Experiment 2

Student Name: Reyansh Arora	UID: 24BAI70273
Branch: CSE - AIML	Section/Group: 24AIT_KRG G1
Semester: 4	Date of Performance: 16/1/26
Subject Name: DBMS	Subject Code: 24CSH-298

Aim

To understand and implement SQL SELECT queries using various clauses such as WHERE, ORDER BY, GROUP BY, and HAVING to retrieve and manipulate data efficiently from relational database tables.

Software Requirements

- Database Management System:
    - PostgreSQL
- Database Administration Tool:
    - pgAdmin

Objectives

- 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().

Problem Statement

An organization maintains an EMPLOYEE table to store details of its employees. The structure of the table is as follows:

Column Name	Data Type
emp_id	NUMBER
emp_name	VARCHAR
Department	VARCHAR
Salary	NUMBER
joining_date	DATE

Practical/Experiment Steps

- Schema Definition: Constructed the fundamental EMPLOYEE table structure, defining specific data types for employee IDs, names, departments, salaries, and joining dates.
- Data Population: Seeded the database with sample employee records across various departments (IT, HR, Finance) to create a functional dataset for testing.
- Aggregate Data Analysis: Implemented GROUP BY operations to calculate the average salary for each department using the AVG() aggregate function.
- Conditional Filtering: Applied high-level filtering logic using the HAVING clause to isolate specific records, such as employees with salaries exceeding 20,000.
- Data Sorting & Grouped Constraints: Configured queries to sort department averages in descending order and practiced applying secondary filters to grouped results.

Procedure

- Logged into the pgAdmin administration tool and established a connection to the PostgreSQL database server.
- Initialized a new database environment to house the employee management system.
- Ran the CREATE TABLE command to define the EMPLOYEE schema, ensuring EMP_ID was set as the Primary Key.
- Executed multiple INSERT statements to populate the table with diverse sample books and visitor profiles—in this case, employee records.
- Used SELECT queries paired with GROUP BY to verify that data was correctly stored and consistent across the table.
- Applied HAVING and WHERE clauses to test how the system handles specific data retrieval conditions.
- Utilized the ORDER BY clause to arrange the output in descending order based on average salaries.

- Tested and verified the effectiveness of security or logic policies by ensuring queries returned expected results or empty sets when conditions weren't met.
- Saved the final SQL script and captured screenshots of the execution results for record maintenance.

Input/Output Analysis

```
CREATE TABLE EMPLOYEE (
   emp_id NUMERIC(10,0) PRIMARY KEY,
   emp_name VARCHAR(50),
   department VARCHAR(30),
   salary NUMERIC(10,0),
   joining_date DATE
);


INSERT INTO EMPLOYEE VALUES (1, 'Aman', 'IT', 55000, '2022-01-10');
INSERT INTO EMPLOYEE VALUES (2, 'Rohit', 'IT', 48000, '2021-07-15');
INSERT INTO EMPLOYEE VALUES (3, 'Neha', 'IT', 62000, '2020-03-20');
INSERT INTO EMPLOYEE VALUES (4, 'Simran', 'HR', 53000, '2021-11-05');
INSERT INTO EMPLOYEE VALUES (5, 'Karan', 'HR', 45000, '2022-06-18');

select * from employee
```

```
-- COUNT NUMBER OF EMPLOYEES IN EACH DEPARTMENT

-- (I)
SELECT DEPARTMENT ,COUNT(*) AS COUNT_EMPLOYEES
FROM EMPLOYEE
GROUP BY DEPARTMENT
-- (II)

SELECT DEPARTMENT ,COUNT(EMP_ID) AS COUNT_EMPLOYEES
```

```
FROM EMPLOYEE
GROUP BY DEPARTMENT


--- SORT ON THE BASIS OF COUNT OF EMPLOYEES IN EACH
DEPARTMENT

SELECT DEPARTMENT ,COUNT(EMP_ID) AS COUNT_EMPLOYEES
FROM EMPLOYEE
GROUP BY DEPARTMENT
ORDER BY COUNT_EMPLOYEES ASC


SELECT DEPARTMENT ,COUNT(*) AS COUNT_EMPLOYEES
FROM EMPLOYEE
GROUP BY DEPARTMENT
ORDER BY COUNT(*) ASC



SELECT DEPARTMENT ,COUNT(EMP_ID) AS COUNT_EMPLOYEES
FROM EMPLOYEE
GROUP BY DEPARTMENT
HAVING COUNT(EMP_ID)>=3


-- FIND AVERAGE SALARY OF EACH DEPARTMENT


SELECT DEPARTMENT ,AVG(SALARY)::NUMERIC(10,2) AS
AVERAGE_SALARY
FROM EMPLOYEE
GROUP BY DEPARTMENT

--SUM,MIN,MAX
```

