Chapter 3: Displaying Query Results

Chapter 3: Displaying Query Results

Objectives

- Display a query's results in a specified order.
- Use SAS formats, labels, and titles to enhance the appearance and usability of a query's output.

Business Scenario

You need a report that shows the employee ID of each Orion Star employee who makes charitable donations, and lists the amount of the highest quarterly donation. Rows should be sorted first in descending order of amount, and then by employee ID.

Here is a sketch of the desired report:

Employee ID	
120005	25
120006	25
120001	20
120002	20
120003	20

Ordering Data

Use the ORDER BY clause to sort query results in a specific order.

- Ascending order (No keyword; this is the default.)
- Descending order (by following the column name with the DESC keyword)

Ordering Data

In an ORDER BY clause, order the query results by specifying the following:

- any column name from any table in the FROM clause, even if the column is not in the SELECT list
- a column name or a number representing the position of an item in the SELECT list
- an expression
- a combination of any of the above, with individual items separated by commas

Ordering Data

Example: From the **orion.Employee_payroll** table, list the employee ID and salary of all employees hired prior to January 1, 1979, in descending salary order.

```
proc sql;
   select Employee_ID, Salary
      from orion.Employee_payroll
      where Employee_Hire_Date < '01JAN1979'd
      order by Salary desc;
quit;
```

Ordering Data

Partial PROC SQL Output

```
                    The SAS System

              Employee_ID   Salary
              ------------------------
                   121141   194885
                   120659   161290
                   120103    87975
                   120670    65420
                   120712    63640
                   120804    55400
                   120269    52540
                   120765    51950
                   120265    51950
                   120691    49240
                   120270    48435
```

`Salary` is in descending order.

Poll

Quiz

3.01 Multiple Choice Poll

Which ORDER BY clause orders a report by descending **`State`** and descending **`City`**?

a. **`order by state, city`**

b. **`order by desc state, city`**

c. **`order by state, city desc`**

d. **`order by state desc, city desc`**

e. **`order by desc state, desc city`**

3.01 Multiple Choice Poll – Correct Answer

Which ORDER BY clause orders a report by descending **`State`** and descending **`City`**?

a. **`order by state, city`**

b. **`order by desc state, city`**

c. **`order by state, city desc`**

d. **`order by state desc, city desc`**

e. **`order by desc state, desc city`**

Producing an Ordered Report

Remember to sort output in descending order of amount and then by employee ID.

```
proc sql;
select Employee_ID,
       max(Qtr1,Qtr2,Qtr3,Qtr4)
   from orion.Employee_donations
   where Paid_By="Cash or Check"
   order by 2 desc, Employee_ID;
quit;
```

Mix and match!

Producing an Ordered Report

Partial PROC SQL Output

Ascending Employee_ID

Descending Donation Amounts

Employee ID	
120265	25
120736	25
120270	20
120679	20
120759	20
120760	20
120681	15
120734	15
120757	15
120777	15
120662	10
120742	10
120808	10
120994	10
121013	10
121086	10

Enhancing Query Output

You can use SAS formats and labels to customize PROC SQL output. In the SELECT list, after the column name, but before the commas that separate the columns, you can include the following:

- text in quotation marks (ANSI) or the LABEL= column modifier (SAS enhancement) to alter the column heading
- the FORMAT= column modifier to alter the appearance of the values in that column

Enhancing Query Output

You can enhance a report by displaying column labels instead of variable names, and formatting cash amounts with dollar signs and commas.

```
proc sql;
   select Employee_ID
          label='Employee Identifier',
          sum(Qtr1,Qtr2,Qtr3,Qtr4)
          'Annual Donation' format=dollar7.2,
          Recipients
   from orion.Employee_donations
   where Paid_By="Cash or Check"
   order by 2 desc
;
quit;
```

