



FLIPCART SQL PROJECT

Intermediate SQL Querying on E-commerce Dataset

By : Roshni Kumari



SQL Project Report: Flipkart Sales Analysis Using Intermediate Queries

Objective : To create a small Flipkart-style database with sample data and use SQL to answer real-world business questions. This helps in understanding how to analyze e-commerce data using intermediate SQL queries.

Tools & Technologies Used:

- MySQL (Querying & Data Analysis)
- ChatGPT (Data generation, prompt-based assistance)
- MySQL Workbench (Query execution)



◆ Intermediate Queries

- Get the total number of orders placed by each customer.
- Find the total sales amount for each product.
- List the top 3 most expensive products.
- Get the details of customers who ordered more than one item in a single order.
- Show the total quantity sold for each category.
- List the top 3 customers who spent the most in total on orders.



- Find all products that have never been ordered.
- Get the average order value per customer.
- Show the most frequently ordered product based on total quantity sold.

Dataset Overview :

I asked ChatGPT to help me create a sample Flipkart-style dataset.

Together, we created four tables:

- **customers** - stores customer details like name, city, and email
- **products** - contains product names, categories, and prices
- **orders** - tracks order date, total amount, and which customer placed the order
- **order_items** - lists the individual products in each order, along with quantity and price

I also asked for over 40 sample entries in total to make the data look like real Flipkart transactions.



Get the total number of orders placed by each customer.

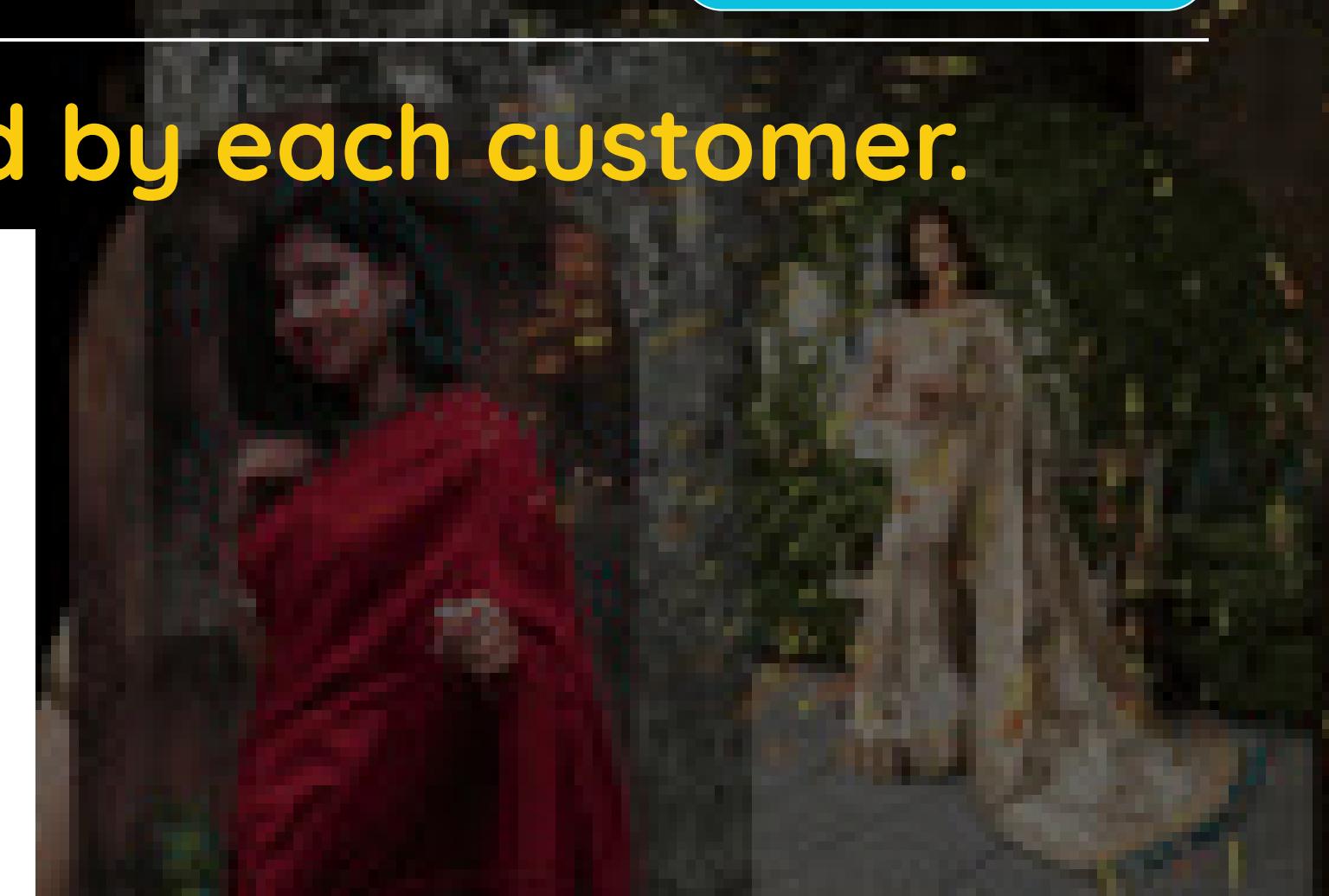
```
-- Get the total number of orders placed by each customer.
```

```
SELECT
```

```
    customers.customer_id,  
    customers.name,  
    customers.email,  
    COUNT(orders.order_id) AS total_order
```

```
FROM
```

```
customers  
    JOIN  
    orders ON customers.customer_id = orders.customer_id  
GROUP BY customers.customer_id , customers.name , customers.email;
```



