TASK-4

Aggregate Functions: perform calculations on multiple rows and return a single summary value. Common functions include **COUNT()** for counting rows, **SUM()** for totals, **AVG()** for average values, **MIN()** and **MAX()** for minimum and maximum values respectively. They are crucial in generating insights like total sales, number of users, or average order values. These functions are typically used with SELECT statements to analyze datasets efficiently. For example, *SELECT SUM(total_amount) FROM orders* calculates total revenue. They ignore NULLs by default, except COUNT(*), which includes them.

Query Query History

```
1  SELECT COUNT(*) AS total_customers FROM customers;
```

Data Output Messages Notifications

	total_customers bigint
1	4

Query Query History

```
1  SELECT MIN(price) AS min_price,
2  MAX(price) AS max_price FROM products;
```

Data Output Messages Notifications

	min_price numeric	max_price numeric
1	19.99	967.99

Query Query History

```
1  SELECT AVG(total_amount) AS avg_order_value FROM orders;
```

Data Output Messages Notifications

	avg_order_value numeric
1	244.9825000000000000

Query Query History

```
1  SELECT SUM(total_amount) AS total_revenue FROM orders;
```

Data Output Messages Notifications

	total_revenue numeric
1	979.93

Query Query History

```
1  SELECT c.category_name, ROUND(AVG(p.price), 2)
2  AS avg_price FROM products p JOIN categories c
3  ON p.category_id = c.category_id
4  GROUP BY c.category_name;
```

Data Output Messages Notifications

	category_name character varying (50)	avg_price numeric
1	Home & Kitchen	59.99
2	Electronics	356.94
3	Clothing	29.99
4	Books	24.99

Query Query History

```
1  SELECT c.category_name,
2  STRING_AGG(p.product_name, ', ')
3  AS products FROM products p
4  JOIN categories c
5  ON p.category_id = c.category_id
6  GROUP BY c.category_name;
```

Data Output Messages Notifications

	category_name character varying (50)	products text
1	Home & Kitchen	Blender
2	Electronics	Smartphone X, Wireless Earbuds, Smart Watch, Wireless Charger
3	Clothing	Men's T-Shirt, Women's Jeans
4	Books	Cookbook

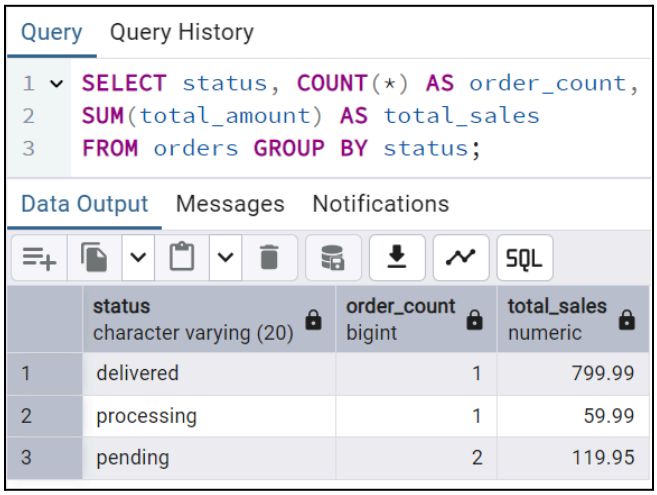
Query Query History

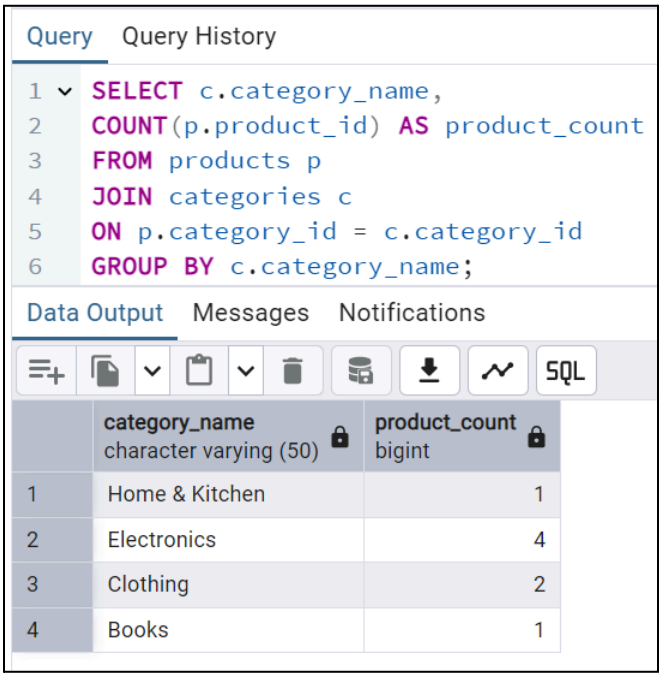
```
1  SELECT COUNT(DISTINCT city)
2  AS unique_cities FROM customers;
```

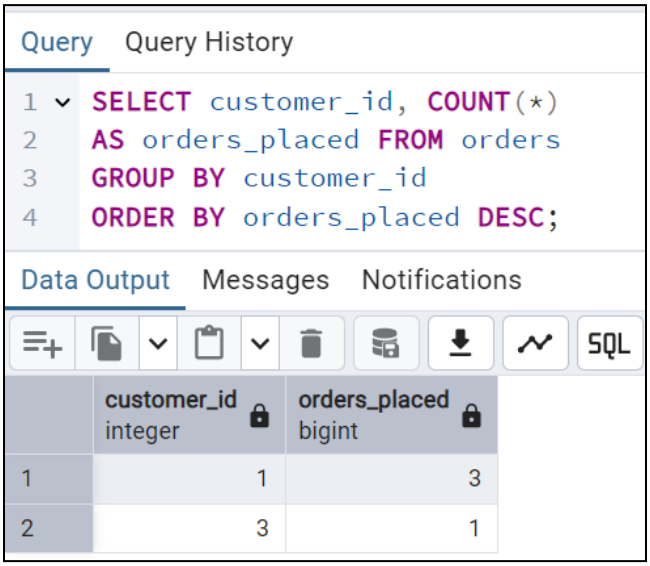
Data Output Messages Notifications

	unique_cities bigint
1	4

GROUP BY Clause: is used to group rows that share a common value into summary rows. It works hand-in-hand with aggregate functions to produce grouped results, such as total sales per region or number of products per category. For example, *SELECT category, COUNT(*) FROM products GROUP BY category* gives the product count in each category. Without GROUP BY, aggregate functions treat the entire table as one group. When grouping, only aggregated values or group-by columns can appear in the SELECT list. It allows multi-level grouping by including multiple columns in the clause.







HAVING Clause: While WHERE filters rows before grouping, HAVING filters groups after aggregation. It is used in combination with GROUP BY to restrict which groups are returned based on aggregate values. For instance, *HAVING COUNT(*) > 2* can exclude categories with less than three products. You can apply conditions like *HAVING SUM(sales) > 1000* to return only high-performing groups. Unlike WHERE, HAVING can reference aggregate functions, making it essential for post-group filtering logic in analytical queries.