QuickKart

QuickKart is a fictional e-commerce company created to demonstrate sales, customer, and operational analytics using SQL.

Prepared by: Deepak
Tools: MySQL, MySQL Workbench

Project Overview

QuickKart is a fictional e-commerce company created to understand how online shopping data can be analyzed using SQL.

In this project, real-life e-commerce data such as customers, products, orders, payments, and returns is stored in a database. SQL queries are used to answer important business questions like total sales, top products, customer spending, payment methods, and returns.

The main goal of this project is to practice writing SQL queries and to learn how data can be used to make better business decisions.
Tools Used

Business Objectives

The purpose of this project is to analyze e-commerce data and answer common business questions using SQL.. These objectives help in understanding overall business performance and customer behavior on the e-commerce platform.

01

- To calculate total sales and revenue generated from orders
- To understand how many orders were delivered and cancelled

02

- To identify top-selling and high-profit products
- To analyze customer spending behavior

03

- To study payment method usage across orders
- To track monthly sales trends

04

- To measure product return rates
- To calculate Average Order Value (AOV)

The database contains below tables:

orders

customers

addresses

categories

payments

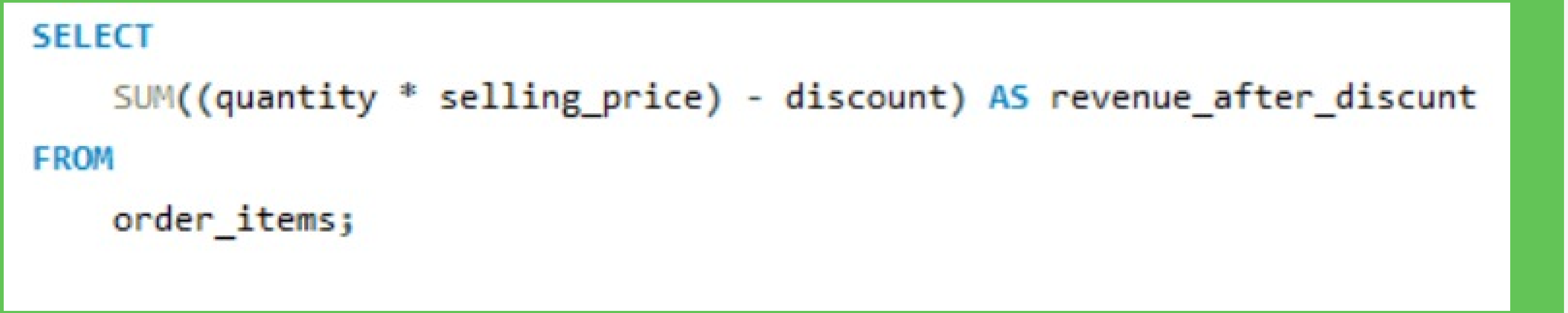
products

returns

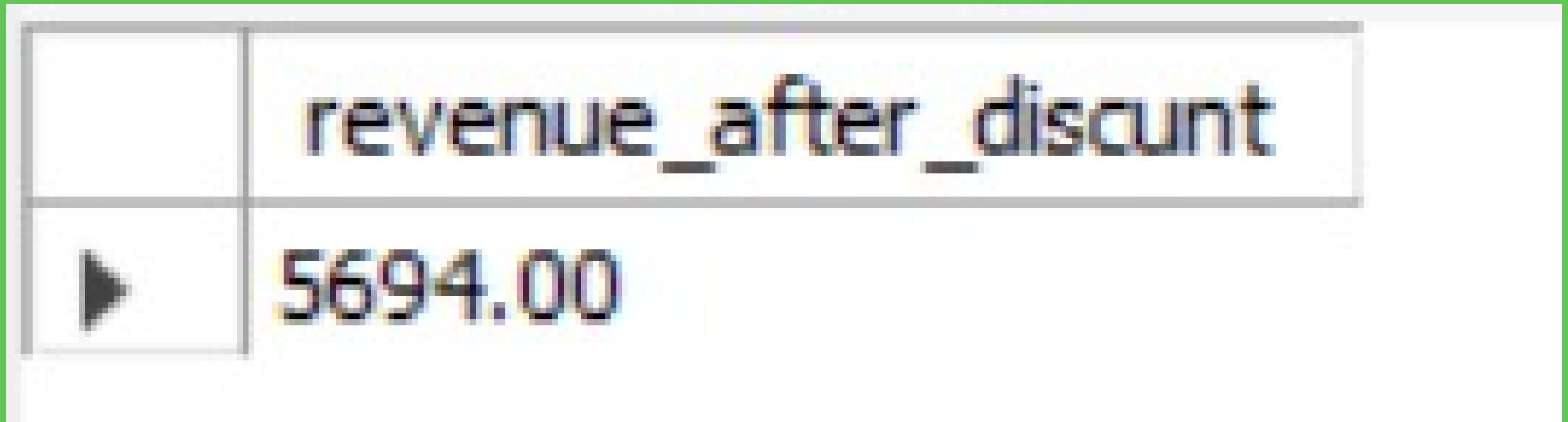
order_items

shipments

Total revenue generated (after discount)?

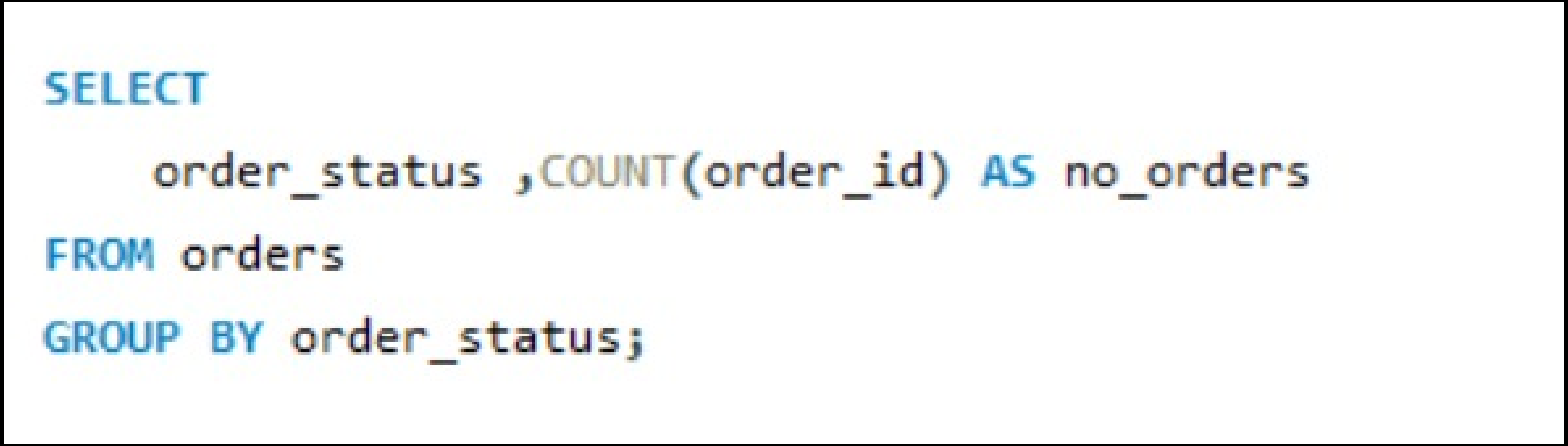


```sql
SELECT
    SUM((quantity * selling_price) - discount) AS revenue_after_discunt
FROM
    order_items;
```