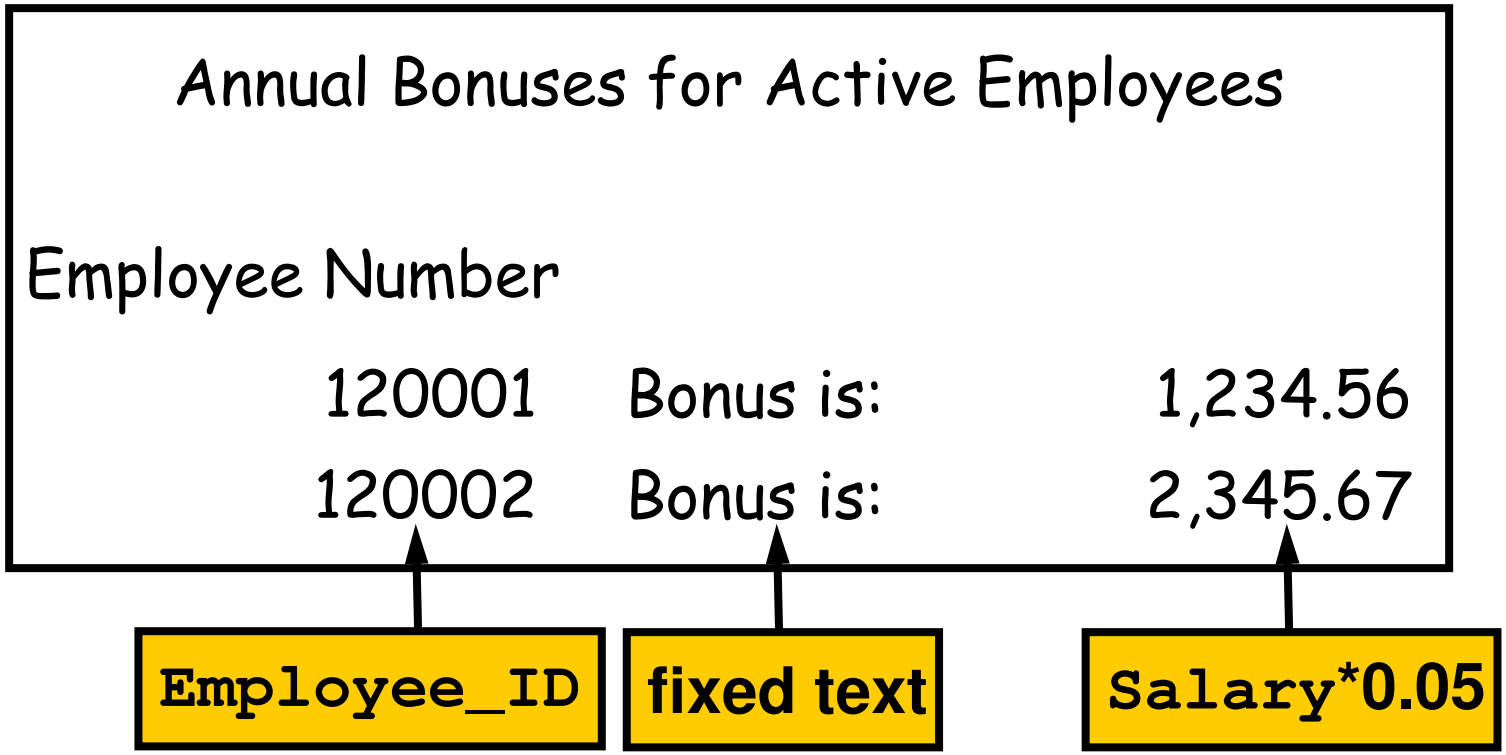
Enhancing Query Output

Partial PROC SQL Output

Employee Identifier	Annual Donation	Recipients
120736	$45.00	Cuidadores Ltd.
120759	$40.00	Child Survivors
120681	$40.00	EarthSalvors 60%, Vox Victimas 40%
120679	$40.00	Cancer Cures, Inc.
120777	$40.00	Cuidadores Ltd. 80%, Mitleid International 20%
120760	$35.00	Cancer Cures, Inc. 40%, Cuidadores Ltd. 60%

Business Scenario

Produce a report of bonus values for all active employees. Bonuses are 5% of salary. The requestor provided this sketch of the desired report.



Enhancing Query Output

You can also enhance the appearance of the query output by doing the following:

- defining a new column containing the same constant character value for every row
- using SAS titles and footnotes

Use a combination of these techniques to produce the Annual Bonuses for Active Employees report.

Enhancing Query Output

The code:

```
proc sql;
title 'Annual Bonuses for Active Employees';
   select Employee_ID label='Employee Number',
          'Bonus is:',
           Salary *.05 format=comma12.2
       from orion.Employee_Payroll
       where Employee_Term_Date is missing
       order by Salary desc
;
quit;
title;
```

Enhancing Query Output

Partial PROC SQL Output

```
Annual Bonuses for Active Employees

Employee
  Number
-------------------------------------------
  120259  Bonus is:      21,690.00
  120262  Bonus is:      13,422.75
  120261  Bonus is:      12,159.50
  120260  Bonus is:      10,394.25
  121141  Bonus is:       9,744.25
  120101  Bonus is:       8,152.00
  120659  Bonus is:       8,064.50
  121142  Bonus is:       7,803.25
  120102  Bonus is:       5,412.75
  121143  Bonus is:       4,754.50
  120103  Bonus is:       4,398.75
  120719  Bonus is:       4,371.00
```

Question & Answer

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 3: Displaying Query Results

Objectives

- Use functions to create summary queries.
- Group data and produce summary statistics for each group.