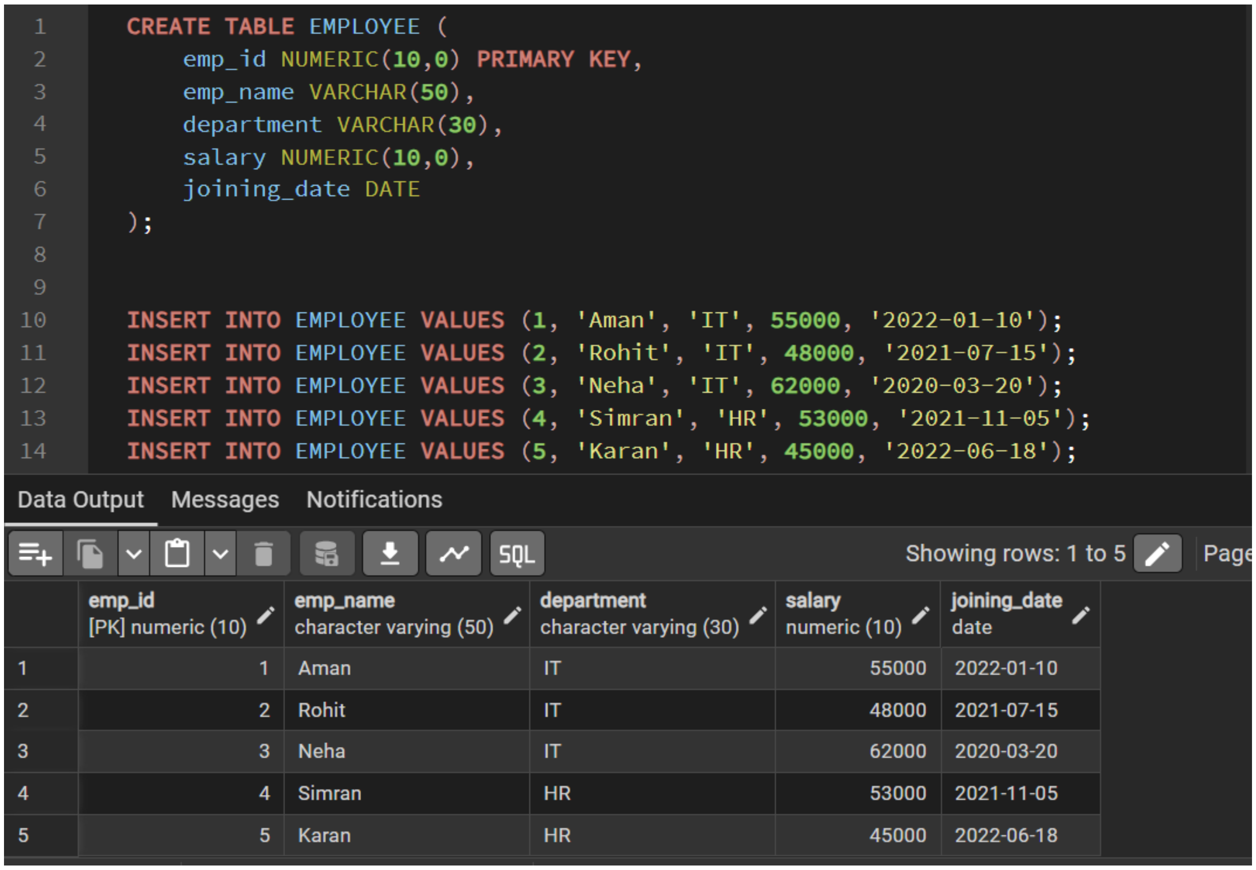
select department, sum(salary):: numeric(10,2) as average_sum
from employee
group by department

