|  |  |  |
| --- | --- | --- |
| **Academic Year:** 2024-25 | **Year:** Second Year | **Semester:** II |
| **PRN No.: 1012412079** | **Name: Ratnajeet Patil** | |
| **Subject:** Database Management System | | |
| **Assignment No.**: 4 |  | |
| **Date:** |  | |

**Lab Assignment: 04**

**Title: Advanced SQL Queries**: Design and execute following SQL queries on suitable applications.

**Joins**: Retrieve data from multiple related tables using INNER JOIN, LEFT JOIN, and RIGHT JOIN.

**Aggregation**: Use aggregate functions like COUNT, AVG, SUM, MIN, MAX in combination with GROUP BY and HAVING clauses.

**Theory:**

**What is JOIN in SQL?**

A JOIN in SQL is used to combine rows from two or more tables based on a related column between them. It allows retrieving data from multiple tables in a single query.

**Different Types of JOINs**

1. INNER JOIN – Returns only the matching rows between the two tables.

1. SELECT A.\*, B.\*

2. FROM TableA A

3. INNER JOIN TableB B

4. ON A.common\_column = B.common\_column;

5.

1. LEFT JOIN (LEFT OUTER JOIN) – Returns all rows from the left table and the matched rows from the right table. If no match is found, NULLs are returned.

1. SELECT A.\*, B.\*

2. FROM TableA A

3. LEFT JOIN TableB B

4. ON A.common\_column = B.common\_column;

5.

1. RIGHT JOIN (RIGHT OUTER JOIN) – Returns all rows from the right table and the matched rows from the left table.

1. SELECT A.\*, B.\*

2. FROM TableA A

3. RIGHT JOIN TableB B

4. ON A.common\_column = B.common\_column;

5.

1. FULL JOIN (FULL OUTER JOIN) – Returns all rows when there is a match in either table. If there’s no match, NULLs are returned.

1. SELECT A.\*, B.\*

2. FROM TableA A

3. FULL JOIN TableB B

4. ON A.common\_column = B.common\_column;

5.

1. CROSS JOIN – Returns the Cartesian product of both tables (every row in TableA is combined with every row in TableB).

1. SELECT A.\*, B.\*

2. FROM TableA A

3. CROSS JOIN TableB B;

4.

1. SELF JOIN – A table joins itself by using aliases.

1. SELECT A.\*, B.\*

2. FROM TableA A

3. INNER JOIN TableA B

4. ON A.column\_name = B.column\_name;

5.

**Aggregate Functions**

Aggregate functions perform calculations on a set of values and return a single value.

| Function | Description |
| --- | --- |
| COUNT() | Returns the number of rows |
| SUM() | Returns the total sum of a column |
| AVG() | Returns the average value of a column |
| MAX() | Returns the highest value in a column |
| MIN() | Returns the lowest value in a column |

**Group By**

The GROUP BY clause is used with aggregate functions to group rows that have the same values in specified columns.

Example:

1. SELECT department, COUNT(\*) AS total\_employees

2. FROM employees

3. GROUP BY department;

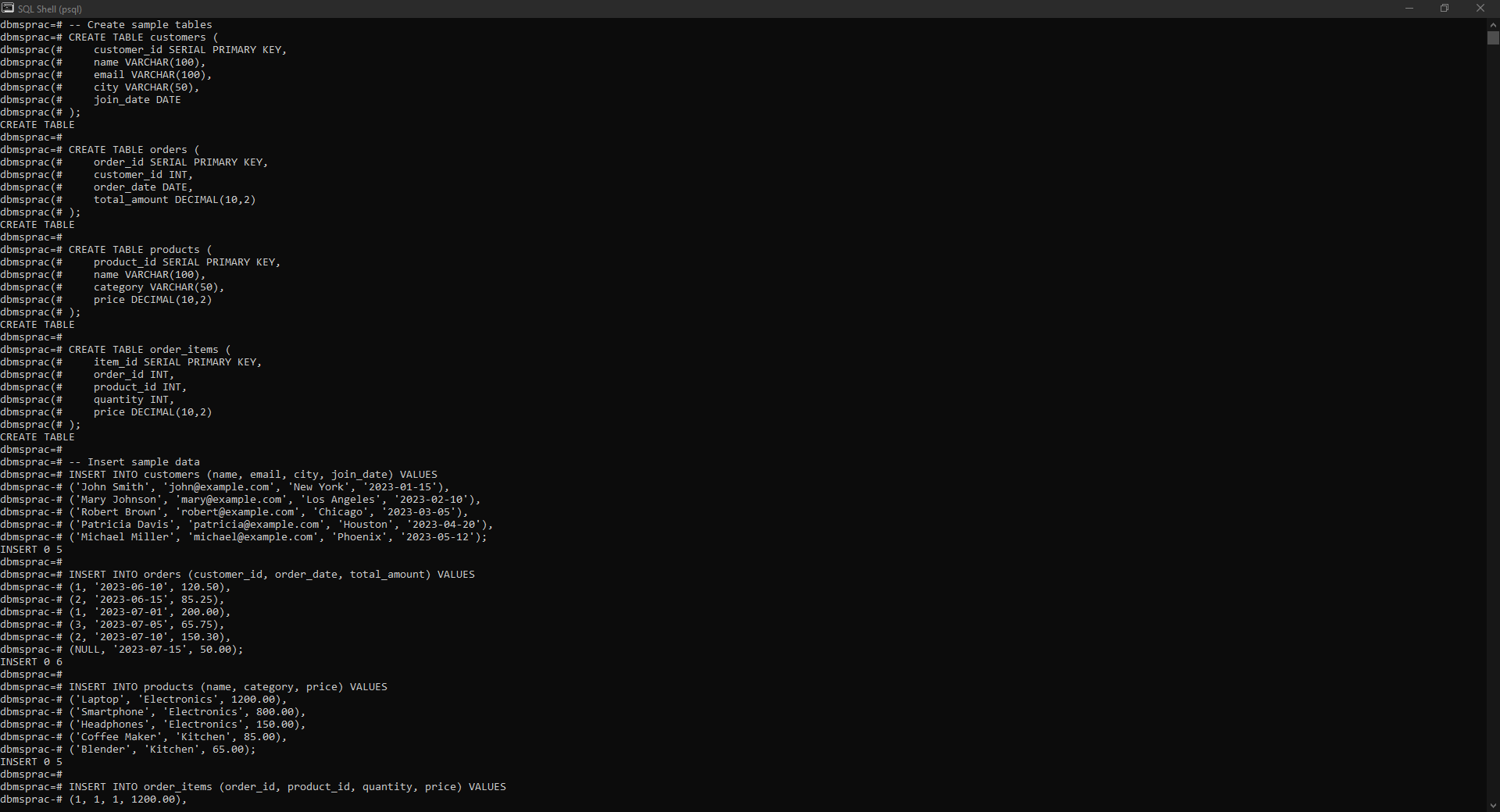
4.

