## 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:

**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.
- The data types for the functions with an expr argument may be CHAR, VARCHAR2, NUMBER, or DATE.
- All group functions ignore null values.

**Using the AVG and SUM Functions**

You can use AVG and SUM for numeric data.

SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary) FROM employees WHERE job_id LIKE '%REP%';

**Using the MIN and MAX Functions**

You can use MIN and MAX for numeric, character, and date data types.

SELECT MIN(hire_date), MAX(hire_date) FROM employees;

You can use the MAX and MIN functions for numeric, character, and date data types. example displays the most junior and most senior employees.

The following example displays the employee last name that is first and the employee last name that is last in an alphabetized list of all employees:

SELECT MIN(last_name), MAX(last_name) FROM employees;

**Note:** The AVG, SUM, VARIANCE, and STDDEV functions can be used only with numeric data types. MAX and MIN cannot be used with LOB or LONG data types.

**Using the COUNT Function**

COUNT(*) returns the number of rows in a table:
SELECT COUNT(*) FROM employees WHERE department_id = 50;
COUNT(*expr*) returns the number of rows with nonnull values for the *expr*:
SELECT COUNT(commission_pct) FROM employees WHERE department_id = 80;

**Using the DISTINCT Keyword**

- COUNT(DISTINCT expr) returns the number of distinct non-null values of the *expr*.
- To display the number of distinct department values in the EMPLOYEES table:

SELECT COUNT(DISTINCT department_id) FROM employees;
Use the DISTINCT keyword to suppress the counting of any duplicate values in a column. **Group Functions and Null Values**

Group functions ignore null values in the column:

SELECT AVG(commission_pct) FROM employees;

The NVL function forces group functions to include null values:

SELECT AVG(NVL(commission_pct, 0)) FROM employees;

**Creating Groups of Data**

To divide the table of information into smaller groups. This can be done by using the GROUP BY clause.

**GROUP BY Clause Syntax**

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

**In the syntax:**
*group_by_expression* specifies columns whose values determine the basis for grouping rows

**Guidelines**

- If you include a group function in a SELECT clause, you cannot select individual results as well, *unless* the individual column appears in the GROUP BY clause. You

receive an error message if you fail to include the column list in the GROUP BY clause.

- Using a WHERE clause, you can exclude rows before dividing them into groups.
- You must include the *columns* in the GROUP BY clause.
- You cannot use a column alias in the GROUP BY clause.

**Using the GROUP BY Clause**

All columns in the SELECT list that are not in group functions must be in the GROUP BY clause.

SELECT department_id, AVG(salary) FROM employees GROUP BY department_id ;
The GROUP BY column does not have to be in the SELECT list.
SELECT AVG(salary) FROM employees GROUP BY department_id ;
You can use the group function in the ORDER BY clause:
SELECT department_id, AVG(salary) FROM employees GROUP BY department_id ORDER BY AVG(salary);

**Grouping by More Than One Column**

SELECT department_id dept_id, job_id, SUM(salary) FROM employees GROUP BY department_id, job_id ;

**Illegal Queries Using Group Functions**

Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP

**BY clause:**

SELECT department_id, COUNT(last_name) FROM employees;
You can correct the error by adding the GROUP BY clause:
SELECT department_id, count(last_name) FROM employees GROUP BY department_id; You cannot use the WHERE clause to restrict groups.

- You use the HAVING clause to restrict groups.

- You cannot use group functions in the WHERE clause.

SELECT department_id, AVG(salary) FROM employees WHERE AVG(salary) > 8000
GROUP BY department_id;
You can correct the error in the example by using the HAVING clause to restrict groups:

SELECT department_id, AVG(salary) FROM employees HAVING AVG(salary) > 8000 GROUP BY department_id;

**Restricting Group Results**

With the HAVING Clause ,When you use the HAVING clause, the Oracle server restricts groups as follows:
1. Rows are grouped.
2. The group function is applied.
3. Groups matching the HAVING clause are displayed.

**Using the HAVING Clause**

SELECT department_id, MAX(salary) FROM employees GROUP BY department_idHAVING MAX(salary)>10000 ;

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**
2. Group functions include nulls in calculations. **False**
3. The WHERE clause restricts rows prior to inclusion in a group calculation. **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

**Ans: ELECT 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.

**Ans: 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.

**Ans: 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.*

**Ans: 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.

**Ans: 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.
**Ans: 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.
**Ans: SELECT COUNT(*) AS Total_Employees, SUM(YEAR(hire_date) = 1995) AS Hired_1995, SUM(YEAR(hire_date) = 1996) AS Hired_1996, SUM(YEAR(hire_date) = 1997) AS Hired_1997, SUM(YEAR(hire_date) = 1998) AS Hired_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.
**Ans: 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, SUM(CASE WHEN department_id = 90 THEN salary ELSE 0 END) AS Dept_90_Salary, SUM(salary) AS Total_Salary FROM employees WHERE department_id IN (20,50,80,90) GROUP BY job_id;**

