E-Commerce Database Analysis using PostgreSQL

Internship Report  
By: RAGAVARSHINI ALAGARSAMY  
Company: ELEVATE LABS  
Date: 08th AUG 2025

# 1. Introduction

This report documents the analysis of an e-commerce database using PostgreSQL. The database contains tables such as customers, categories, products, orders, and order\_items. The objective of this project is to execute SQL queries for retrieving, analyzing, and optimizing business data.

# 2. Tools Used

- PostgreSQL 17

- pgAdmin 4

- Dataset provided by the company

# 3. Dataset Preparation

Before importing the dataset, existing data was truncated to avoid primary key conflicts. The CSV files were then imported using the following commands:

**TRUNCATE TABLE table\_name RESTART IDENTITY CASCADE;  
\copy table\_name FROM 'file\_path.csv' DELIMITER ',' CSV HEADER;**

**Data was verified using:  
SELECT \* FROM table\_name LIMIT 5;**

# 4. Queries and Outputs

Each query is presented with its SQL code and the corresponding output screenshot.

## Query 1

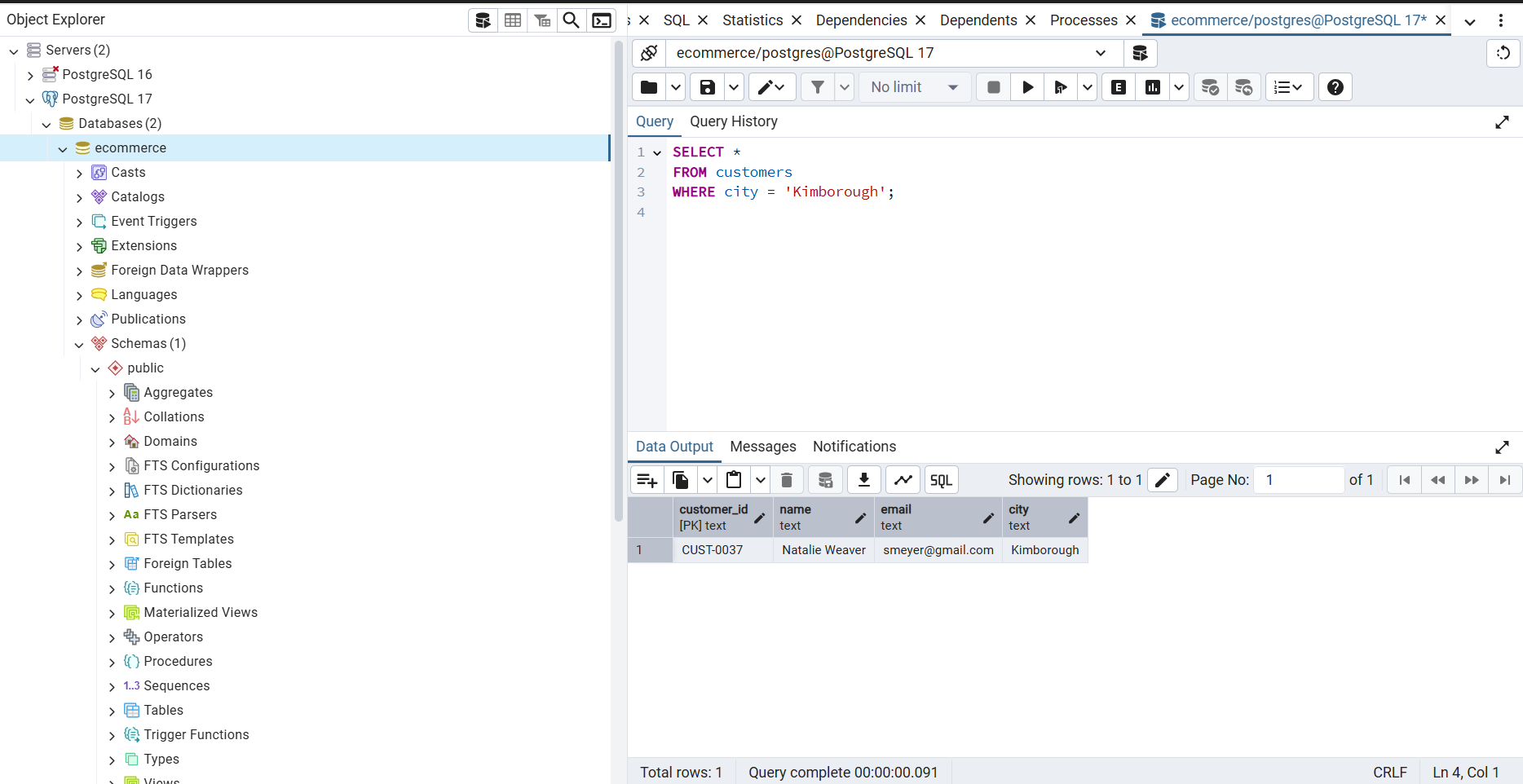
1. List all customers from a specific city

SELECT \*

FROM customers

WHERE city = 'Kimborough';

OUTPUT:



## Query 2

1. Top 10 most expensive products

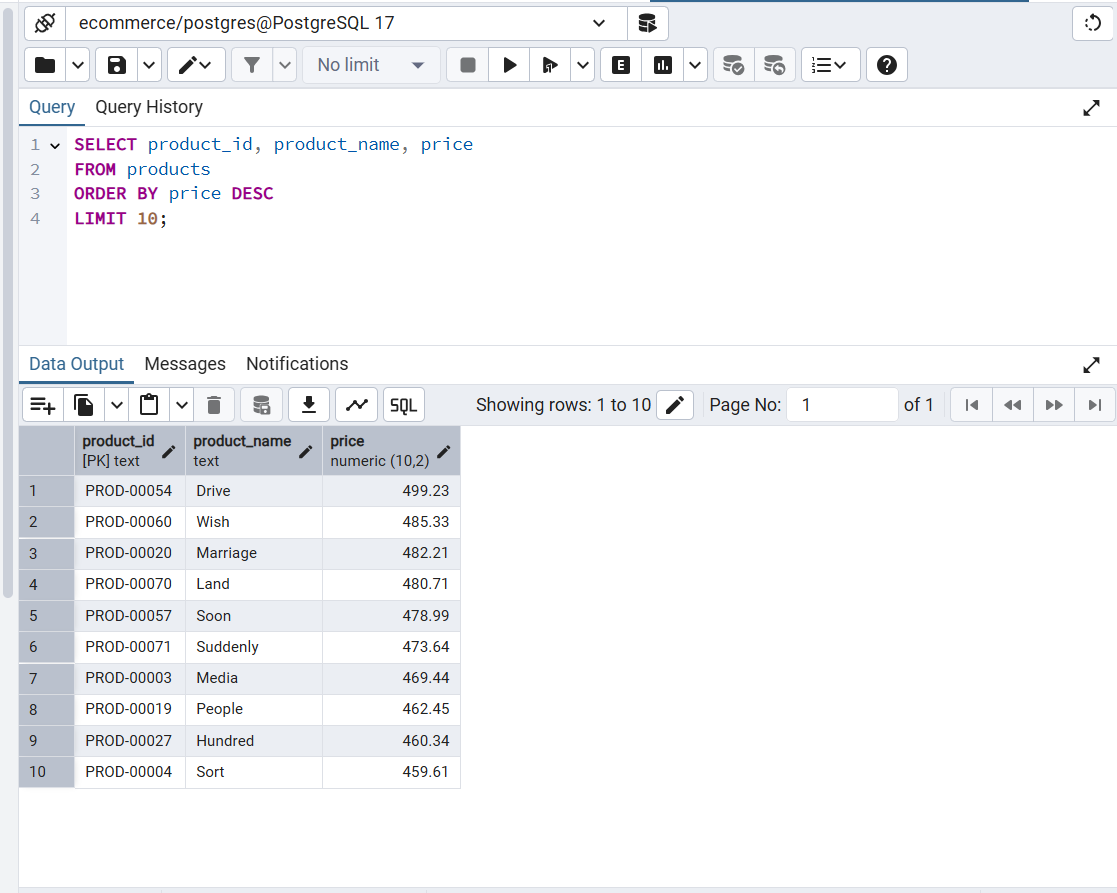
SELECT product\_id, product\_name, price

FROM products

ORDER BY price DESC

LIMIT ;

OUTPUT:



## Query 3

1. Total revenue per category

SELECT c.category\_id, c.category\_name, SUM(oi.price \* oi.quantity) AS total\_revenue

FROM order\_items oi

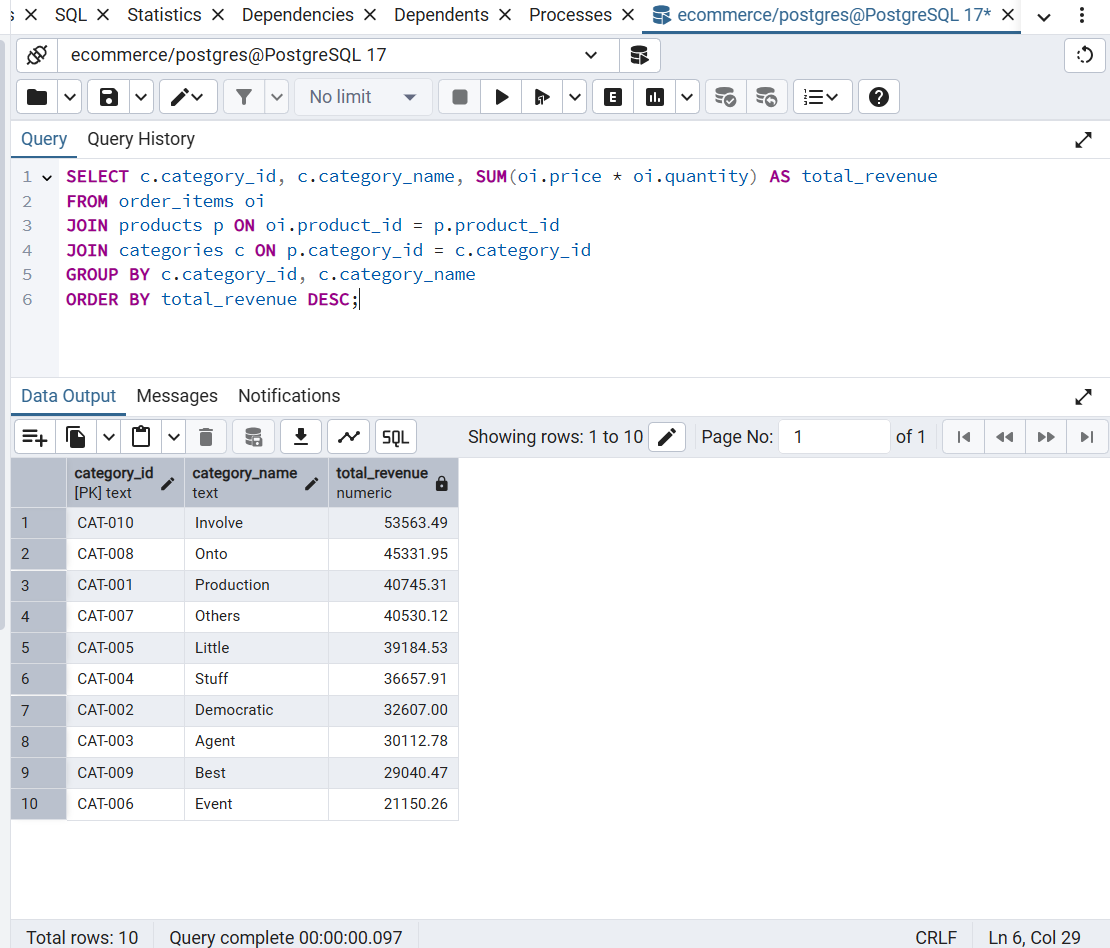
JOIN products p ON oi.product\_id = p.product\_id

JOIN categories c ON p.category\_id = c.category\_id

GROUP BY c.category\_id, c.category\_name

ORDER BY total\_revenue DESC;

OUTPUT:



## Query 4

1. Customers with more than 5 orders

SELECT customer\_id, COUNT(order\_id) AS total\_orders

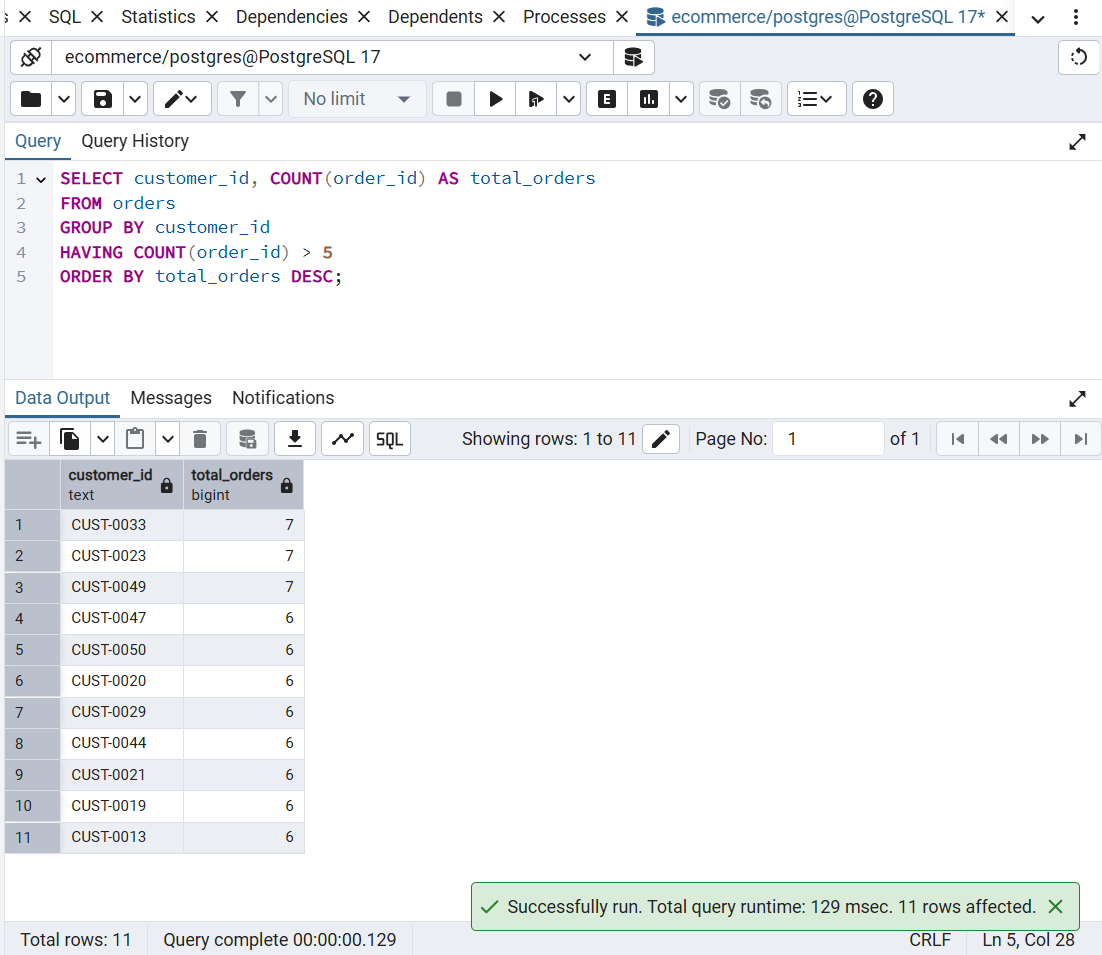
FROM orders

GROUP BY customer\_id

HAVING COUNT(order\_id) > 5

ORDER BY total\_orders DESC;

OUTPUT:



## Query 5

1. Order details (INNER JOIN example)

SELECT o.order\_id, o.order\_date, c.name AS customer\_name, p.product\_name, oi.quantity, oi.price

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

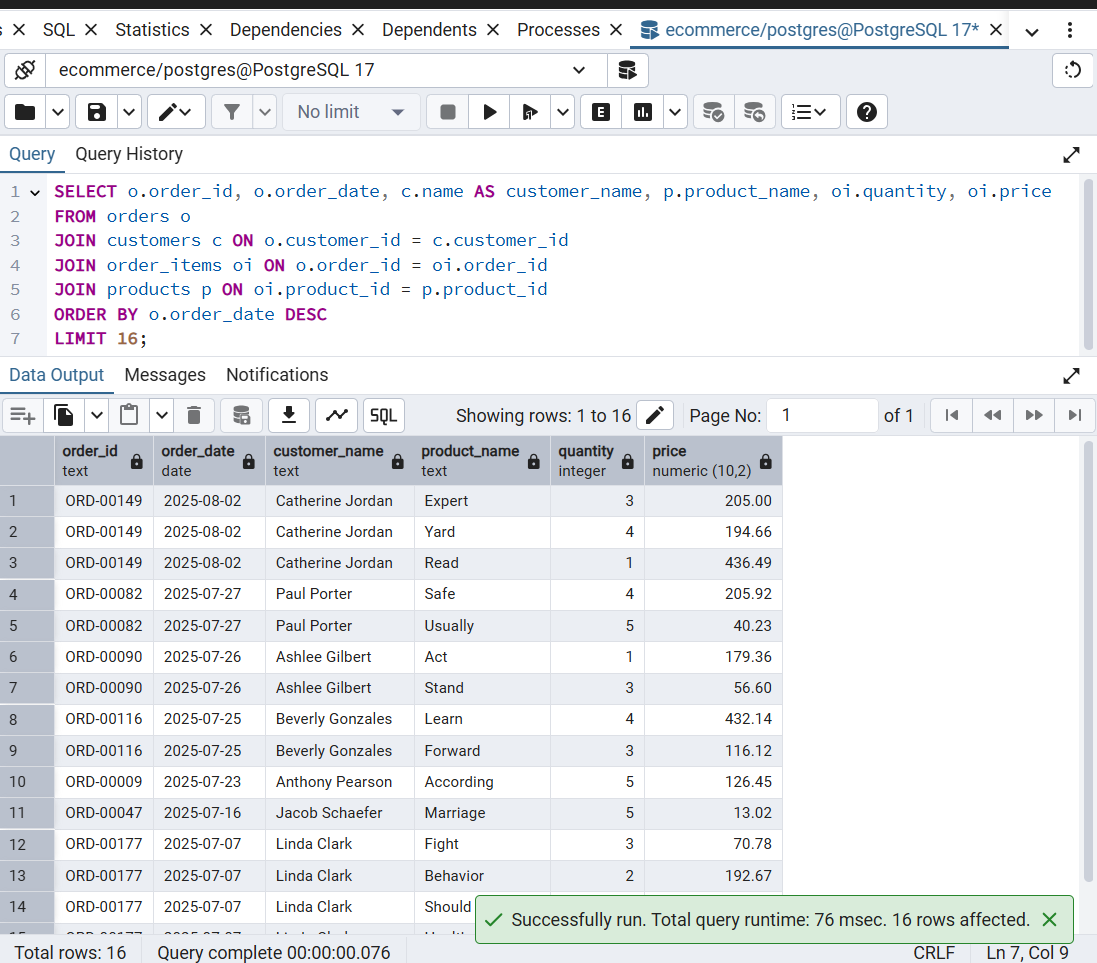
JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products p ON oi.product\_id = p.product\_id

