UCD School of Mathematics and Statistics

STAT40840: Data programming with SAS

Laura Kirwan

Lecture 3

Lecture 3: Producing Reports

3.1 Subsetting Report Data

3.2 Sorting and Grouping Report Data

3.3 Enhancing Reports

Objectives – Part 1

- Create a default PROC PRINT report.
- Select variables with a VAR statement.
- Calculate totals with a SUM statement.
- Select observations with a WHERE statement.
- Define a date constant.
- Identify observations with an ID statement.

Orion Star management wants a report that displays the names, salaries, and a salary total for all sales employees.

orion.sales

PROC PRINT

Obs	Last_Name	First_Name	Salary
1	xxxxxxx	xxxxxxxxxx	99999
2	xxxxxxx	xxxxxxxxxx	99999
3	xxxxxxx	xxxxxxxxxx	99999

			99999

PRINT Procedure

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

```
proc print data=orion.sales;
run;
```

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_ Date	Hire_ Date
1	120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
3	120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114
4	120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-1984	6756
5	120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	1732	9405

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

L2_D1.sas

VAR Statement

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

```
proc print data=orion.sales;
   var Last_Name First_Name Salary;
run;
```

VAR *variable(s)*;

L2_D1.sas

SUM Statement

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

```
proc print data=orion.sales;
   var Last_Name First_Name Salary;
   sum Salary;
run;
```

SUM *variable(s)*;

L2_D1.sas

Viewing the Log

Partial SAS Log

```
84    proc print data=orion.sales;
85       var Last_Name First_Name Salary;
86       sum salary;
87    run;

NOTE: There were 165 observations read from the data set ORION.SALES.
```

- The order of statements in a SAS procedure is usually not important.

Orion Star management wants a report that displays the names and salaries of the sales employees who earn less than $25,500. Suppress the Obs column.

orion.sales

PROC PRINT

Last_Name	First_Name	Salary
xxxxxxx	xxxxxxxx	25000
xxxxxxx	xxxxxxxx	20000
xxxxxxx	xxxxxxxx	23000

WHERE Statement

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

```
proc print data=orion.sales;
    var Last_Name First_Name Salary;
    where Salary<25500;
run;
```

WHERE *WHERE-expression*;

L3_D2.sas

Viewing the Log

Only 7 of the 165 observations from **orion.sales** were selected by the WHERE statement.

```
295  proc print data=orion.sales;
296     var Last_Name First_Name Salary;
297     where Salary<25500;
298  run;

NOTE: There were 7 observations read from the data set ORION.SALES.
      WHERE Salary<25500;
```

Viewing the Output

PROC PRINT Output

Obs	Last_ Name	First_ Name	Salary
49	Tilley	Kimiko	25185
50	Barcoe	Selina	25275
85	Anstey	David	25285
104	Voron	Tachaun	25125
111	Polky	Asishana	25110
131	Ould	Tulsidas	22710
148	Buckner	Burnetta	25390