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, Number of people, and salary respectively. Round the average salary to two decimal places.
**Ans: SELECT d.department_name AS 'Name-Location', l.city AS Location, COUNT(e.employee_id) AS 'Number of People', ROUND(AVG(e.salary), 2) AS Salary FROM employees 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) | 5 |
| Total (15) | 15 |
| Faculty Signature | 9/9/25 |

## Practice Questions

### Date Functions

1. For DJs on Demand, display the number of months between the event_date of the Vigil wedding and today's date. Round to the nearest month.
**Ans: SELECT ROUND(TIMESTAMPDIFF(MONTH, (SELECT event_date FROM events WHERE event_name = 'Vigil wedding'), CURDATE())) AS Months;**

2. Display the days between the start of last summer's school vacation break and the day school started this year. Assume 30.5 days per month. Name the output "Days."
**Ans: SELECT ROUND(DATEDIFF('2025-06-01', '2024-05-15')) AS Days;**

3. Display the days between January 1 and December 31.
**Ans: SELECT DATEDIFF('2025-12-31', '2025-01-01') AS Days;**

4. Using one statement, round today's date to the nearest month and nearest year and truncate it to the nearest month and nearest year. Use an alias for each column.
**Ans: SELECT DATE_FORMAT(CURDATE(), '%Y-%m-01') AS Trunc_Month, DATE_FORMAT(CURDATE(), '%Y-01-01') AS Trunc_Year, DATE_FORMAT(DATE_ADD(CURDATE(), INTERVAL 15 DAY), '%Y-%m-01') AS Round_Month, DATE_FORMAT(DATE_ADD(CURDATE(), INTERVAL 182 DAY), '%Y-01-01') AS Round_Year;**

5. What is the last day of the month for June 2005? Use an alias for the output.
**Ans: SELECT LAST_DAY('2005-06-01') AS Last_Day_June_2005;**

6.Display the number of years between the Global Fast Foods employee Bob Miller's birthday and today. Round to the nearest year.
**Ans: SELECT ROUND(TIMESTAMPDIFF(YEAR, '1980-04-15', CURDATE())) AS Years FROM employees WHERE first_name = 'Bob' AND last_name = 'Miller';**

7.Your next appointment with the dentist is six months from today. On what day will you go to the dentist? Name the output, "Appointment."
**Ans: SELECT DATE_ADD(CURDATE(), INTERVAL 6 MONTH) AS Appointment;**

8.The teacher said you have until the last day of this month to turn in your research paper. What day will this be? Name the output, "Deadline."
**Ans: SELECT LAST_DAY(CURDATE()) AS Deadline;**

9.How many months between your birthday this year and January 1 next year?
**Ans: SELECT TIMESTAMPDIFF(MONTH, '2025-08-20', '2026-01-01') AS Months;**

10.What's the date of the next Friday after your birthday this year? Name the output, "First Friday."
**Ans: SELECT DATE_ADD('2025-08-20', INTERVAL (5 - WEEKDAY('2025-08-20') + 7) % 7 DAY) AS First_Friday;**

11. Name a date function that will return a number.
**Ans: `TIMESTAMPDIFF( )` returns a number (e.g., months, years, days).`**

12.Name a date function that will return a date.
**Ans: `DATE_ADD( )` returns a date.`**

13.Give one example of why it is important for businesses to be able to manipulate date data?
**Ans: Businesses need to manipulate date data to track employee tenure, calculate billing cycles, schedule promotions, and ensure compliance with deadlines.**

### Conversion Functions

In each of the following exercises, feel free to use labels for the converted column to make the output more readable.

1. List the last names and birthdays of Global Fast Food Employees. Convert the birth dates to character data in the Month DD, YYYY format. Suppress any leading zeros.
**Ans: SELECT last_name, DATE_FORMAT(birth_date, '%M %e, %Y') AS Birthday FROM employees;**

2. Convert January 3, 04, to the default date format 03-Jan-2004.
**Ans: SELECT STR_TO_DATE('January 3, 04', '%M %e, %y') AS Converted_Date;**

3. Format a query from the Global Fast Foods f_promotional_menus table to print out the start_date of promotional code 110 as: The promotion began on the tenth of February 2004
**Ans: SELECT CONCAT('The promotion began on the ', DATE_FORMAT(start_date, '%D of %M %Y')) AS Message FROM f_promotional_menus WHERE promo_code = 110;.**

4. Convert today's date to a format such as: "Today is the Twentieth of March, Two Thousand Four"
**Ans: SELECT CONCAT('Today is the ', DATE_FORMAT(CURDATE(), '%D of %M, %Y')) AS Today;**

5. List the ID, name and salary for all Global Fast Foods employees. Display salary with a $ sign and two decimal places.
**Ans: SELECT employee_id, CONCAT(first_name, ' ', last_name) AS Name, CONCAT('$', FORMAT(salary, 2)) AS Salary FROM employees;**

| Evaluation Procedure | Marks awarded |
|---|---|
| Practice Evaluation (5) | 5 |
| Viva(5) | 5 |
| Total (10) | 10 |
| Faculty Signature | 9/9/25 |