ORDER BY o.order\_date DESC

LIMIT 20;

OUTPUT:



## Query 6

1. All customers and their orders (LEFT JOIN example)

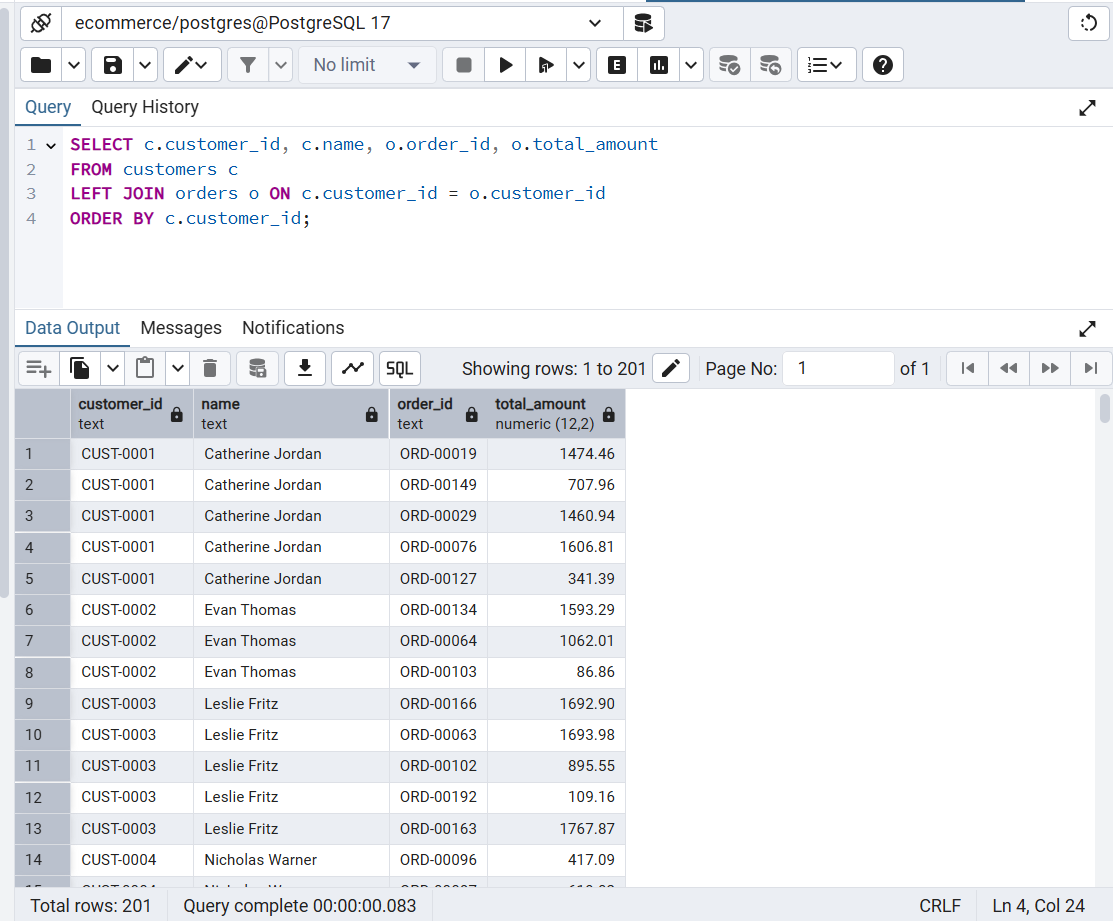
SELECT c.customer\_id, c.name, o.order\_id, o.total\_amount

FROM customers c

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

ORDER BY c.customer\_id;

OUTPUT:



## Query 7

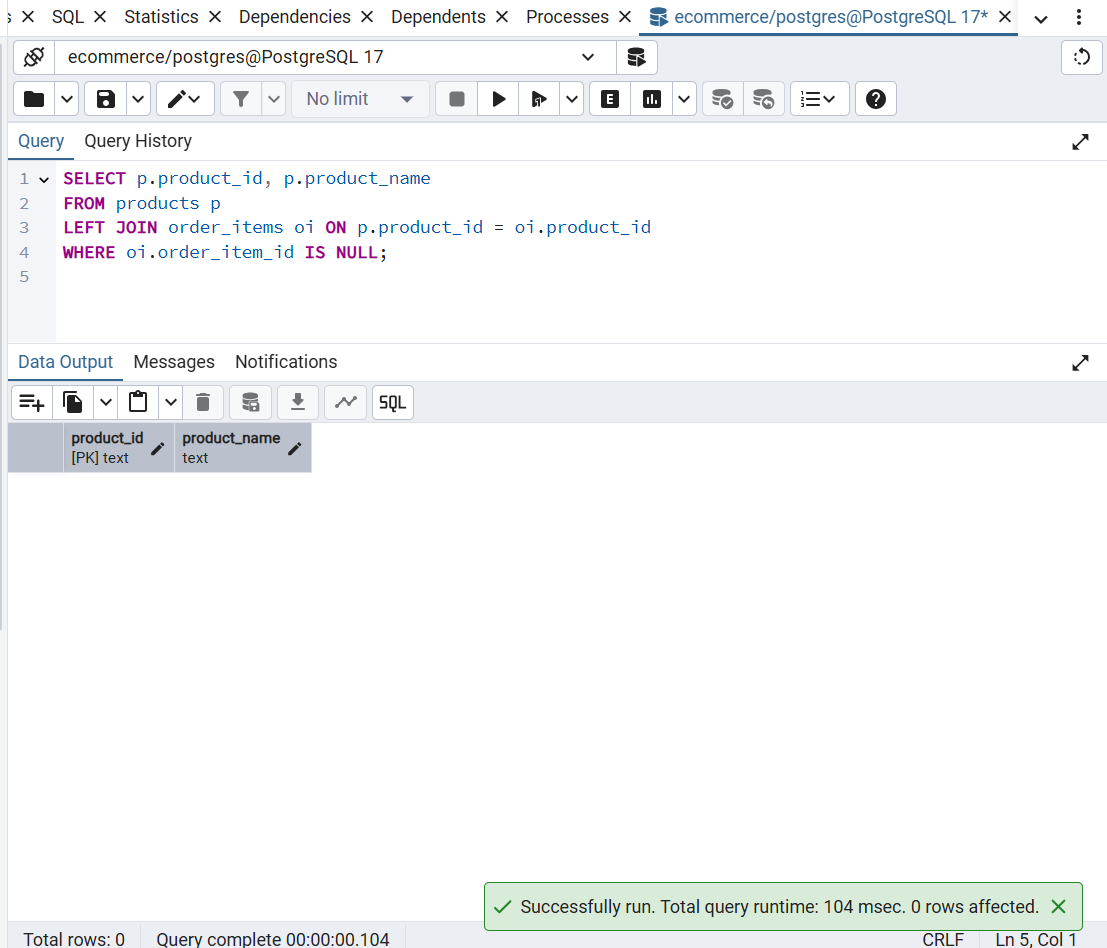
1. Products that have never been ordered (RIGHT JOIN example)