original observation numbers

Suppressing the Obs Column

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

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Salary;
   where Salary<25500;
run;
```

PROC PRINT DATA=*SAS-data-set* **NOOBS;**

L3_D2.sas

WHERE Statement

The WHERE expression defines the condition (or conditions) for selecting observations.

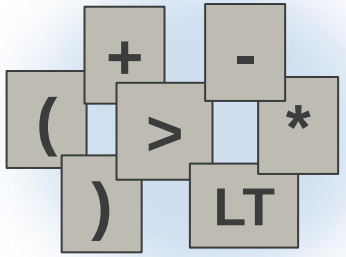
WHERE *WHERE-expression*;

Operands

- character constants
- numeric constants
- date constants
- character variables
- numeric variables

Operators

- symbols that represent a comparison, calculation, or logical operation



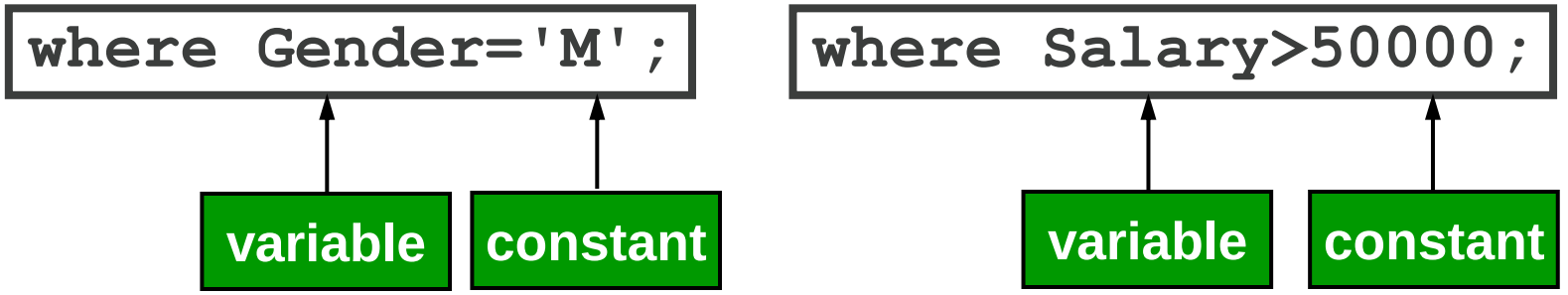
- SAS functions
- special WHERE operators

Operands

Constants are fixed values.

- Character values are enclosed in quotation marks and are case sensitive.
- Numeric values do not use quotation marks or special characters.

Variables must exist in the input data set.



SAS Date Constant

A *SAS date constant* is a date written in the following form: **'ddmmm<yy>yy'd**

Examples
'01JAN2000'd
'31Dec11'D
'1jan04'd
'06NOV2000'D

SAS automatically converts a date constant to a SAS date value.

Comparison Operators

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

Symbol	Mnemonic	Definition
=	EQ	Equal to
^= ¬= ~=	NE	Not equal to
>	GT	Greater than
<	LT	Less than
>=	GE	Greater than or equal
<=	LE	Less than or equal
	IN	Equal to one of a list

Comparison Operators

Examples
`where Gender eq ' ';`
`where Salary ne .;`
`where Salary>=50000;`
`where Hire_Date<'01Jan2000'd;`
`where Country in ('AU','US');`
`where Country in ('AU' 'US');`
`where Order_Type in (1,2,3);`

The value list in the IN operator must be enclosed in parentheses and separated by either commas or blanks. Character values must be enclosed in quotation marks.

Exercise 1

Run the program L3_E1.sas

Which of the following is true?

a. The program executes, and applies both WHERE conditions successfully.
b. The program fails and an error message is written to the log.
c. The program executes, but only the first WHERE condition is applied.
d. The program executes, but only the second WHERE condition is applied.

Logical Operators

Logical operators combine or modify WHERE expressions.

```
proc print data=orion.sales;
   where Country='AU' and
         Salary<30000;
run;
```

WHERE *WHERE-expression-1* AND | OR *WHERE-expression-n*;

L3_D3.sas

Viewing the Log

```
67   proc print data=orion.sales;
68      where Country='AU' and
69            Salary<30000;
70   run;

NOTE: There were 51 observations read from the data set ORION.SALES.
WHERE (Country='AU') and (Salary<30000);
```

Logical Operators

The operators can be written as symbols or mnemonics, and parentheses can be added to modify the order of evaluation.

Symbol	Mnemonic	Priority
^ ¬ ~	NOT	I
&	AND	II
|	OR	III

The NOT operator modifies a condition by finding the complement of the specified criteria.

```
where City not in ('London','Rome','Paris');
```

Logical Operators

Examples
`where Country ne 'AU' and Salary>=50000;`
`where Gender eq 'M' or Salary ge 50000;`
`where Country='AU' or Country='US';`
`where Country in ('AU','US');`
`where Country not in ('AU','US');`

equivalent expressions

You should use only one WHERE statement in a step.

Exercise 2

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

- a.
```
where Month in (5-7)
      and Names=.;
```
- b.
```
where Month in (5,6,7)
      and Names=' ';
```
- c.
```
where Month in ('5','6','7')
      and Names='.';
```

Business Scenario

Orion Star management wants a report that lists only the Australian sales representatives.

orion.sales

Last_Name	First_ Name	Country	Job_Title
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxx
xxxxxxxxxx	xxxxxxx	xx	xxxxxxxxxxx

Exploring the Data

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Country
       Job_Title;
run;
```

Partial PROC PRINT Output

```
Plested       Billy      AU    Sales Rep. II
Wills         Matsuoka   AU    Sales Rep. III
George        Vino       AU    Sales Rep. II
Body          Meera      AU    Sales Rep. III
Highpoint     Harry      US    Chief Sales Officer
Magolan       Julienne   US    Sales Rep. II
Desanctis     Scott      US    Sales Rep. IV
Ridley        Cherda     US    Sales Rep. IV
```

L3_D4.sas

Subsetting in a PROC PRINT Step

Include a WHERE statement to subset by **Country** and **Job_Title**.

```
proc print data=orion.sales noobs;
   var Last_Name First_Name Country
       Job_Title;
   where Country='AU' and
         Job_Title contains 'Rep';
run;
```

CONTAINS is a special WHERE operator.

