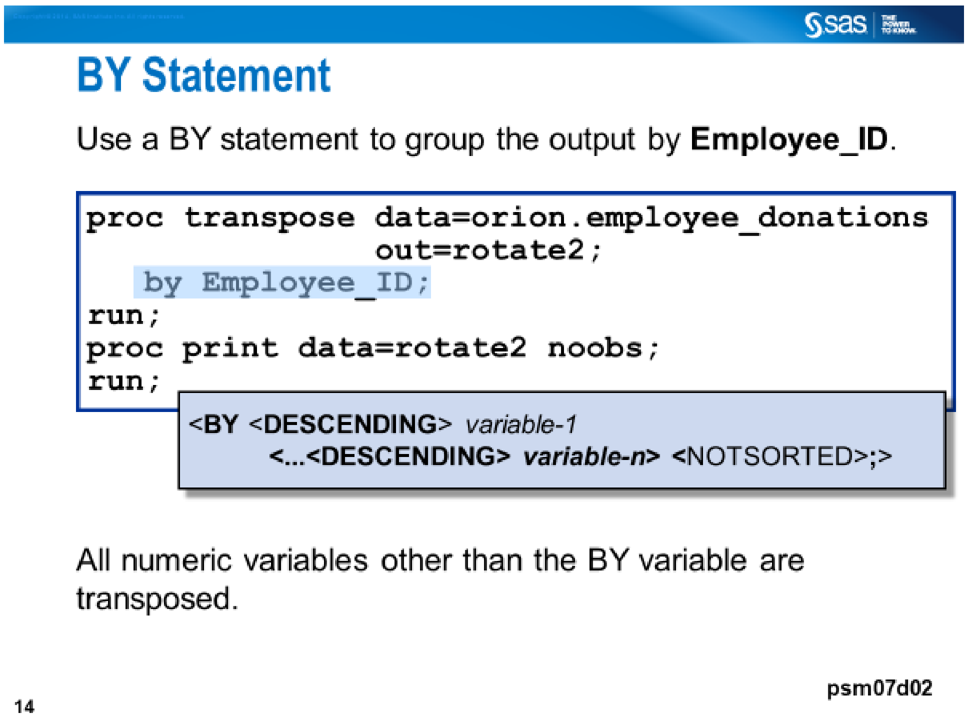


TRANSPOSE Procedure

The TRANSPOSE procedure

- transposes selected variables into observations
- transposes numeric variables by default
- transposes character variables only if explicitly listed in a VAR statement
- creates a new data set, not a report.

13



BY Statement

Use a BY statement to group the output by **Employee_ID**.

```
proc transpose data=orion.employee_donations
               out=rotate2;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

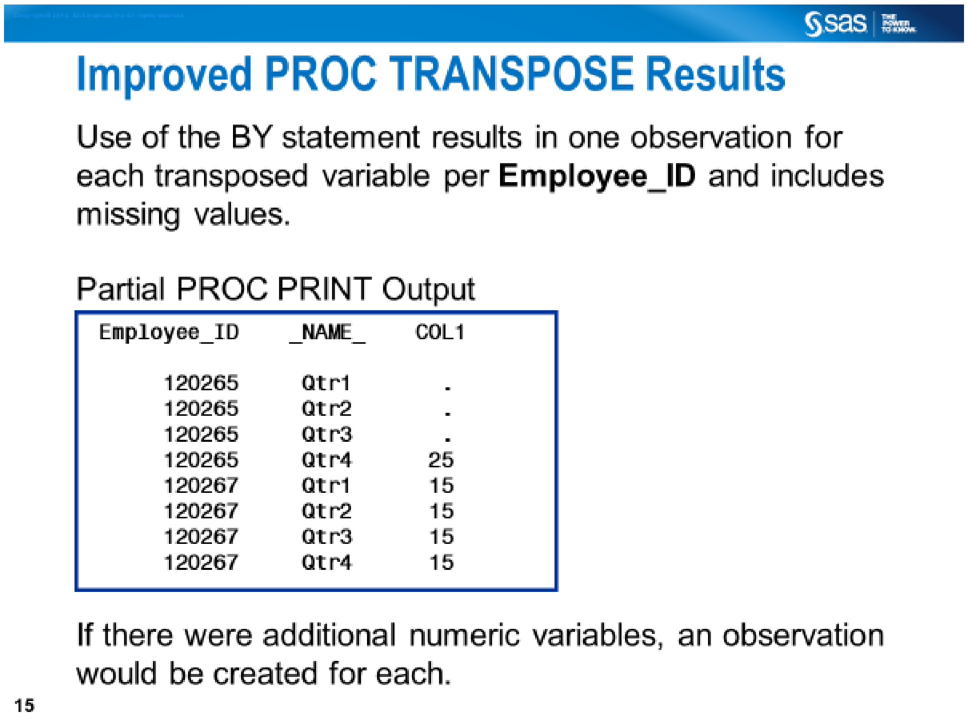
<**BY** <**DESCENDING**> *variable-1*
    <...<**DESCENDING**> ***variable-n***> <NOTSORTED>;>

All numeric variables other than the BY variable are transposed.

psm07d02

14

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Improved PROC TRANSPOSE Results

Use of the BY statement results in one observation for each transposed variable per **Employee_ID** and includes missing values.

Partial PROC PRINT Output

```
 Employee_ID     _NAME_     COL1

      120265      Qtr1        .
      120265      Qtr2        .
      120265      Qtr3        .
      120265      Qtr4       25
      120267      Qtr1       15
      120267      Qtr2       15
      120267      Qtr3       15
      120267      Qtr4       15
```

If there were additional numeric variables, an observation would be created for each.

15

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

VAR Statement

The VAR statement is used to specify which variables to transpose. It can include character and numeric variables.

```
proc transpose data=orion.employee_donations
               out=rotate2;
   by Employee_ID;
   var Qtr1-Qtr4;
run;
proc print data=rotate2 noobs;
run;
```

<**VAR** *variable(s)*;>

psm07d03

16

Enhancing PROC TRANSPOSE Results

The final step is to change the default names of the new variables.

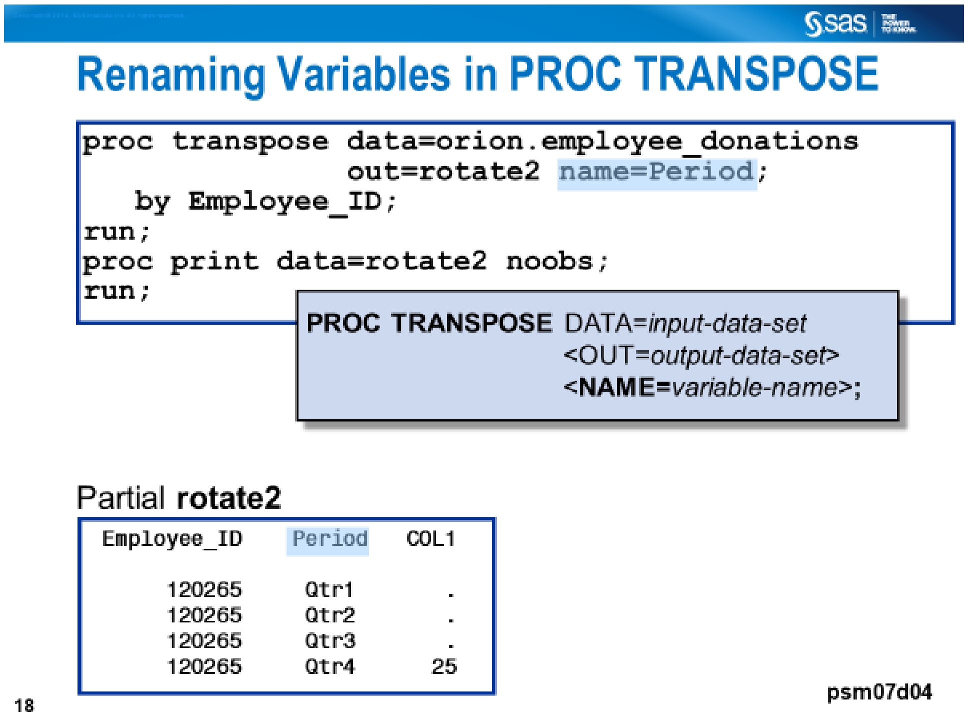
Partial PROC PRINT Output