SELECT p.product\_id, p.product\_name

FROM products p

LEFT JOIN order\_items oi ON p.product\_id = oi.product\_id

WHERE oi.order\_item\_id IS NULL;

OUTPUT:

## Query 8

1. Subquery – Top 5 customers by total spend

SELECT customer\_id, total\_spent

FROM (

SELECT customer\_id, SUM(total\_amount) AS total\_spent

FROM orders

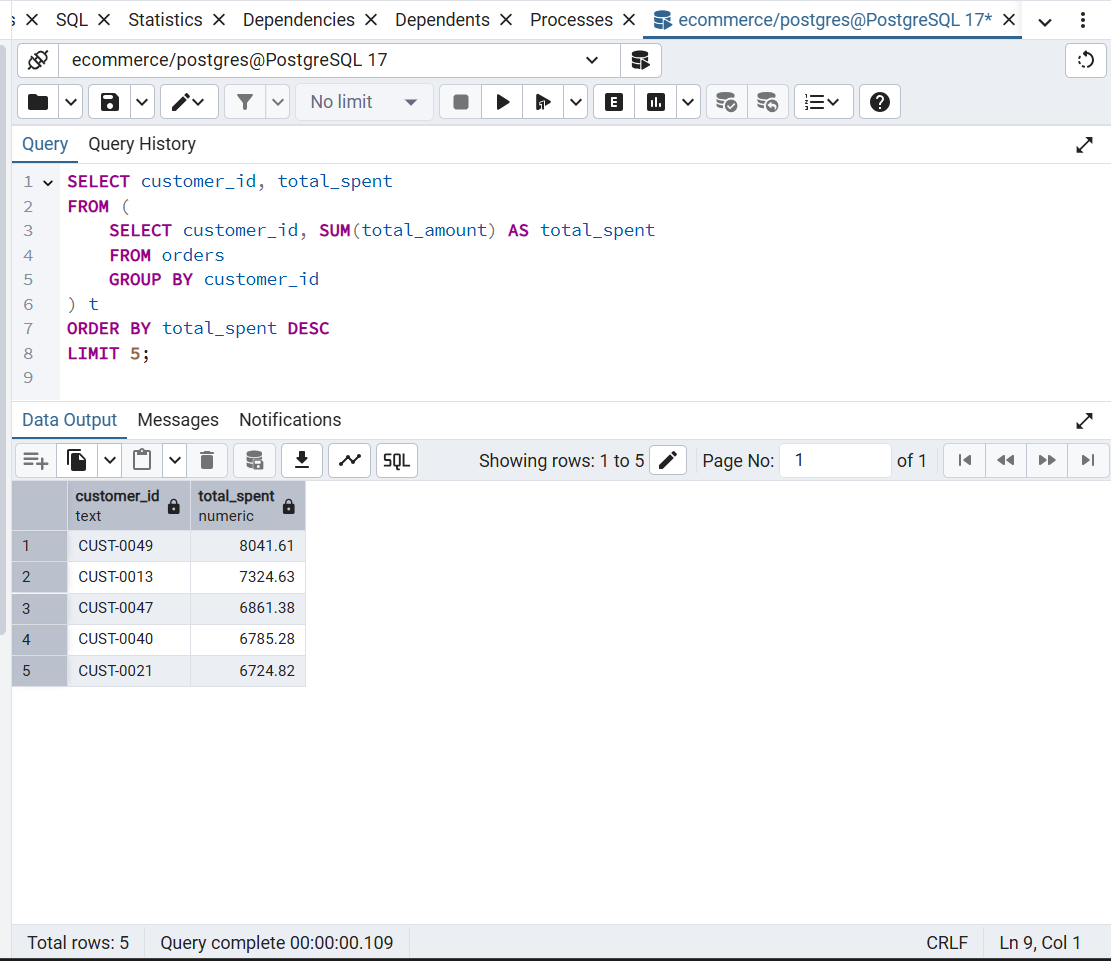
GROUP BY customer\_id

) t

ORDER BY total\_spent DESC

LIMIT 5;

OUTPUT:



## Query 9

1. Create a view for monthly revenue

CREATE OR REPLACE VIEW monthly\_revenue AS

SELECT DATE\_TRUNC('month', order\_date) AS month, SUM(total\_amount) AS revenue

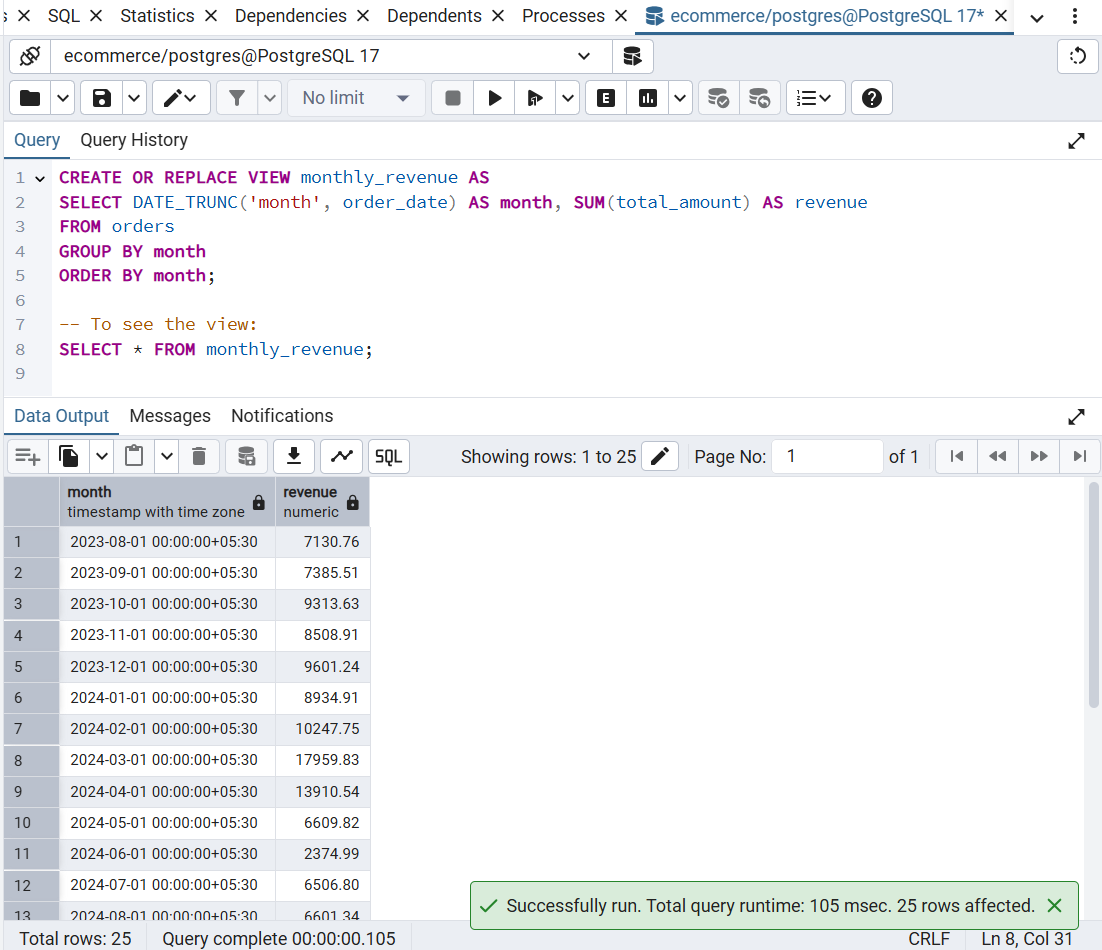
FROM orders

GROUP BY month

ORDER BY month;

-- To see the view:

SELECT \* FROM monthly\_revenue;

OUTPUT:

## Query 10

1. Index + EXPLAIN ANALYZE

-- Create index on orders table

CREATE INDEX idx\_orders\_customer\_date ON orders(customer\_id, order\_date);

-- Analyze query performance

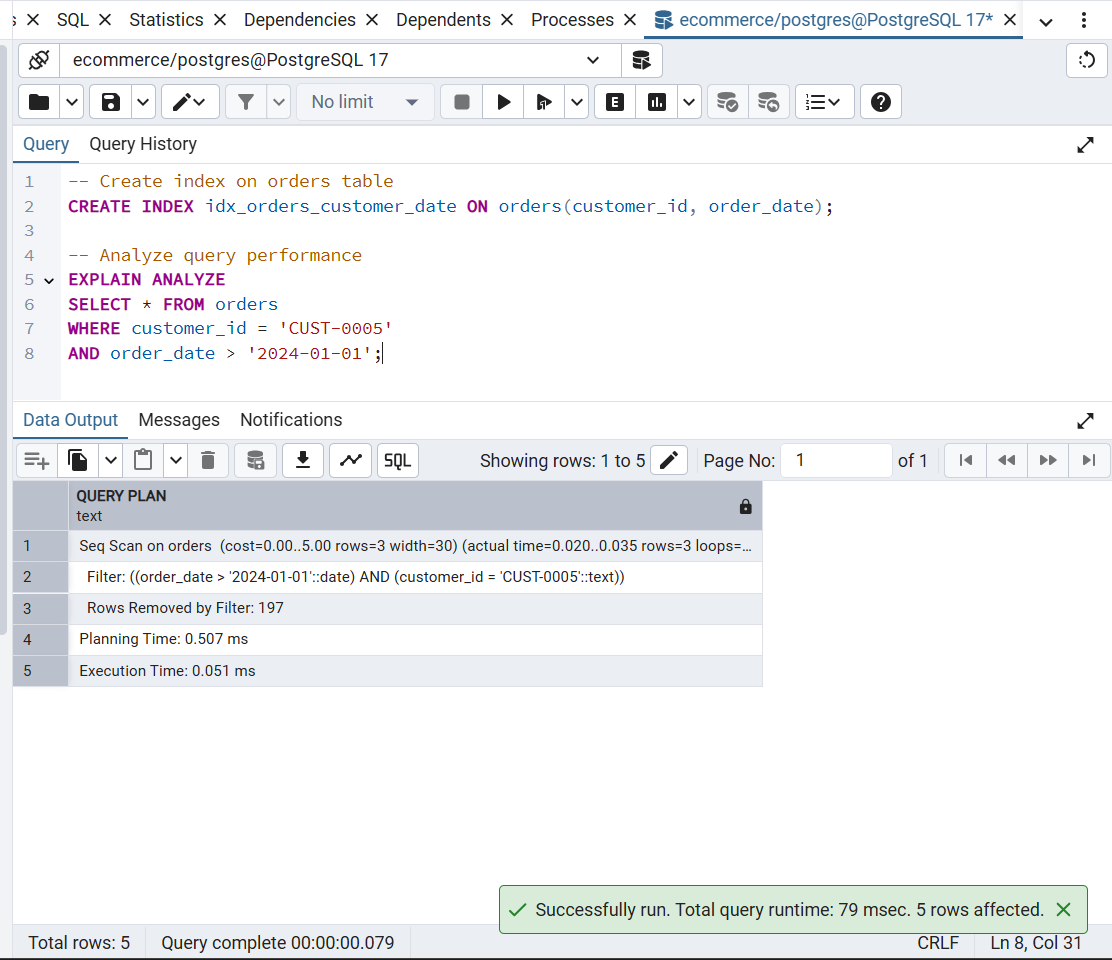
EXPLAIN ANALYZE

SELECT \* FROM orders

WHERE customer\_id = 'CUST-0005'