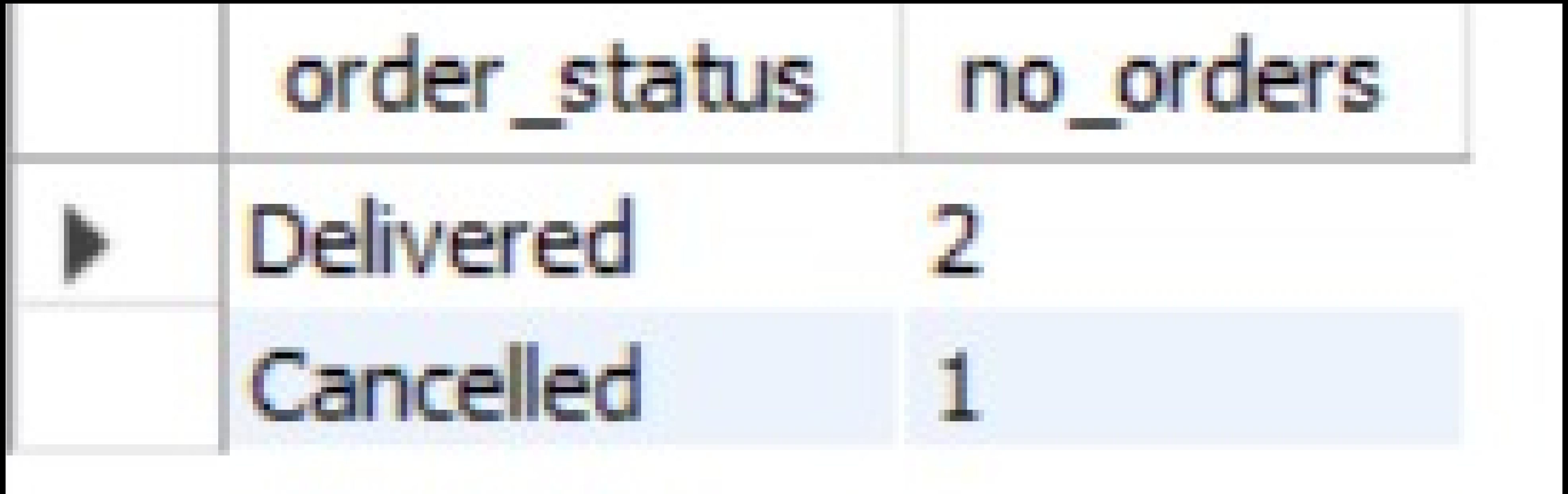


	revenue_after_discunt
▶	5694.00

Total orders count by order_status

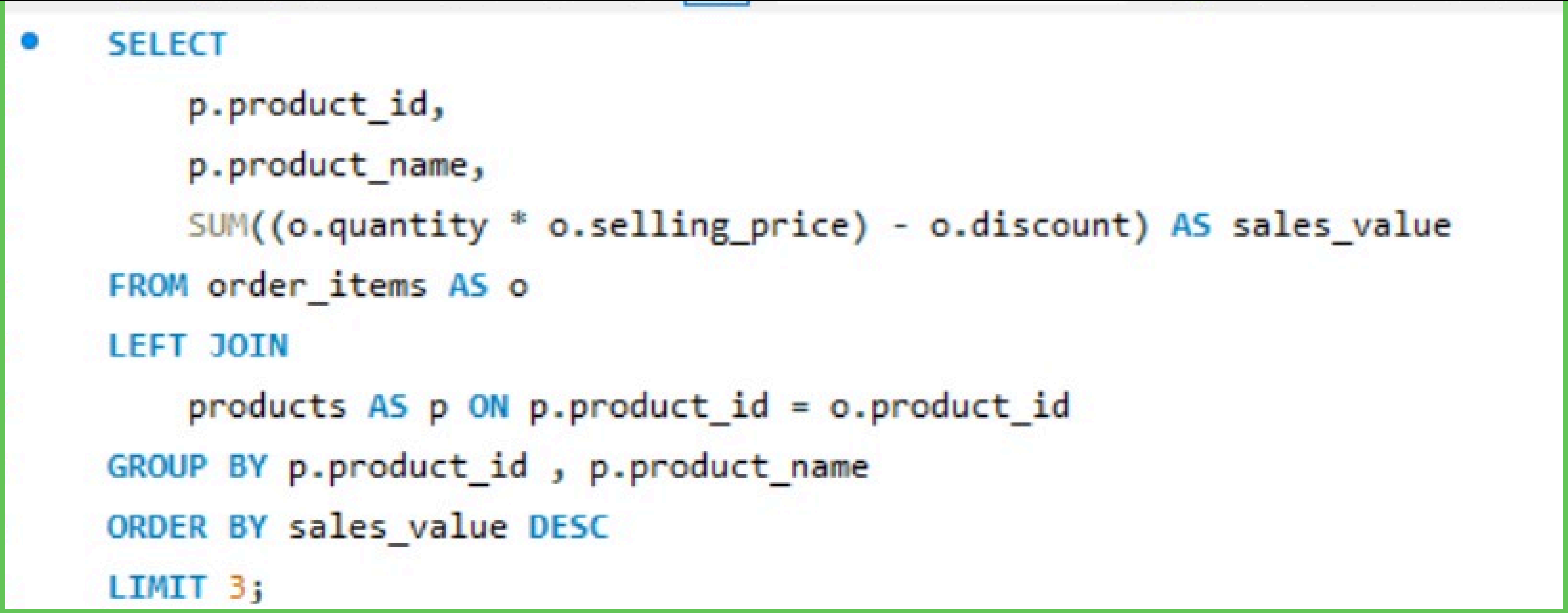


```
SELECT
    order_status ,COUNT(order_id) AS no_orders
FROM orders
GROUP BY order_status;
```