Summary Functions

How a summary function works in SQL depends on the number of columns specified.

- If the summary function specifies only one column, the statistic is calculated for the column (using values from one or more rows).
- If the summary function specifies more than one column, the statistic is calculated for the row (using values from the listed columns).

The SUM Function (Review)

The SUM function returns the sum of the non-missing arguments.

General form of the SUM function:

SUM(*argument1*<,*argument2*, ...>)

argument includes numeric constants, expressions, or variable names. Only when all arguments are missing will the SUM function return a missing value.

Summary Functions

Example: Total each individual's annual cash donations.
Order the results by decreasing total donation.

```
proc sql;
   select Employee_ID
          label='Employee Identifier',
          Qtr1,Qtr2,Qtr3,Qtr4,
          sum(Qtr1,Qtr2,Qtr3,Qtr4)
          label='Annual Donation'
          format=comma9.2
      from orion.Employee_donations
      where Paid_By="Cash or Check"
      order by 6 desc
;
quit;
```

Summary Functions

Non-missing values are totaled across columns by row. In SQL, specifying multiple columns in a summary function returns results similar to that of a DATA step.

Partial PROC SQL Output

Employee Identifier	Qtr1		Qtr2		Qtr3		Qtr4		Annual Donation
120736	25		.		.		20		45.00
120759	15		20		5		.		40.00
120681	10	+	10	+	5	+	15	=	40.00
120679	.		20		5		15		40.00
120777	5		15		5		15		40.00
120760	.		15		20		.		35.00
120270	20		10		5		.		35.00
120994	5		5		10		10		30.00

Summary Functions

If a summary function specifies only one column name, the statistic is calculated down the column (across rows). This technique compares to using the MEANS procedure.

Example: Determine the total of all charitable donations in quarter 1.

```
proc sql;
   select sum(Qtr1)
          'Total Quarter 1 Donations'
      from orion.Employee_Donations
;
quit;
```

Summary Functions

SUM(Qtr1) calculates the sum of the values in this column for all rows in the table.

Partial Listing of **`orion.Employee_Donations`**

Employee Identifier	Qtr1	Qtr2	Qtr3	Qtr4
120265	.	.	.	25
120270	20	10	5	.
120662	10	.	5	5
120663	.	.	5	.
120679	.	20	5	15
120681	10	10	5	15
120734	.	.	15	10
120736	25	.	.	20
120742	.	.	10	10
120757	.	.	15	5
120759	15	20	5	.

Summary Functions

PROC SQL Output

```
Total
Quarter 1
Donations
_________
    1515
```

Summary Functions

Example: Determine the total of all charitable donations in quarter 1.

```
proc means data=orion.Employee_donations
           sum maxdec=0;
   var Qtr1;
run;
```

PROC MEANS Output

```
Analysis Variable : Qtr1

                 Sum
_______________
            1515
_______________
```

The COUNT Function

The COUNT function returns the number of rows returned by a query.

General form of the COUNT function:

COUNT(*|*argument*)

argument can be the following:

- * (asterisk), which counts all rows
- a column name, which counts the number of non-missing values in that column

Summary Functions

Example: Determine the total number of current employees.

```
proc sql;
   select count(*) as Count
      from orion.Employee_Payroll
      where Employee_Term_Date is missing
;
quit;
```

PROC SQL Output

Count
308

Summary Functions

A few commonly used summary functions are listed. Both ANSI SQL and SAS functions can be used in PROC SQL.

SQL	SAS	Description
AVG	MEAN	returns the mean (average) value.
COUNT	FREQ, N	returns the number of non-missing values.
MAX	MAX	returns the largest value.
MIN	MIN	returns the smallest non-missing value.
SUM	SUM	returns the sum of non-missing values.
	NMISS	counts the number of missing values.
	STD	returns the standard deviation.
	VAR	returns the variance.

Poll

Quiz

3.02 Quiz

Open the program file **s103a01**. Submit the program and review the output.

1. How many rows did the first query create?
2. How many rows did the second query create?
3. In the second query's results, was the value in the average column different for every gender listed?

3.02 Quiz – Correct Answer

1. How many rows did the first query create?

```
proc sql;
   select 'The Average Salary is:',
          avg(Salary)
      from orion.Employee_Payroll
      where Employee_Term_Date is missing
;
quit;
```

Only one row, which displays the average salary for the entire table, was created.

```
                The SAS System
          -------------------------------
          The Average Salary is:  40476.92
```

3.02 Quiz – Correct Answer

2. How many rows did the second query create?

```
proc sql;
   select Employee_Gender,
          avg(Salary) as Average
      from orion.Employee_Payroll
      where Employee_Term_Date is missing
;
quit;
```

The output contains 308 rows. This is the number of rows returned by the COUNT(*) function in program s103d07.

3.02 Quiz – Correct Answer

3. In the second query's results, was the value in the average column different for every gender listed? **No.**

Every row contained the same `Average` value, which is the overall average salary for the entire table.

Gender	Employee_ Average
M	40476.92
M	40476.92
M	40476.92
F	40476.92
F	40476.92

Remerging Summary Statistics

When a SELECT list contains both a column created by a summary function and a column that is **not** summarized, by default, the summarized data is appended to each row of the original data table (remerged) in order to produce the output.

SAS informs you of this by placing this note in the log.

Partial SAS Log

```
NOTE: The query requires remerging summary statistics back
      with the original data.
```

Remerging Summary Statistics

To change the default behavior, use either of the following:

- NOSQLREMERGE SAS system option
- PROC SQL NOREMERGE option

Resubmitting the query with the NOREMERGE option in effect produces no output and results in this SAS log error message:

```
ERROR: The query requires remerging summary statistics back with the
       original data. This is disallowed due to the NOREMERGE proc
       option or NOSQLREMERGE system option.
```

Poll

Quiz

3.03 Quiz

Open the program file **s103a02**. Submit the query and review the output and the log. Answer the following questions:

1. How many rows of output were created?
2. What major difference was there in the log between this query's results and the second query in the previous activity?

3.03 Quiz – Correct Answer

1. How many rows of output were created?

```
proc sql;
   select Employee_Gender,
          avg(Salary) as Average
      from orion.Employee_Payroll
      where Employee_Term_Date is missing
      group by Employee_Gender
;
quit;
```

Two rows

Row	Employee_ Gender	Average
1	F	37002.88
2	M	43334.26

s103a02

3.03 Quiz – Correct Answer

2. What major difference was there in the log between this query's results and the second query in the previous activity?

SAS log notes from the previous activity:

```
NOTE: The query requires remerging summary statistics back
with the original data.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

SAS log notes from this activity:

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

There was no note about remerging statistics.

Question & Answer

Grouping Data

You can use the GROUP BY clause to do the following:

- classify the data into groups based on the values of one or more columns
- calculate statistics for each unique value of the grouping columns

Grouping Data

Example: Calculate the average salary by gender.

```
proc sql;
title "Average Salary by Gender";
   select Employee_Gender as Gender,
          avg(Salary) as Average
      from orion.Employee_Payroll
      where Employee_Term_Date is missing
      group by Employee_Gender
;
quit;
title;
```

Grouping Data

PROC SQL Output

Average Salary by Gender

Gender	Average
F	37002.88
M	43334.26

Poll

Quiz

3.04 Poll

Can you group by more than one column?

- ❍ Yes
- ❍ No

3.04 Poll – Correct Answer

Can you group by more than one column?

- ❍ Yes
- ❍ No

Analyzing Groups of Data

Example: Determine the total number of employees in each department.

```
proc sql;
   select Department, count(*) as Count
      from orion.Employee_Organization
      group by Department
;
quit;
```

Analyzing Groups of Data

PROC SQL Output

Department	Count
Accounts	17
Accounts Management	9
Administration	34
Concession Management	11
Engineering	9
Executives	4
Group Financials	3
Group HR Management	18
IS	25
Logistics Management	14
Marketing	20
Purchasing	18
Sales	201
Sales Management	11
Secretary of the Board	2
Stock & Shipping	26
Strategy	2

Analyzing Groups of Data

Example: Calculate each male employee's salary as a percentage of all male employees' salaries. Display the employee ID, salary, and percentage in decreasing order of percentage.

```
proc sql;
title "Male Employee Salaries";
   select Employee_ID, Salary format=comma12.,
          Salary / sum(Salary)
          format=percent6.2
      from orion.Employee_Payroll
      where Employee_Gender="M"
            and Employee_Term_Date is missing
      order by 3 desc
;
quit;
title;
```

Analyzing Groups of Data

Example: Calculate each male employee's salary as a percentage of all male employees' salaries. Display the employee ID, salary, and percentage in decreasing order of percentage.

```
proc sql;
title "Male Employee Salaries";
   select Employee_ID, Salary format=comma12.,
          Salary / sum(Salary)
          format=percent6.2
      from orion.Employee_Payroll
      where Employee_Gender="M"
            and Employee_Term_Date is missing
      order by 3 desc
;
quit;
title;
```

Select only the group of rows that you want to analyze.

Analyzing Groups of Data

Example: Calculate each male employee's salary as a percentage of all male employees' salaries. Display the employee ID, salary, and percentage in decreasing order of percentage.

```
proc sql;
title "Male Employee Salaries";
   select Employee_ID, Salary format=comma12.,
          Salary / sum(Salary)
          format=percent6.2
      from orion.Employee_Payroll
      where Employee_Gender="M"
            and Employee_Term_Date is missing
      order by
;
quit;
title;
```

Individual salary value for each row

Divided by a remerged summary value (sum of all salaries)

Analyzing Groups of Data

Partial PROC SQL Output

Male Employee Salaries

Employee_ID	Salary	
120259	433,800	5.9%
120262	268,455	3.7%
120261	243,190	3.3%
121141	194,885	2.7%
120101	163,040	2.2%
120659	161,290	2.2%
121142	156,065	2.1%
120102	108,255	1.5%
121143	95,090	1.3%
120103	87,975	1.2%
121145	84,260	1.2%
120268	76,105	1.0%
120724	63,705	.87%
120714	62,625	.86%
120660	61,125	.83%

Selecting Groups with the HAVING Clause

- The WHERE clause is processed **before** a GROUP BY clause and determines which individual rows are available for grouping.
- The HAVING clause is processed **after** the GROUP BY clause and determines which groups will be displayed.

Selecting Groups with the HAVING Clause

Example: Display the names of the departments and the number of employees for departments with 25 or more employees. List the department with the highest count first.

```
proc sql;
   select Department, count(*) as Count
      from orion.Employee_Organization
      group by Department
      having Count ge 25
      order by Count desc
;
quit;
```

Selecting Groups with the HAVING Clause

PROC SQL Output

Department	Count
Sales	201
Administration	34
Stock & Shipping	26
IS	25

Poll

Quiz

3.05 Quiz

Which syntax will select employee IDs having bonuses greater than $1000?

1.
```
select Employee_ID, Salary*0.1 as Bonus
       from orion.Employee_Payroll
       where calculated Bonus > 1000;
```
2.
```
select Employee_ID, Salary*0.1 as Bonus
       from orion.Employee_Payroll
       having Bonus > 1000;
```
3. Both of the above
4. Neither of the above

3.05 Quiz – Correct Answer

Which syntax will select employee IDs having bonuses greater than $1000?

1.
```
select Employee_ID, Salary*0.1 as Bonus
       from orion.Employee_Payroll
       where calculated Bonus > 1000;
```
2.
```
select Employee_ID, Salary*0.1 as Bonus
       from orion.Employee_Payroll
       having Bonus > 1000;
```

Both of the these queries produce the desired results.

In the second query, the HAVING clause can be used without the GROUP BY clause to filter the calculated columns row-by-row without specifying the CALCULATED keyword.

Business Scenario

Create a report that lists, for each department, the total number of managers, total number of employees, and the Manager-to-Employee (M/E) ratio. Calculate the M/E ratio as follows:

M/E Ratio= # Managers / # non-Manager Employees

Below is a rough sketch of the desired report.

Department	Managers	Employees	M/E Ratio
Accounts	1	5	20%
Administration	2	20	10%

Department **# Mgrs** **# Emps** **Mgrs/Emps**

Counting Rows Meeting a Specified Criteria

This request is complicated by the need, in the same query, to count rows that **do** have **`Manager`** in the title, as well as rows that **do not**. You cannot use a WHERE clause to exclude either group.

Instead, use the FIND function in a Boolean expression to simplify the query.

The FIND Function

The FIND function returns the starting position of the first occurrence of a substring within a string (character value).

General form of the FIND function:

FIND(*string*, *substring*<,*modifier(s)*><,*startpos*>)

string	constant, variable, or expression to be searched
substring	constant, variable, or expression sought within the string
modifiers	i=ignore case, t=trim trailing blanks
startpos	an integer specifying the start position and direction of the search

The FIND Function

Example: Find the starting position of the substring **`Manager`** in the character variable **`Job_Title`**.

```
find(Job_Title,"manager","i")
```

Job_Title									1											2					
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
A	d	m	i	n	i	s	t	r	a	t	i	o	n		M	a	n	a	g	e	r			

The value returned by the FIND function is 16.

Using Boolean Expressions

Boolean expressions evaluate to TRUE (1) or FALSE (0). They are used in this SELECT list to distinguish rows that have **`Manager`** in the **`Job_Title`** column.

```
proc sql;
   select Department,Job_Title,
          (find(Job_Title,"manager","i") >0)
          "Manager"
      from orion.Employee_Organization
;
quit;
```

The Boolean expression will produce the value 1 when **`Job_Title`** contains the word **`Manager`** and 0 when it does not.

Using Boolean Expressions

Partial PROC SQL Output (Rows 4–14)

Department	Job_Title	Manager
Administration	Administration Manager	1
Administration	Secretary I	0
Administration	Office Assistant II	0
Administration	Office Assistant III	0
Administration	Warehouse Assistant II	0
Administration	Warehouse Assistant I	0
Administration	Warehouse Assistant III	0
Administration	Security Guard II	0
Administration	Security Guard I	0
Administration	Security Guard II	0
Administration	Security Manager	1

To count the managers, you can add the values in the column produced by the Boolean expression. In this segment, the Administration department has two managers.

Using Boolean Expressions

Example: For each department, calculate the percentage of people with the word *Manager* in the job title.

```
proc sql;
title "Manager to Employee Ratios";
   select Department,
          sum((find(Job_Title,"manager","i") >0))
            as Managers,
          sum((find(Job_Title,"manager","i") =0))
            as Employees,
          calculated Managers/calculated Employees
          "M/E Ratio" format=percent8.1
      from orion.Employee_Organization
      group by Department
;
quit;
```

Using Boolean Expressions

PROC SQL Output

Manager to Employee Ratios

Department	Managers	Employees	M/E Ratio
Accounts	3	14	21.4%
Accounts Management	1	8	12.5%
Administration	5	29	17.2%
Concession Management	1	10	10.0%
Engineering	1	8	12.5%
Executives	0	4	0.0%
Group Financials	0	3	0.0%
Group HR Management	3	15	20.0%
IS	2	23	8.7%
Logistics Management	6	8	75.0%
Marketing	6	14	42.9%
Purchasing	3	15	20.0%
Sales	0	201	0.0%
Sales Management	5	6	83.3%
Secretary of the Board	0	2	0.0%
Stock & Shipping	5	21	23.8%
Strategy	0	2	0.0%

Question & Answer

Exercise

This exercise reinforces the concepts discussed previously.

Chapter Review

1. Which of these ORDER BY clauses will display the results ordered by decreasing **`Salary`** and then by increasing **`Name`**?

   a. **`order by descending Salary, Name`**

   b. **`order by Salary desc, Name`**

   c. **`order by desc Salary, ascending Name`**

Chapter Review Answers

1. Which of these ORDER BY clauses will display the results ordered by decreasing **`Salary`** and then by increasing **`Name`**?

   a. **`order by descending Salary, Name`**

   b. **`order by Salary desc, Name`**

   c. **`order by desc Salary, ascending Name`**

Chapter Review

2. How would you modify this SELECT statement to display the **`Salary`** column using the EUROX10. format?

```
proc sql;
   select First_Name, Last_Name,
          Job_Title,
          Salary
      from orion.Sales
;
quit;
```

Chapter Review Answers

2. How would you modify this SELECT statement to display the **`Salary`** column using the EUROX10. format? **format=eurox10.**

```
proc sql;
   select First_Name, Last_Name,
          Job_Title,
          Salary format=eurox10.
      from orion.Sales
;
quit;
```

Chapter Review

3. The SAS RANGE() function returns the difference between the largest and the smallest of the non-missing arguments. What clause can you add to this query to produce a listing of the salary range for each value of **`Job_Title`**?

```
proc sql;
   select Job_Title, range(Salary)
      from orion.Sales

;
quit;
```

Chapter Review Answers

3. The SAS RANGE() function returns the difference between the largest and the smallest of the non-missing arguments. What clause can you add to this query to produce a listing of the salary range for each value of **`Job_Title`**? **GROUP BY clause**

```
proc sql;
   select Job_Title, range(Salary)
      from orion.Sales
      group by Job_Title
;
quit;
```