select department, min(salary):: numeric(10,2) as average_min
from employee
group by department

select department, max(salary):: numeric(10,2) as average_max
from employee
group by department

Output

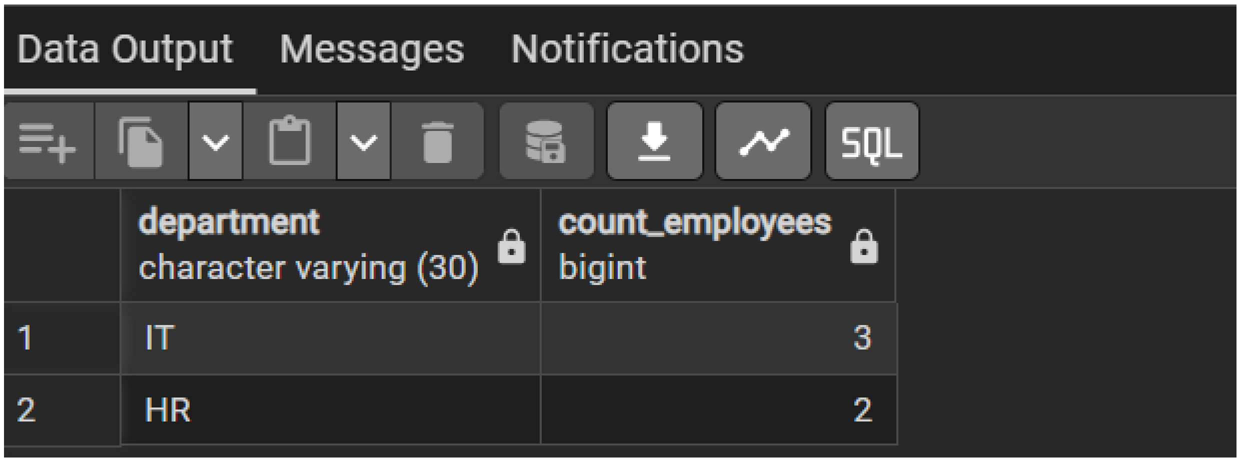
Table created



```
CREATE TABLE EMPLOYEE (
    emp_id NUMERIC(10,0) PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(30),
    salary NUMERIC(10,0),
    joining_date DATE
);


INSERT INTO EMPLOYEE VALUES (1, 'Aman', 'IT', 55000, '2022-01-10');
INSERT INTO EMPLOYEE VALUES (2, 'Rohit', 'IT', 48000, '2021-07-15');
INSERT INTO EMPLOYEE VALUES (3, 'Neha', 'IT', 62000, '2020-03-20');
INSERT INTO EMPLOYEE VALUES (4, 'Simran', 'HR', 53000, '2021-11-05');
INSERT INTO EMPLOYEE VALUES (5, 'Karan', 'HR', 45000, '2022-06-18');
```