L3_D4.sas

CONTAINS Operator

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

Equivalent Statements
`where Job_Title contains 'Rep';`
`where Job_Title ? 'Rep';`

- ? can be used instead of the mnemonic.
- The position of the substring within the variable's values is not important.
- Comparisons made with the CONTAINS operator are case sensitive.

Viewing the Output

- P

Last_Name	First_ Name	Country	Job_Title
Elvish	Irenie	AU	Sales Rep. II
Ngan	Christina	AU	Sales Rep. II
Hotstone	Kimiko	AU	Sales Rep. I
Daymond	Lucian	AU	Sales Rep. I
Hofmeister	Fong	AU	Sales Rep. IV

Special WHERE Operators

Special WHERE operators are operators that can be used only in WHERE expressions.

Operator	Definition	Char	Num
CONTAINS	Includes a substring	x	
BETWEEN-AND	An inclusive range	x	x
WHERE SAME AND	Augment a WHERE expression	x	x
IS NULL	A missing value	x	x
IS MISSING	A missing value	x	x
LIKE	Matches a pattern	x	

BETWEEN-AND Operator

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

Examples
`where salary between 50000 and 100000;`
`where salary not between 50000 and 100000;`
`where Last_Name between 'A' and 'L';`
`where Last_Name between 'Baker' and 'Gomez';`

BETWEEN-AND Operator

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

WHERE SAME AND Operator

Use the *WHERE SAME AND operator* to add more conditions to an existing WHERE expression.

```
proc print data=orion.sales;
    where Country='AU' and Salary<30000;
    where same and Gender='F';
    var First_Name Last_Name Gender
        Salary Country;
run;
```

The WHERE SAME AND condition ***augments*** the original condition.

L3_D5.sas

Viewing the Log

```
22   proc print data=orion.sales;
23        where Country='AU'  and Salary<30000;
24        where same and Gender='F';
NOTE: WHERE clause has been augmented.
25        var First_Name Last_Name Gender Salary Country;
26   run;

NOTE: There were 23 observations read from the data set ORION.SALES.
      WHERE (Country='AU') and (Gender='F') and (Salary<30000);
```

Exercise 3

Open **L3_D5.sas**. Change WHERE SAME AND to WHERE ALSO. Submit the program and view the log.

What message is written to the log?

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=.;
```

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=.;
```

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 must occupy that position.

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

Exercise 4

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

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

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

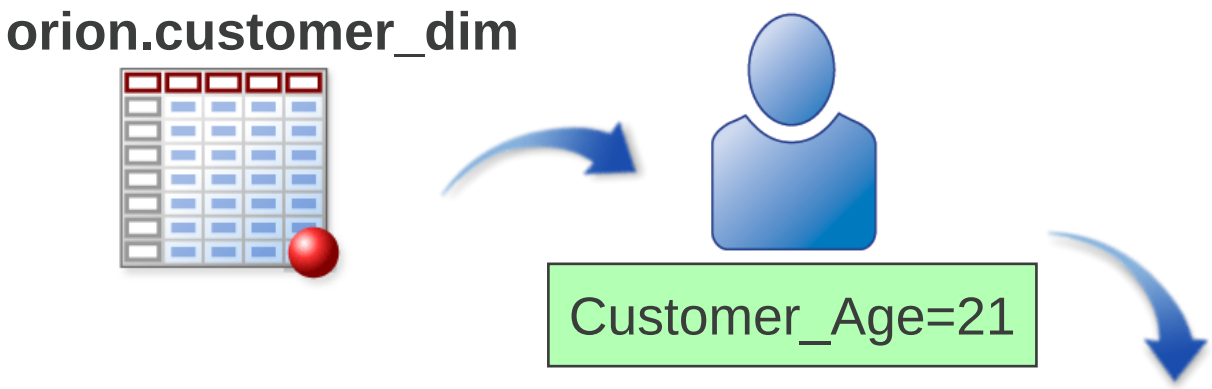
c.
```
where Name like '_, M%';
```

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

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

last name, first name

The Sales Manager wants a report that includes only the customers who are 21 years old.



