Experiment 1.2 – SQL SELECT Queries with WHERE, GROUP BY, HAVING, ORDER BY

Experiment

Experiment 1.2: Practicing SQL SELECT queries with WHERE, GROUP BY, HAVING, and ORDER BY clauses to retrieve and analyze data from the EMPLOYEE table.

Aim

The aim of this experiment is to practice writing SQL SELECT statements to filter, group, sort, and calculate average salary for meaningful analysis of employee data.

Objective

- Practice writing SQL SELECT statements.
- Apply filtering conditions using the WHERE clause.
- Group records using the GROUP BY clause.
- Filter grouped data using the HAVING clause.
- Sort query results using the ORDER BY clause.
- Calculate average salary using the AVG() aggregate function.

Software Requirements

- **Database: Oracle XE or PostgreSQL (PgAdmin)**

Practical / Experiment Steps

1. Display the department name and the average salary of employees for each department.
2. Consider only those employees whose salary is greater than 20,000.
3. Display only those departments where the average salary is greater than 30,000.
4. Arrange the final output in descending order of average salary.

Procedure of the Experiment

1. Start the system and log in to the computer.
2. Open the required database tool (Oracle XE or PgAdmin).
3. Connect to the database containing the EMPLOYEE table.

4. Examine the EMPLOYEE table structure and data.
5. Write SQL SELECT queries according to the practical steps.
6. Execute each query and verify the output.
7. Take screenshots of all four queries in order (s1 to s4) for record.
8. Save the work.

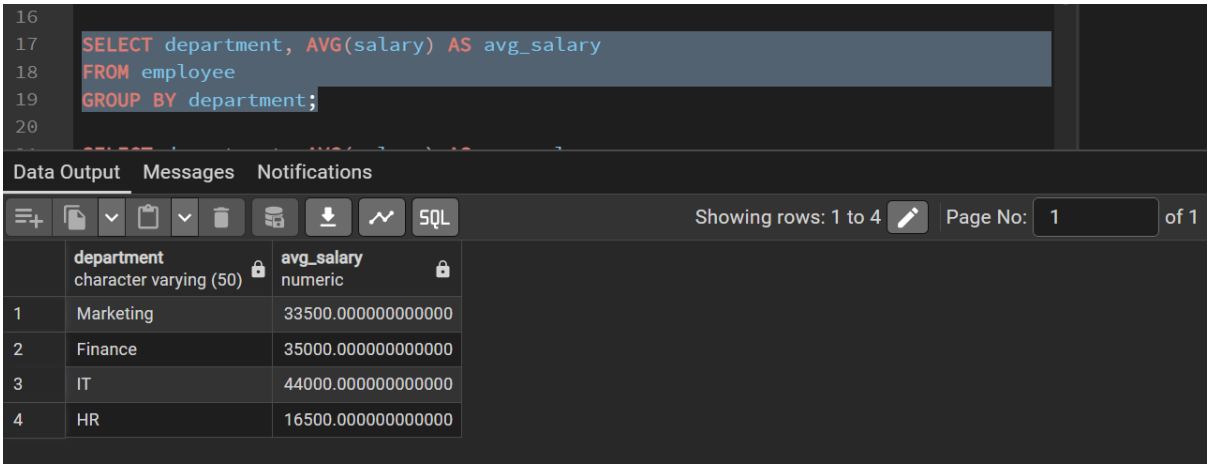
Input / Output Details

Input

- EMPLOYEE table with columns: emp_id, emp_name, department, salary, joining_date.
- SQL SELECT queries using WHERE, GROUP BY, HAVING, ORDER BY, and AVG().

Output

- Step 1: Average salary of employees by department.



	department character varying (50)	avg_salary numeric
1	Marketing	33500.000000000000
2	Finance	35000.000000000000
3	IT	44000.000000000000
4	HR	16500.000000000000

- Step 2: Average salary by department for employees with salary > 20,000.

```
21  SELECT department, AVG(salary) AS avg_salary
22  FROM employee
23  WHERE salary > 20000
24  GROUP BY department;
25
26  SELECT department, AVG(salary) AS avg_salary
27  FROM employee
```

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1 of 1

	department character varying (50)	avg_salary numeric
1	Marketing	33500.000000000000
2	Finance	35000.000000000000
3	IT	44000.000000000000

- Step 3: Departments where average salary > 30,000.

```
SELECT department, AVG(salary) AS avg_salary
FROM employee
GROUP BY department
HAVING AVG(salary) > 30000;

SELECT department, AVG(salary) AS avg_salary
FROM employee
WHERE salary > 20000
GROUP BY department
HAVING AVG(salary) > 30000
ORDER BY avg_salary DESC;
```

Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1

department character varying (50)	avg_salary numeric
Marketing	33500.000000000000
Finance	35000.000000000000
IT	44000.000000000000

- Step 4: Departments with average salary > 30,000 for salary > 20,000, sorted in descending order.

```sql
SELECT department, AVG(salary) AS avg_salary
FROM employee
WHERE salary > 20000
GROUP BY department
HAVING AVG(salary) > 30000
ORDER BY avg_salary DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1

	department character varying (50)	avg_salary numeric
1	IT	44000.000000000000
2	Finance	35000.000000000000
3	Marketing	33500.000000000000

- Screenshots (s1 to s4) attached showing query execution results in order.

Learning Outcome

After completing this experiment, the student will be able to:

- Filter records using the WHERE clause.
- Group records using the GROUP BY clause.
- Apply conditions on grouped data using the HAVING clause.
- Sort query results using ORDER BY.
- Analyze data using AVG() for meaningful insights.
- Write complex queries combining multiple SQL clauses efficiently.