AIM

To study and perform SQL **SELECT** operations using clauses like **WHERE, ORDER BY, GROUP BY, and HAVING**, and to analyze table data using aggregate functions.

Software Requirements

Database System

- PostgreSQL

Tool Used

- pgAdmin

Objective

- To write and execute different **SELECT** queries.
- To apply conditions using the **WHERE** clause.
- To arrange query output using **ORDER BY**.
- To group data city-wise using **GROUP BY**.
- To filter grouped results using **HAVING**.
- To use aggregate functions such as **COUNT(), SUM(), AVG(), MIN(), MAX()** for analysis.

Implementation / Work Performed

1) Table Creation

A table named **Students** was created to store basic student details.

It includes:

- **id** → unique student number (**Primary Key**)
- **name** → student name
- **city** → student's city
- **marks** → marks scored

2) Inserting Records

Multiple entries were added into the **Students** table with values for **id, name, city, and marks**.

3) Counting Students City-wise

To find how many students belong to each city:

- **COUNT(*)** was used to count total records
- **COUNT(id)** was also tested (same output here since **id** is always filled)

Grouping was done using:

- **GROUP BY city**

4) Sorting the Result

The city-wise student count was arranged based on number of students.

For sorting, the query used:

- **ORDER BY** (ascending order)

5) Selecting Cities with Minimum 3 Students

To display only those cities where student count is **3 or more**, filtering was done using:

- **HAVING COUNT(*) >= 3**

(Used **HAVING** because the condition is applied after grouping.)

6) Average Marks by City

The average marks of students from each city were calculated using:

- **AVG(marks)**

To show a cleaner output, the average was displayed up to **2 decimal places**.

Procedure (Step-by-Step)

1. Open **pgAdmin** and connect to the PostgreSQL database.
2. Create a table **Students** with columns: id, name, city, marks and set id as **Primary Key**.
3. Insert student data into the table using **INSERT** statements.
4. Run a query to get total students per city using **COUNT()** and **GROUP BY city**.
5. Sort the city-wise count result using **ORDER BY**.
6. Use **HAVING** to display only cities with at least **3 students**.
7. Find average marks per city using **AVG(marks)** with **GROUP BY city**.

Input / Output

Students

id	name	city	marks
1	King	WAKANDA	75
2	President	AMERICA	71
3	Minister	GUJRAT	99
4	Mayor	LONDON	78
5	Sarpanch	DHOLAKPUR	65

CODE:

```
CREATE TABLE Students (
  id NUMERIC PRIMARY KEY,
  name VARCHAR(50),
  city VARCHAR(30),
  marks NUMERIC(10,0)
);
```

```
INSERT INTO Students VALUES (1, 'Aman', 'Mohali', 85);
INSERT INTO Students VALUES (2, 'Rohit', 'Mohali', 78);
INSERT INTO Students VALUES (3, 'Neha', 'Mohali', 92);
INSERT INTO Students VALUES (4, 'Simran', 'Amritsar', 88);
INSERT INTO Students VALUES (5, 'Karan', 'Amritsar', 75);
```

-- COUNT NUMBER OF STUDENT IN EACH CITY

-- (I)

```
SELECT CITY ,COUNT(*) AS COUNT_STUDNETS
FROM STUDENTS
GROUP BY CITY
```

-- (II)

```
SELECT CITY ,COUNT(ID) AS COUNT_STUDNETS
FROM STUDENTS
GROUP BY CITY
```

--- SORT ON THE BASIS OF COUNT OF STUDENTS IN EACH CITY

-- (I)

```
SELECT CITY ,COUNT(ID) AS COUNT_STUDNETS
FROM STUDENTS
GROUP BY CITY
ORDER BY COUNT_STUDNETS ASC
```

```
-- (II)

SELECT CITY ,COUNT(*) AS COUNT_STUDNETS
FROM STUDENTS
GROUP BY CITY
ORDER BY COUNT(*) ASC

-- FIND CITIES HAVING COUNT AT LEAST 3


SELECT CITY ,COUNT(ID) AS COUNT_STUDNETS
FROM STUDENTS
GROUP BY CITY
HAVING COUNT(ID)>=3

-----
-- FIND AVERAGE MARKS OF EACH CITY


SELECT CITY ,AVG(MARKS)::NUMERIC(10,2) AS AVERAGE_MARKS
FROM STUDENTS
GROUP BY CITY
```

Learning Outcomes

- Understood how to filter rows using the **WHERE** clause.
- Learned to group records and apply aggregate functions using **GROUP BY**.
- Understood the difference between **WHERE vs HAVING** for filtering.
- Learned how to sort query output properly using **ORDER BY**.