Obs	Customer_ID	Customer_Name	Customer_ Gender	Customer_ Country	Customer_Group	Customer_ Age_Group	Customer_ Type
1	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX
2	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX
3	999	XXXXXXXXXXXX	X	XX	XXXXXXXXXXXX	XXXXXXXXX	XXXXXXXX

Subsetting the Data Set

Display the required rows and variables.

```
proc print data=orion.customer_dim;
    where Customer_Age=21;
    var Customer_ID Customer_Name
        Customer_Gender Customer_Country
        Customer_Group Customer_Age_Group
        Customer_Type;
run;
```

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

L3_D6.sas

Viewing the Output

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

Obs	Customer_ID	Customer_Name	Customer_ Gender	Customer_ Country	Customer_Group
37	79	Najma Hicks	F	US	Orion Club members
58	11171	Bill Cuddy	M	CA	Orion Club Gold members
66	46966	Lauren Krasowski	F	CA	Orion Club members
...					
76	70210	Alex Santinello	M	CA	Orion Club members

Obs	Customer_ Age_Group	Customer_Type
37	15-30 years	Orion Club members medium activity
58	15-30 years	Orion Club Gold members low activity
66	15-30 years	Orion Club members high activity
...		
76	15-30 years	Orion Club members medium activity

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

ID Statement

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

```
proc print data=orion.customer_dim;
   where Customer_Age=21;
   id Customer_ID;
   var Customer_Name Customer_Gender
       Customer_Country Customer_Group
       Customer_Age_Group Customer_Type;
run;
```

ID *variables*;

Choose ID variables that uniquely identify observations.

L3_D7.sas

Viewing the Output

Customer_ID	Customer_Name	Customer_ Gender	Customer_ Country	Customer_Group
79	Najma Hicks	F	US	Orion Club members
11171	Bill Cuddy	M	CA	Orion Club Gold members
46966	Lauren Krasowski	F	CA	Orion Club members
70079	Lera Knott	F	CA	Orion Club members
70187	Soberina Berent	F	CA	Orion Club members
70210	Alex Santinello	M	CA	Orion Club members

Customer_ID	Customer_ Age_Group		Customer_Type
79	15-30 years	Orion	Club members medium activity
11171	15-30 years	Orion	Club Gold members low activity
46966	15-30 years	Orion	Club members high activity
70079	15-30 years	Orion	Club members medium activity
70187	15-30 years	Orion	Club members medium activity
70210	15-30 years	Orion	Club members medium activity

Lecture 3: Producing Reports

3.1 Subsetting Report Data

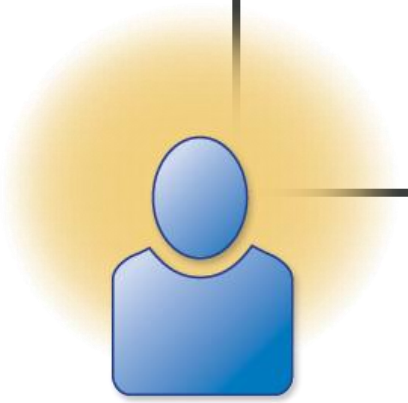
3.2 Sorting and Grouping Report Data

3.3 Enhancing Reports

Objectives – Part 2

- Sort the observations in a SAS data set based on the values of one or more variables.
- Display the sorted observations.
- Display a data set with report totals and subtotals for each BY group.

Display observations from **orion.sales** in ascending order by the variable **Salary**.



Employee_ID	Last_Name	Salary
999999	xxxxxxxxxx	99999
999999	xxxxxxxxxx	99999
999999	xxxxxxxxxx	99999

Creating a Sorted Report

Step 1 Use the SORT procedure to create a new data set, **work.sales**. Order the observations by the value of **Salary**.

Creating a Sorted Report

Step 2 Use the PRINT procedure to display the sorted data set, **work.sales**.