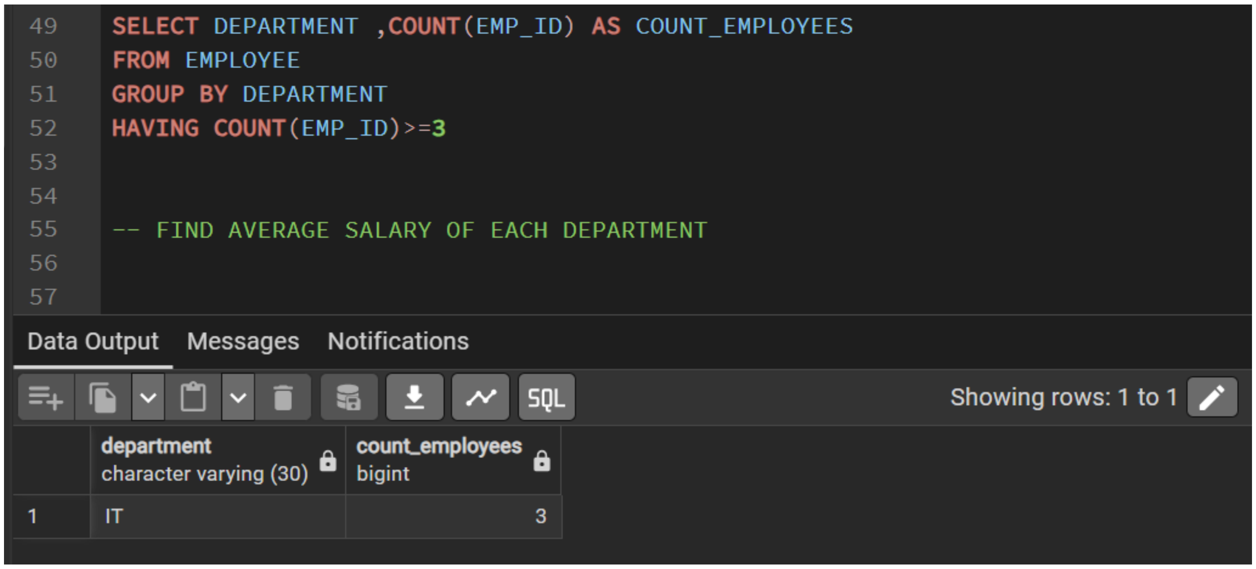
	emp_id [PK] numeric (10)	emp_name character varying (50)	department character varying (30)	salary numeric (10)	joining_date date
1	1	Aman	IT	55000	2022-01-10
2	2	Rohit	IT	48000	2021-07-15
3	3	Neha	IT	62000	2020-03-20
4	4	Simran	HR	53000	2021-11-05
5	5	Karan	HR	45000	2022-06-18

Count of Employees



	department character varying (30)	count_employees bigint
1	IT	3
2	HR	2

Company having count>=3

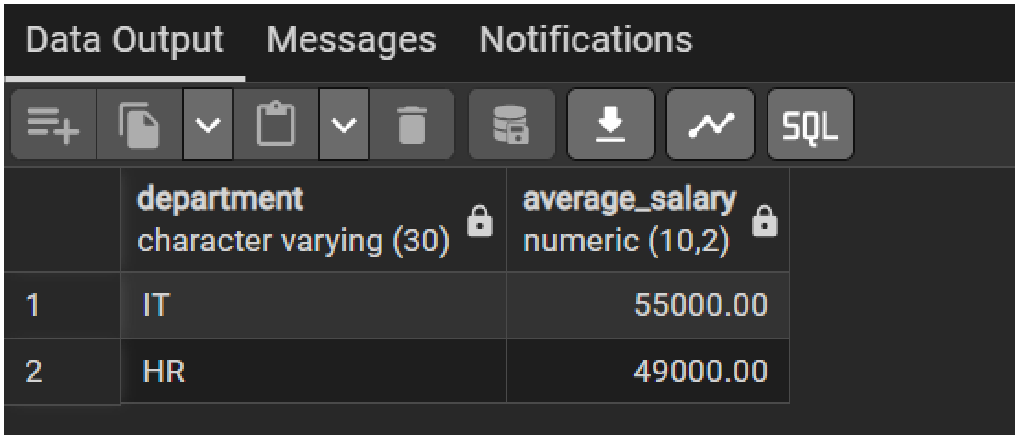


```
SELECT DEPARTMENT ,COUNT(EMP_ID) AS COUNT_EMPLOYEES
FROM EMPLOYEE
GROUP BY DEPARTMENT
HAVING COUNT(EMP_ID)>=3


-- FIND AVERAGE SALARY OF EACH DEPARTMENT