```
Employee_ID    _NAME_     COL1

   120265       Qtr1        .
   120265       Qtr2        .
   120265       Qtr3        .
   120265       Qtr4       25
   120267       Qtr1       15
   120267       Qtr2       15
   120267       Qtr3       15
   120267       Qtr4       15
```

- Change **_NAME_** to **Period**.
- Change **COL1** to **Amount**.

17

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Renaming Variables in PROC TRANSPOSE

```
proc transpose data=orion.employee_donations
               out=rotate2 name=Period;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

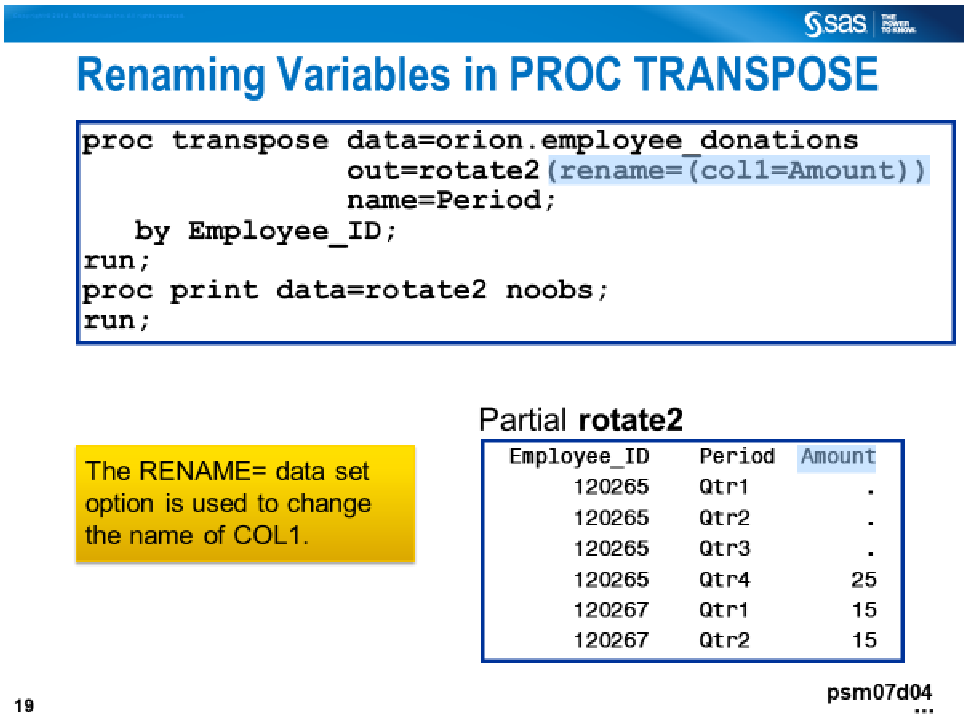
PROC TRANSPOSE DATA=*input-data-set*
<OUT=*output-data-set*>
<**NAME=***variable-name*>;

Partial **rotate2**

```
Employee_ID    Period    COL1

   120265       Qtr1        .
   120265       Qtr2        .
   120265       Qtr3        .
   120265       Qtr4       25
```

18

psm07d04



Renaming Variables in PROC TRANSPOSE

```
proc transpose data=orion.employee_donations
               out=rotate2 (rename=(col1=Amount))
               name=Period;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

The RENAME= data set option is used to change the name of COL1.

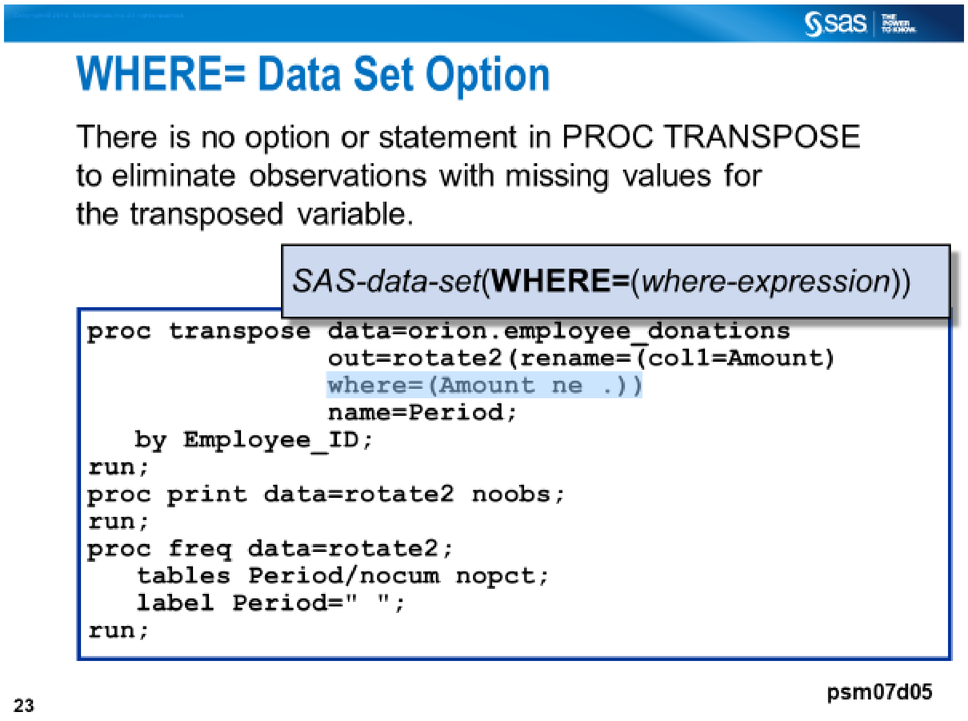
Partial **rotate2**

```
Employee_ID    Period    Amount
   120265       Qtr1        .
   120265       Qtr2        .
   120265       Qtr3        .
   120265       Qtr4       25
   120267       Qtr1       15
   120267       Qtr2       15
```

19

psm07d04
...

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



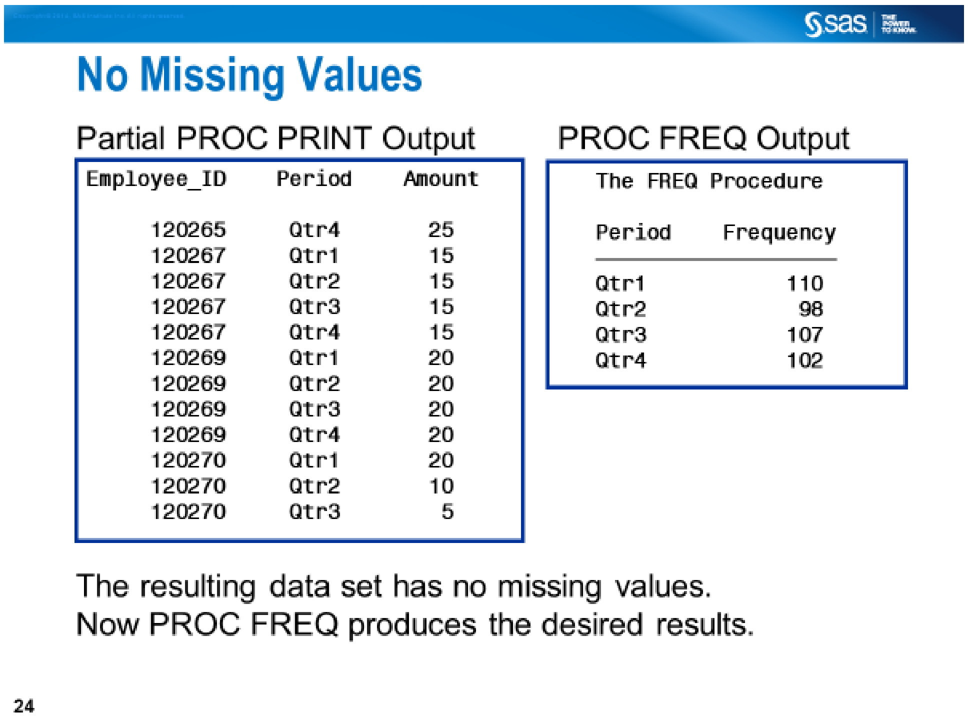
WHERE= Data Set Option

There is no option or statement in PROC TRANSPOSE to eliminate observations with missing values for the transposed variable.

SAS-data-set(**WHERE=**(*where-expression*))