Step 1: SORT Procedure

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

```
proc sort data=orion.sales
          out=work.sales;
   by Salary;
run;
```

L3_D8.sas

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

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

Viewing the Log

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

Partial SAS Log

```
34   proc sort data=orion.sales
35             out=work.sales;
36      by Salary;
37   run;

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

Step 2: Viewing the Output

```
proc print data=work.sales noobs;
   var Employee_ID Last_Name Salary;
run;
```

Partial PROC PRINT Output

Employee_ID	Last_Name	Salary
121084	Ould	22710
121064	Polky	25110
121057	Voron	25125
...		
121143	Favaron	95090
120102	Zhou	108255
120261	Highpoint	243190

L3_D8.sas

SORT Procedure

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

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

Exercise 5

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

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

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

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

Exercise 5

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

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

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

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

Creating a Grouped Report

Step 1	Use the SORT procedure to group data in a data set. This scenario requires two variables to be sorted: ■ **Country** ■ descending **Salary** within **Country**
Step 2	Use a BY statement in PROC PRINT to display the sorted observations grouped by **Country**.

Step 1: Sort the Data

Sort the data set to group the observations.

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

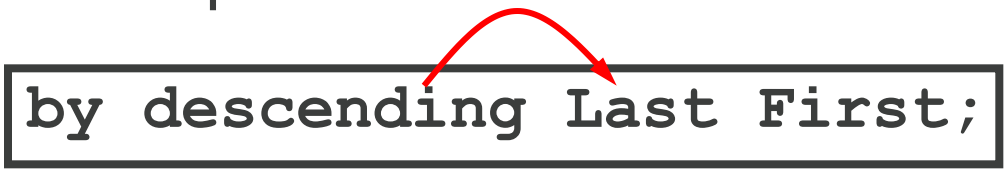
BY <**DESCENDING**> *variables*;

L3_D9.sas

Specifying Sort Order

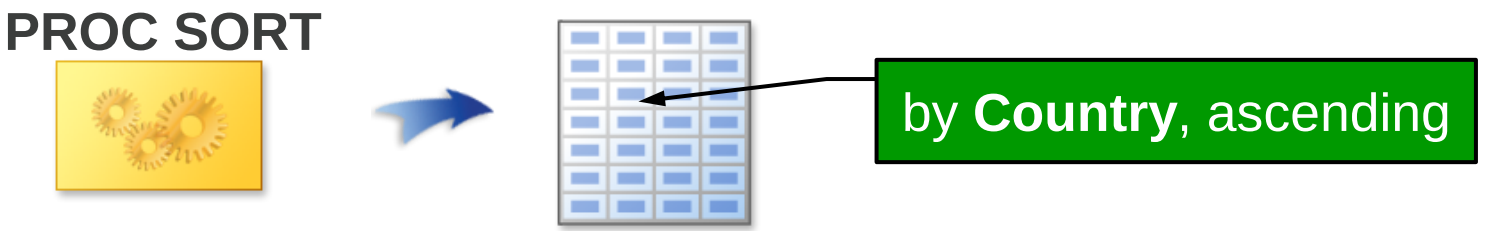
The *DESCENDING option* reverses the sort order for the variable that immediately follows it. The observations are sorted from the largest value to the smallest value.

Examples:

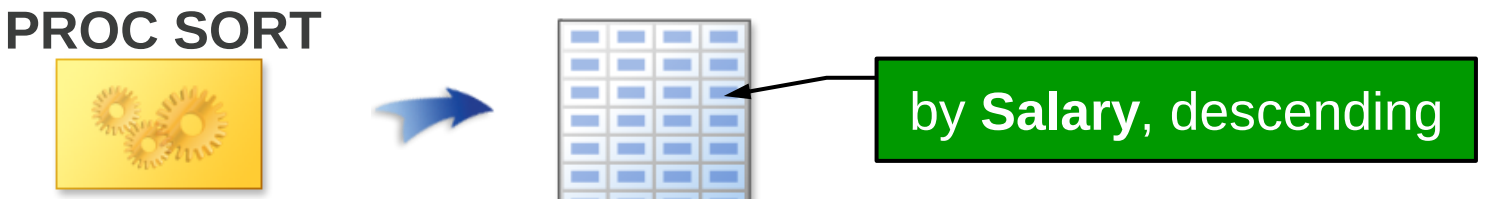


Specifying Multiple BY Variables

- PROC SORT first arranges the data set by the values of the first BY variable.



- PROC SORT then arranges any observations that have the same value as the first BY variable by the values of the second BY variable.



- This sorting continues for every specified BY variable.

Step 2: Specify Report Groupings

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