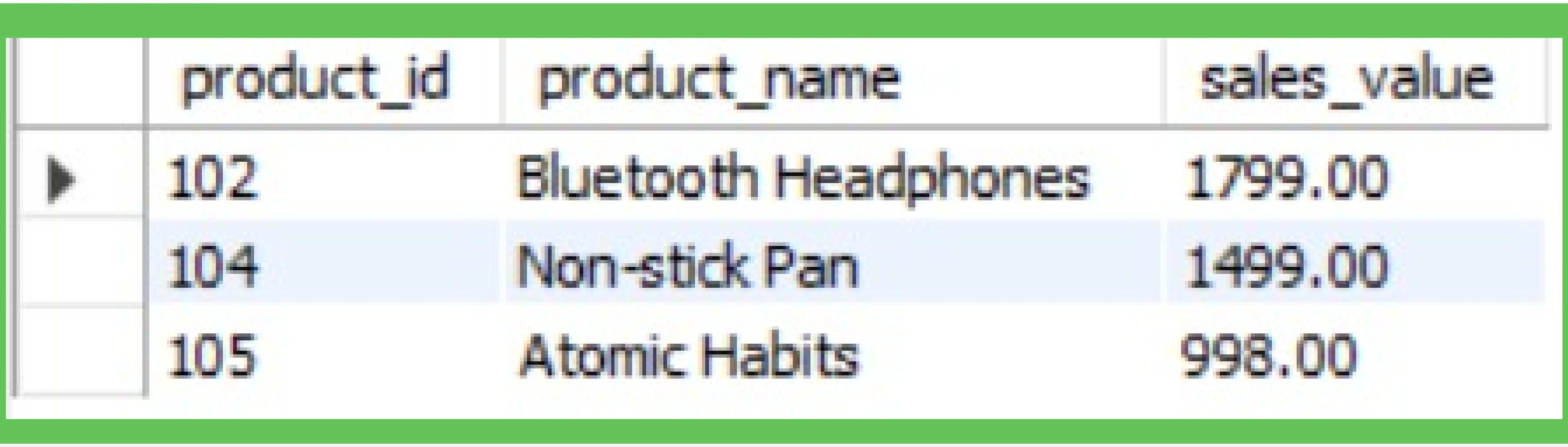


	order_status	no_orders
▶	Delivered	2
	Cancelled	1

Top 3 products by sales value

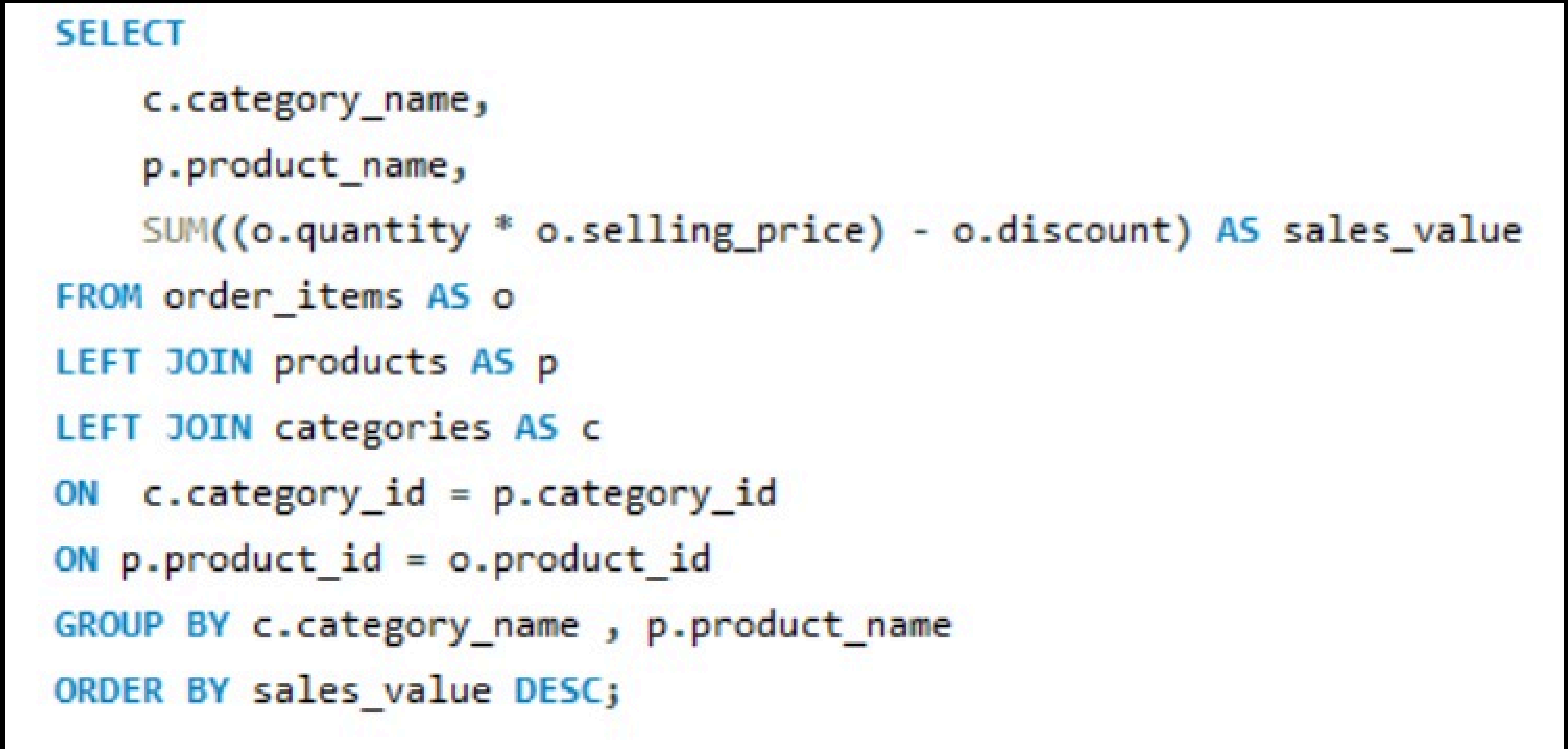


```
SELECT
    p.product_id,
    p.product_name,
    SUM((o.quantity * o.selling_price) - o.discount) AS sales_value
FROM order_items AS o
LEFT JOIN
    products AS p ON p.product_id = o.product_id
GROUP BY p.product_id , p.product_name
ORDER BY sales_value DESC
LIMIT 3;
```