```

Showing rows: 1 to 1

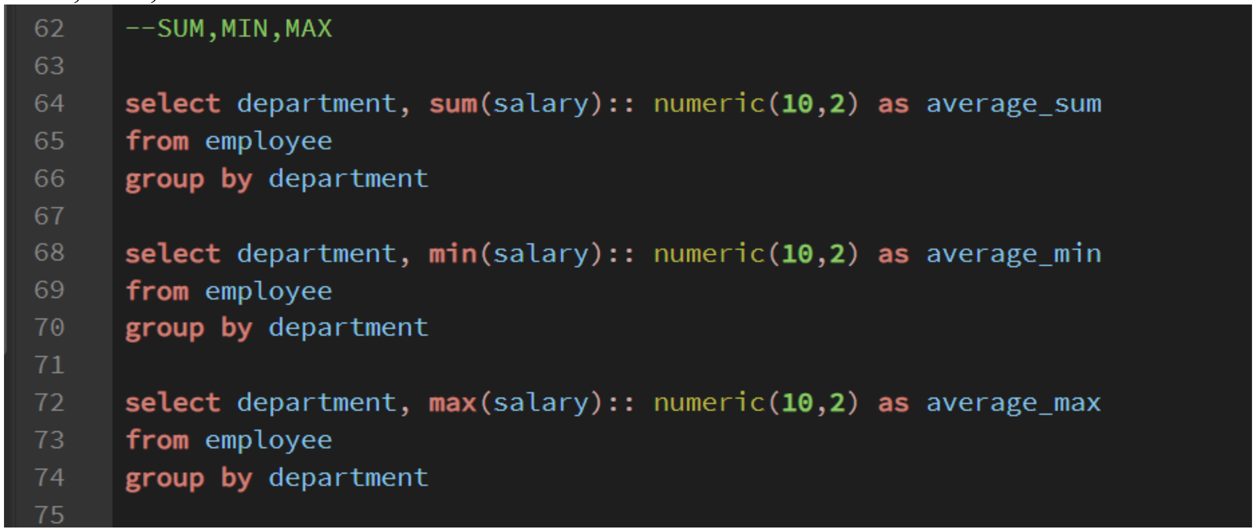
	department character varying (30)	count_employees bigint
1	IT	3

Average salaries of department


Data Output Messages Notifications

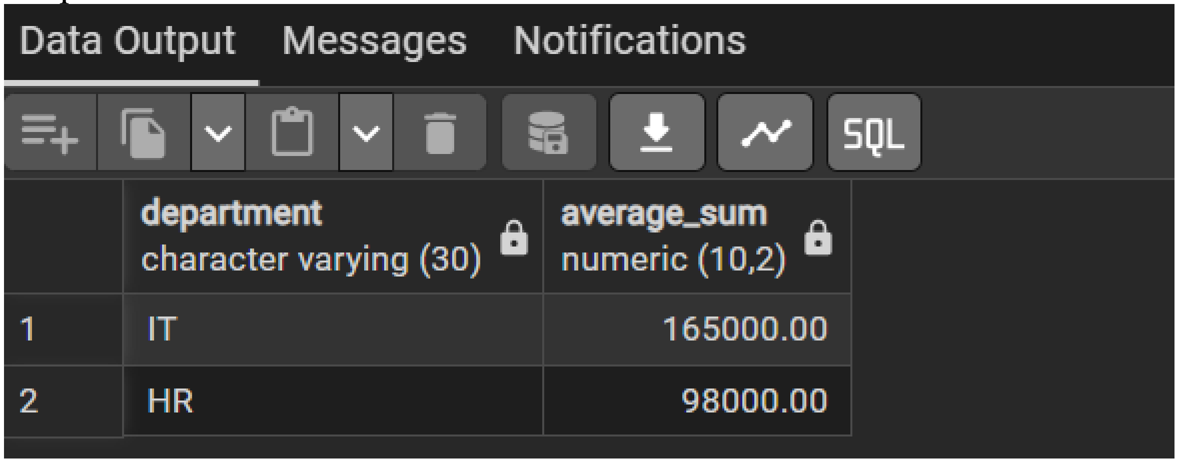
	department character varying (30)	average_salary numeric (10,2)
1	IT	55000.00
2	HR	49000.00

