Name: Kamal Sharma

UID: 24BAI70380

Course: BE-CSE (AI&ML)

Subject: Database Management System

Experiment: Advanced Data Aggregation and filtering

1. Aim of the Session

The aim of this lab session was to understand and implement SQL SELECT queries using clauses such as WHERE, ORDER BY, GROUP BY, and HAVING to efficiently retrieve and manipulate data from relational database tables.

2. Software Requirements

- **Database**:
    - Oracle Database Express Edition (Oracle XE)
    - PostgreSQL Database (PgAdmin)

3. Objective of the Session

By the end of the session, the following objectives were achieved:

- 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()`**.

4. Practical / Experiment Steps

The experiment was carried out through the following activities:

1. **Schema Design:** Created the EMPLOYEE table with constraints (PRIMARY KEY, NOT NULL, CHECK).
2. **Data Insertion:** Inserted sample employee records into the table.
3. **Basic Retrieval:** Displayed all records using SELECT *.
4. **Aggregate Analysis:** Applied GROUP BY to calculate average salaries per department.
5. **Conditional Filtering:** Used WHERE to filter employees with salary greater than 20,000.
6. **Grouped Filtering:** Applied HAVING to restrict results to departments with average salary above 30,000.
7. **Sorting:** Ordered results in descending order of average salary using ORDER BY.

5. Procedure of the Practical

Execution was performed in the following order:

1. **Environment Setup:** Logged into DBMS interface and accessed the server instance.
2. **Database Setup:** Created a dedicated database for the library system.
3. **Schema Execution:** Executed CREATE TABLE commands ensuring parent tables were defined first.
4. **Data Entry Phase**: Inserted multiple employee records across IT, HR, and Finance departments.
5. **Verification Queries:** Verified data using SELECT queries
6. .Executed queries step by step:

    - Step 1: Grouped salaries by department.
    - Step 2: Applied `WHERE` clause to filter salaries > 20,000.
    - Step 3: Applied `HAVING` clause to restrict average salary > 30,000.
    - Step 4: Ordered results in descending order of average salary.

7. **Documentation:** Saved final SQL script and captured outputs for reporting.

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

Input Queries

SQL

```
CREATE TABLE EMPLOYEE(

      EMP_ID INT PRIMARY KEY,
```

```
      EMP_NAME VARCHAR(30) NOT NULL,
      DEPARTMENT VARCHAR(30) NOT NULL,
      SALARY INT CHECK(SALARY>0) NOT NULL,
      JOINING_DATE DATE NOT NULL
)

SELECT * FROM EMPLOYEE

INSERT INTO EMPLOYEE
VALUES(101,'KRRISH','IT',47850,'01-08-2022')
INSERT INTO EMPLOYEE
VALUES(102,'NISHANT','HR',37000,'01-01-2024')
INSERT INTO EMPLOYEE
VALUES(103,'ROHIT','FINANCE',18000,'15-04-2025')
INSERT INTO EMPLOYEE
VALUES(104,'LAKSHAY','IT',27850,'01-09-2024')
INSERT INTO EMPLOYEE
VALUES(105,'ARYA','HR',28000,'05-11-2021')
INSERT INTO EMPLOYEE
VALUES(106,'HARSH','FINANCE',16000,'21-07-2023')
INSERT INTO EMPLOYEE
VALUES(107,'SHIVAM','IT',24000,'20-11-2020')
INSERT INTO EMPLOYEE
VALUES(108,'MANAV','FINANCE',21700,'14-11-2022')
INSERT INTO EMPLOYEE
VALUES(109,'VIPUL','HR',31700,'10-10-2025')
```

```
SELECT * FROM EMPLOYEE


--STEP 1
SELECT DEPARTMENT,AVG(SALARY)::NUMERIC(10,2) AS AVG_SALARY
FROM EMPLOYEE
GROUP BY DEPARTMENT


--STEP 2
SELECT DEPARTMENT,AVG(SALARY)::NUMERIC(10,2) AS AVG_SALARY
FROM EMPLOYEE
WHERE SALARY>20000
GROUP BY DEPARTMENT


--STEP 3
SELECT DEPARTMENT,AVG(SALARY)::NUMERIC(10,2) AS AVG_SALARY
FROM EMPLOYEE
WHERE SALARY>20000
GROUP BY DEPARTMENT
HAVING AVG(SALARY)>30000


--STEP 4
SELECT DEPARTMENT,AVG(SALARY)::NUMERIC(10,2) AS AVG_SALARY
FROM EMPLOYEE
WHERE SALARY>20000
GROUP BY DEPARTMENT
```

```
HAVING AVG(SALARY)>30000

ORDER BY AVG(SALARY) DESC
```

Output Details

1. Schema Creation

- EMPLOYEE table created successfully with constraints.
- CHECK(SALARY > 0) ensured valid salary entries.

✓ Result: Schema creation completed without errors.

2. **Data Insertion:**

Records inserted for employees across IT, HR, and Finance departments.

	emp_id [PK] integer	emp_name character varying (30)	department character varying (30)	salary integer	joining_date date
1	101	KRRISH	IT	47850	2022-08-01
2	102	NISHANT	HR	37000	2024-01-01
3	103	ROHIT	FINANCE	18000	2025-04-15
4	104	LAKSHAY	IT	27850	2024-09-01
5	105	ARYA	HR	28000	2021-11-05
6	106	HARSH	FINANCE	16000	2023-07-21
7	107	SHIVAM	IT	24000	2020-11-20
8	108	MANAV	FINANCE	21700	2022-11-14
9	109	VIPUL	HR	31700	2025-10-10

3. Perfomed Stepwise Operations :

Step 1 Output:

- Displayed average salary per department.

	department character varying (30)	avg_salary numeric (10,2)
1	FINANCE	18566.67
2	IT	33233.33
3	HR	32233.33

Step 2 Output:

- Filtered employees with salary > 20,000 before grouping.

	department character varying (30)	avg_salary numeric (10,2)
1	FINANCE	21700.00
2	IT	33233.33
3	HR	32233.33

Step 3 Output:

- Displayed only departments with average salary > 30,000.

	department character varying (30)	avg_salary numeric (10,2)
1	IT	33233.33
2	HR	32233.33

Step 4 Output:

- Final result sorted in descending order of average salary.

	department character varying (30)	avg_salary numeric (10,2)
1	IT	33233.33
2	HR	32233.33

7. Learning Outcome

From this practical, the following knowledge and skills were gained:

- **Learned how to filter records using the WHERE clause.**
- **Understood grouping of records using GROUP BY.**
- **Applied conditions on grouped data using HAVING.**
- **Practiced sorting results using ORDER BY.**
- **Gained insight into aggregate functions (COUNT, SUM, AVG, MIN, MAX) for data analysis.**