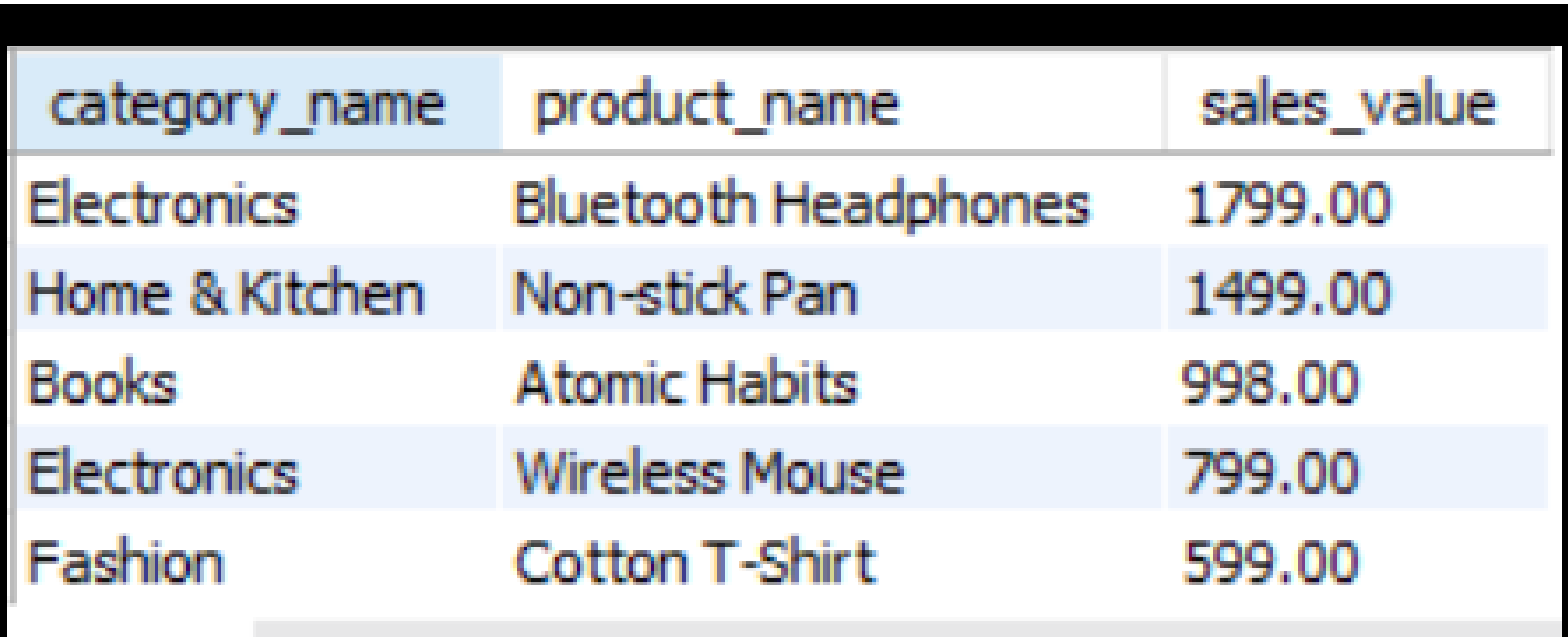


	product_id	product_name	sales_value
▶	102	Bluetooth Headphones	1799.00
	104	Non-stick Pan	1499.00
	105	Atomic Habits	998.00

Total revenue per category

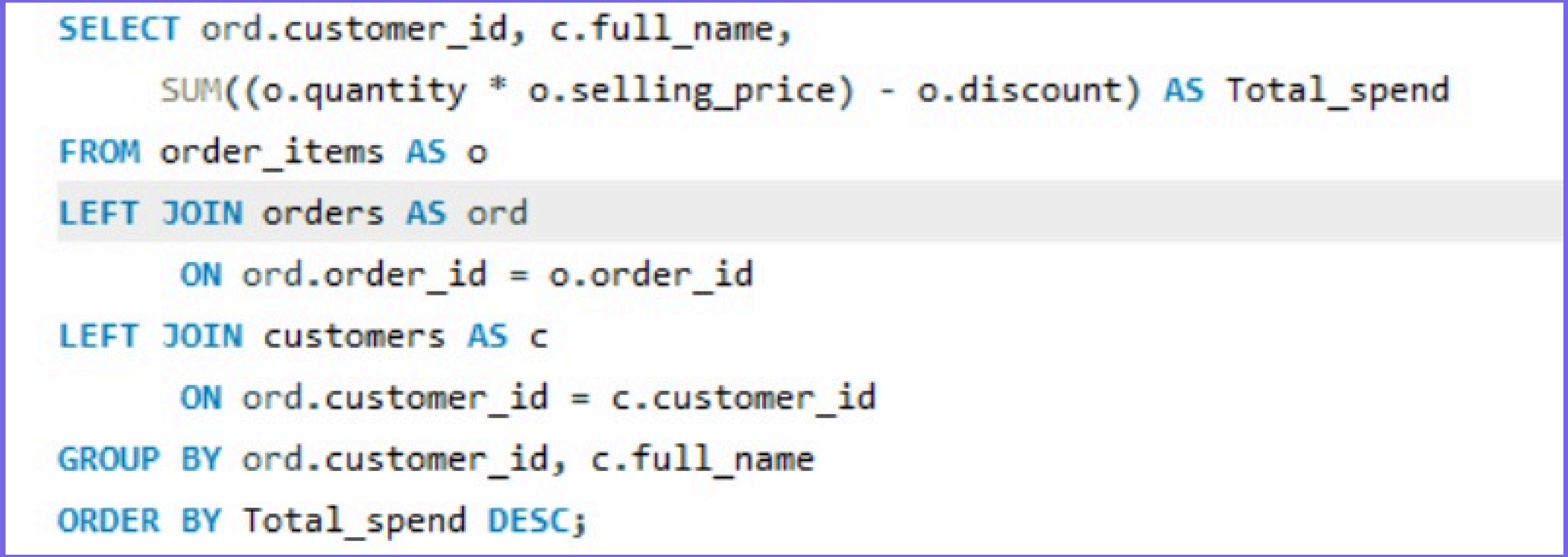


```
SELECT
    c.category_name,
    p.product_name,
    SUM((o.quantity * o.selling_price) - o.discount) AS sales_value
FROM order_items AS o
LEFT JOIN products AS p
LEFT JOIN categories AS c
ON  c.category_id = p.category_id
ON p.product_id = o.product_id
GROUP BY c.category_name , p.product_name
ORDER BY sales_value DESC;
```

