

Ex.No.: 10	AGGREGATING DATA USING GROUP FUNCTIONS
Date: 20/09/2024	

### 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 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 True

2. Group functions include nulls in calculations.

True/False False

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

True/False True

#### 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(MIN(salary)) AS "Minimum",  
       ROUND(MAX(salary)) AS "Maximum",  
       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 "Number of people"
FROM employees
WHERE job-id = '&job-title';
```

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 "Minimum Salary"
FROM employees WHERE manager_id IS NOT NULL
GROUP BY manager_id HAVING MIN(salary) > 6000
ORDER BY MIN(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(CASE WHEN TO-CHAR(hire-date, 'YYYY') = '1995' THEN 1 ELSE 0 END)
AS "Hired in 1995"
```

```
SUM(CASE WHEN TO-CHAR(hire-date, 'YYYY') = '1996' THEN 1 ELSE 0 END) AS "Hired in 1996",
```

```
SUM(CASE WHEN TO-CHAR(hire-date, 'YYYY') = '1997' THEN 1 ELSE 0 END)
```



AS hired in 1997"

SUM(CASE WHEN TO\_CHAR(hire\_date, 'YYYY') = 1997  
THEN 1 ELSE 0 END)

AS hired in 1998"

FROM employees;

11. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

SELECT Job-Id, SUM(CASE WHEN department\_id = 20 THEN  
Salary ELSE 0 END) AS "Dept 20 Salary", SUM(CASE WHEN  
department\_id = 50 THEN Salary ELSE 0 END) AS "Dept 50  
Salary", SUM(CASE WHEN department\_id = 80 THEN Salary ELSE  
0 END) AS "Dept 80 Salary", FROM employees WHERE department\_id  
IN(20, 50, 80, 90);

12. Write a query to display each department's name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Group By Job. Number of people, and salary respectively. Round the average salary to two decimal places.

SELECT d.department\_name || ', ' || l.city AS "Location",

COUNT(e.employee\_id) AS "Number of People",

ROUND(AVG(e.salary), 2) AS "Average Salary"

FROM employee e

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

GROUP BY d.department\_name, l.city;

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	