Student Name: Diwansh	UID: 24BAI70294
Branch: CSE - AIML	Section/Group: 24AIT_KRG G1
Semester: 4	Date of Performance: 16/1/26
Subject Name: DBMS	Subject Code: 24CSH-298

Experiment 2: SQL GROUP BY, ORDER BY, and HAVING with Aggregation Functions

---

1. Aim of the Session

The aim of this practical is to gain hands-on experience in using advanced SQL clauses for **data grouping, sorting, aggregation, and filtering**. Students learn how to generate meaningful statistical reports from database records using **GROUP BY, ORDER BY, HAVING**, and aggregation functions.

---

2. Objectives of the Session

After completing this experiment, students will be able to:

- Understand **GROUP BY** for organizing records into categories
- Use aggregation functions: **COUNT, AVG**
- Apply **ORDER BY** to sort grouped results
- Use **HAVING** to filter grouped data
- Generate statistical reports from student records
- Write multi-clause SQL queries

---

3. Practical / Experiment Steps

Step 1: Create Students Table

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

   marks NUMERIC(10,0)

);
```

Step 2: Insert Sample Student Data

```
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);

INSERT INTO Students VALUES (6, 'Diwansh', 'Chandigarh', 90);
```

Step 3: Retrieve All Records

```
SELECT * FROM Students;
```

Step 4: GROUP BY with COUNT

```
SELECT city, COUNT(*) AS COUNT_STUDENTS

FROM Students

GROUP BY city;
```

Purpose: Counts number of students in each city.

Step 5: GROUP BY + ORDER BY

```
SELECT city, COUNT(id) AS COUNT_STUDENTS

FROM Students

GROUP BY city

ORDER BY COUNT_STUDENTS ASC;
```

Purpose:
Groups students city-wise and sorts cities in ascending order of student count.

Step 6: GROUP BY + HAVING

```
SELECT city, COUNT(*) AS COUNT_STUDENTS

FROM Students

GROUP BY city

HAVING COUNT(*) >= 3;
```

Purpose:
Displays only cities having **3 or more students**.

Step 7: GROUP BY with AVG

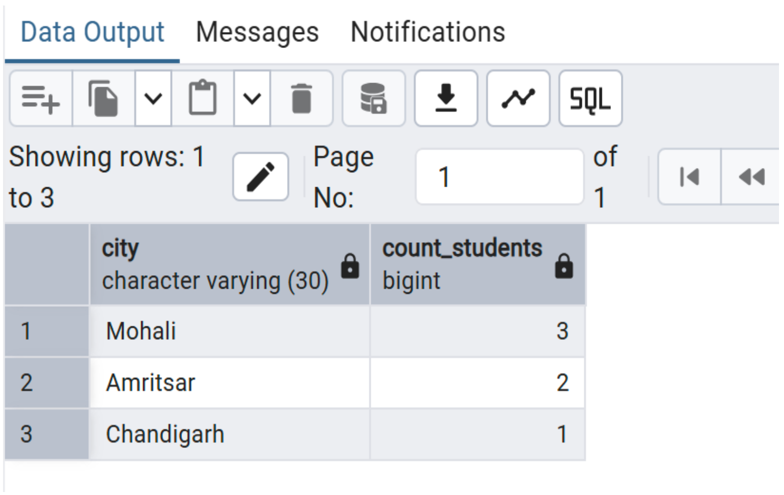
```
SELECT city, AVG(marks)::NUMERIC(10,2) AS AVERAGE_MARKS

