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 PRIMARY KEY,

EMP\_NAME VARCHAR(20),

DEPARTMENT VARCHAR(20),

SALARY NUMERIC(10,2),

JOINING\_DATE DATE

)

INSERT INTO EMPLOYEE VALUES(1, 'Aman', 'IT', 30000, '2023-05-23');

INSERT INTO EMPLOYEE VALUES(2, 'Sam', 'IT', 25000, '2016-05-23');

INSERT INTO EMPLOYEE VALUES(3, 'Neha', 'HR', 18000, '2025-09-19');

INSERT INTO EMPLOYEE VALUES(4, 'Suman', 'Finance', 20000, '2021-11-06');

INSERT INTO EMPLOYEE VALUES(5, 'Rohan', 'Finance', 24500, '2023-10-23');

INSERT INTO EMPLOYEE VALUES(6, 'Aditi', 'HR', 28000, '2018-04-16');

INSERT INTO EMPLOYEE VALUES(7, 'Aanya', 'IT', 26000, '2022-07-07')

SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE

GROUP BY DEPARTMENT

SELECT EMP\_ID, EMP\_NAME, SALARY

FROM EMPLOYEE

GROUP BY EMP\_ID

HAVING SALARY>20000

SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE

GROUP BY DEPARTMENT

HAVING AVG(SALARY)>30000

SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE

GROUP BY DEPARTMENT

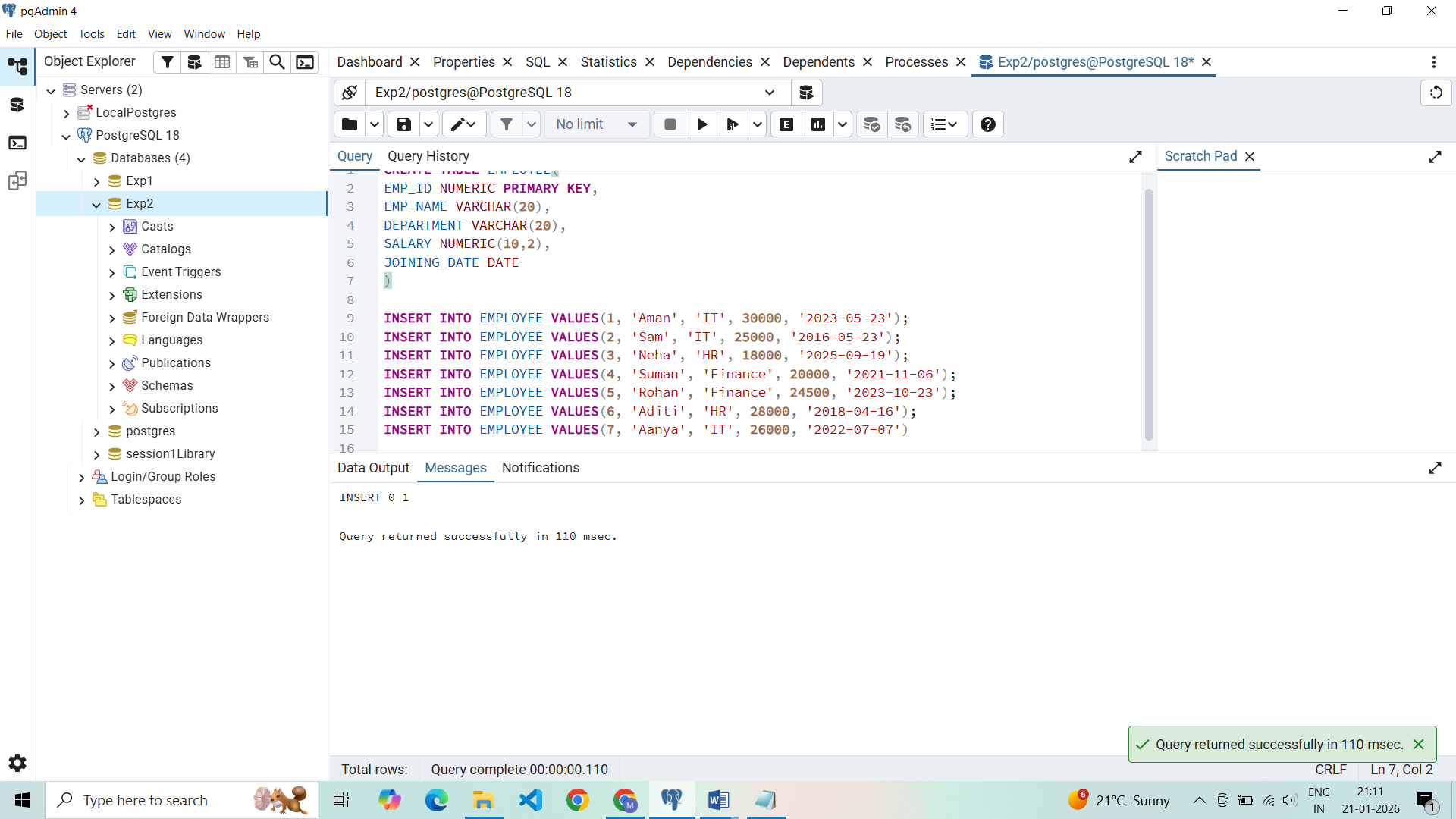
ORDER BY AVG(SALARY) DESC

## **Output**

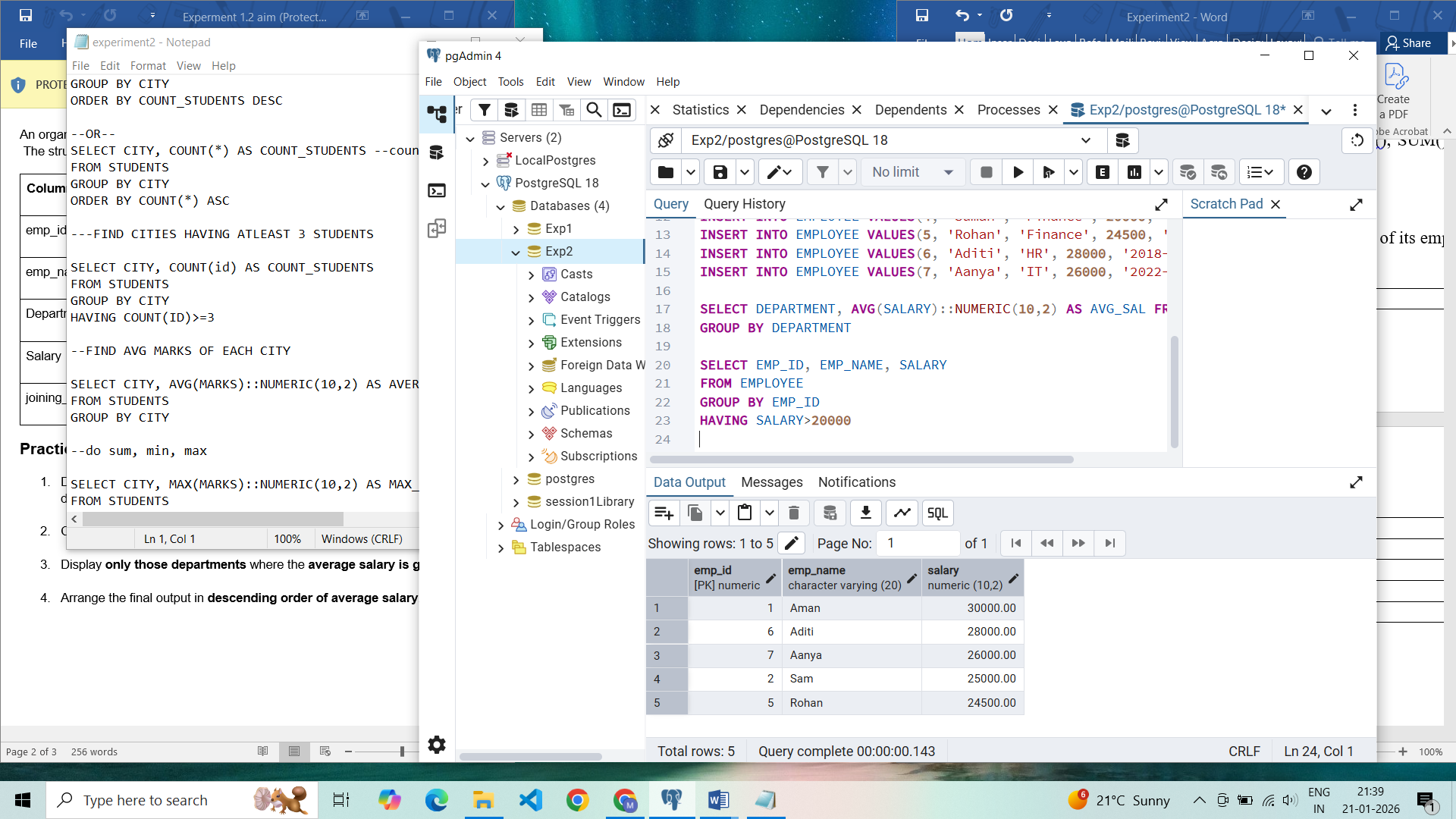
Table created



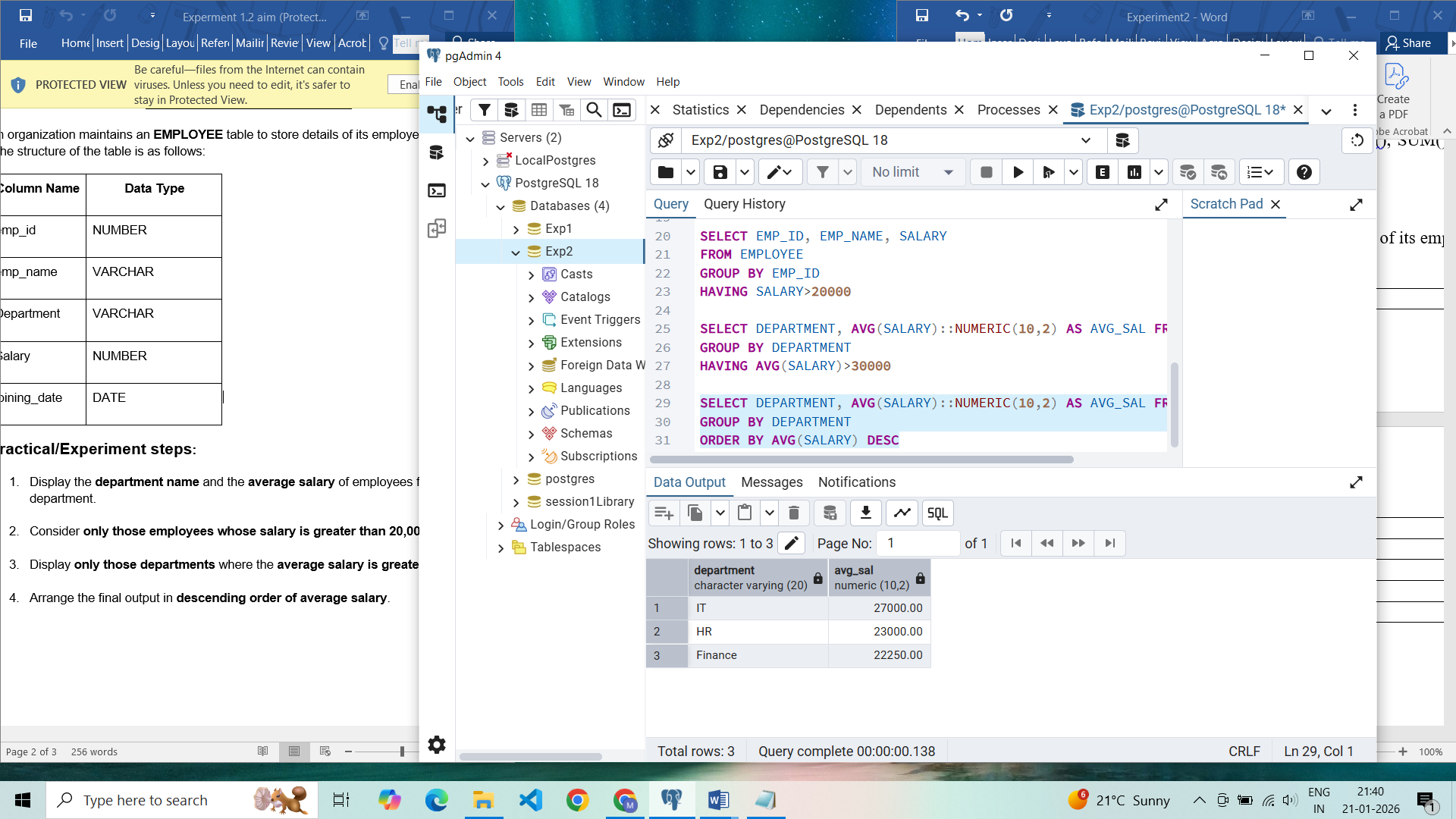
Records inserted



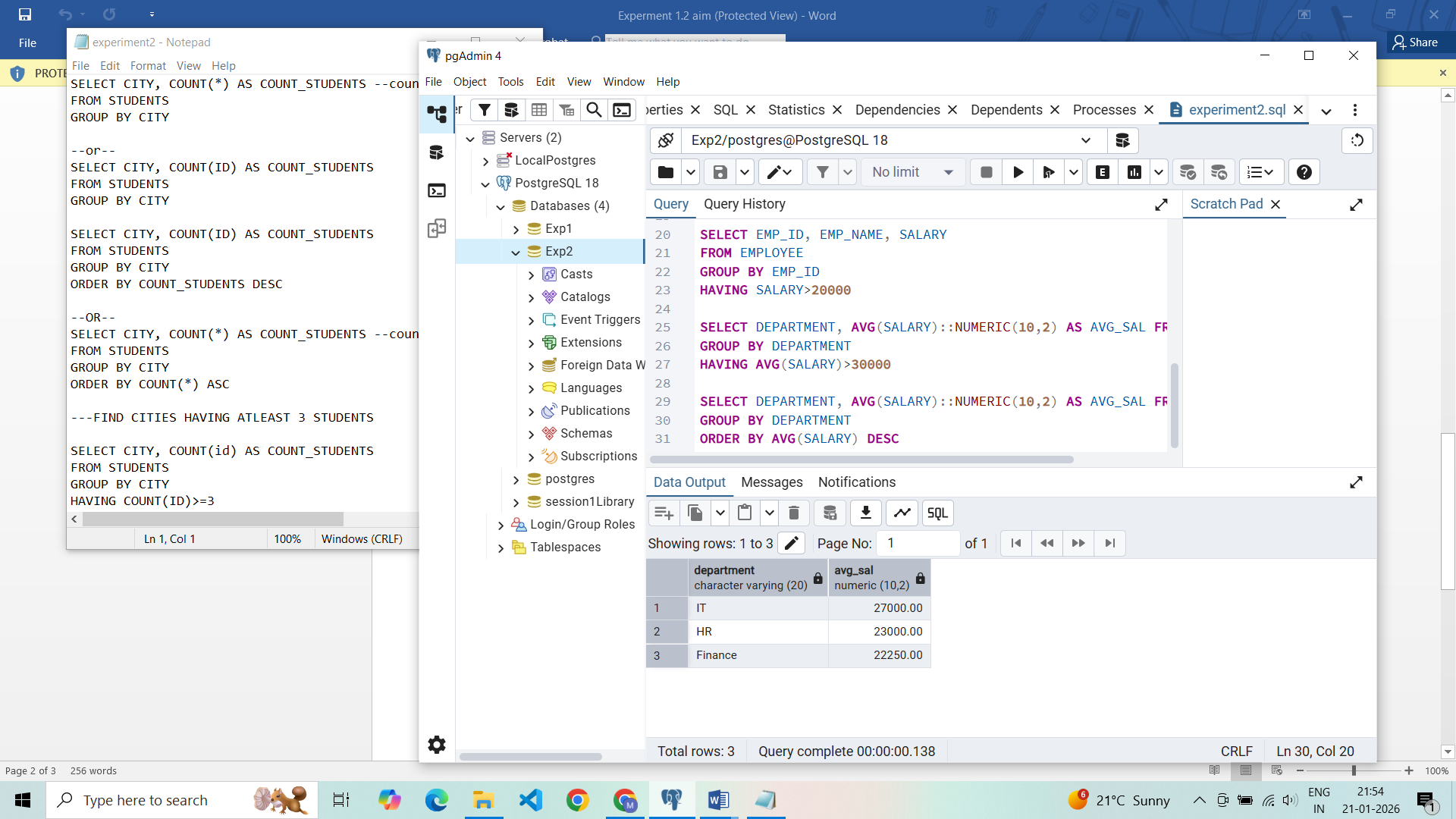
Employees with salaries greater than 20,000