A screenshot of a database query results grid. The grid has columns for customer_id, name, email, and total_order. The data shows 10 rows of customer information with their respective order counts.

	customer_id	name	email	total_order
1	Ravi Kumar	ravi@example.com	3	
2	Priya Singh	priya@example.com	2	
3	Amit Shah	amit@example.com	2	
4	Neha Gupta	neha@example.com	1	
5	Karan Mehta	karan@example.com	1	
6	Meera Joshi	meera@example.com	1	
7	Arjun Verma	arjun@example.com	1	
8	Simran Kaur	simran@example.com	1	
9	Rahul Das	rahul@example.com	1	
10	Sneha Rao	sneha@example.com	1	



Find the total sales amount for each product.

-- Find the total sales amount for each product.

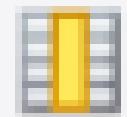
SELECT

SUM(price) AS total_sales_amount

FROM

products;

Result Grid



Filter Rows

total_sales_amount

263990.00



List the top 3 most expensive products.

-- List the top 3 most expensive products.

SELECT

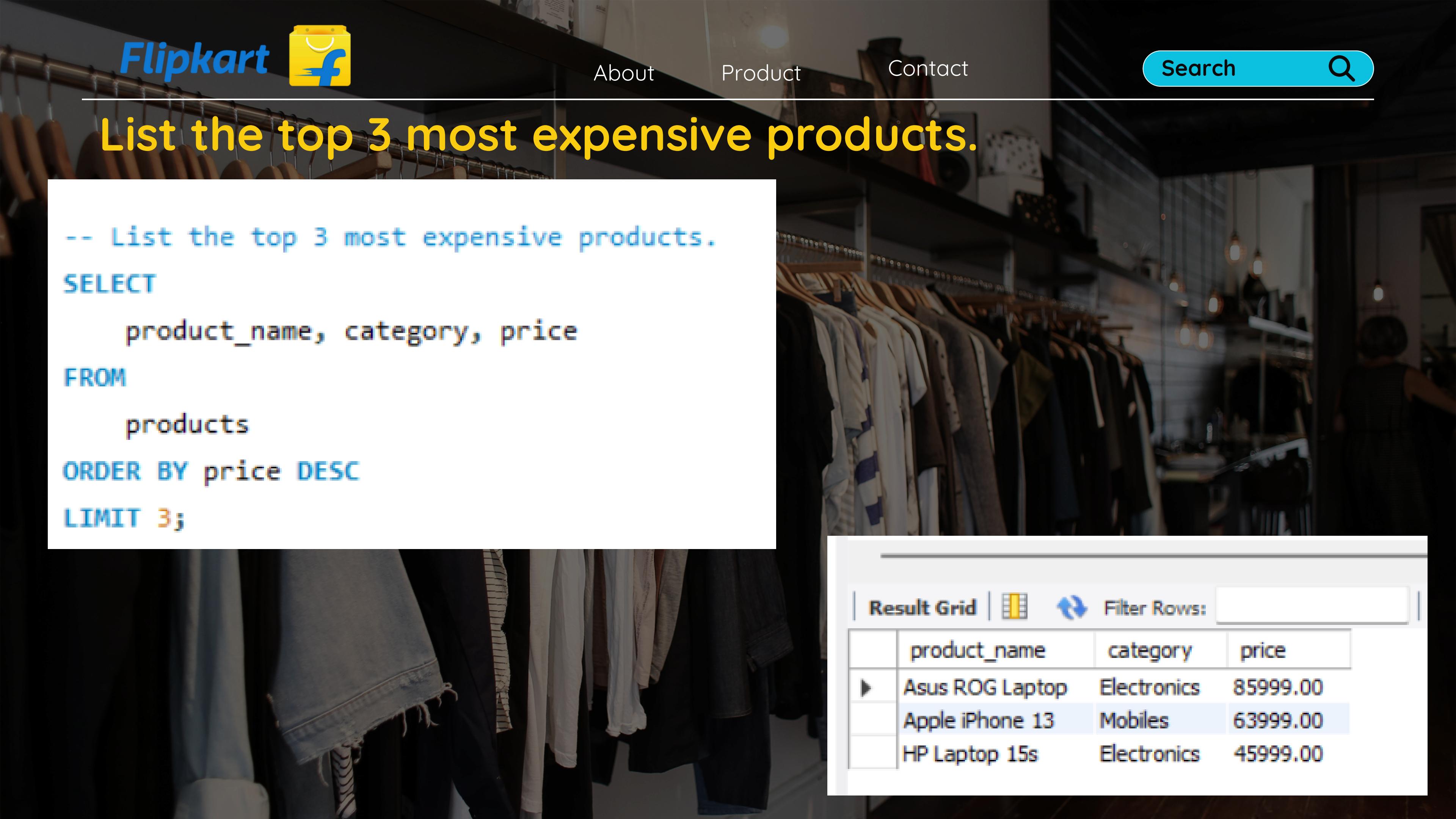
product_name, category, price

FROM

products

ORDER BY price **DESC**

LIMIT 3;



The background of the slide shows a blurred image of a clothing store interior, with rows of clothes hanging on wooden racks.

Result Grid | Filter Rows:

	product_name	category	price
▶	Asus ROG Laptop	Electronics	85999.00
	Apple iPhone 13	Mobiles	63999.00
	HP Laptop 15s	Electronics	45999.00



Get the details of customers who ordered more than one item in a single order.

-- Get the details of customers who ordered more than one item in a single order.

SELECT

c.name, c.email, COUNT(oi.quantity) AS no_of_items_ordered

FROM

customers AS c

JOIN

orders AS o ON c.customer_id = o.customer_id

JOIN

order_items AS oi ON o.order_id = oi.order_id

GROUP BY c.name , c.email

HAVING no_of_items_ordered > 1;

Result Grid			
	name	email	no_of_items_ordered
▶	Ravi Kumar	ravi@example.com	5
	Priya Singh	priya@example.com	4
	Amit Shah	amit@example.com	3
	Karan Mehta	karan@example.com	2
	Meera Joshi	meera@example.com	3
	Arjun Verma	arjun@example.com	2
	Simran Kaur	simran@example.com	2
	Rahul Das	rahul@example.com	2
	Sneha Rao	sneha@example.com	2



List the top 3 most expensive products.