FROM Students

GROUP BY city;
```

Purpose:
Calculates **average marks** of students city-wise.

---

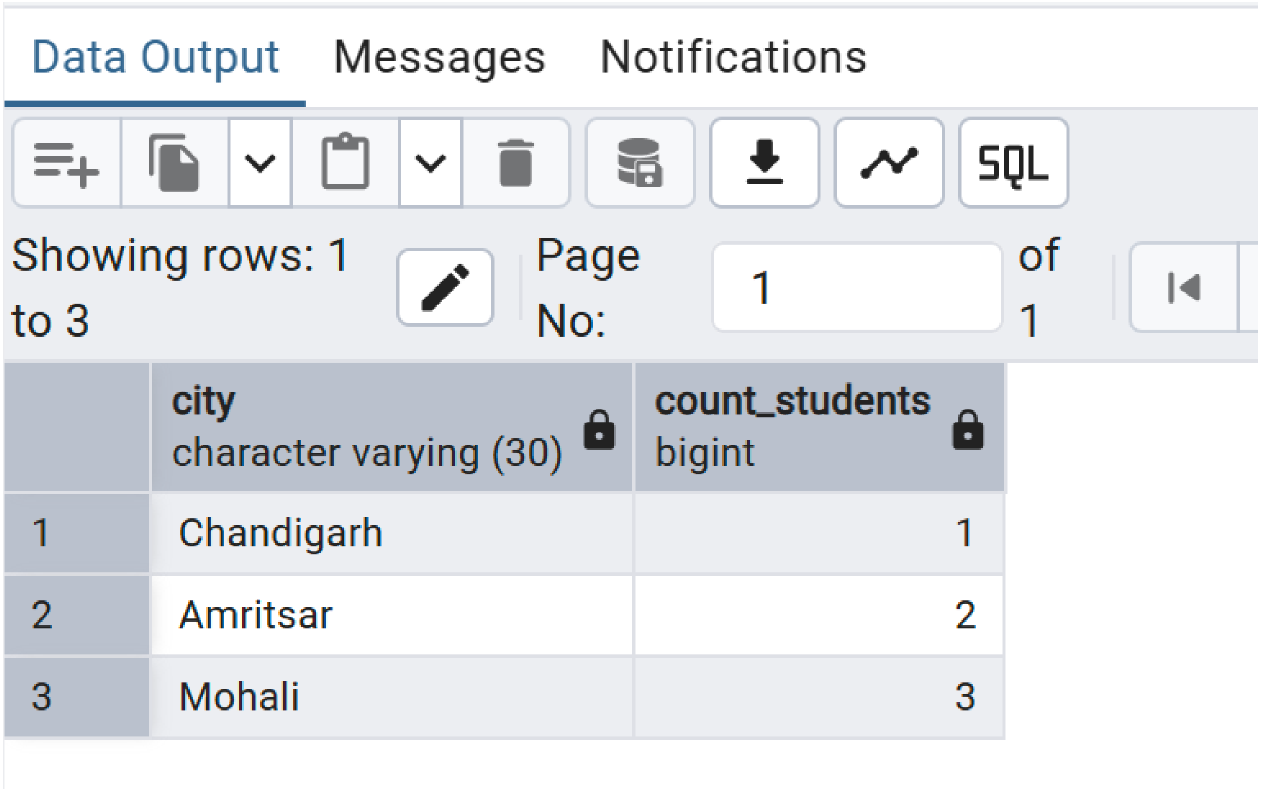
4. Output Analysis

Query 1 – COUNT by City



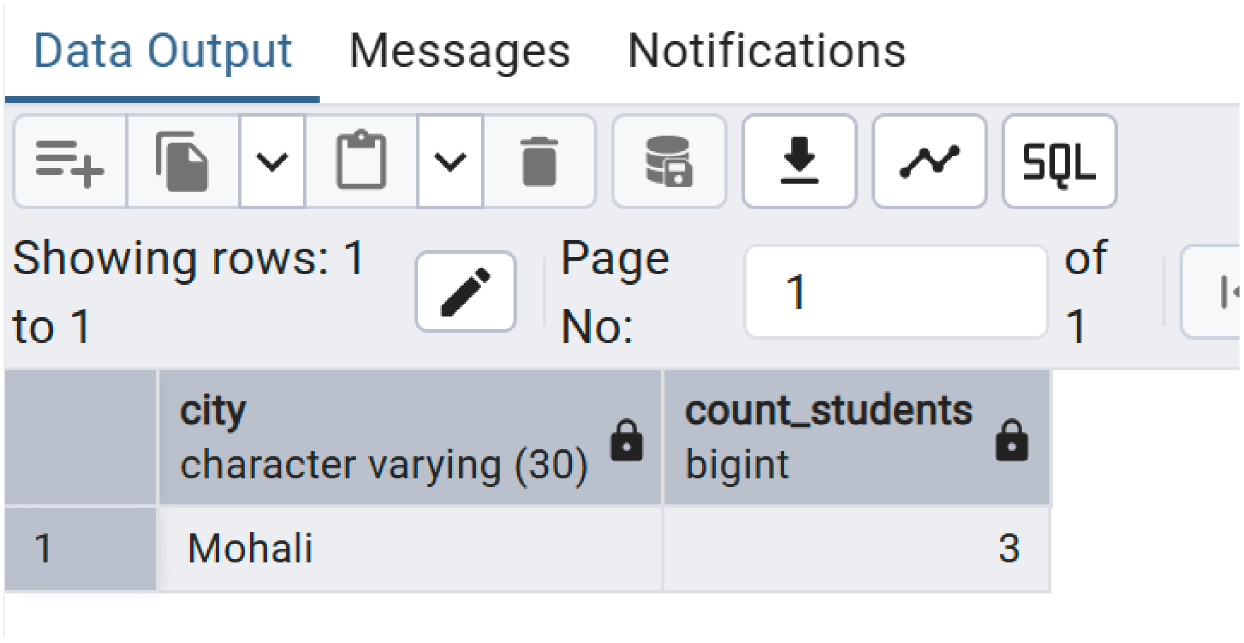
	city character varying (30)	count_students bigint
1	Mohali	3
2	Amritsar	2
3	Chandigarh	1

Query 2 – Sorted Count



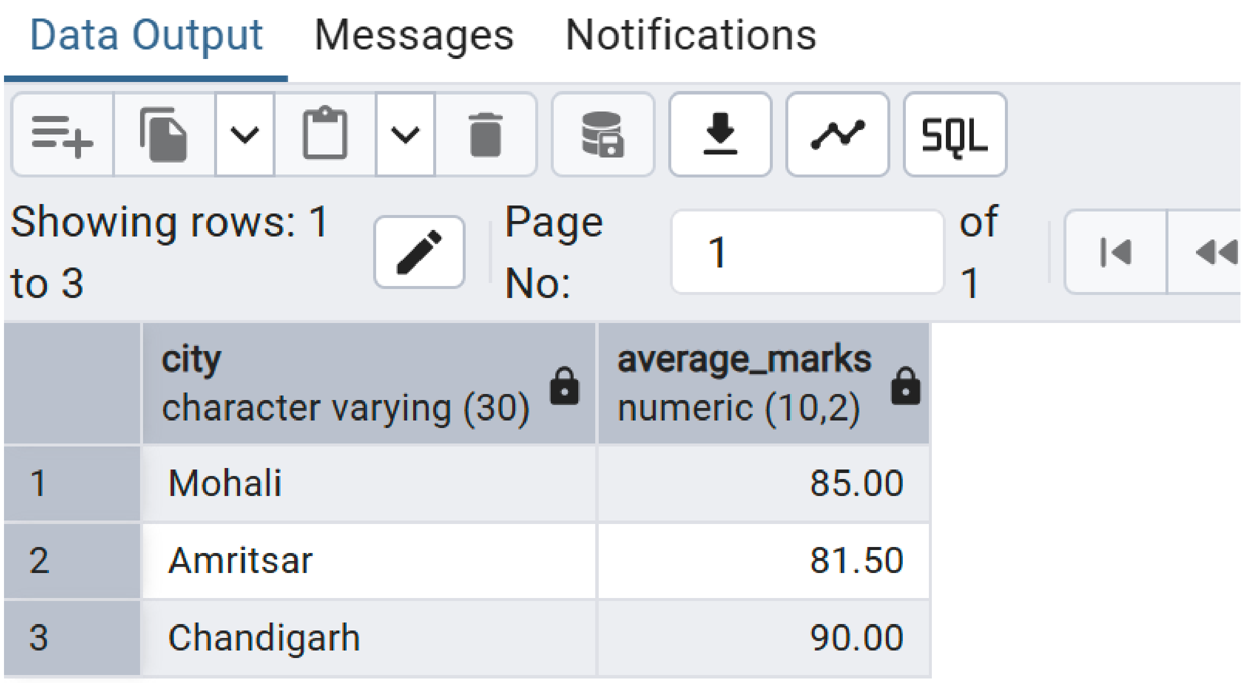
	city character varying (30)	count_students bigint
1	Chandigarh	1
2	Amritsar	2
3	Mohali	3

Query 3 – HAVING Filter



	city character varying (30)	count_students bigint
1	Mohali	3

Query 4 – Average Marks



	city character varying (30)	average_marks numeric (10,2)
1	Mohali	85.00
2	Amritsar	81.50
3	Chandigarh	90.00

---

5. Key Concept Difference

WHERE	**HAVING**
Filters rows before grouping	Filters groups after aggregation
Cannot use aggregate functions	Can use aggregate functions

---

6. Learning Outcomes

Students learned to:

- Organize data using **GROUP BY**
- Perform statistical analysis using **COUNT & AVG**

- Sort grouped data using **ORDER BY**
- Filter grouped results using **HAVING**
- Generate analytical SQL reports