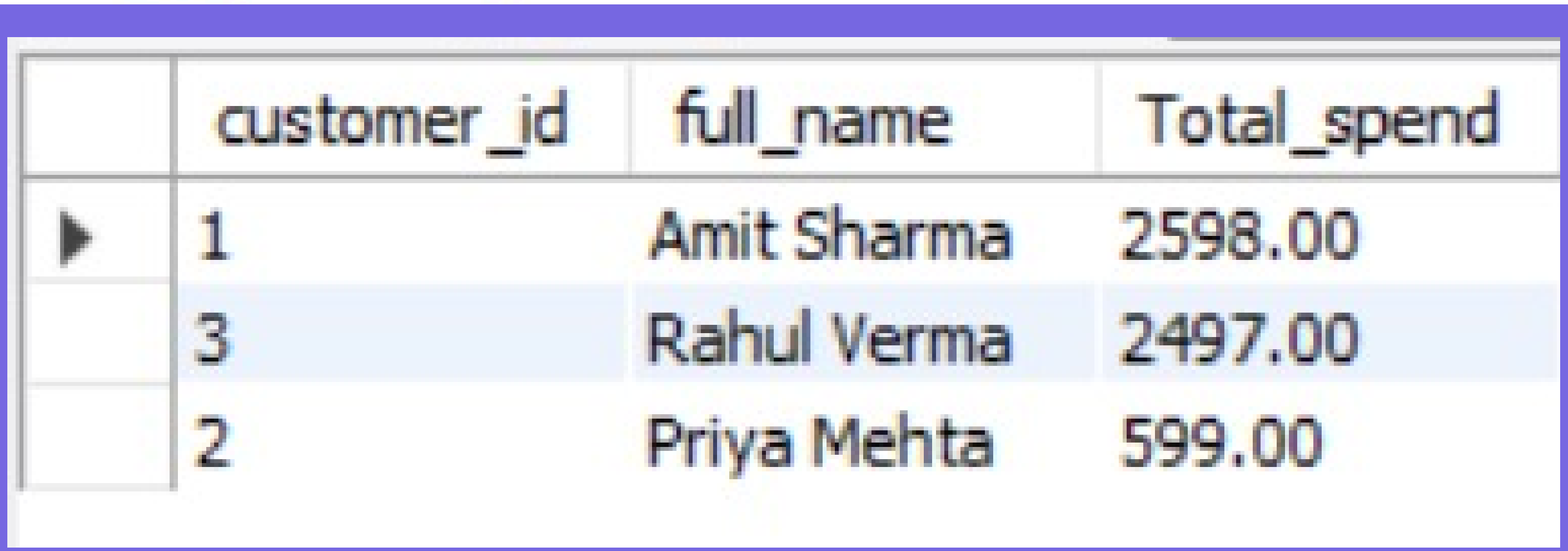


category_name	product_name	sales_value
Electronics	Bluetooth Headphones	1799.00
Home & Kitchen	Non-stick Pan	1499.00
Books	Atomic Habits	998.00
Electronics	Wireless Mouse	799.00
Fashion	Cotton T-Shirt	599.00

Top customers by total spend

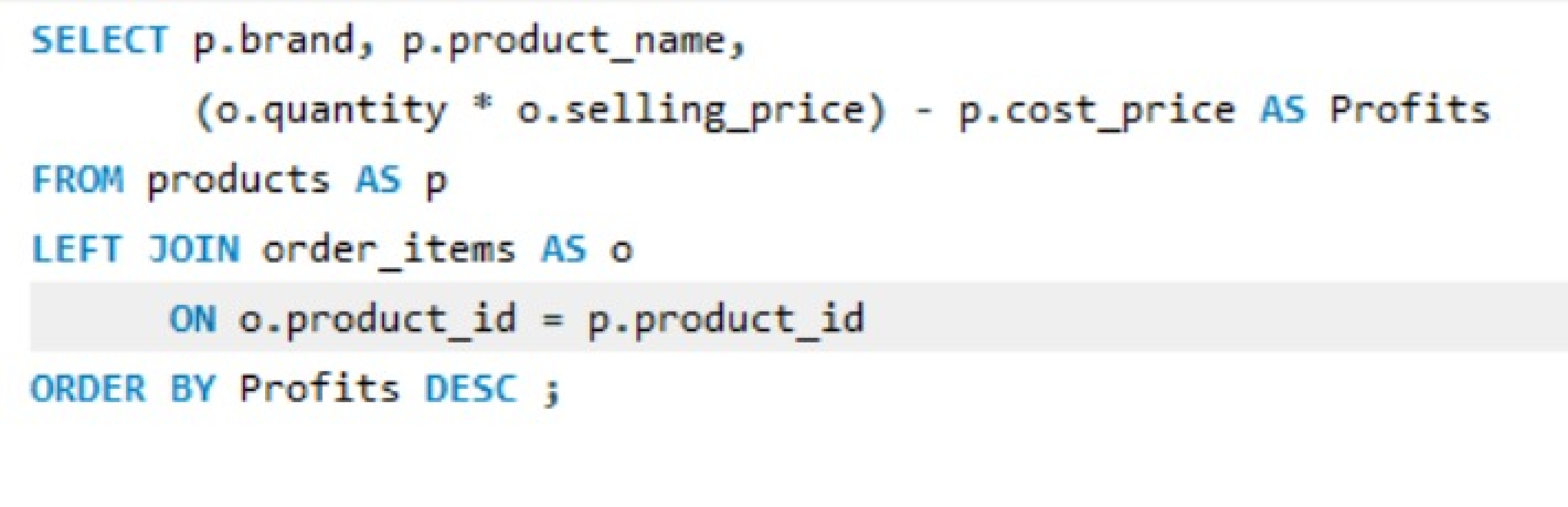


```
SELECT ord.customer_id, c.full_name,
     SUM((o.quantity * o.selling_price) - o.discount) AS Total_spend
FROM order_items AS o
LEFT JOIN orders AS ord
      ON ord.order_id = o.order_id
LEFT JOIN customers AS c
      ON ord.customer_id = c.customer_id
GROUP BY ord.customer_id, c.full_name
ORDER BY Total_spend DESC;
```

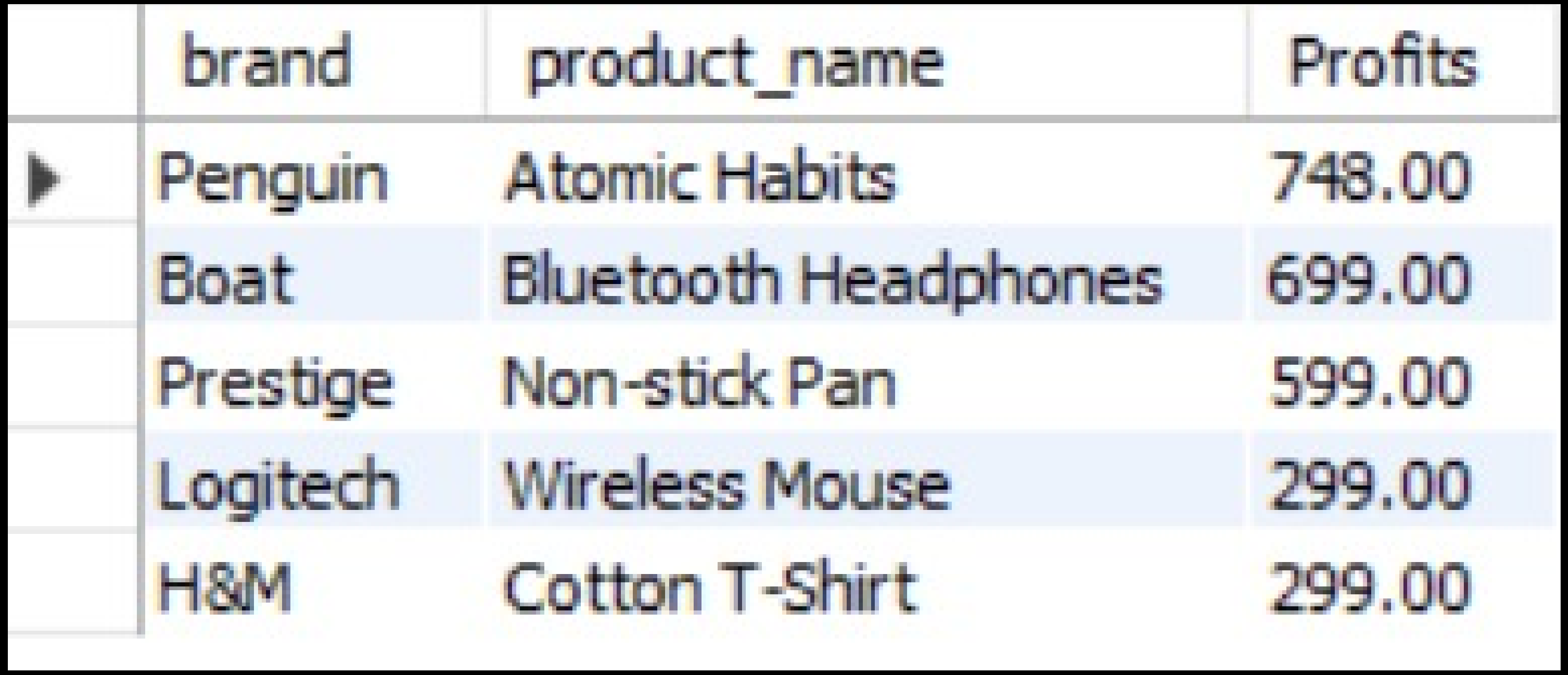


	customer_id	full_name	Total_spend
▶	1	Amit Sharma	2598.00
	3	Rahul Verma	2497.00
	2	Priya Mehta	599.00