```
proc print data=work.sales noobs;
    by Country;
run;
```

BY <**DESCENDING**> *variables*;

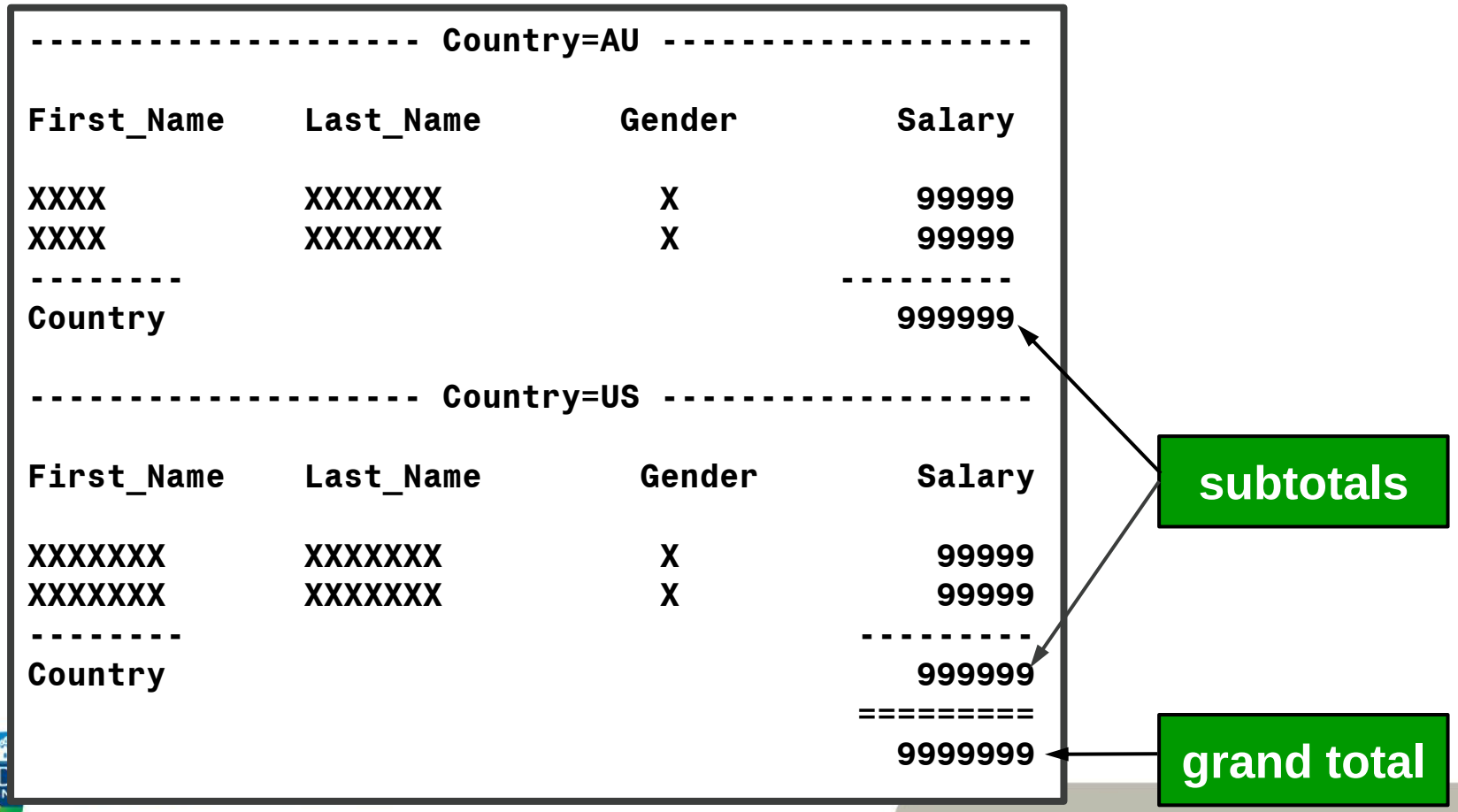
The variables in the BY statement are called *BY variables*.

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

Exercise 6

•Open and submit **L3_E6.sas**. View the log. Why did the program fail?

