Student Name: Bhavya
Branch: CSE(AI & ML)
Semester: 4
Subject Name: Database Management System

UID: 24BAI70791
Section/Group: 24AIT_KRG-1/G2
Date of Performance: 13/01/2026
Subject Code: 24CSH-298

Experiment 2

1. Aim of the Session

To understand and implement SQL **SELECT queries** using clauses such as **WHERE, GROUP BY, HAVING, and ORDER BY** for retrieving and analyzing employee data from a relational database table.

---

2. Software Requirements

- PostgreSQL Database (pgAdmin)
- MS Word / PDF editor

---

3. Objectives

- To practice writing SQL SELECT queries.
- To apply filtering conditions using the **WHERE** clause.
- To group records using the **GROUP BY** clause.
- To filter grouped data using the **HAVING** clause.

- To sort query results using the **ORDER BY** clause.
- To use aggregate functions such as **AVG()** for data analysis.

---

4. Procedure of the Experiment

1. Start the system and log in to the database.
2. Create the EMPLOYEE table using SQL commands.
3. Insert employee records into the table.
4. Execute SELECT queries using GROUP BY, WHERE, HAVING, and ORDER BY clauses.
5. Verify the output after execution.
6. Save the work and take screenshots for record.

---

5. Practical / Experiment Steps

(A) Table Creation

```
CREATE TABLE EMPLOYEE (
    emp_id NUMERIC PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(30),
    salary NUMERIC,
    joining_date DATE
);
```

(B) Insert Records

```
INSERT INTO EMPLOYEE VALUES (1, 'Bhavya', 'IT', 48000, DATE '2022-02-10');
INSERT INTO EMPLOYEE VALUES (2, 'Myra', 'IT', 42000, DATE '2021-07-15');
INSERT INTO EMPLOYEE VALUES (3, 'Saksham', 'HR', 35000, DATE '2020-04-20');
INSERT INTO EMPLOYEE VALUES (4, 'Anant', 'HR', 30000, DATE '2019-09-12');
INSERT INTO EMPLOYEE VALUES (5, 'Srijan', 'Sales', 26000, DATE '2021-11-05');
INSERT INTO EMPLOYEE VALUES (6, 'Yash', 'Sales', 38000, DATE '2020-03-18');
```

(C) Queries Using GROUP BY, WHERE, HAVING and ORDER BY

1. Display department-wise average salary

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

2. Display average salary considering only employees with salary > 20000

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

3. Display only departments with average salary > 30000

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

```
GROUP BY department
HAVING AVG(salary) > 30000;
```