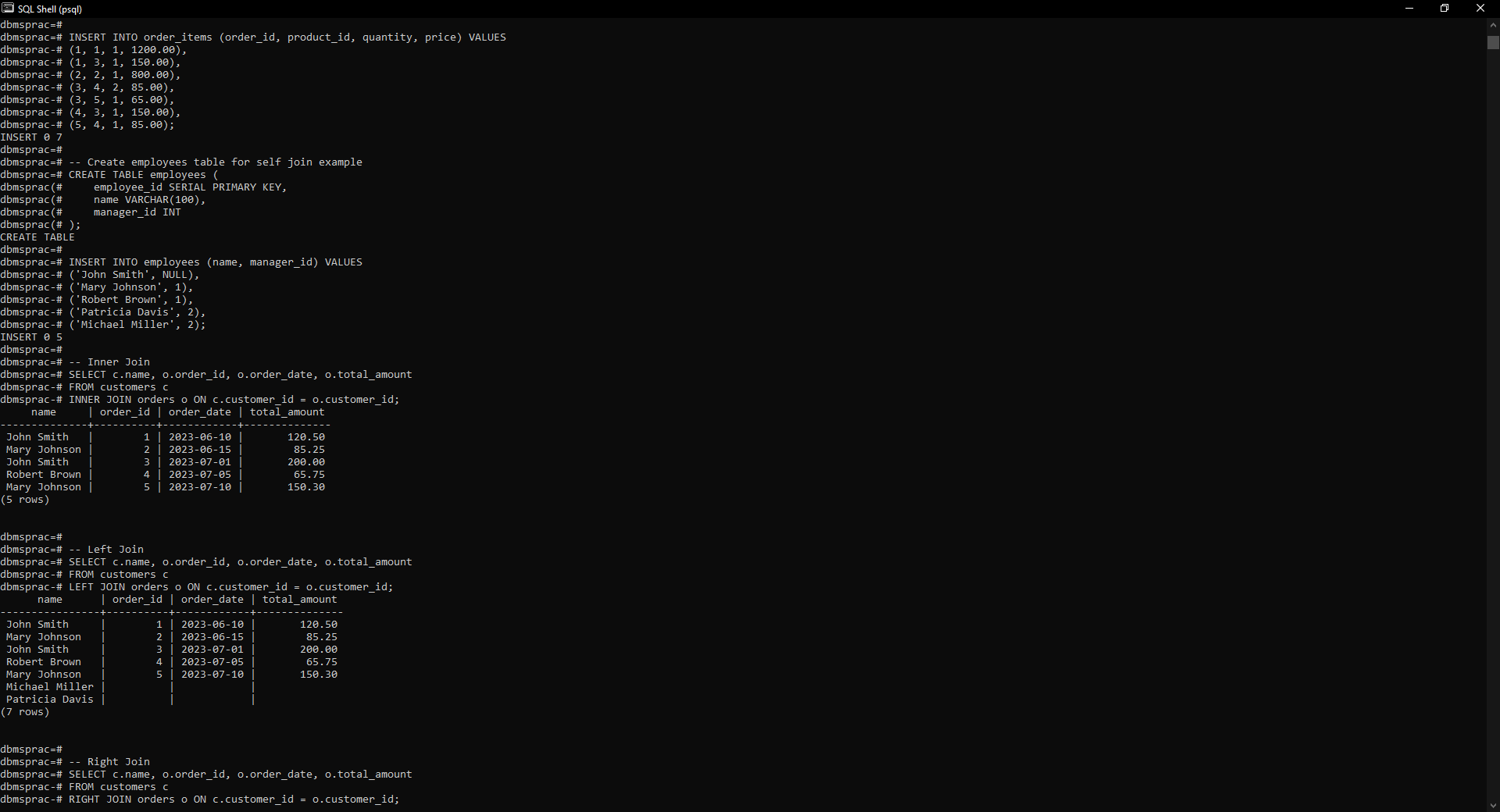
**Having**

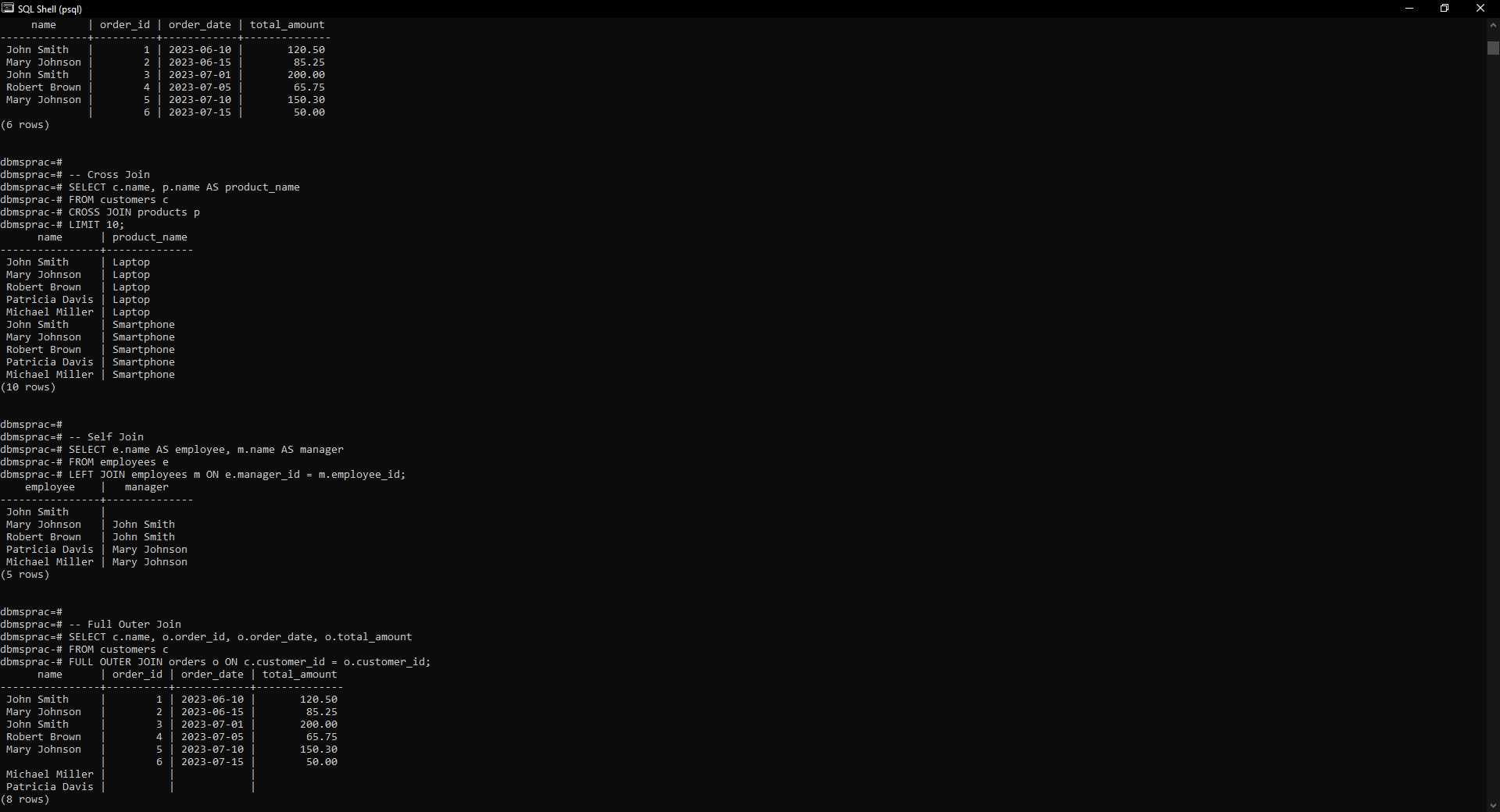
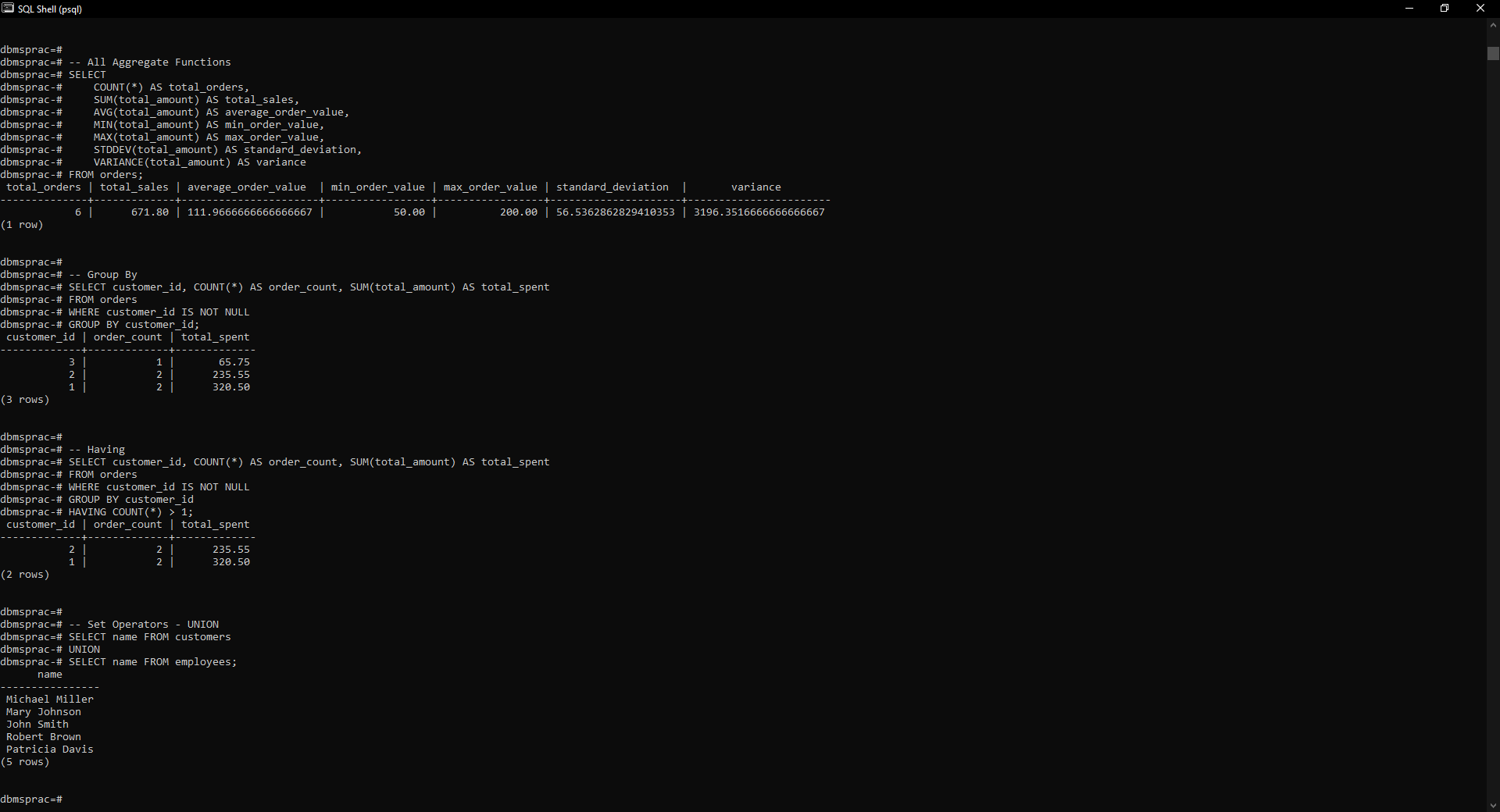
The HAVING clause is used to filter groups created by GROUP BY based on aggregate function results (since WHERE cannot be used with aggregate functions).