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



Generating Subtotals

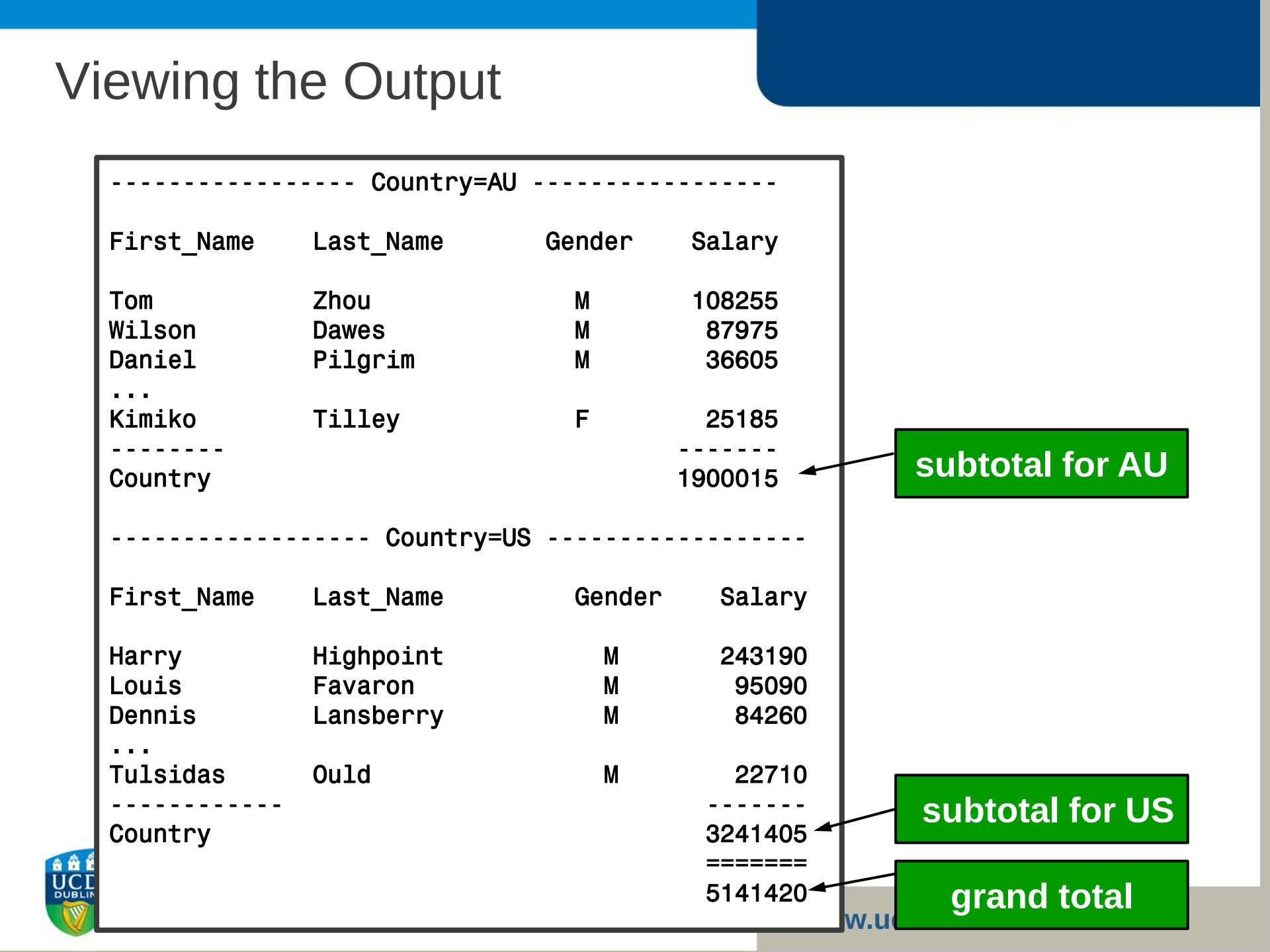
Use a BY statement and a SUM statement in a PROC PRINT step.

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

proc print data=work.sales noobs;
   by Country;
   sum Salary;
   var First_Name Last_Name Gender Salary;
run;
```