Profit per product

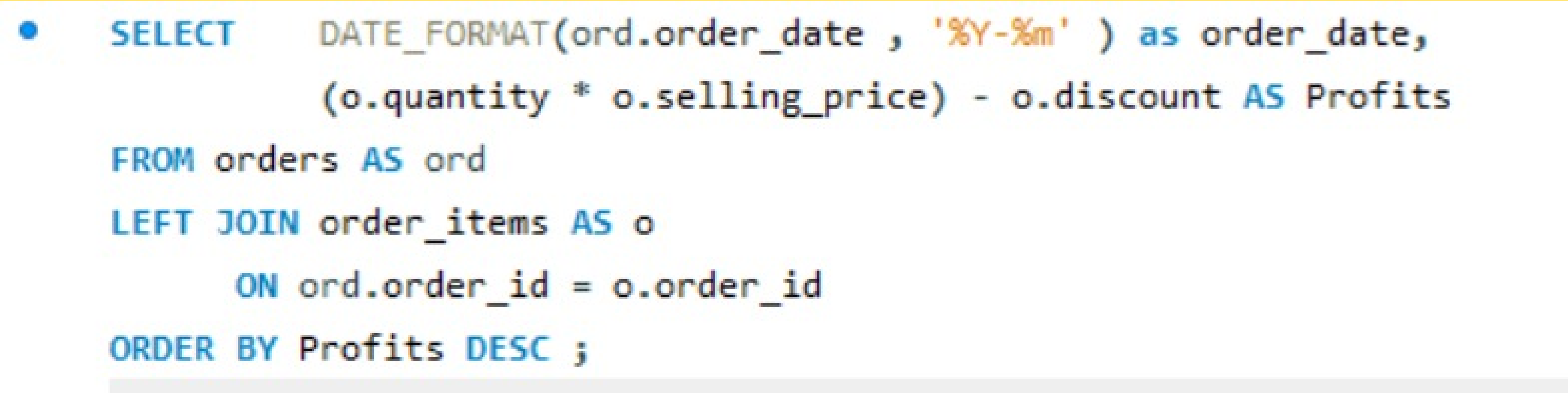


```
SELECT p.brand, p.product_name,
       (o.quantity * o.selling_price) - p.cost_price AS Profits
FROM products AS p
LEFT JOIN order_items AS o
      ON o.product_id = p.product_id
ORDER BY Profits DESC ;
```

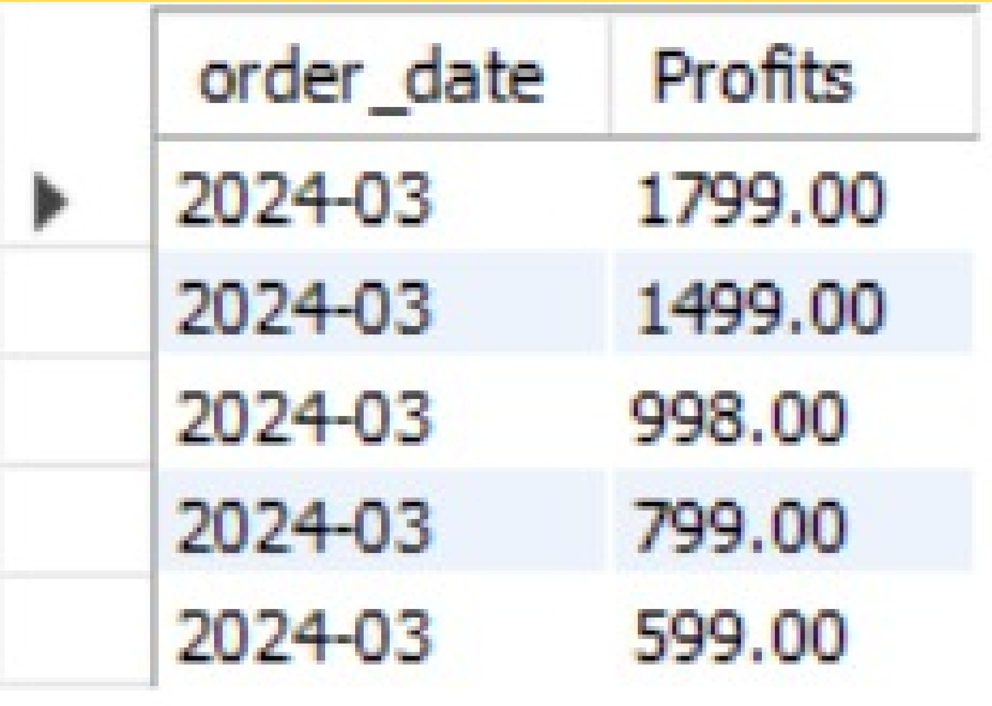


brand	product_name	Profits
Penguin	Atomic Habits	748.00
Boat	Bluetooth Headphones	699.00
Prestige	Non-stick Pan	599.00
Logitech	Wireless Mouse	299.00
H&M	Cotton T-Shirt	299.00