```
proc transpose data=orion.employee_donations
               out=rotate2(rename=(col1=Amount)
               where=(Amount ne .))
               name=Period;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
proc freq data=rotate2;
   tables Period/nocum nopct;
   label Period=" ";
run;
```

23 psm07d05



No Missing Values

Partial PROC PRINT Output

```
Employee_ID    Period    Amount

     120265     Qtr4        25
     120267     Qtr1        15
     120267     Qtr2        15
     120267     Qtr3        15
     120267     Qtr4        15
     120269     Qtr1        20
     120269     Qtr2        20
     120269     Qtr3        20
     120269     Qtr4        20
     120270     Qtr1        20
     120270     Qtr2        10
     120270     Qtr3         5
```

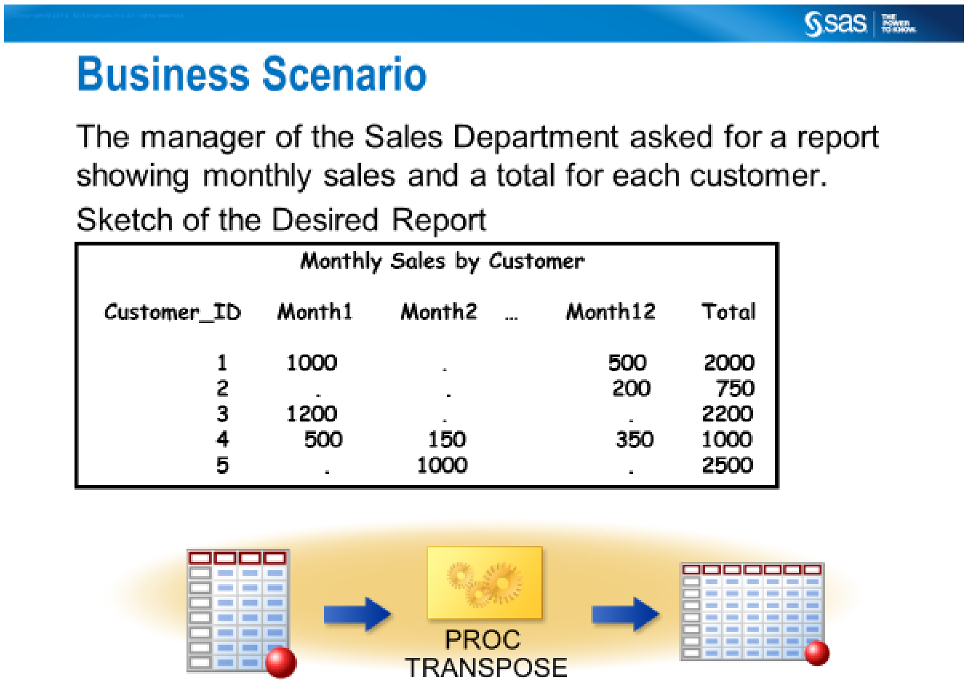
PROC FREQ Output