AND order\_date > '2024-01-01';

OUTPUT:



## Query 11

1. Top 5 Customers by Lifetime Value

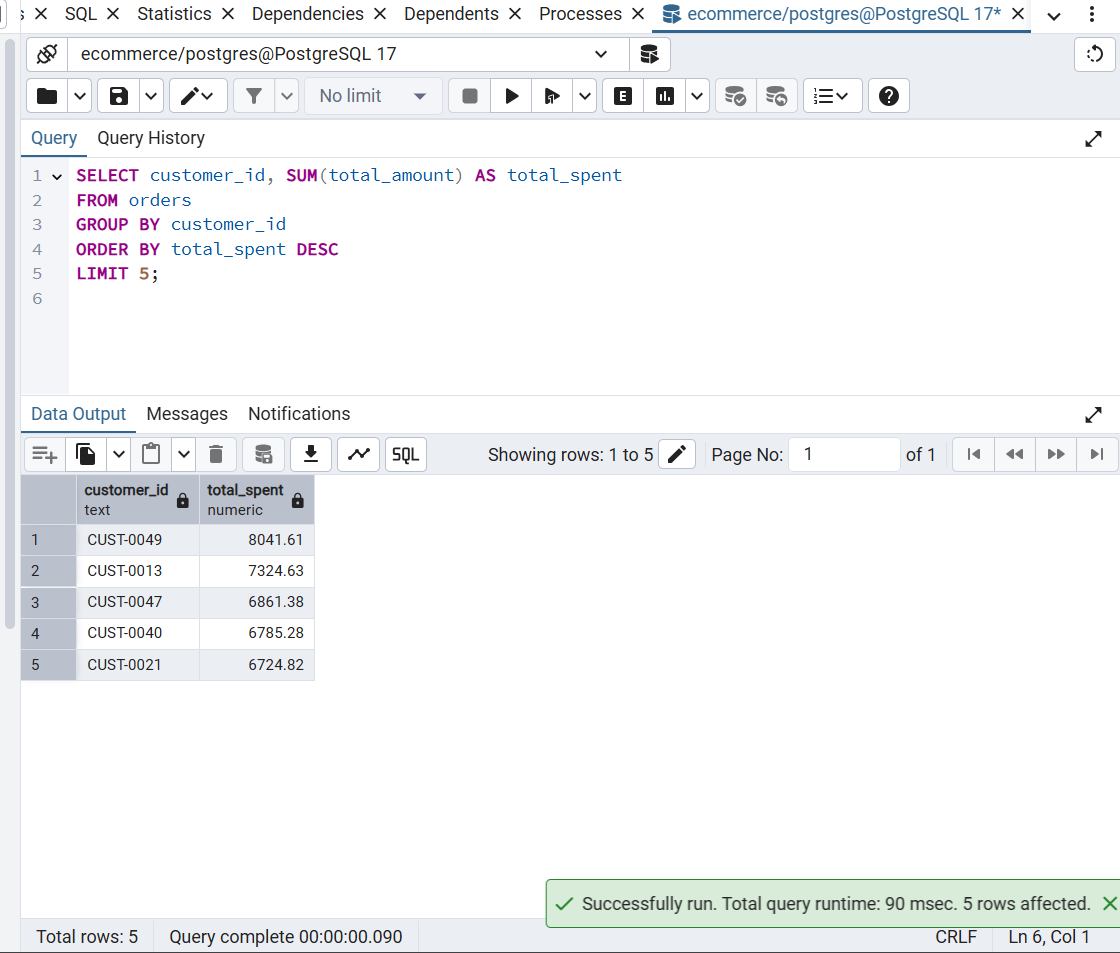
SELECT customer\_id, SUM(total\_amount) AS total\_spent

FROM orders

GROUP BY customer\_id

ORDER BY total\_spent DESC

LIMIT 5;

OUTPUT:

## Query 12

1. **Monthly Revenue Trend** (shows business growth)