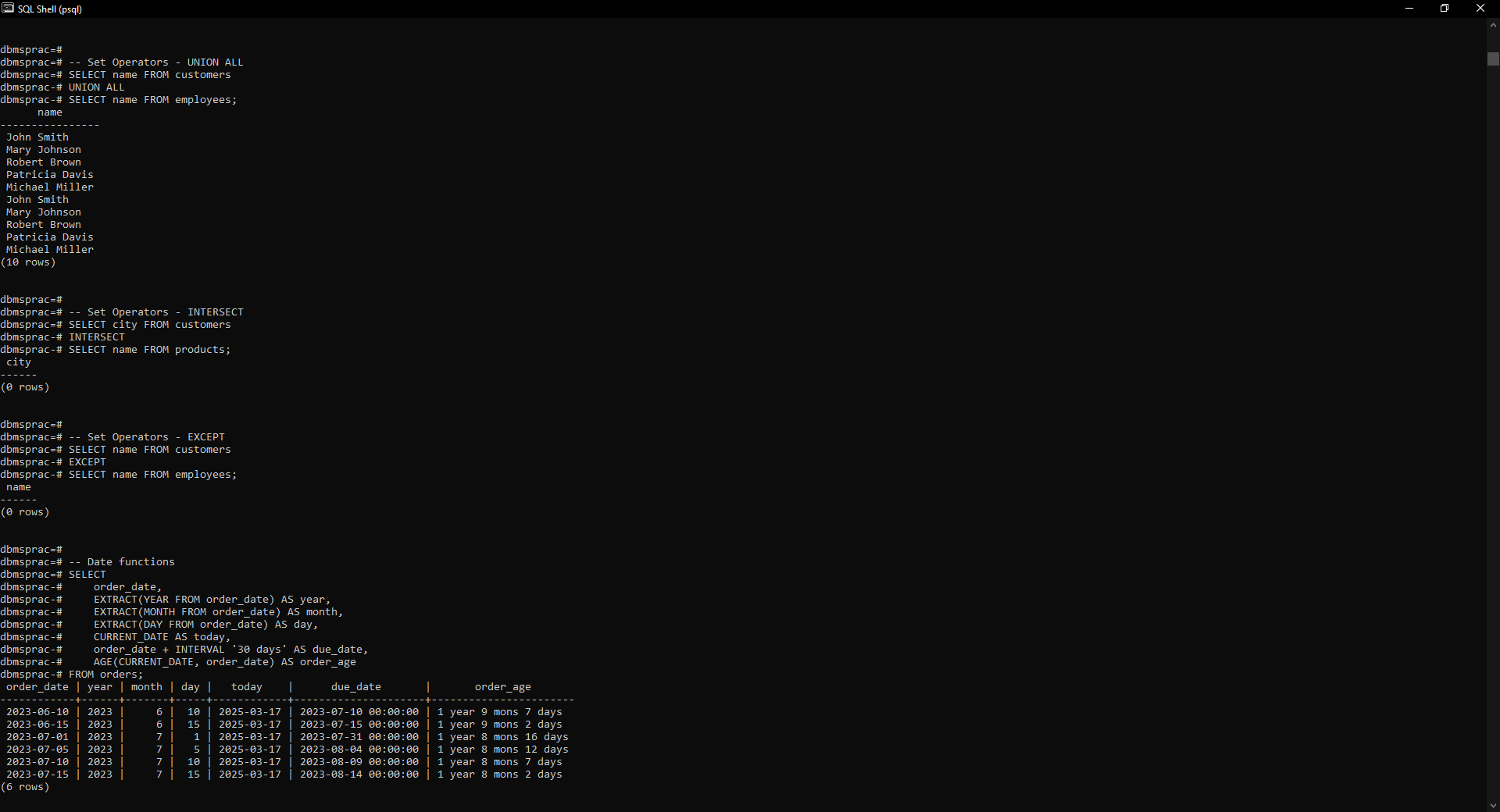
**Show Query Execution Screenshots for:**

