Name: Bhavik Pahuja

UID: 24BAI70522

Course: Bachelor of CSE (AI) – 2nd Year

Subject: Database Management Systems

Experiment 2: Advanced Data Aggregation and Filtering

1. Aim of the Session

The aim of this practical is to implement and analyze Group Functions and Conditional Filtering in SQL. The session focuses on using GROUP BY, HAVING, and ORDER BY clauses to extract meaningful insights from an employee dataset.

2. Objective of the Session

By completing this practical, I have achieved the following:

- Developed a schema for employee management using appropriate data types like NUMERIC and DATE.
- Mastered the use of Aggregate Functions (specifically AVG) to perform calculations on data groups.
- Learned to differentiate between the WHERE clause (row-level filtering) and the HAVING clause (group-level filtering).
- Gained proficiency in sorting aggregated results using the ORDER BY clause.

3. Practical / Experiment Steps

The following implementation tasks were completed:

1. Schema Definition: Created the employee table with constraints and precise numeric scaling for salaries.
2. Data Population: Inserted diverse records representing various departments (IT, HR, Sales, Finance) and salary ranges.
3. Basic Aggregation: Calculated the average salary per department using the GROUP BY clause.
4. Advanced Filtering: Applied the HAVING clause to filter out departments where the average salary did not meet a specific threshold.

5. Complex Querying: Combined WHERE, GROUP BY, HAVING, and ORDER BY into a single query to refine results based on individual salaries and group averages simultaneously.

4. Procedure of the Practical

The experiment was conducted following these sequential steps:

1. System Initialization: Logged into the PostgreSQL environment via pgAdmin 4 using localhost as the host server.
2. Table Construction: Executed the CREATE TABLE command to define the structure for the employee dataset.
3. Data Insertion: Ran multiple INSERT statements to populate the table with the provided employee data.
4. Initial Verification: Used SELECT * to confirm that all employee records were correctly stored and formatted.
5. Group Analysis: Executed a GROUP BY query to observe the distribution of average salaries across different departments.
6. Applying Group Filters: Integrated the HAVING clause to restrict the output to high-paying departments (Average > 30,000).
7. Final Refinement: Executed a comprehensive query that filtered individual employees (Salary > 20,000), grouped them by department, and sorted the results in descending order.
8. Output Recording: Captured screenshots of the query results and saved the final SQL script for documentation.

5. I/O Analysis (Input / Output Analysis)

Input Queries

SQL

```
-- Table creation
CREATE TABLE employee(
    emp_id NUMERIC PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(50),
```

```
    salary NUMERIC(10,2),
    joining_date DATE
);


-- Advanced Aggregation Query
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 Details

- Aggregate Results: The system successfully grouped employees by department.
- Filtering Logic: The WHERE clause correctly excluded employees with salaries under 20,000 (like Sara and Vikram) before calculating averages.
- Group Filtering: The HAVING clause ensured only departments with an average salary exceeding 30,000 were displayed in the final output.
- Sorting: The ORDER BY clause successfully sorted the final results from highest to lowest average salary.

```
Run | Tab | JSON | Select | Ask AI
SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
FROM EMPLOYEE
GROUP BY DEPARTMENT
ORDER BY AVG(SALARY) DESC;  19ms      You, now • Uncommitted changes
```

EMPLOYEE

Search Results | Export | Cost: 26ms | 1 | Total 3

department	avg_sal
IT	27000.00
HR	23000.00
Finance	22250.00

```
Run | Tab | JSON | Select | Ask AI
SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
FROM EMPLOYEE
GROUP BY DEPARTMENT
HAVING AVG(SALARY)>30000;  4ms

Run | Tab | JSON | Select | Ask AI
SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
```

EMPLOYEE

Search Results | Export | Cost: 10ms | 1 | Total 0

department	avg_sal

```
Run | Tab | JSON | Select | Ask AI
SELECT EMP_ID, EMP_NAME, SALARY
FROM EMPLOYEE
GROUP BY EMP_ID
HAVING SALARY>20000;  6ms

Run | Tab | JSON | Select | Ask AI
SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
FROM EMPLOYEE
```

EMPLOYEE

Search Results | Export | Cost: 12ms | 1 | Total 5

emp_id	emp_nam	salary
1	Aman	30000.00
6	Aditi	28000.00
7	Aanya	26000.00
2	Sam	25000.00
5	Rohan	24500.00

```
Run | Tab | JSON | Ask AI
19  SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
20  FROM EMPLOYEE
21  GROUP BY DEPARTMENT;  11ms
22
    Run | Tab | JSON | Select | Ask AI
23  SELECT EMP_ID, EMP_NAME, SALARY
24  FROM EMPLOYEE
25  GROUP BY EMP_ID
26  HAVING SALARY>20000;
27
    Run | Tab | JSON | Select | Ask AI
28  SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
29  FROM EMPLOYEE
```

EMPLOYEE

Search Results | Export | Cost: 75ms | 1 | Total 3

department	avg_sal
Finance	22250.00
IT	27000.00
HR	23000.00

```
    Run | Select | Ask AI
9   INSERT INTO EMPLOYEE
10  VALUES
11  (1, 'Aman', 'IT', 30000, '2023-05-23'),
12  (2, 'Sam', 'IT', 25000, '2016-05-23'),
13  (3, 'Neha', 'HR', 18000, '2025-09-19'),
14  (4, 'Suman', 'Finance', 20000, '2021-11-06'),
15  (5, 'Rohan', 'Finance', 24500, '2023-10-23'),
16  (6, 'Aditi', 'HR', 28000, '2018-04-16'),
17  (7, 'Aanya', 'IT', 26000, '2022-07-07');  20ms  AffectedRows: 7
18
    Run | Tab | JSON | Ask AI
19  SELECT DEPARTMENT, AVG(SALARY)::NUMERIC(10,2) AS AVG_SAL
```

Result

Search Results | Export | Cost: 233ms

Execution completed in 233ms

6. Learning Outcome

Through this session, I have developed the following competencies:

- Analytical Skills: Gained the ability to transform raw row-level data into high-level summary reports using aggregation.
- Query Logic: Understood the logical execution order of SQL clauses: FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY.
- Practical Exposure: Experienced handling real-world data scenarios, such as department-wise salary analysis and performance-based filtering in a professional database environment.