-- List the top 3 most expensive products.

SELECT

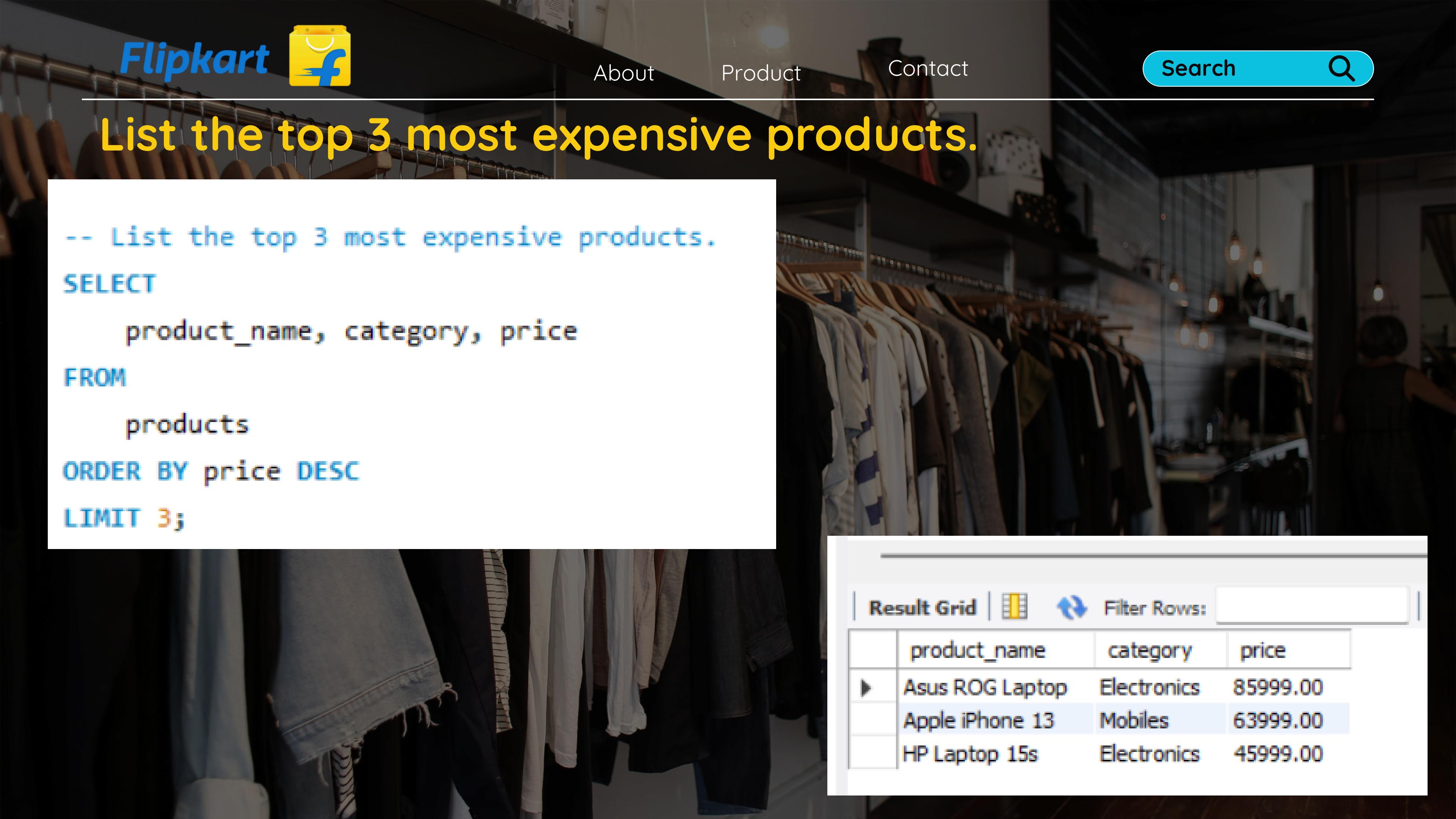
product_name, category, price

FROM

products

ORDER BY price **DESC**

LIMIT 3;



