Experiment 2 – SQL SELECT Queries with WHERE, GROUP BY, HAVING, ORDER BY

Experiment

Experiment 2: Understanding and implementing SQL SELECT queries using WHERE, ORDER BY, GROUP BY, and HAVING clauses to retrieve and manipulate data efficiently from relational database tables.

Aim

The aim of this experiment is to practice writing SQL SELECT statements with filtering, grouping, sorting, and aggregate functions to analyze data from relational tables.

Objective

- To practice writing SQL SELECT statements.
- To apply filtering conditions using the WHERE clause.
- To sort query results using the ORDER BY clause.
- To group records using the GROUP BY clause.
- To filter grouped data using the HAVING clause.
- To analyze data using aggregate functions like COUNT(), SUM(), AVG(), MIN(), and MAX().

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 the SQL SELECT query using WHERE, GROUP BY, HAVING, and ORDER BY clauses according to the practical steps.
6. Execute the query and verify the output.
7. Note down the results obtained.
8. Save the work and take screenshots for record.

Input / Output Details

Input

EMPLOYEE table containing columns:

- emp_id
- emp_name
- department
- salary
- joining_date

SQL SELECT queries using WHERE, GROUP BY, HAVING, ORDER BY, and aggregate functions.

Output

Departments and their average salary for employees with salary > 20,000 and average salary > 30,000.

Output arranged in descending order of average salary.

Screenshots of query execution and results are attached in the repository.

SQL Source Code

CREATING TABLE

```
CREATE TABLE EMPLOYEE (
   emp_id INT PRIMARY KEY,
   emp_name VARCHAR(50),
   department VARCHAR(50),
   salary INT,
   joining_date DATE
);
```

INSERT DATA

```
(1, 'Kaif',   'HR',       25000, '2022-01-15'),
(2, 'Neha',   'HR',       35000, '2021-03-10'),
(3, 'Vinay',  'IT',       45000, '2020-07-22'),
(4, 'Meehul',  'IT',       55000, '2019-11-05'),
(5, 'Debyindu',  'Finance',  18000, '2023-02-01'),
(6, 'Tejas',  'Finance',  42000, '2021-06-18'),
(7, 'Mudasar',  'Sales',    30000, '2020-09-12'),
(8, 'Sohail', 'Sales',    48000, '2018-04-25');
```

Average salary of employees for each department

```
SELECT department, AVG(salary) AS avg_salary
FROM EMPLOYEE
GROUP BY department;
```

Consider only employees with salary > 20000

```
SELECT department, AVG(salary) AS avg_salary
FROM EMPLOYEE
WHERE salary > 20000
GROUP BY department;
```

Departments where avg salary > 30000

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

Final output in descending order of average salary

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

Learning Outcome

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

- Filter records using the WHERE clause.
- Group records using GROUP BY.
- Apply conditions on grouped data using HAVING.
- Sort query results using ORDER BY.

- Analyze data using aggregate functions for meaningful insights.