4. Arrange result 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;
```

---

6. Input / Output Details and Screenshots

Input:

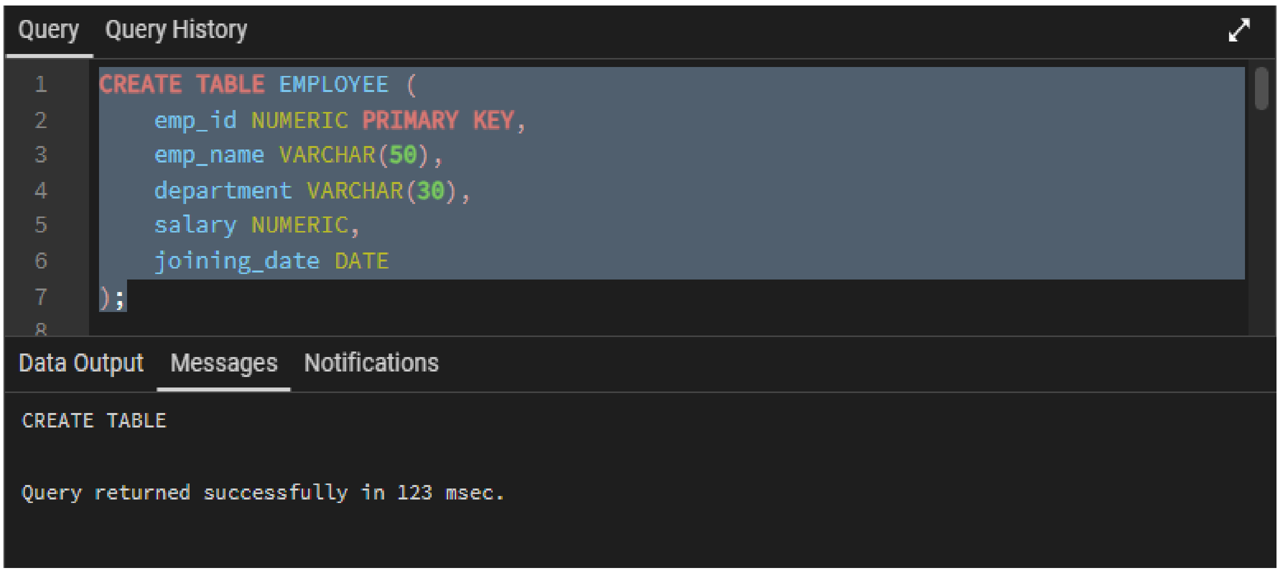
- Employee ID
- Employee Name
- Department
- Salary
- Joining Date

Output:

- Displays department-wise average salary
- Filters employees with salary greater than 20000

- Displays only those departments whose average salary is greater than 30000
- Output is sorted in descending order of average salary

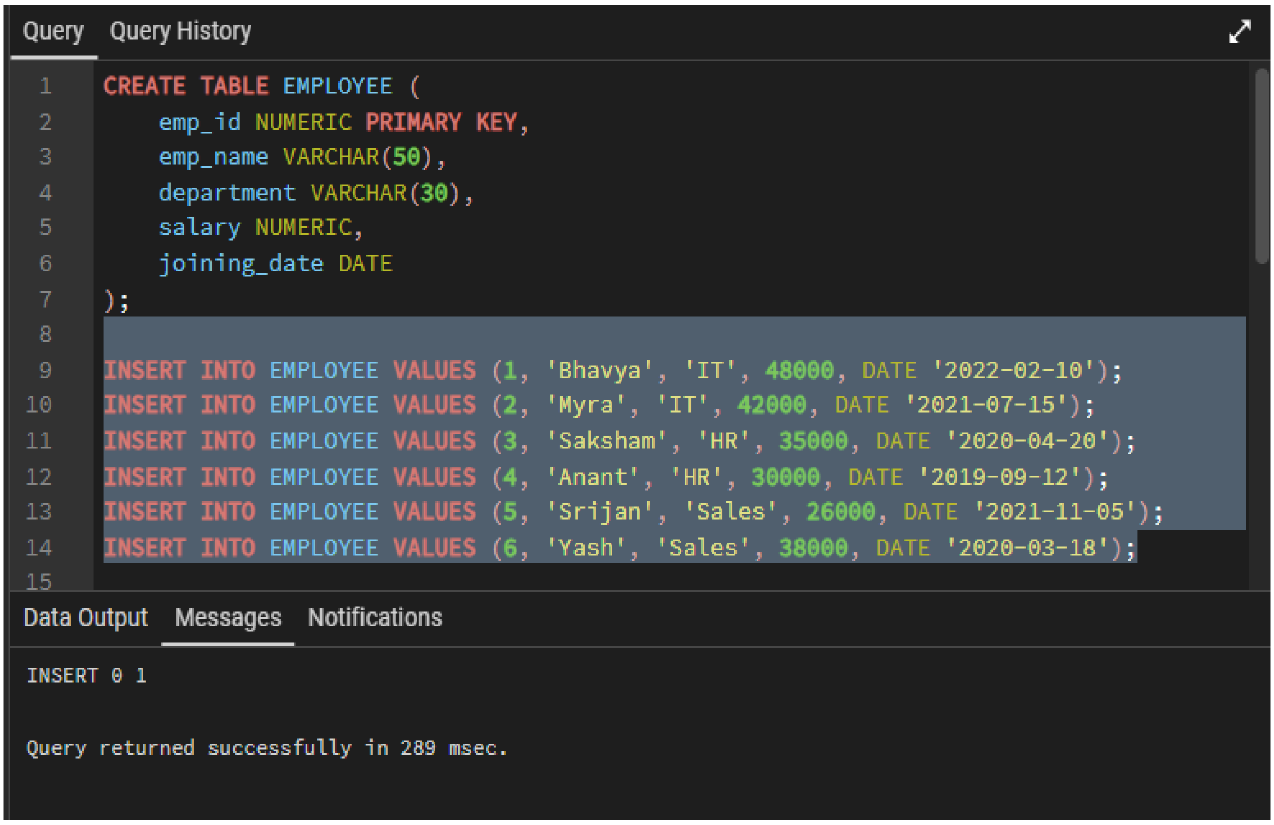
Screenshots:



```
CREATE TABLE EMPLOYEE (
    emp_id NUMERIC PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(30),
    salary NUMERIC,
    joining_date DATE
);
```

```
CREATE TABLE

Query returned successfully in 123 msec.
```

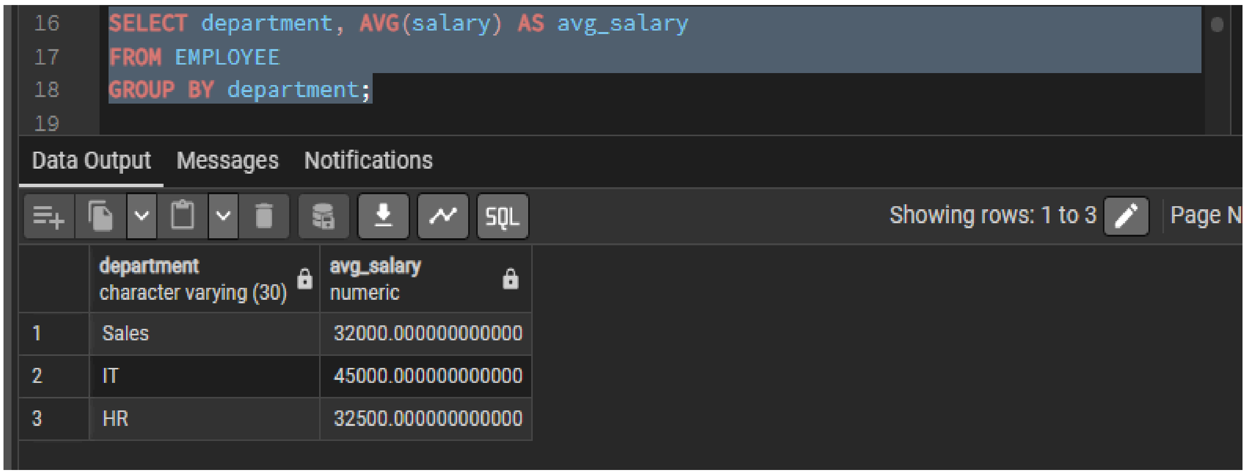


```
CREATE TABLE EMPLOYEE (
    emp_id NUMERIC PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(30),
    salary NUMERIC,
    joining_date DATE
);

