Name: Mimanshu Gahlaut

UID: 24BAI70038

Course: BE-CSE (AI&ML)

Subject: Database Management System

Experiment 2: Advanced Data Aggregation and Filtering

1. Aim of the Session

The objective of this experiment is to perform advanced data analysis in SQL using grouping and filtering techniques. The practical focuses on applying aggregate functions along with GROUP BY, HAVING, and ORDER BY clauses to analyze departmental salary data from an employee table.

2. Software Requirements

- **Database:** PostgreSQL Database (PgAdmin)

3. Objective of the Session

After completing this experiment, the following objectives were fulfilled:

- Designed an employee database table using suitable data types such as NUMERIC and DATE.
- Applied aggregate functions to compute summary statistics on grouped records.
- Understood the functional difference between WHERE (record-level filtering) and HAVING (group-level filtering).
- Implemented sorting mechanisms on aggregated results using the ORDER BY clause.

4. Practical / Experiment Steps

The experiment was carried out through the following steps:

1. **Table Design:** Defined the structure of the employee table with appropriate constraints and precision for salary values.

2. **Data Insertion:** Added multiple employee records belonging to different departments such as IT, HR, Finance, and Sales.
3. **Group-Based Calculation:** Computed the average salary for each department using SQL aggregate functions.
4. **Conditional Group Filtering:** Used the `HAVING` clause to remove departments that did not satisfy the required average salary condition.
5. **Combined Query Execution:** Executed a single SQL query incorporating `WHERE`, `GROUP BY`, `HAVING`, and `ORDER BY` clauses for refined output.

5. Procedure of the Practical

The following procedure was adopted to perform the experiment:

1. **Environment Setup:** Logged into the PostgreSQL server using pgAdmin.
2. **Table Creation:** Executed the CREATE TABLE command to define the employee schema.
3. **Data Population:** Inserted sample employee records into the table.
4. **Data Verification:** Displayed table contents using SELECT * to ensure correctness.
5. **Aggregation Execution:** Applied GROUP BY to calculate department-wise average salaries.
6. **Applying Filters:** Used HAVING to select only departments with high average salaries.
7. **Final Query Execution:** Applied salary constraints using WHERE and sorted results using ORDER BY.
8. **Documentation:** Captured screenshots of outputs and saved the SQL script.

6. 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
)
-- Data Entry
INSERT INTO employee VALUES (101, 'Amit', 'IT', 19000, '2020-02-12');
INSERT INTO employee VALUES (102, 'Priya', 'HR', 22000, '2019-03-10');
INSERT INTO employee VALUES (103, 'Rahul', 'Sales', 35000, '2021-07-18');
INSERT INTO employee VALUES (104, 'Neha', 'IT', 55000, '2018-09-22');
INSERT INTO employee VALUES (105, 'Rohan', 'Finance', 32000, '2022-01-05');
INSERT INTO employee VALUES (106, 'Sara', 'Sales', 13000, '2020-12-03');
INSERT INTO employee VALUES (107, 'Vikram', 'HR', 12000, '2017-04-11');
SELECT * FROM employee

-- Advanced Aggregation Query
```

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

SELECT department, AVG(salary) AS avg_salary
FROM employee
GROUP BY department
HAVING AVG(salary) > 30000


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

1. Table Creation Output

The employee table was created successfully with the following attributes:

- emp_id defined as PRIMARY KEY, ensuring unique identification.
- salary stored with precision using NUMERIC(10,2).
- joining_date stored using the DATE data type.

✓ No errors occurred during table creation.

2. Data Insertion & Verification Output

After executing the INSERT statements, the following query was executed:

```sql
CREATE TABLE employee(
    emp_id NUMERIC PRIMARY KEY,
    emp_name VARCHAR(50),
    department VARCHAR(50),
    salary NUMERIC(10,2),
    joining_date DATE
)

INSERT INTO employee VALUES (101, 'Amit', 'IT', 19000, '2020-02-12');
INSERT INTO employee VALUES (102, 'Priya', 'HR', 22000, '2019-03-10');
INSERT INTO employee VALUES (103, 'Rahul', 'Sales', 35000, '2021-07-18');
INSERT INTO employee VALUES (104, 'Neha', 'IT', 55000, '2018-09-22');
INSERT INTO employee VALUES (105, 'Rohan', 'Finance', 32000, '2022-01-05');
INSERT INTO employee VALUES (106, 'Sara', 'Sales', 13000, '2020-12-03');
INSERT INTO employee VALUES (107, 'Vikram', 'HR', 12000, '2017-04-11');

SELECT * FROM employee
```

3. Grouping Output (GROUP BY Clause)

```sql
19  SELECT department, AVG(salary) AS avg_salary
20  FROM employee
21  GROUP BY department
```

Data Output | Messages | Notifications

	department character varying (50)	avg_salary numeric
1	Finance	32000.000000000000
2	Sales	24000.000000000000
3	IT	37000.000000000000
4	HR	17000.000000000000

	emp_id [PK] numeric	emp_name character varying (50)	department character varying (50)	salary numeric (10,2)	joining_date date
1	101	Amit	IT	19000.00	2020-02-12
2	102	Priya	HR	22000.00	2019-03-10
3	103	Rahul	Sales	35000.00	2021-07-18
4	104	Neha	IT	55000.00	2018-09-22
5	105	Rohan	Finance	32000.00	2022-01-05
6	106	Sara	Sales	13000.00	2020-12-03
7	107	Vikram	HR	12000.00	2017-04-11

4. Group Filtering Output (HAVING Clause)

```
23  SELECT department, AVG(salary) AS avg_salary
24  FROM employee
25  GROUP BY department
26  HAVING AVG(salary) > 30000
```

Data Output | Messages | Notifications

	department character varying (50)	avg_salary numeric
1	Finance	32000.000000000000
2	IT	37000.000000000000

5. Combined Filtering & Sorting Output (WHERE + HAVING + ORDER BY)

```
29  SELECT department, AVG(salary) AS avg_salary
30  FROM employee
31  WHERE salary > 20000
32  GROUP BY department
33  HAVING AVG(salary) > 30000
34  ORDER BY avg_salary DESC
```

Data Output | Messages | Notifications

	department character varying (50)	avg_salary numeric
1	IT	55000.000000000000
2	Sales	35000.000000000000
3	Finance	32000.000000000000

7. Learning Outcome

- Learned to use aggregate functions like AVG() for data analysis.
- Understood how GROUP BY organizes data into meaningful groups.
- Differentiated between WHERE and HAVING clauses.
- Practiced filtering and sorting results using HAVING and ORDER BY.
- Gained hands-on experience in analyzing real-world employee salary data.