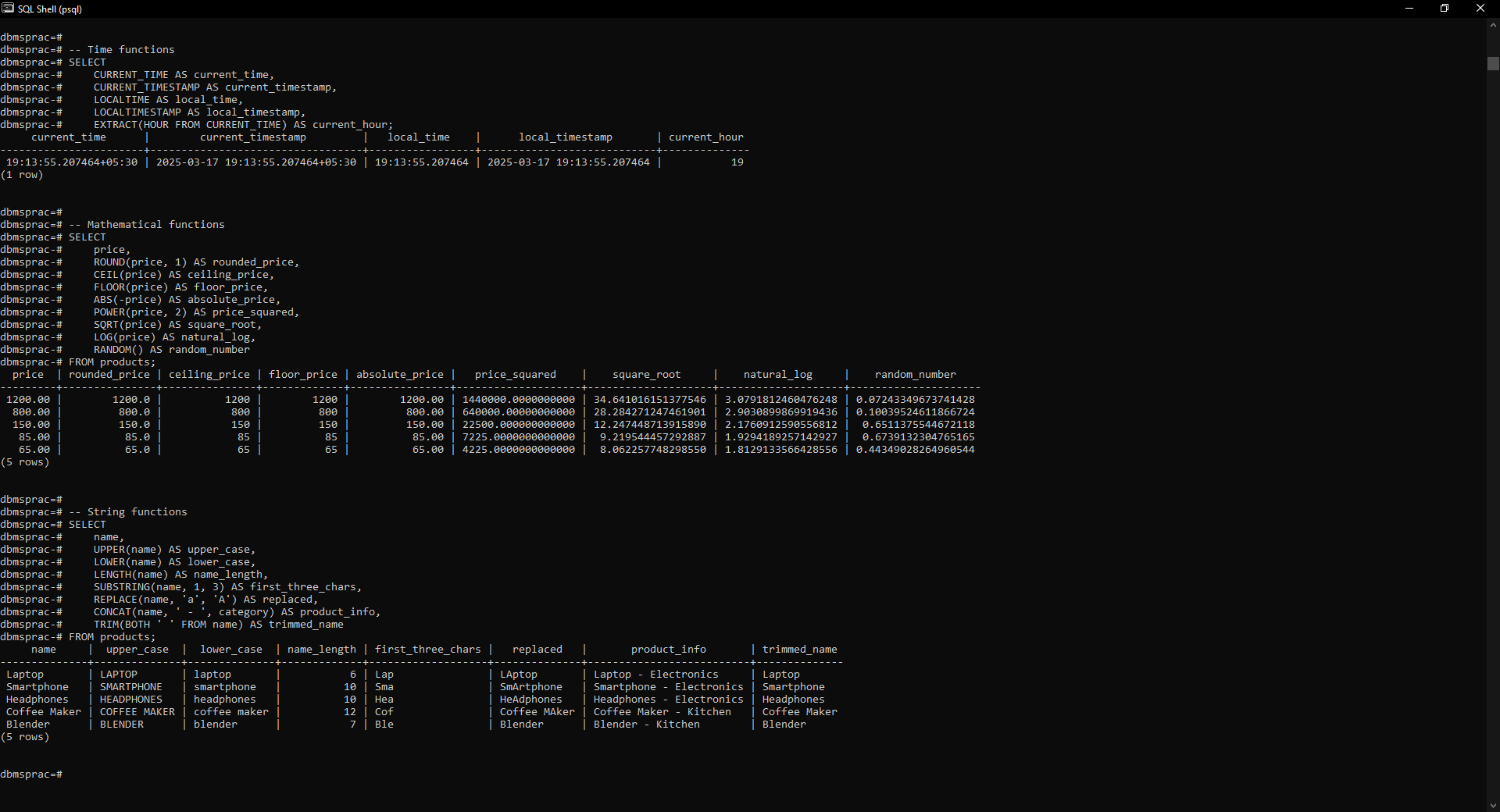
* Inner join
* Left join
* Right join
* Cross join
* Self join
* Full Outer join
* All Aggregate Functions
* Group By
* Having
* Set Operators (Union, Intersect and Difference)
* Use of Date, Time, Mathematical and String Functions