Monthly sales trend

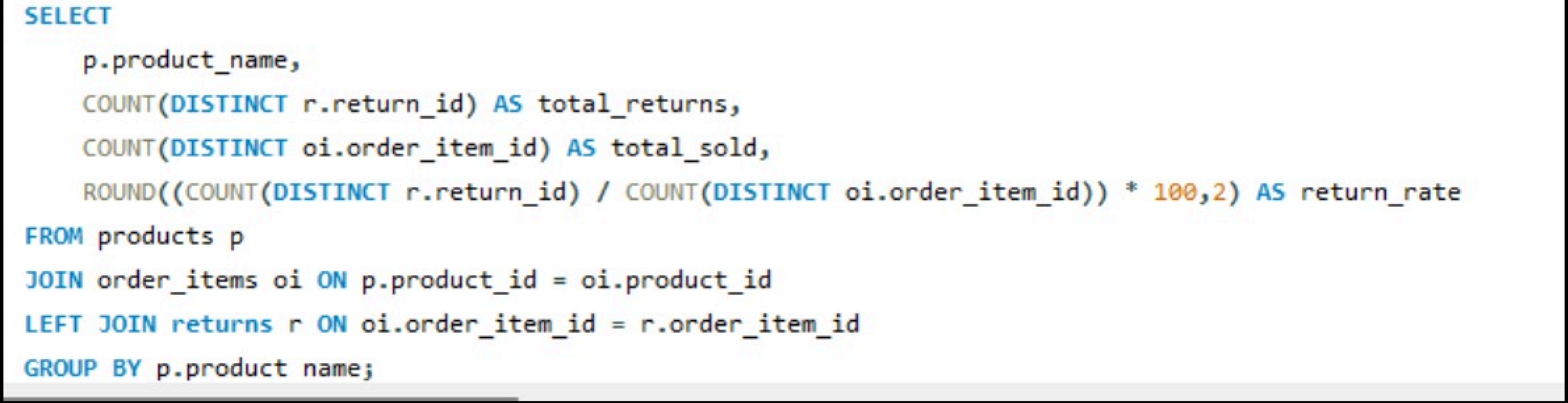


```
SELECT    DATE_FORMAT(ord.order_date , '%Y-%m' ) as order_date,
          (o.quantity * o.selling_price) - o.discount AS Profits
FROM orders AS ord
LEFT JOIN order_items AS o
      ON ord.order_id = o.order_id
ORDER BY Profits DESC ;
```

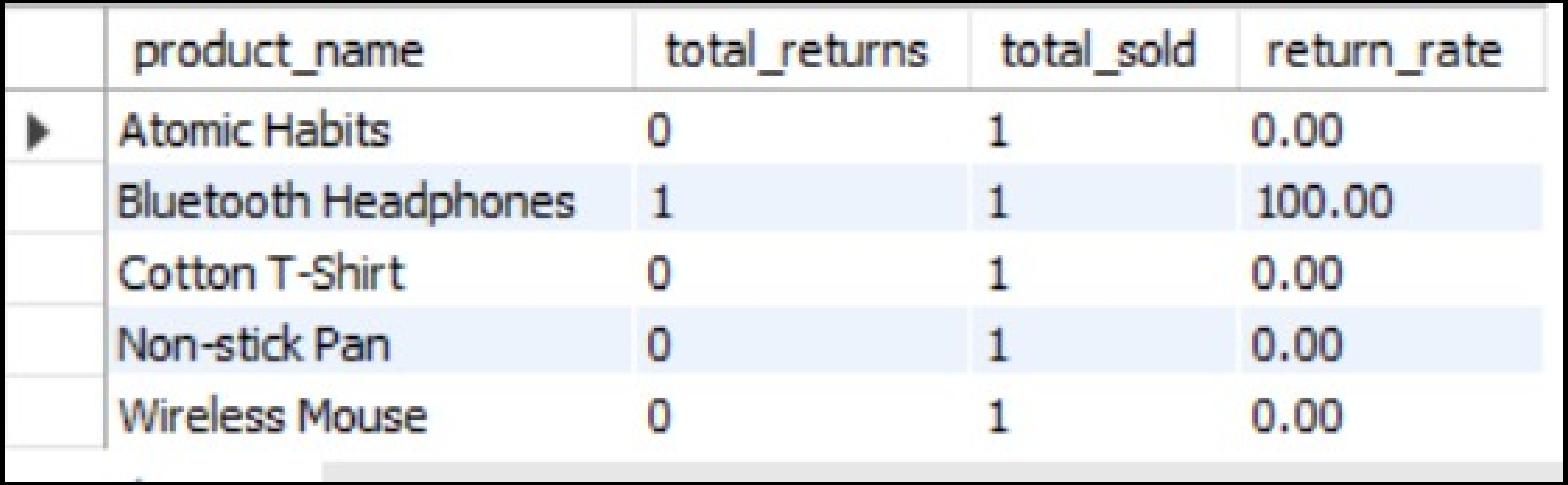


order_date	Profits
2024-03	1799.00
2024-03	1499.00
2024-03	998.00
2024-03	799.00
2024-03	599.00

Return rate per product

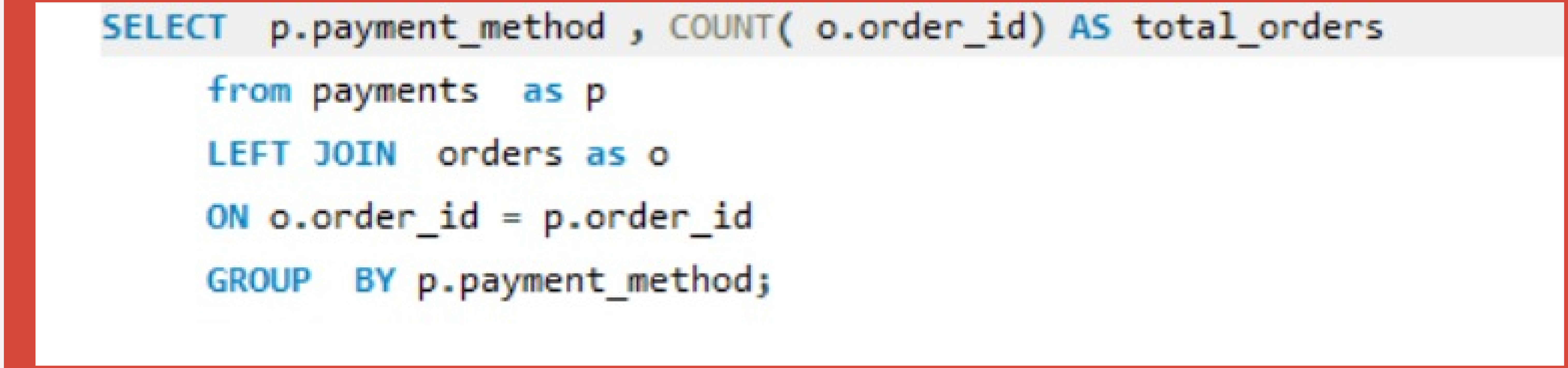


```
SELECT
    p.product_name,
    COUNT(DISTINCT r.return_id) AS total_returns,
    COUNT(DISTINCT oi.order_item_id) AS total_sold,
    ROUND((COUNT(DISTINCT r.return_id) / COUNT(DISTINCT oi.order_item_id)) * 100,2) AS return_rate
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
LEFT JOIN returns r ON oi.order_item_id = r.order_item_id
GROUP BY p.product name;
```