```
 The FREQ Procedure

 Period     Frequency
 --------------------
 Qtr1            110
 Qtr2             98
 Qtr3            107
 Qtr4            102
```

The resulting data set has no missing values.
Now PROC FREQ produces the desired results.

24

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Business Scenario

The manager of the Sales Department asked for a report showing monthly sales and a total for each customer.

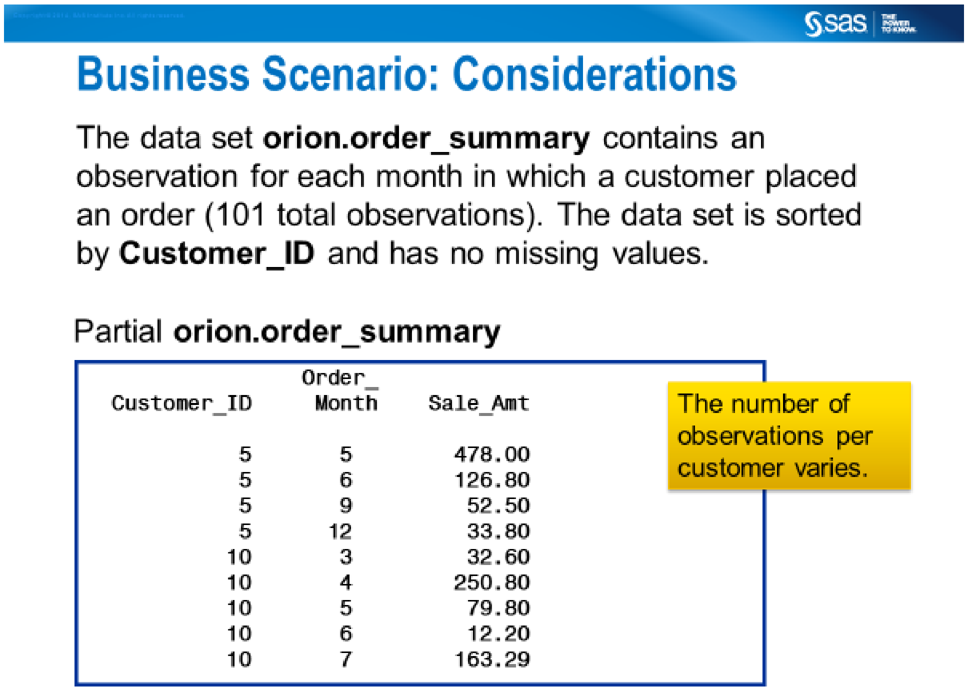
Sketch of the Desired Report

Monthly Sales by Customer

Customer_ID	Month1	Month2	...	Month12	Total
1	1000	.		500	2000
2	.	.		200	750
3	1200	.		.	2200
4	500	150		350	1000
5	.	1000		.	2500

PROC TRANSPOSE

26



Business Scenario: Considerations

The data set **orion.order_summary** contains an observation for each month in which a customer placed an order (101 total observations). The data set is sorted by **Customer_ID** and has no missing values.

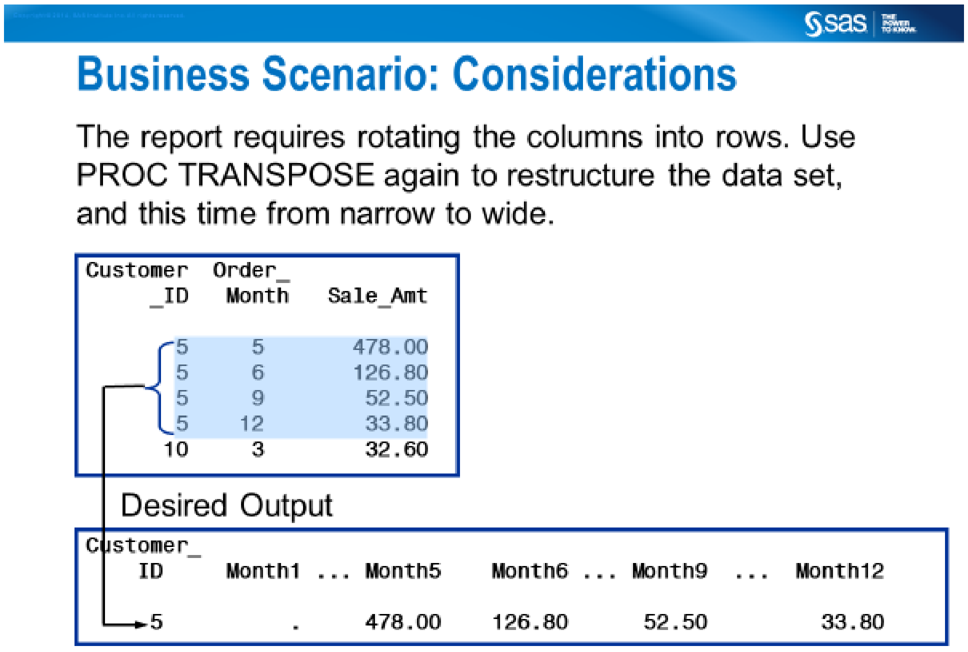
Partial **orion.order_summary**

Customer_ID	Order_ Month	Sale_Amt
5	5	478.00
5	6	126.80
5	9	52.50
5	12	33.80
10	3	32.60
10	4	250.80
10	5	79.80
10	6	12.20
10	7	163.29

The number of observations per customer varies.

27

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Business Scenario: Considerations

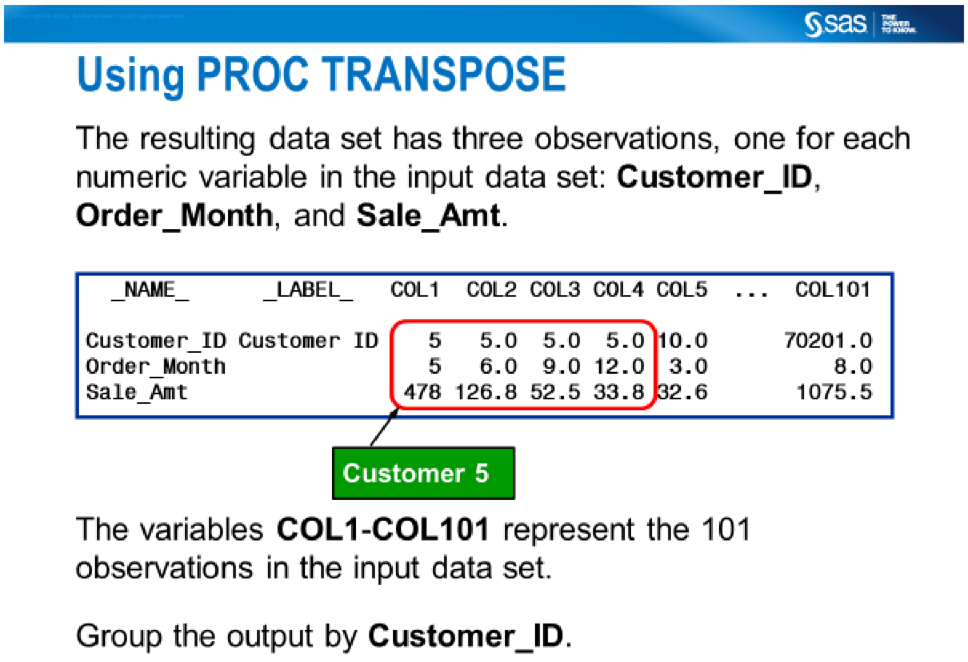
The report requires rotating the columns into rows. Use PROC TRANSPOSE again to restructure the data set, and this time from narrow to wide.

```
Customer  Order_
     _ID   Month    Sale_Amt

       5       5      478.00
       5       6      126.80
       5       9       52.50
       5      12       33.80
      10       3       32.60
```

Desired Output

```
Customer_
   ID     Month1 ... Month5    Month6 ... Month9   ...  Month12

    5          .     478.00    126.80      52.50          33.80
```

28



Using PROC TRANSPOSE

The resulting data set has three observations, one for each numeric variable in the input data set: **Customer_ID**, **Order_Month**, and **Sale_Amt**.

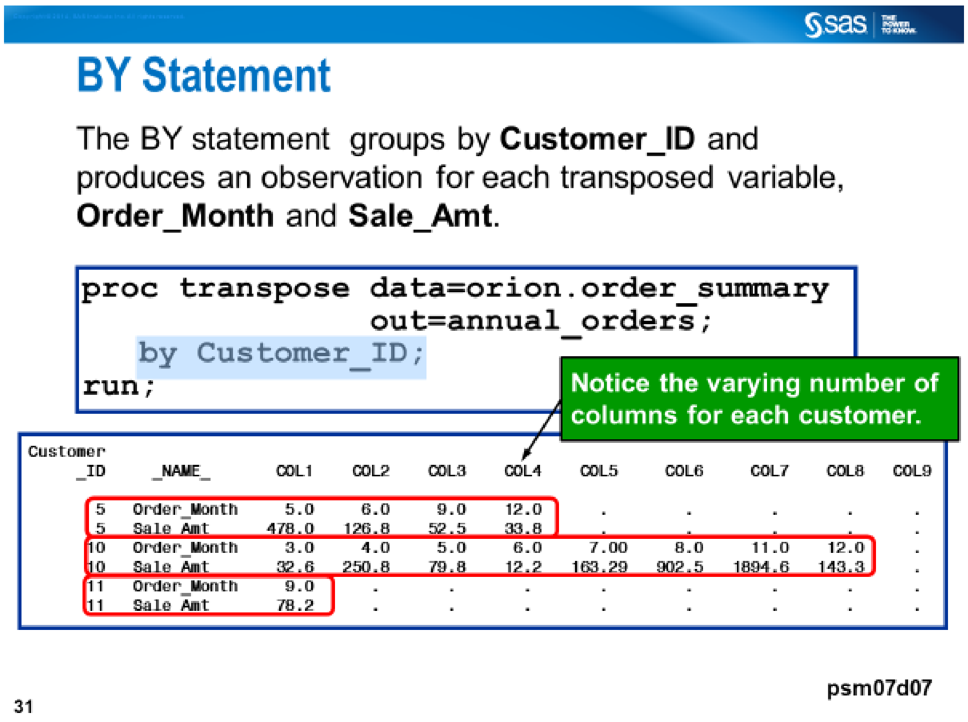
```
  _NAME_        _LABEL_     COL1   COL2  COL3  COL4  COL5   ...   COL101