A dark, atmospheric photograph of a clothing store interior. In the foreground, several racks of clothes are visible, including shirts and jackets. In the background, a person is seen from behind, walking through the store. The lighting is low, creating a moody atmosphere.

Result Grid			
	product_name	category	price
▶	Asus ROG Laptop	Electronics	85999.00
	Apple iPhone 13	Mobiles	63999.00
	HP Laptop 15s	Electronics	45999.00



Show the total quantity sold for each category.

-- Show the total quantity sold for each category.

SELECT

p.category, SUM(oi.quantity) AS total_quantity

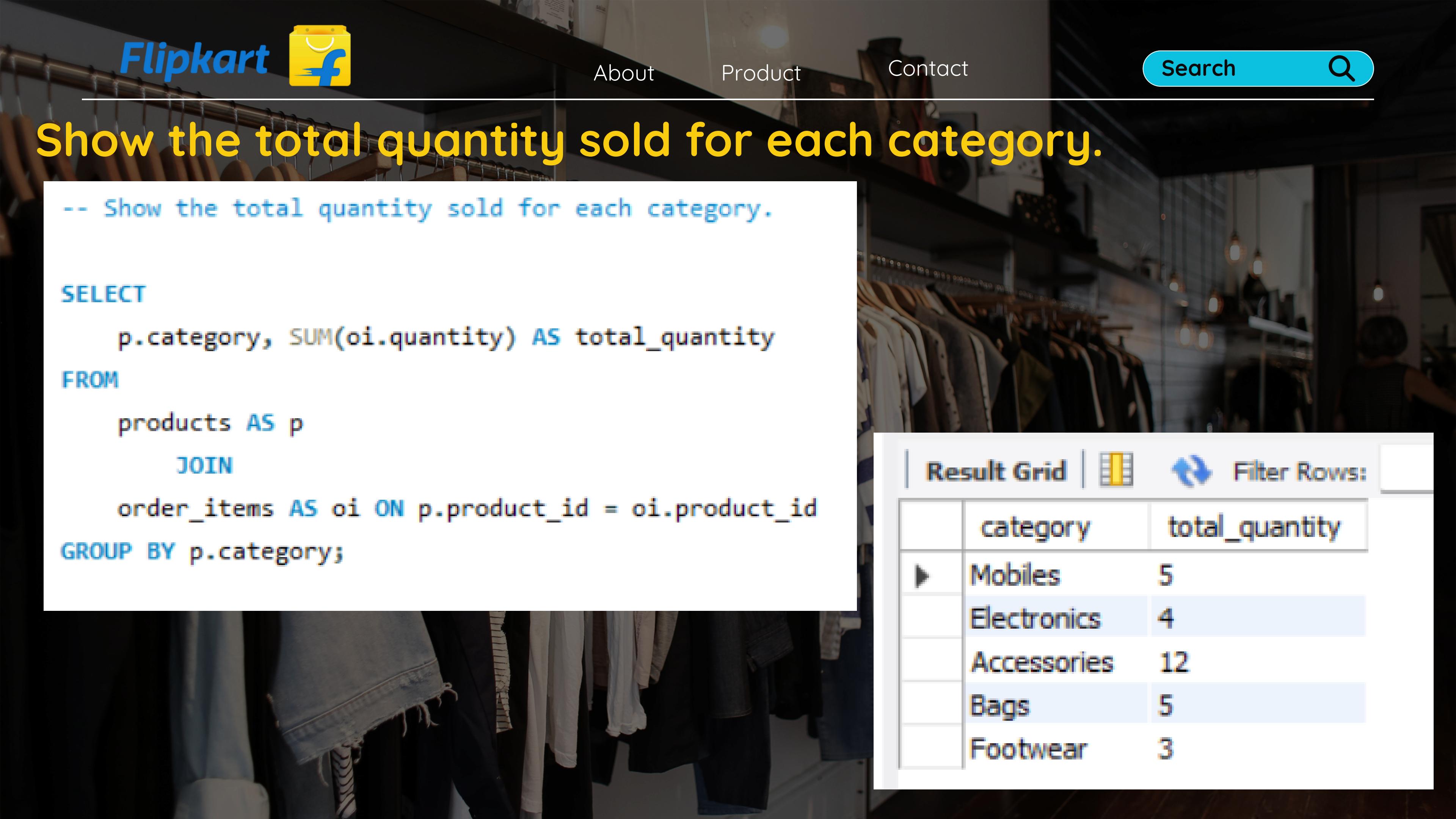
FROM

products AS p

JOIN

order_items AS oi ON p.product_id = oi.product_id

GROUP BY p.category;



The background of the slide features a photograph of a clothing store interior. In the foreground, there's a rack of various shirts and blouses. In the background, more racks of clothes are visible, along with some shelves and a person walking away from the camera.

	category	total_quantity
▶	Mobiles	5
	Electronics	4
	Accessories	12
	Bags	5
	Footwear	3



List the top 3 customers who spent the most in total on orders.

```
-- List the top 3 customers who spent the most in total on orders.
```

```
SELECT
```

```
    c.name, SUM(o.total_amount) AS total_amount_spent
```

```
FROM
```

```
customers AS c
```

```
JOIN
```

```
orders AS o ON c.customer_id = o.customer_id
```

```
group by c.name;
```

	name	total_amount_spent
▶	Ravi Kumar	34996.00
	Priya Singh	62997.00
	Amit Shah	126997.00
	Neha Gupta	4999.00
	Karan Mehta	14999.00
	Meera Joshi	14998.00