**FAQs:**

1. **Can aggregate functions be used without the GROUP BY clause?**

Yes, aggregate functions can be used without GROUP BY. When used without GROUP BY, they apply to the entire result set and return a single row. For example:

1. SELECT COUNT(\*), AVG(total\_amount), SUM(total\_amount) FROM orders;

**2. What is the difference between COUNT(\*) and COUNT(column\_name)?**

COUNT(\*) counts all rows in the table, including rows with NULL values

COUNT(column\_name) counts only non-NULL values in the specified column For example, if a column has NULL values, COUNT(column\_name) will return a smaller number than COUNT(\*).

**3. Can we use multiple aggregate functions in a single query?** Yes, you can use multiple aggregate functions in a single query. For example:

1. SELECT COUNT(\*) AS total\_orders,

2. SUM(total\_amount) AS total\_sales,

3. AVG(total\_amount) AS average\_order

4. FROM orders;

5.

**4. Can we use WHERE and HAVING together in a query with aggregate functions?** Yes, you can use both WHERE and HAVING together. The key difference is:

WHERE filters rows before they're grouped

HAVING filters groups after they're formed For example:

1. SELECT customer\_id, COUNT(\*) AS order\_count

2. FROM orders

3. WHERE order\_date > '2023-01-01'

4. GROUP BY customer\_id

5. HAVING COUNT(\*) > 5;

6.

**5. What is the difference between INNER JOIN and LEFT JOIN?**

INNER JOIN: Returns only rows that have matching values in both tables

LEFT JOIN: Returns all rows from the left table and matching rows from the right table. If no match is found, NULL values are returned for right table columns

**6. What is the difference between ON and USING in joins?**

ON: Allows you to specify any join condition (with any columns)

1. -- Using ON

2. SELECT \* FROM orders o JOIN customers c ON o.customer\_id = c.customer\_id;

3.

4.

USING: Simplifies the syntax when joining tables on columns with the same name For example

-- 1. Using USING (works only when column names are identical)

2. SELECT \* FROM orders JOIN customers USING (customer\_id);

3.

**7. Can we use aggregate functions with joins?**

Yes, you can use aggregate functions with joins. This is often used to aggregate data from related tables. For example:

1. SELECT c.name, COUNT(o.order\_id) AS order\_count, SUM(o.total\_amount) AS total\_spent

2. FROM customers c

3. LEFT JOIN orders o ON c.customer\_id = o.customer\_id

4. GROUP BY c.name;

5.

This query joins customers and orders tables, then aggregates the order data for each customer.