Customer_ID  Customer ID      5     5.0   5.0   5.0  10.0        70201.0
Order_Month                   5     6.0   9.0  12.0   3.0            8.0
Sale_Amt                    478   126.8  52.5  33.8  32.6         1075.5
```

The variables **COL1-COL101** represent the 101 observations in the input data set.

Group the output by **Customer_ID**.

30

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



BY Statement

The BY statement groups by **Customer_ID** and produces an observation for each transposed variable, **Order_Month** and **Sale_Amt**.

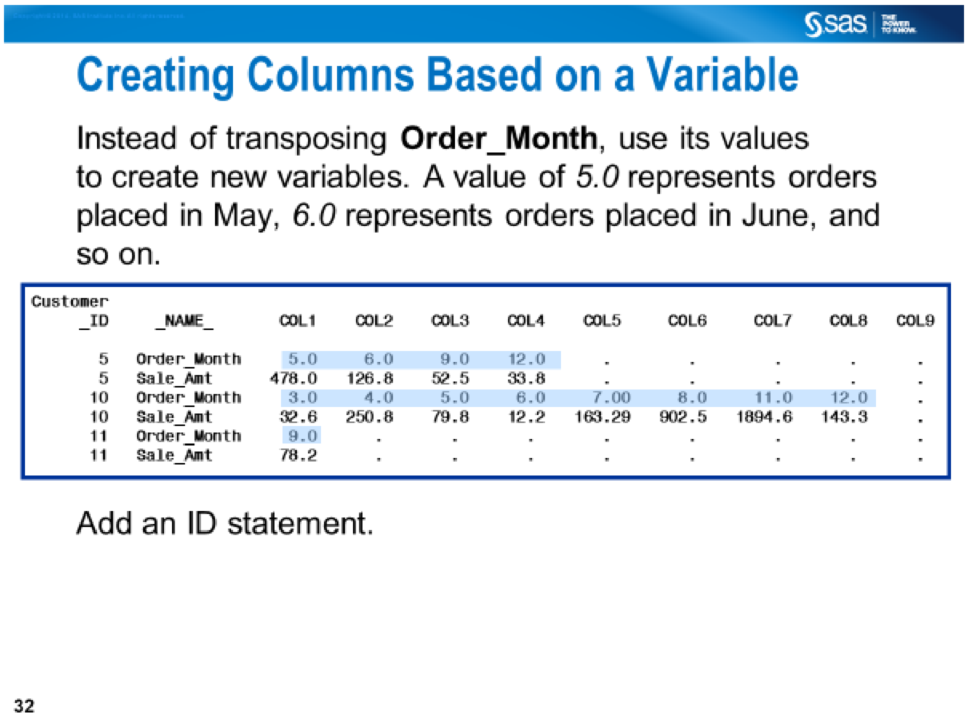
```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
run;
```

Notice the varying number of columns for each customer.

```
Customer
  _ID     _NAME_         COL1    COL2    COL3    COL4    COL5     COL6     COL7     COL8    COL9

   5     Order_Month      5.0     6.0     9.0    12.0     .        .        .        .       .
   5     Sale_Amt       478.0   126.8    52.5    33.8     .        .        .        .       .
  10     Order_Month      3.0     4.0     5.0     6.0    7.00      8.0     11.0     12.0     .
  10     Sale_Amt        32.6   250.8    79.8    12.2  163.29    902.5   1894.6    143.3     .
  11     Order_Month      9.0     .       .       .       .        .        .        .       .
  11     Sale_Amt        78.2     .       .       .       .        .        .        .       .
```

31

psm07d07



Creating Columns Based on a Variable

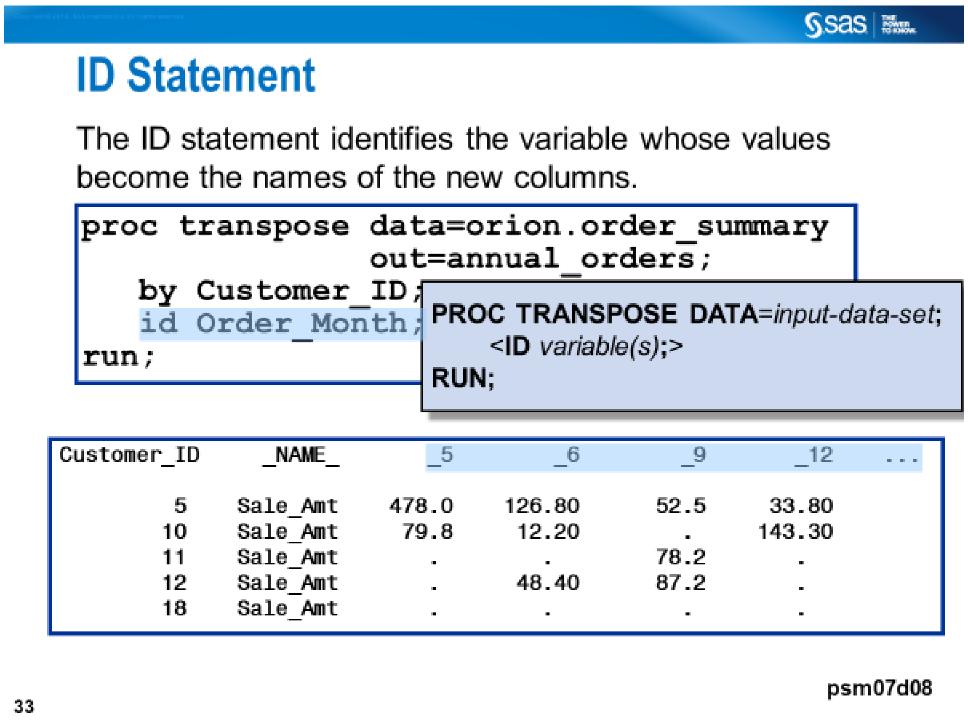
Instead of transposing **Order_Month**, use its values to create new variables. A value of *5.0* represents orders placed in May, *6.0* represents orders placed in June, and so on.

```
Customer
  _ID     _NAME_         COL1    COL2    COL3    COL4    COL5     COL6     COL7     COL8    COL9

   5     Order_Month      5.0     6.0     9.0    12.0     .        .        .        .       .
   5     Sale_Amt       478.0   126.8    52.5    33.8     .        .        .        .       .
  10     Order_Month      3.0     4.0     5.0     6.0    7.00      8.0     11.0     12.0     .
  10     Sale_Amt        32.6   250.8    79.8    12.2  163.29    902.5   1894.6    143.3     .
  11     Order_Month      9.0     .       .       .       .        .        .        .       .
  11     Sale_Amt        78.2     .       .       .       .        .        .        .       .
```

Add an ID statement.

32

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



ID Statement

The ID statement identifies the variable whose values become the names of the new columns.

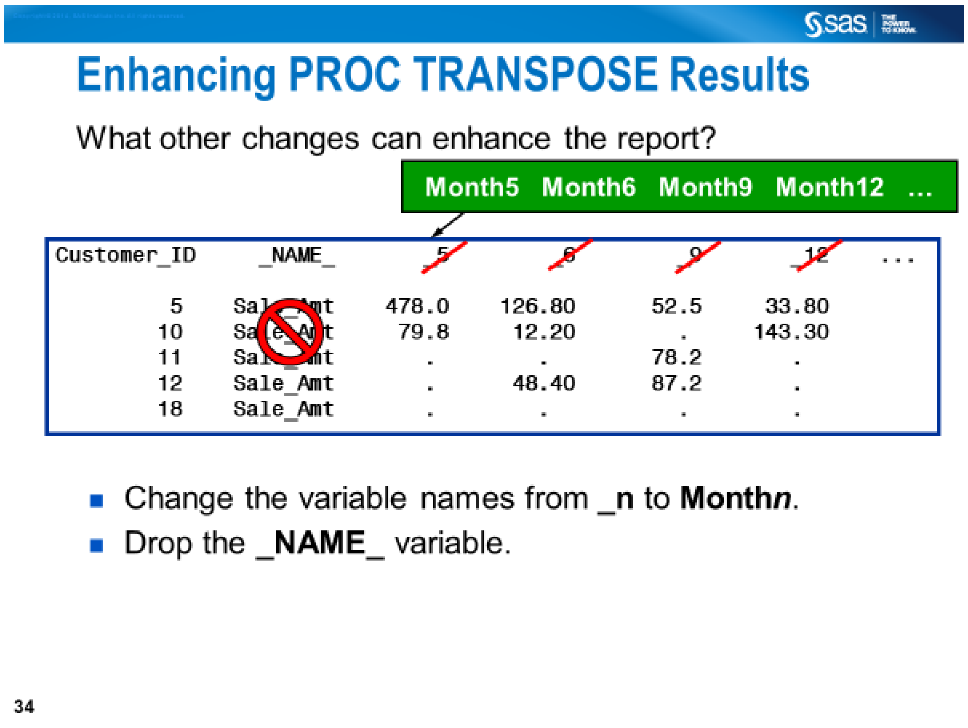
```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
   id Order_Month;
run;
```

PROC TRANSPOSE DATA=*input-data-set***;**
    <ID *variable(s)***;>**
RUN;

```
Customer_ID     _NAME_       _5        _6        _9       _12     ...