INSERT INTO EMPLOYEE VALUES (1, 'Bhavya', 'IT', 48000, DATE '2022-02-10');
INSERT INTO EMPLOYEE VALUES (2, 'Myra', 'IT', 42000, DATE '2021-07-15');
INSERT INTO EMPLOYEE VALUES (3, 'Saksham', 'HR', 35000, DATE '2020-04-20');
INSERT INTO EMPLOYEE VALUES (4, 'Anant', 'HR', 30000, DATE '2019-09-12');
INSERT INTO EMPLOYEE VALUES (5, 'Srijan', 'Sales', 26000, DATE '2021-11-05');
INSERT INTO EMPLOYEE VALUES (6, 'Yash', 'Sales', 38000, DATE '2020-03-18');
```

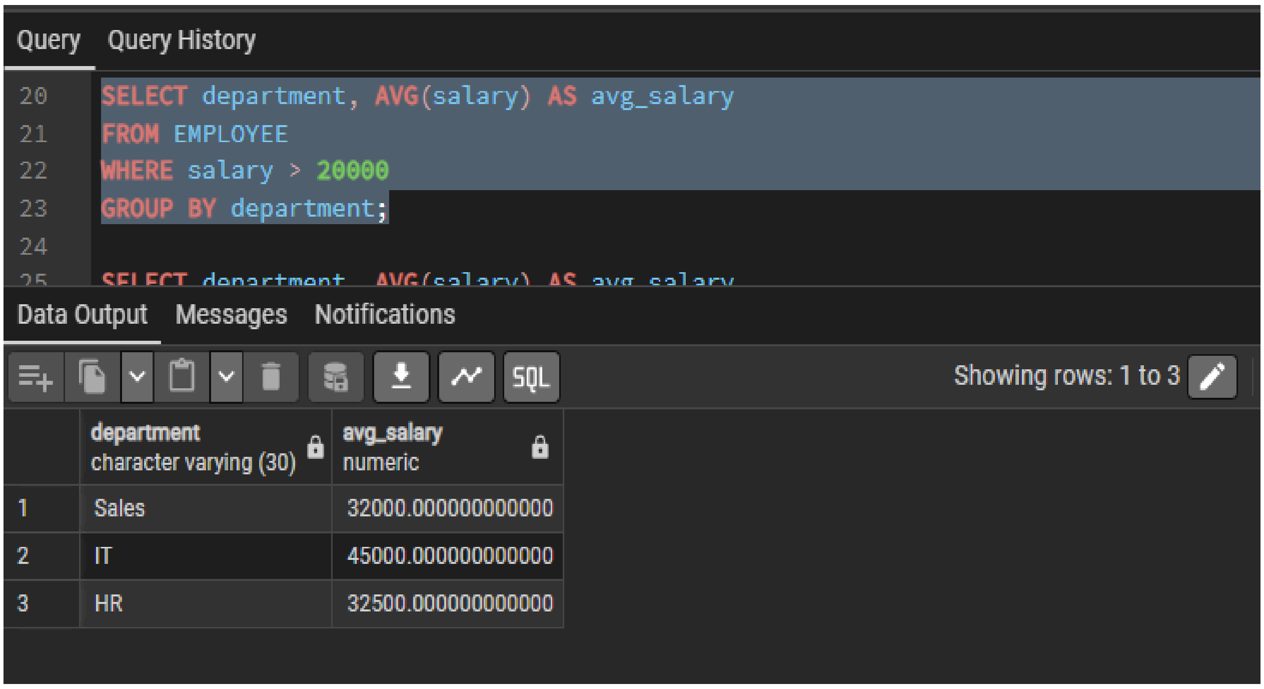
```
INSERT 0 1