L3_D10.sas

Viewing the Output

```
----------------- Country=AU -----------------

First_Name     Last_Name         Gender     Salary

Tom            Zhou                M        108255
Wilson         Dawes               M         87975
Daniel         Pilgrim             M         36605
...
Kimiko         Tilley              F         25185
--------                                    -------
Country                                     1900015

------------------ Country=US ------------------

First_Name     Last_Name         Gender     Salary

Harry          Highpoint           M        243190
Louis          Favaron             M         95090
Dennis         Lansberry           M         84260
...
Tulsidas       Ould                M         22710
------------                                -------
Country                                     3241405
                                            =======
                                            5141420
```



Lecture 3: Producing Reports

3.1 Subsetting Report Data

3.2 Sorting and Grouping Report Data

3.3 Enhancing Reports

Objectives – Part 3

- Include titles and footnotes in a report.
- Use the LABEL statement to define descriptive column headings.
- Control the use of column headings with the LABEL and SPLIT= options.

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

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

Orion Star Sales Staff
Salary Report

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

Confidential

Displaying Titles and Footnotes

Use TITLE and FOOTNOTE statements to enhance the report.

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

footnote1 'Confidential';

proc print data=orion.sales;
   var Employee_ID Last_Name Salary;
run;

title;
footnote;
```

TITLE*n* *'text'*;

FOOTNOTE*n* *'text'*;

L3_D11.sas

Viewing the Output

Orion Star Sales Staff
Salary Report

Obs	Employee_ID	Last_Name	Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
...			
164	121144	Capachietti	83505
165	121145	Lansberry	84260

Confidential

TITLE Statement

The global *TITLE statement* specifies title lines for SAS output.

TITLE*n* '*text* '**;**

- Titles appear at the top of the page.
- The default title is **The SAS System**.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **TITLE** is equivalent to **TITLE1**.
- Titles remain in effect until they are changed, canceled, or you end your SAS session.

FOOTNOTE Statement

The global *FOOTNOTE statement* specifies footnote lines for SAS output.

FOOTNOTE*n* '*text* '**;**

- Footnotes appear at the bottom of the page.
- No footnote is printed unless one is specified.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **FOOTNOTE** is equivalent to **FOOTNOTE1**.
- Footnotes remain in effect until they are changed, canceled, or you end your SAS session.

Changing Titles and Footnotes

To change a title line, submit a TITLE statement with the same number but different text.

- This replaces a previous title with the same number.
- It cancels all titles with higher numbers.

```
title1 'ABC Company';
title2 'Sales Division';
title3 'Salary Report';
```

```
title1 'Salary Report';
```

This statement changes title 1 and cancels titles 2 and 3.

Footnotes are changed the same way.

Cancelling All Titles and Footnotes

```
title;
```

– The null TITLE statement cancels all titles.

```
footnote;
```

– The null FOOTNOTE statement cancels all footnotes.

Exercise 7 – code run sequentially

PROC PRINT Code	Resultant Title(s)

`title1 'The First Line';`
`title2 'The Second Line';`
`proc print data=orion.sales;`
`run;`	

`title2 'The Next Line';`
`proc print data=orion.sales;`
`run;`	

`title 'The Top Line';`
`proc print data=orion.sales;`
`run;`	

`title3 'The Third Line';`
`proc print data=orion.sales;`
`run;`	

`title;`
`proc print data=orion.sales;`
`run;`	

Exercise 8

Which footnote or footnotes appear in the second procedure output?

- a.

```
Non Sales Employees
```

- b.

```
Orion Star
Non Sales Employees
```

- c.

```
Non Sales Employees
Confidential
```

- d.

```
Orion Star
Non Sales Employees
Confidential
```

```
footnote1 'Orion Star';
footnote2 'Sales Employees';
footnote3 'Confidential';
proc print data=orion.sales;
run;

footnote2 'Non Sales Employees';
proc print data=orion.nonsales;
run;
```

LABEL Statement and Option

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

L3_D12.sas

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

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

title;
footnote;
```

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

LABEL Statement

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

Viewing the Output

Orion Star Sales Staff
Salary Report

Obs	Sales ID	Last Name	Annual Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
...			
164	121144	Capachietti	83505
165	121145	Lansberry	84260

Confidential

SPLIT= Option

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

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

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

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

L3_D13.sas

Viewing the Output

Orion Star Sales Staff
Salary Report

Obs	Sales ID	Last Name	Annual Salary
1	120102	Zhou	108255
2	120103	Dawes	87975
3	120121	Elvish	26600
...			
164	121144	Capachietti	83505
165	121145	Lansberry	84260

Confidential