       5      Sale_Amt    478.0    126.80      52.5     33.80
      10      Sale_Amt     79.8     12.20        .     143.30
      11      Sale_Amt       .        .        78.2       .
      12      Sale_Amt       .      48.40      87.2       .
      18      Sale_Amt       .        .          .        .
```

33

psm07d08



Enhancing PROC TRANSPOSE Results

What other changes can enhance the report?

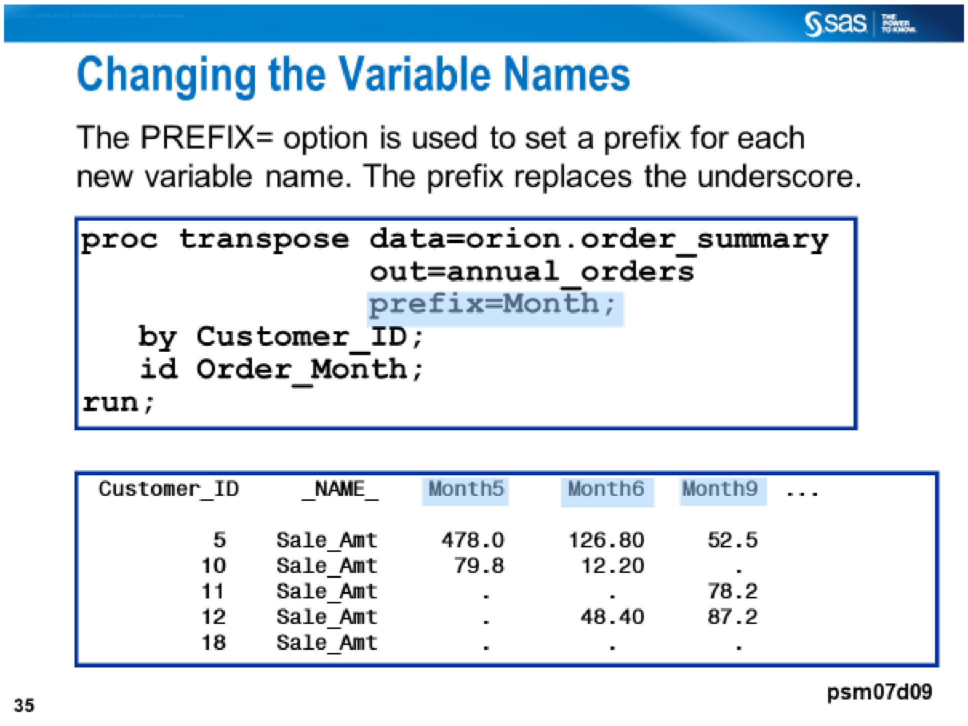
Month5 Month6 Month9 Month12 ...

```
Customer_ID     _NAME_       5         6         9        12     ...

       5      Sale_Amt    478.0    126.80      52.5     33.80
      10      Sale_Amt     79.8     12.20        .     143.30
      11      Sale_Amt       .        .        78.2       .
      12      Sale_Amt       .      48.40      87.2       .
      18      Sale_Amt       .        .          .        .
```

- Change the variable names from **_n** to **Month*n***.
- Drop the **_NAME_** variable.

34

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



Changing the Variable Names

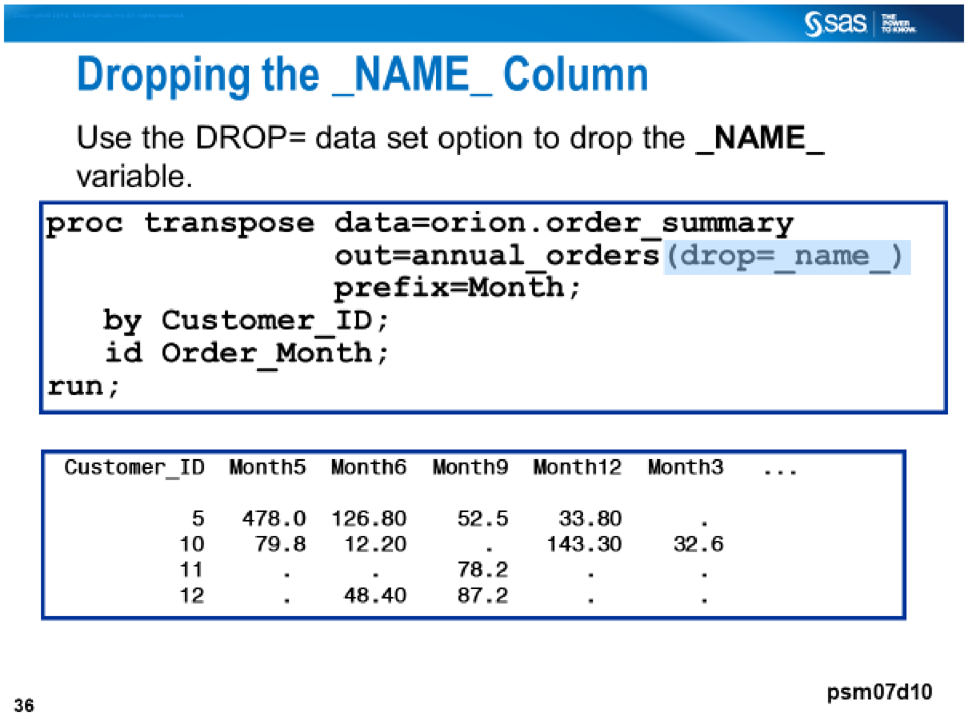
The PREFIX= option is used to set a prefix for each new variable name. The prefix replaces the underscore.

```
proc transpose data=orion.order_summary
               out=annual_orders
               prefix=Month;
   by Customer_ID;
   id Order_Month;
run;
```

```
Customer_ID    _NAME_     Month5     Month6    Month9   ...

         5    Sale_Amt    478.0     126.80     52.5
        10    Sale_Amt     79.8      12.20       .
        11    Sale_Amt       .          .      78.2
        12    Sale_Amt       .       48.40     87.2
        18    Sale_Amt       .          .        .
```

35

psm07d09



Dropping the _NAME_ Column

Use the DROP= data set option to drop the **_NAME_** variable.

```
proc transpose data=orion.order_summary
               out=annual_orders(drop=_name_)
               prefix=Month;
   by Customer_ID;
   id Order_Month;
run;
```

```
Customer_ID  Month5  Month6  Month9  Month12  Month3   ...