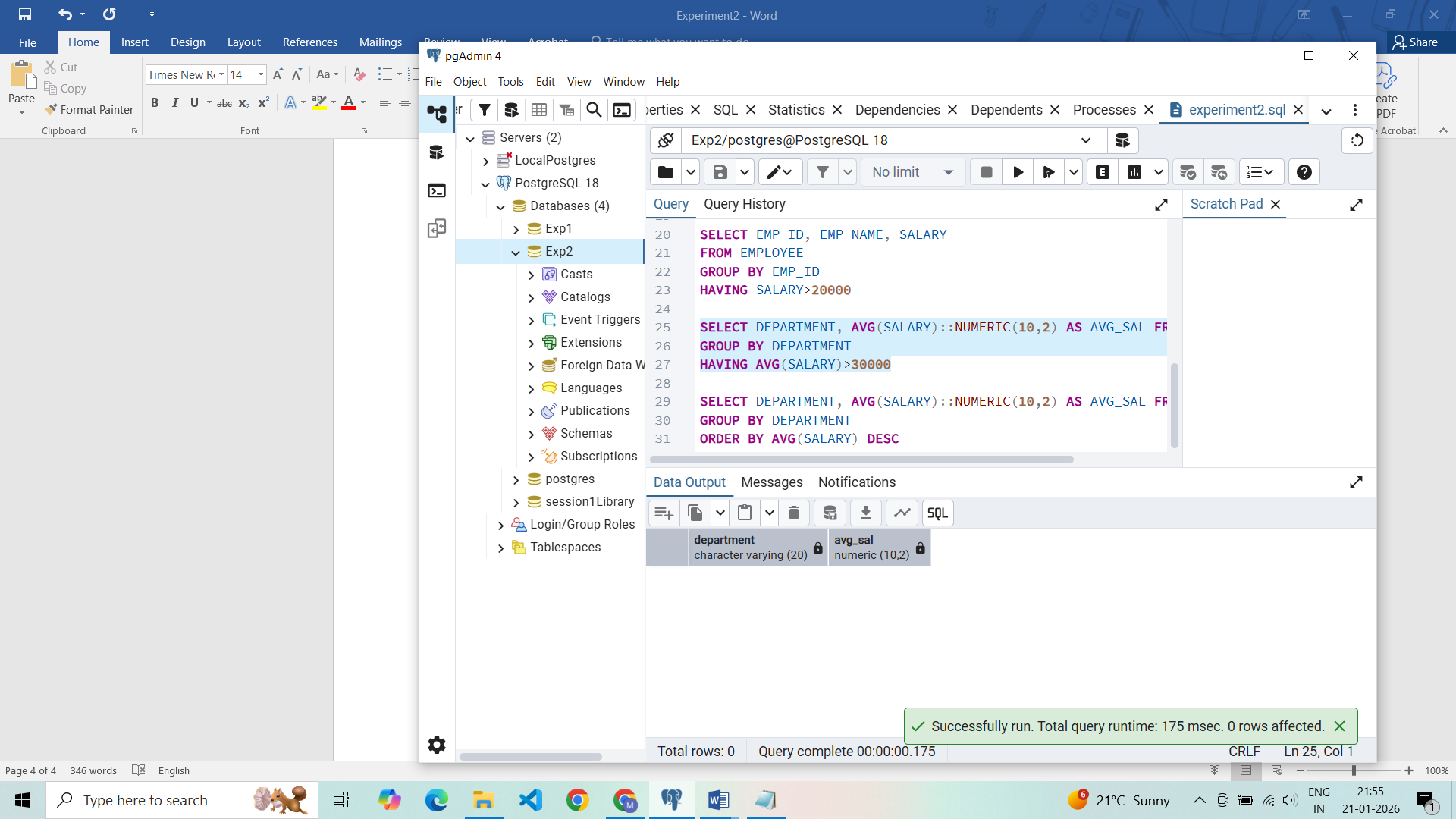
Average salaries of department



Sorting average salaries in descending order:



Departments with average salary more than 30,000 (empty because none)



**I/O ANALYSIS TABLE**

|  |  |
| --- | --- |
| **Input (SQL Query)** | **Output/Purpose** |
| CREATE TABLE EMPLOYEE(...) | Table created. |
| INSERT INTO EMPLOYEE VALUES(...) (All seven records) | Records inserted. |
| SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE GROUP BY DEPARTMENT | Displays the average salary for each department. |
| SELECT EMP\_ID, EMP\_NAME, SALARY FROM EMPLOYEE GROUP BY EMP\_ID HAVING SALARY>20000 | Displays employee IDs, names, and salaries for employees whose salary is greater than 20,000. |
| SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE GROUP BY DEPARTMENT HAVING AVG(SALARY)>30000 | Attempts to display departments with an average salary greater than 30,000 (Result is empty). |
| SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG\_SAL FROM EMPLOYEE GROUP BY DEPARTMENT ORDER BY AVG(SALARY) DESC | Displays the average salary for each department, sorted in descending order of average salary. |

## **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.