Query returned successfully in 289 msec.
```



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

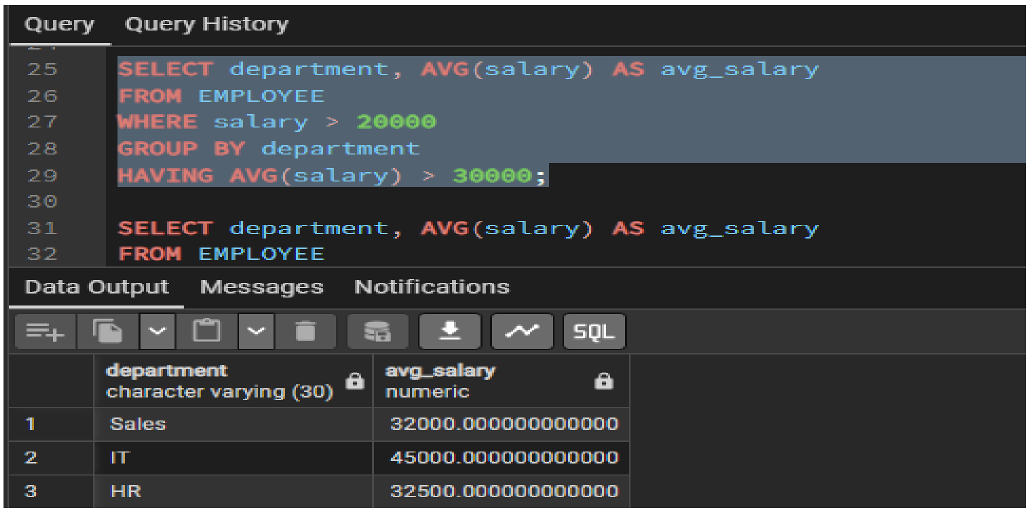
	department character varying (30)	avg_salary numeric
1	Sales	32000.000000000000
2	IT	45000.000000000000
3	HR	32500.000000000000



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

SELECT department, AVG(salary) AS avg_salary
```

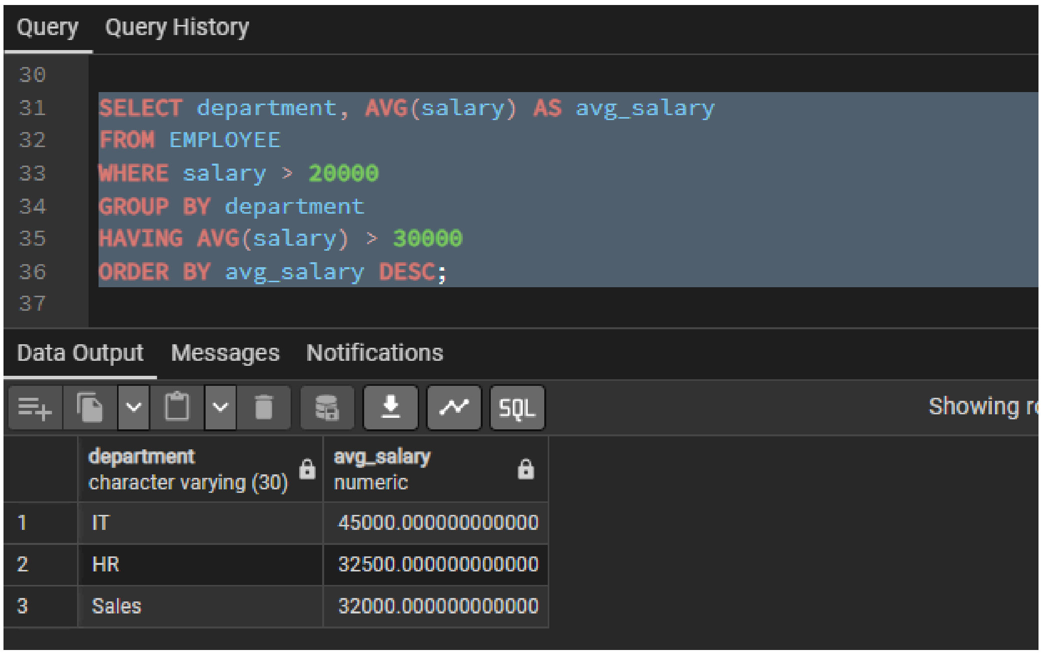
	department character varying (30)	avg_salary numeric
1	Sales	32000.000000000000
2	IT	45000.000000000000
3	HR	32500.000000000000



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

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

	department character varying (30)	avg_salary numeric
1	Sales	32000.000000000000
2	IT	45000.000000000000
3	HR	32500.000000000000



```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;
```

	department character varying (30)	avg_salary numeric
1	IT	45000.000000000000
2	HR	32500.000000000000
3	Sales	32000.000000000000

7. Learning Outcome

After completing this experiment, students 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 the **ORDER BY** clause.
- Use aggregate functions like **AVG()** for data analysis.