          5   478.0  126.80    52.5    33.80     .
         10    79.8   12.20      .    143.30   32.6
         11      .       .     78.2      .       .
         12      .    48.40    87.2      .       .
```

36

psm07d10

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Example on PROC TRANSPOSE

```
 1  *Ex6_long_to_wide.sas;
 2  options nocenter nodate;
 3  data long;
 4   input year $ gdp cpi ;
 5   datalines;
 6  2010    101.226 218.056
 7  2011    103.315 224.939
 8  2012    105.220 229.594
 9  2013    106.935 232.957
10  2014    108.694 236.736
11  2015    109.782 237.017
12  ;
```

`* Ex6_long_to_wide.sas;`

```
19  proc transpose data=LONG out=wide1;run;
20  proc print data=WIDE1 noobs; run;
```

```
_NAME_      COL1       COL2       COL3       COL4       COL5       COL6

 year     2010.00    2011.00    2012.00    2013.00    2014.00    2015.00
 gdp       101.23     103.32     105.22     106.94     108.69     109.78
 cpi       218.06     224.94     229.59     232.96     236.74     237.02
```

Line 19: The values of all numeric variables are transposed in the output data set.

`* Ex6_long_to_wide.sas;`

```
23  proc transpose data=LONG out=wide2;
24  var GDP; run;
25  proc print data=WIDE2 noobs; run;
```

Line 24: Only one numeric variable is declared in the VAR statement. Therefore, the values of that variable (i.e., GDP) only are transposed in the output data set. Because there is no ID statement added to PROC TRANSPOSE, the transposed columns are named as COL1- COL10.

```
_NAME_      COL1       COL2      COL3       COL4       COL5       COL6

 gdp      101.226    103.315    105.22    106.935    108.694    109.782
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

```
* Ex6_long_to_wide.sas;
```

```
31 proc transpose data=LONG out=wide3;
32 id year;
33 var GDP; run;
34 proc print data=WIDE3 noobs; run;
```

Line 32: Since the variable in the ID statement is numeric, the values are printed with an underscore (_).

```
_NAME_      _2010      _2011      _2012      _2013      _2014      _2015

 gdp       101.226    103.315    105.22    106.935    108.694    109.782
```

```
* Ex6_long_to_wide.sas;
```

```
36 proc transpose data=LONG out=wide4 prefix=Year;
37 id year;
38 var GDP CPI ; run;
39 proc print data=WIDE4 noobs; run;
```

Line 36: The PREFIX= option is declared in the PROC statement to attach a prefix to the value of the variable (YEAR) in the ID statement.

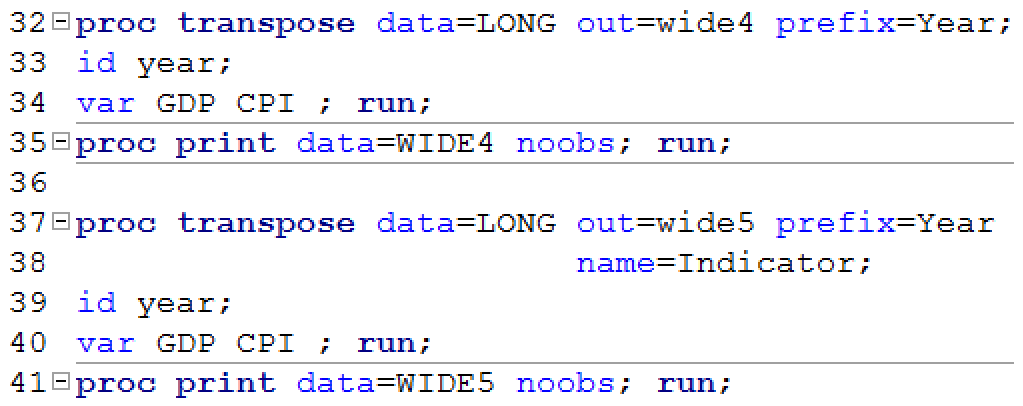
Line 38: There are only two numeric variables other than the YEAR variable, which is in the ID statement, so we could omit the VAR statement. But, we have kept the VAR statement here (no harm!) for syntax-description purposes.

NAME is an automatic variable in the output data set that holds the variable name(s) in the input data set from which the values originate.

```
_NAME_     Year2010    Year2011    Year2012    Year2013    Year2014    Year2015

 gdp        101.226     103.315     105.220     106.935     108.694     109.782
 cpi        218.056     224.939     229.594     232.957     236.736     237.017
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

*** Ex6_long_to_wide.sas;**

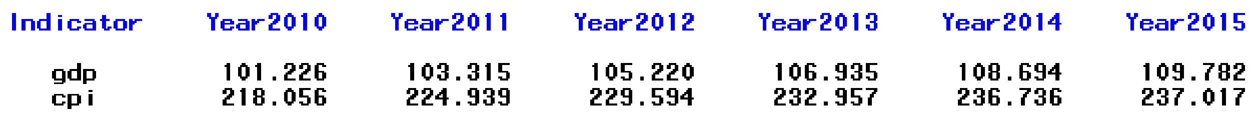


```
32 proc transpose data=LONG out=wide4 prefix=Year;
33 id year;
34 var GDP CPI ; run;
35 proc print data=WIDE4 noobs; run;
36
37 proc transpose data=LONG out=wide5 prefix=Year
38                              name=Indicator;
39 id year;
40 var GDP CPI ; run;
41 proc print data=WIDE5 noobs; run;
```

Line 37: Remember that SAS automatically adds _NAME_ in the output data set that holds the variable name(s) in the input data set from which the values originate. We have used the NAME= option to replace _NAME_. The _NAME_ is now named as “Indicator” in the output data set (shown below).

Line 39: As stated earlier, the ID statement is used specify a variable whose values name the transposed variables. The column used for the ID statement cannot have any duplicate values (Zirbel, 2009).

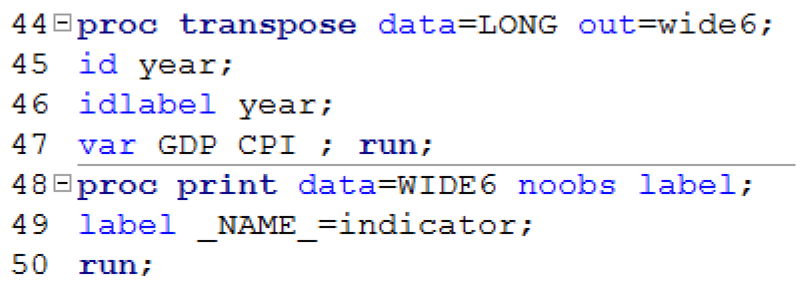
Line 40: As noted earlier, the VAR statement is used to list the variables that need to be transposed.



```
Indicator    Year2010    Year2011    Year2012    Year2013    Year2014    Year2015

   gdp        101.226     103.315     105.220     106.935     108.694     109.782
   cpi        218.056     224.939     229.594     232.957     236.736     237.017
```

Acknowledgments: Portions of SAS’ copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

*** Ex6_long_to_wide.sas;**



```
44 proc transpose data=LONG out=wide6;
45  id year;
46  idlabel year;
47  var GDP CPI ; run;
48 proc print data=WIDE6 noobs label;
49  label _NAME_=indicator;
50  run;
```

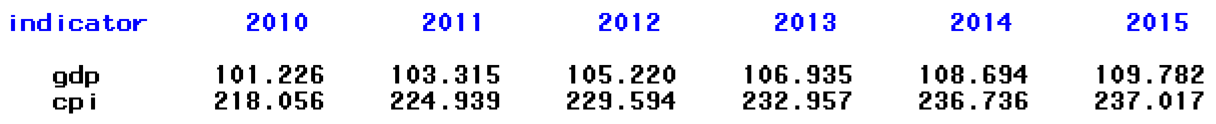
Line 45: As stated earlier, the ID statement is used specify a variable whose values name the transposed variables. The column used for the ID statement cannot have any duplicate values (Zirbel, 2009).

Line 46: The IDLABEL statement is used to create labels for the transposed variables.

Line 47: As noted earlier, the VAR statement is used to list the variables that need to be transposed.

Line 49: The LABEL statement in the PROC PRINT step assigns a temporary label INDICATOR to the automatic variable _NAME_ in the output data set. As note earlier, this automatic variable holds the variable name(s) in the input data set from which the values originate.

Note that the NAME= option with PROC TRANSPOSE does not work when the IDLABEL statement is added to the step.