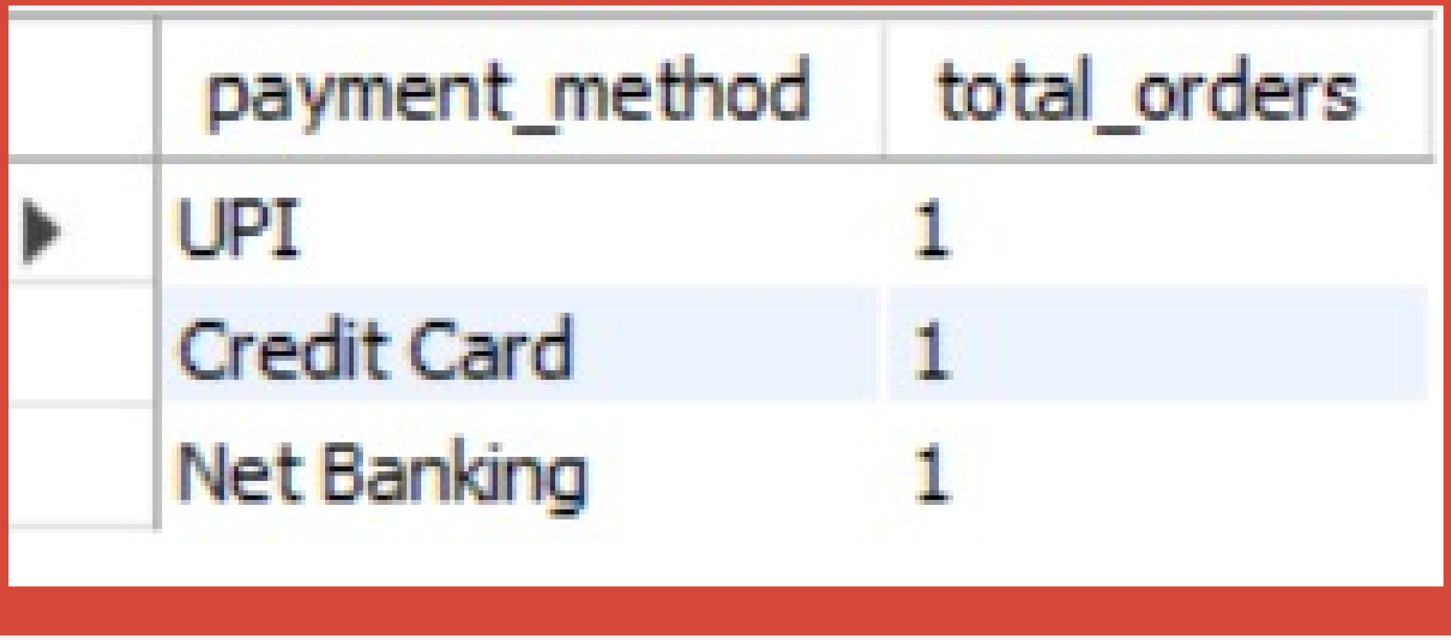


product_name	total_returns	total_sold	return_rate
Atomic Habits	0	1	0.00
Bluetooth Headphones	1	1	100.00
Cotton T-Shirt	0	1	0.00
Non-stick Pan	0	1	0.00
Wireless Mouse	0	1	0.00

Payment method usage count

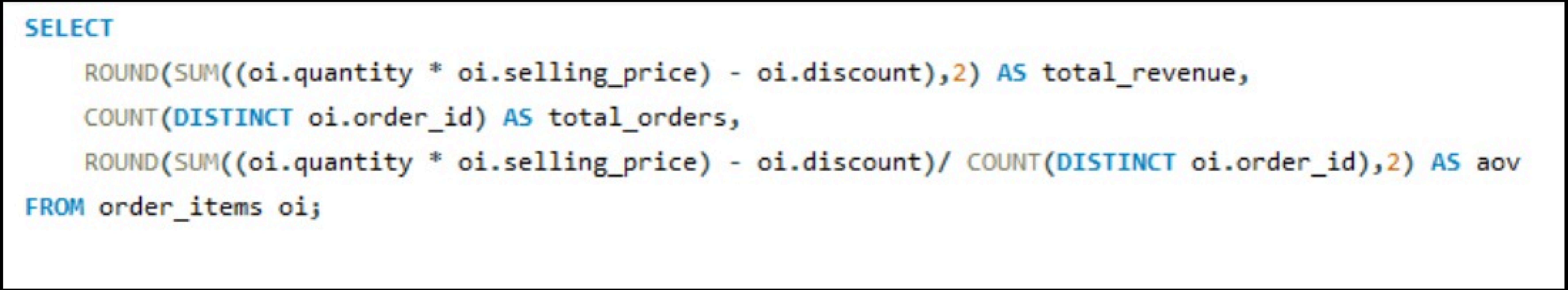


```
SELECT  p.payment_method , COUNT( o.order_id) AS total_orders
     from payments  as p
     LEFT JOIN  orders as o
     ON o.order_id = p.order_id
     GROUP  BY p.payment_method;
```

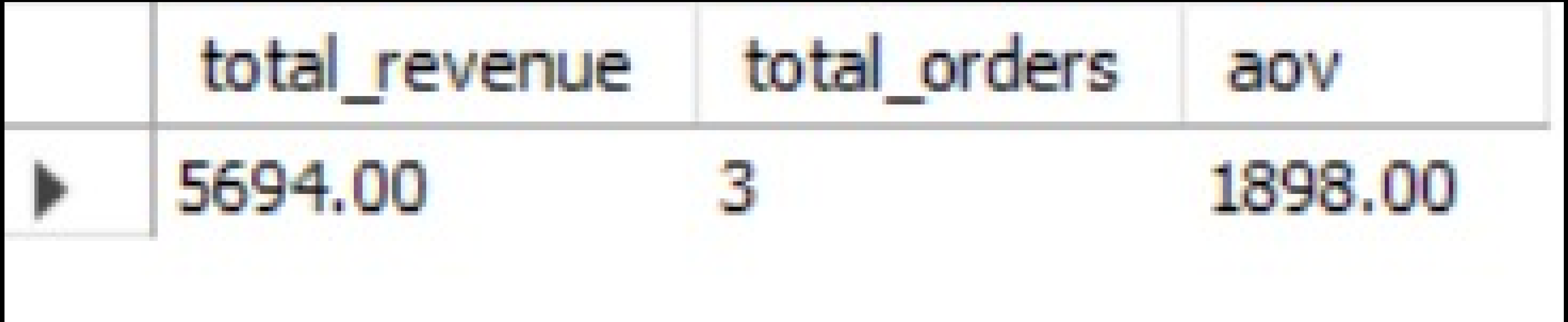


	payment_method	total_orders
▶	UPI	1
	Credit Card	1
	Net Banking	1

Average order value (AOV)



```
SELECT
    ROUND(SUM((oi.quantity * oi.selling_price) - oi.discount),2) AS total_revenue,
    COUNT(DISTINCT oi.order_id) AS total_orders,
    ROUND(SUM((oi.quantity * oi.selling_price) - oi.discount)/ COUNT(DISTINCT oi.order_id),2) AS aov
FROM order_items oi;
```



	total_revenue	total_orders	aov
▶	5694.00	3	1898.00

Business Recommendations

- Improve product quality to reduce returns
- Promote high-profit products through offers
- Focus marketing on top-performing categories
- Optimize discounts to increase order value

Conclusion

This project demonstrates the use of SQL to analyze real-world e-commerce data. By designing a relational database and answering business-driven questions, meaningful insights were derived to support data-driven decision making.