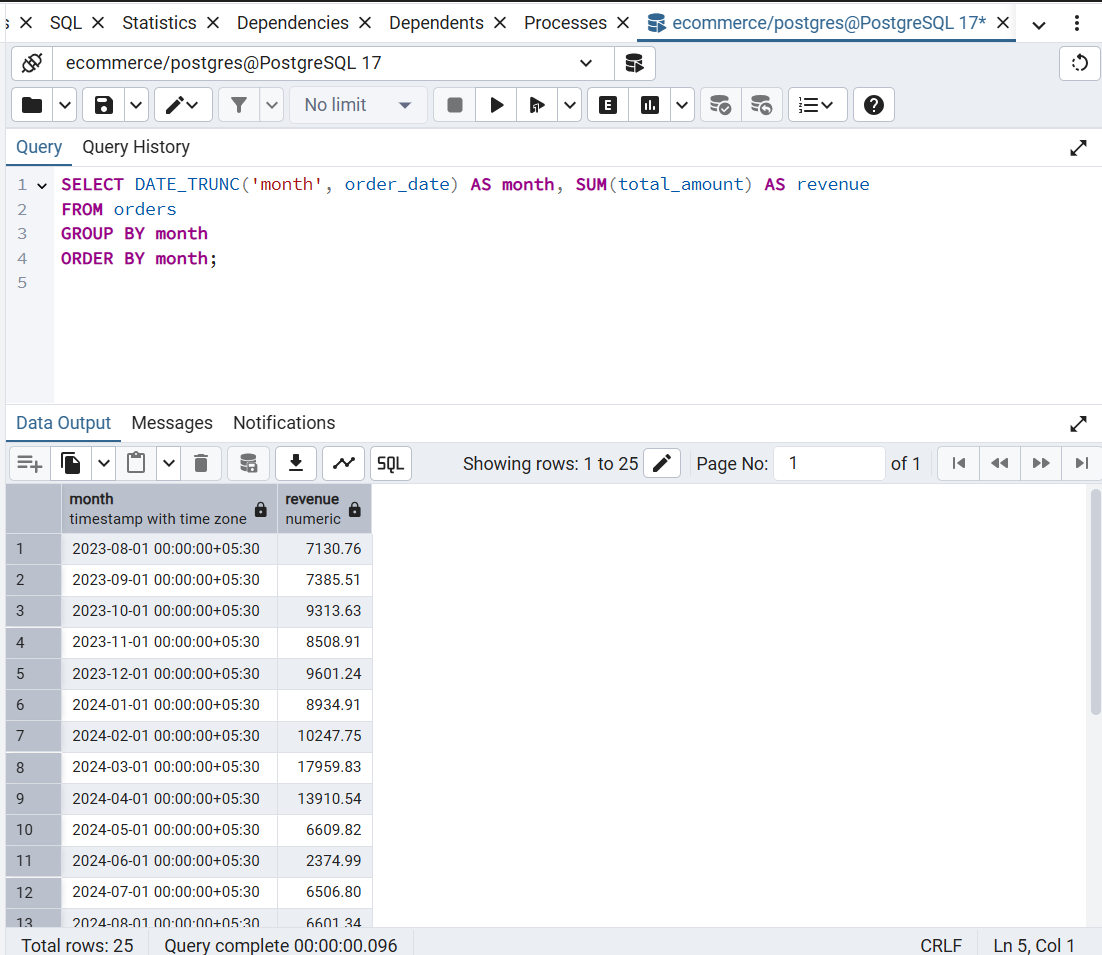
SELECT DATE\_TRUNC('month', order\_date) AS month, SUM(total\_amount) AS revenue

FROM orders

GROUP BY month

ORDER BY month;

OUTPUT:



## Query 13

13 Most Profitable Category

SELECT c.category\_name, SUM(oi.price \* oi.quantity) AS profit

FROM order\_items oi

JOIN products p ON oi.product\_id = p.product\_id

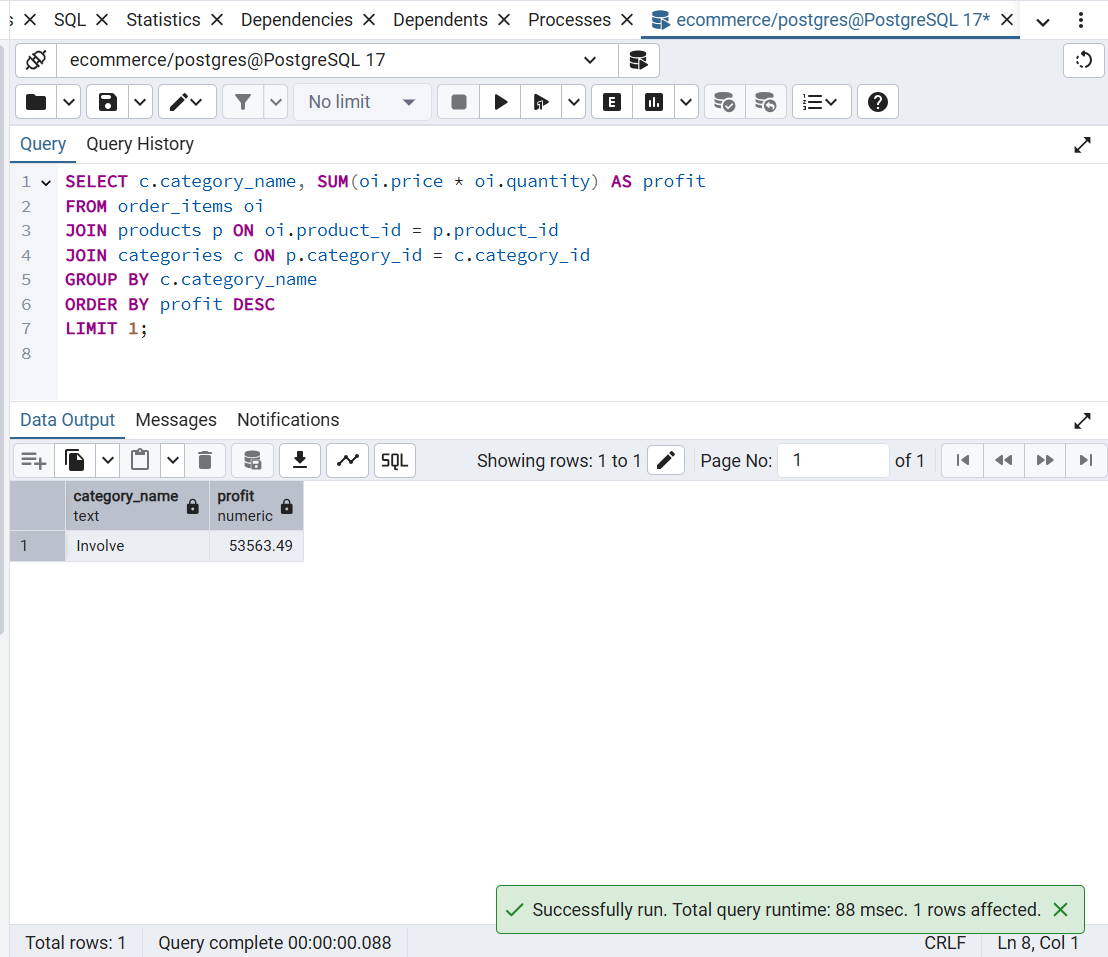
JOIN categories c ON p.category\_id = c.category\_id

GROUP BY c.category\_name

ORDER BY profit DESC

LIMIT 1;

OUTPUT:



# 5. Observations

From the analysis, we observed trends in customer purchasing patterns, monthly revenues, and top spenders. For example, certain months such as March and April showed higher revenues, and a few customers contributed to a large percentage of sales.

# 6. Conclusion

Through this project, I gained hands-on experience in SQL queries, joins, views, indexing, and data analysis in PostgreSQL. These skills are essential for real-world database management and business intelligence.