Sum, min, max



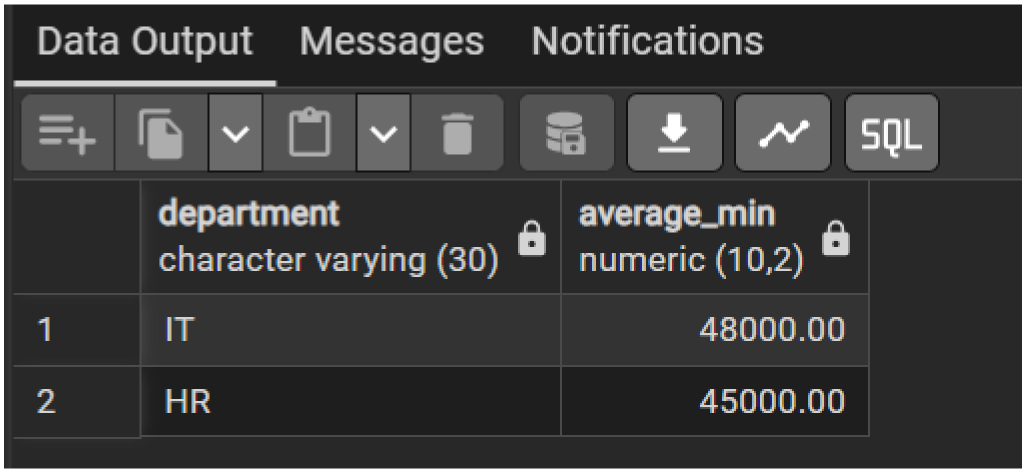
```
--SUM,MIN,MAX

select department, sum(salary):: numeric(10,2) as average_sum
from employee
group by department

select department, min(salary):: numeric(10,2) as average_min
from employee
group by department

select department, max(salary):: numeric(10,2) as average_max
from employee
group by department
```

Output: 1. Sum


Data Output Messages Notifications

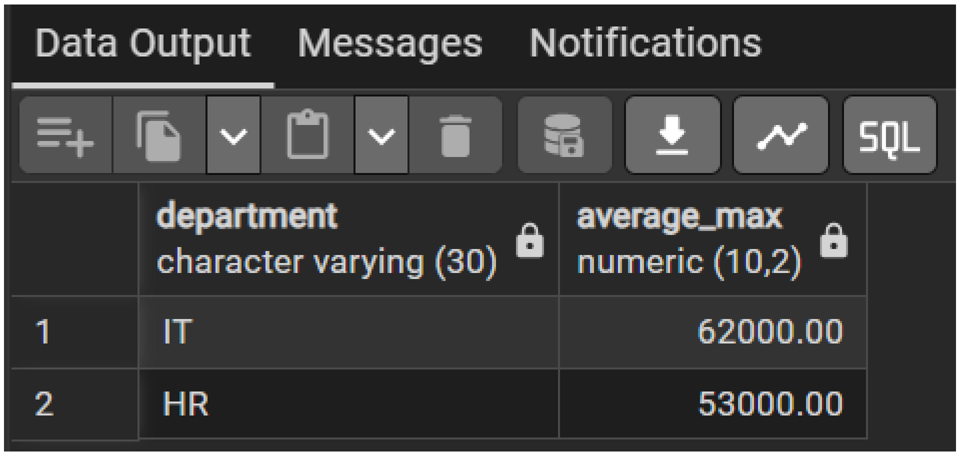
	department character varying (30)	average_sum numeric (10,2)
1	IT	165000.00
2	HR	98000.00

2. min



	department character varying (30)	average_min numeric (10,2)
1	IT	48000.00
2	HR	45000.00

3.max



	department character varying (30)	average_max numeric (10,2)
1	IT	62000.00
2	HR	53000.00

Learning Outcomes

- Learn 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.