	Arjun Verma	4999.00
	Simran Kaur	19999.00
	Rahul Das	85999.00
	Sneha Rao	12999.00



Get the average order value per customer.

-- Get the average order value per customer.

SELECT

c.name, round(AVG(o.total_amount),2) AS Average_order_value

FROM

customers AS c

JOIN

orders AS o ON c.customer_id = o.customer_id

GROUP BY c.name;

name	Average_order_value
Ravi Kumar	11665.33
Priya Singh	31498.50
Amit Shah	63498.50
Neha Gupta	4999.00
Karan Mehta	14999.00
Meera Joshi	14998.00
Arjun Verma	4999.00
Simran Kaur	19999.00
Rahul Das	85999.00
Sneha Rao	12999.00



Show the most frequently ordered product based on total quantity sold.

-- Show the most frequently ordered product based on total quantity sold.

SELECT

p.product_name, SUM(oi.quantity) AS total_quantity_sold

FROM

products AS p

JOIN

order_items AS oi ON p.product_id = oi.product_id

GROUP BY p.product_name

ORDER BY total_quantity_sold DESC

LIMIT 1;

Result Grid		Filter Rows:
	product_name	total_quantity_sold
▶	Boat Headphones	10



Key Learnings:

- Strengthened understanding of SQL joins (especially LEFT JOIN and multi-table joins)
- Gained practical skills in grouping, filtering, and aggregating data
- Learned how to structure real-world business questions into SQL
- Practised handling product-category analysis and customer behaviour tracking

Conclusion:

This project successfully demonstrated how SQL can be applied to simulate real business questions in an e-commerce setting. It helped develop proficiency in writing clean, efficient queries that solve practical data analysis problems.

Thank You