```
indicator      2010       2011       2012       2013       2014       2015

   gdp       101.226    103.315    105.220    106.935    108.694    109.782
   cpi       218.056    224.939    229.594    232.957    236.736    237.017
```

Also see the example code Ex7_wide_to_long.sas on GitHub.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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.   Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

Double Transposition

http://support.sas.com/kb/44/637.html

Sample *44637:* Double PROC TRANSPOSE method for reshaping your data set with multiple BY variables

"You might have the need to reshape your data set to get one unique row for each unique value of a BY variable. When a second BY variable is involved, you can use a double PROC TRANSPOSE method to reshape your data set." (SAS® Documentation).

For example, you have this data set (see the instream data in **`Ex8_multi_transpose_x.sas – next page)`.**

```
family_
  id          month      Ins_paid      copay

 F002           1           350          60
 F002           2           100          30
 F002           3            88          20
 F002           4            20           0
 F002           5           450          90
 F002           6            70          30
 F001           1           245          60
 F001           2           100           0
 F001           3             0           0
 F001           4           120          30
 F001           5           345          60
 F001           6            95          30
```

You want to convert the above data set to the following data set.

```
family_   Ins_             Ins_             Ins_             Ins_             Ins_             Ins_
  id      paid1   copay1   paid2   copay2   paid3   copay3   paid4   copay4   paid5   copay5   paid6   copay6

 F001      245      60      100       0        0       0      120      30      345      60       95      30
 F002      350      60      100      30       88      20       20       0      450      90       70      30
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

```
13 proc sort data=have;  by family_id; run;
14 proc transpose data= have
15             out=have_t name=stat; ;
16     by family_id month;
17     var Ins_paid copay;
18 run;
19 proc print data=have_t noobs; run;
20 proc contents data=have_t varnum; run;
```

```
family_
  id        month      stat        COL1

 F001         1        Ins_paid     245
 F001         1        copay         60
 F001         2        Ins_paid     100
 F001         2        copay          0
 F001         3        Ins_paid       0
 F001         3        copay          0
 F001         4        Ins_paid     120
 F001         4        copay         30
 F001         5        Ins_paid     345
 F001         5        copay         60
 F001         6        Ins_paid      95
 F001         6        copay         30
 F002         1        Ins_paid     350
 F002         1        copay         60
 F002         2        Ins_paid     100
 F002         2        copay         30
 F002         3        Ins_paid      88
 F002         3        copay         20
 F002         4        Ins_paid      20
 F002         4        copay          0
 F002         5        Ins_paid     450
 F002         5        copay         90
 F002         6        Ins_paid      70
 F002         6        copay         30
```

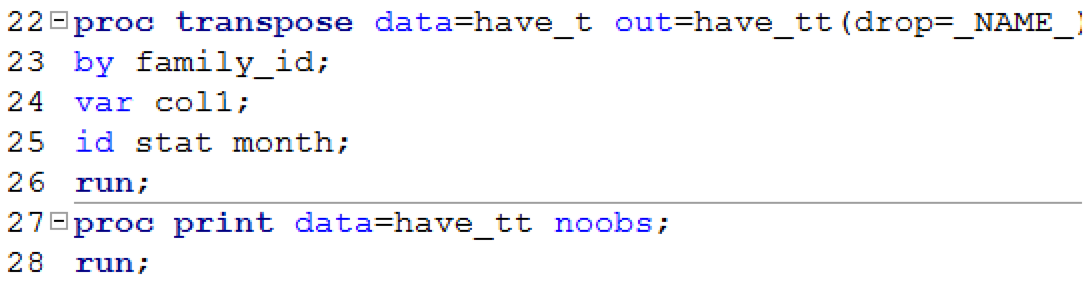
Line 15: We have used the NAME= option to replace _NAME_. The _NAME_ is now named as “stat”. Note that the “stat” variable is labelled as NAME OF FORMER VARIABLE by default (see PROC CONTENTS output below).

Line 16: We want SAS to transpose the input the data file per BY group (i.e., the combination of family_id and month). This required sorting the data using the BY variable as done prior to the TRANSPOSE step in line 13.

Line 17: As noted earlier, the VAR statement is used to list the variables that need to be transposed.

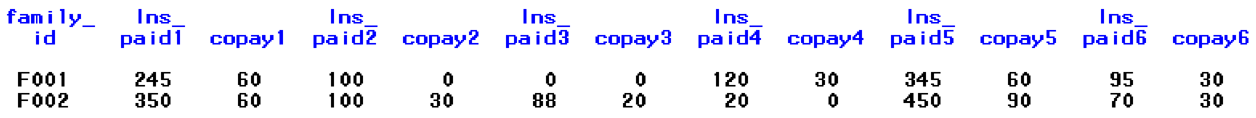
```
                 Variables in Creation Order

 #    Variable      Type    Len    Label

 1    family_id     Char      8
 2    month         Num       8
 3    stat          Char      8    NAME OF FORMER VARIABLE
 4    COL1          Num       8
```

Acknowledgments: Portions of SAS’ copyrighted SAS course content are reproduced here 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.   Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.



```
22 proc transpose data=have_t out=have_tt(drop=_NAME_
23 by family_id;
24 var col1;
25 id stat month;
26 run;
27 proc print data=have_tt noobs;
28 run;
```

Line 25: The ID statement includes multiple variables. The column used for the ID statement cannot have any duplicate values.



```
family_   Ins_             Ins_             Ins_             Ins_             Ins_             Ins_
  id     paid1   copay1   paid2   copay2   paid3   copay3   paid4   copay4   paid5   copay5   paid6   copay6

 F001     245      60      100       0       0        0      120      30      345      60       95      30
 F002     350      60      100      30      88       20       20       0      450      90       70      30
```

PROC DATASETS

The DATASETS procedure can be used to do the following among many tasks:

- list data sets in the memory in the existing SAS session
- manage SAS data sets (e.g., copying, updating, deleting indexes, catalogs)
- rename variables in a SAS data set
- add/change formats and labels, etc. to a SAS data set
- remove formats from a SAS data set

Ex13_proc_datasets.sas

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.

SAS Transport Files

According to the SAS Institute, SAS transport files are the "best overall format" for interfacing with other systems because they are consistent across all host environments. SAS transport files can be converted into a variety of system files (e.g., SAS, Stata, and R).

https://meps.ahrq.gov/data_stats/download_data_files.jsp

There are many SAS transport files (public-use files) available for download from the MEPS (The Medical Expenditure Panel Survey) web site.

"The Medical Expenditure Panel Survey, which began in 1996, is a set of large-scale surveys of families and individuals, their medical providers (doctors, hospitals, pharmacies, etc.), and employers across the United States. MEPS collects data on the specific health services that Americans use, how frequently they use them, the cost of these services, and how they are paid for, as well as data on the cost, scope, and breadth of health insurance held by and available to U.S. workers."

`Ex11_download_unzip_create.sas`

- can be used to download/unzip a single SAS transport file from the above web site and convert into a SAS data set
-

`Ex12_download_unzip_create_macro.sas` can be used to:

- download multiple SAS transport files from the MEPS website
- unzip those files
- convert the SAS transport files into SAS data sets

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here 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. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts.