

## EXERCISE-8

### Aggregating Data Using Group Functions

#### Objectives

After the completion of this exercise, the students be will be able to do the following:

- Identify the available group functions
- Describe the use of group functions
- Group data by using the GROUP BY clause
- Include or exclude grouped rows by using the HAVING clause

#### What Are Group Functions?

Group functions operate on sets of rows to give one result per group

#### Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Each of the functions accepts an argument. The following table identifies the options that you can use in the syntax:

Function	Description
AVG ( [DISTINCT ALL] n)	Average value of n, ignoring null values
COUNT ( { *   [DISTINCT ALL] expr } )	Number of rows, where expr evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX ( [DISTINCT ALL] expr)	Maximum value of expr, ignoring null values
MIN ( [DISTINCT ALL] expr)	Minimum value of expr, ignoring null values
STDDEV ( [DISTINCT ALL] x)	Standard deviation of n, ignoring null values
SUM ( [DISTINCT ALL] n)	Sum values of n, ignoring null values
VARIANCE ( [DISTINCT ALL] x)	Variance of n, ignoring null values

#### Group Functions: Syntax

```
SELECT [column,] group_function(column), ...  
FROM table  
[WHERE condition]  
[GROUP BY column]  
[ORDER BY column];
```

#### Guidelines for Using Group Functions

- DISTINCT makes the function consider only nonduplicate values; ALL makes it consider every value, including duplicates. The default is ALL and therefore does not need to be specified.

```
SELECT department_id, AVG(salary) FROM employees GROUP BY department_id
HAVING max(salary)>10000;
```

Example displays the job ID and total monthly salary for each job that has a total payroll exceeding \$13,000. The example excludes sales representatives and sorts the list by the total monthly salary.

```
SELECT job_id, SUM(salary) PAYROLL FROM employees WHERE job_id NOT LIKE
'%REP%'
GROUP BY job_id HAVING SUM(salary) > 13000 ORDER BY SUM(salary);
```

### Nesting Group Functions

Display the maximum average salary:

Group functions can be nested to a depth of two. The slide example displays the maximum average salary.

```
SELECT MAX(AVG(salary)) FROM employees GROUP BY department_id;
```

#### **Summary**

In this exercise, students should have learned how to:

- Use the group functions COUNT, MAX, MIN, and AVG
- Write queries that use the GROUP BY clause
- Write queries that use the HAVING clause

```
SELECT column, group_function
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

#### Find the Solution for the following:

Determine the validity of the following three statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group.

True/False

2. Group functions include nulls in calculations.

True/False

3. The WHERE clause restricts rows prior to inclusion in a group calculation.

True/False

The HR department needs the following reports:

4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number

```
SELECT ROUND (MAX (salary)) AS "Maximum", ROUND (MIN(salary)) AS
"Minimum", ROUND (SUM (salary)) AS "Sum", ROUND (AVG (salary)) AS
"Average" FROM employees;
```



5. Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

```
SELECT job_id, ROUND(MAX(salary)) AS "Maximum", ROUND(MIN(salary)) AS  
"Minimum", ROUND(SUM(salary)) AS "Sum", ROUND(AVG(salary)) AS "Average"  
FROM employees GROUP BY job_id;
```

6. Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

```
SELECT COUNT (*) AS COUNT FROM employees WHERE job_id = ?;
```

7. Determine the number of managers without listing them. Label the column Number of Managers. *Hint: Use the MANAGER\_ID column to determine the number of managers.*

```
SELECT COUNT(DISTINCT manager_id) AS 'Number of Managers'  
FROM employees WHERE manager_id IS NOT NULL;
```

8. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SELECT MAX(salary) - MIN(salary) AS DIFFERENCE FROM  
employees;
```

9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is \$6,000 or less. Sort the output in descending order of salary.

```
SELECT manager_id, MIN(salary) AS lowest_salary FROM  
employees WHERE manager_id IS NOT NULL GROUP BY manager_id  
HAVING MIN(salary) > 6000 ORDER BY lowest_salary DESC;
```

10. Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT COUNT (*) AS Total_Employees, SUM (YEAR
```