DATA ANALYST

Intenship Task 4

DESCRIPTION

This task focuses on using intermediate SQL JOIN operations to combine multiple tables and answer business-oriented questions. The objective is to practice INNER JOIN and LEFT JOIN, analyze customer and sales data, calculate revenue metrics, and derive meaningful insights to support data-driven decision-making.

PREPARED BY

Reema Safrin M
(20-01-2026)

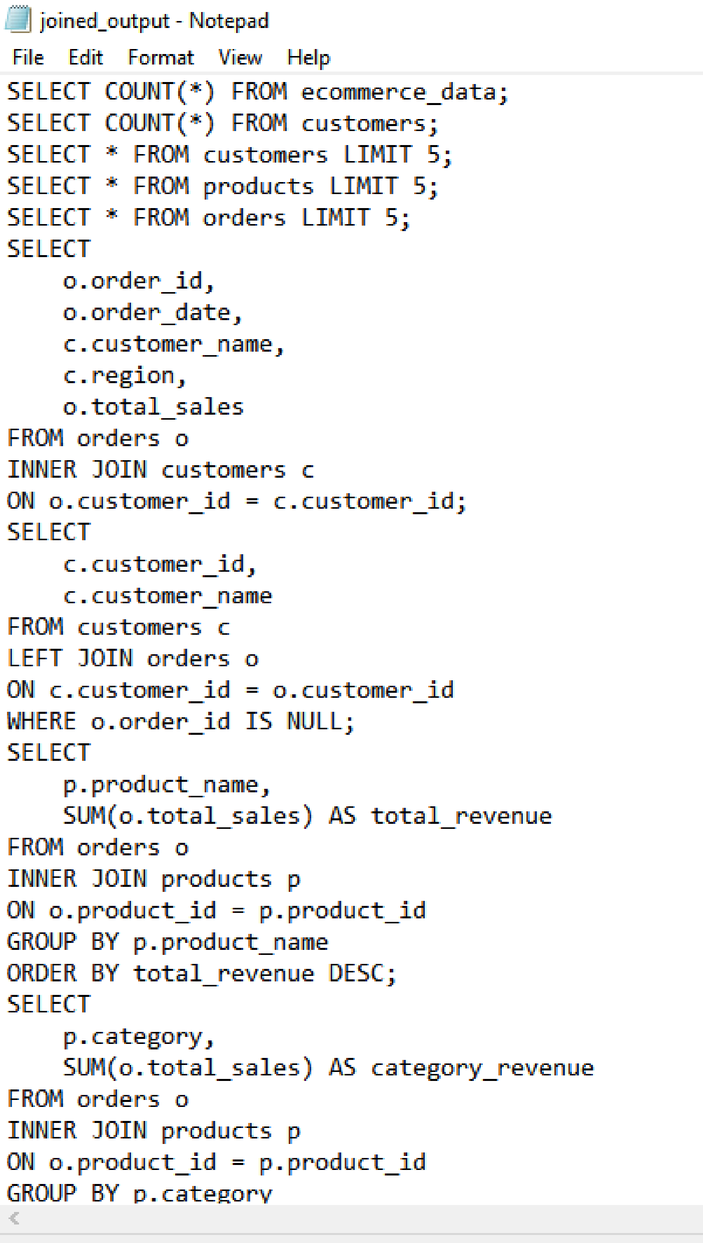
MY WORK

AIn this task, I imported an e-commerce dataset into SQLite and verified data integrity using record counts. I normalized the data by creating separate customers, products, and orders tables from the raw dataset. Using INNER JOIN, I combined orders with customers and products to analyze sales performance and revenue distribution. LEFT JOIN was applied to identify customers who have not placed any orders, which is useful for marketing analysis. I also performed aggregations to calculate product-wise and category-wise revenue and applied filters to answer region-specific business questions. All query results were exported and documented systematically.

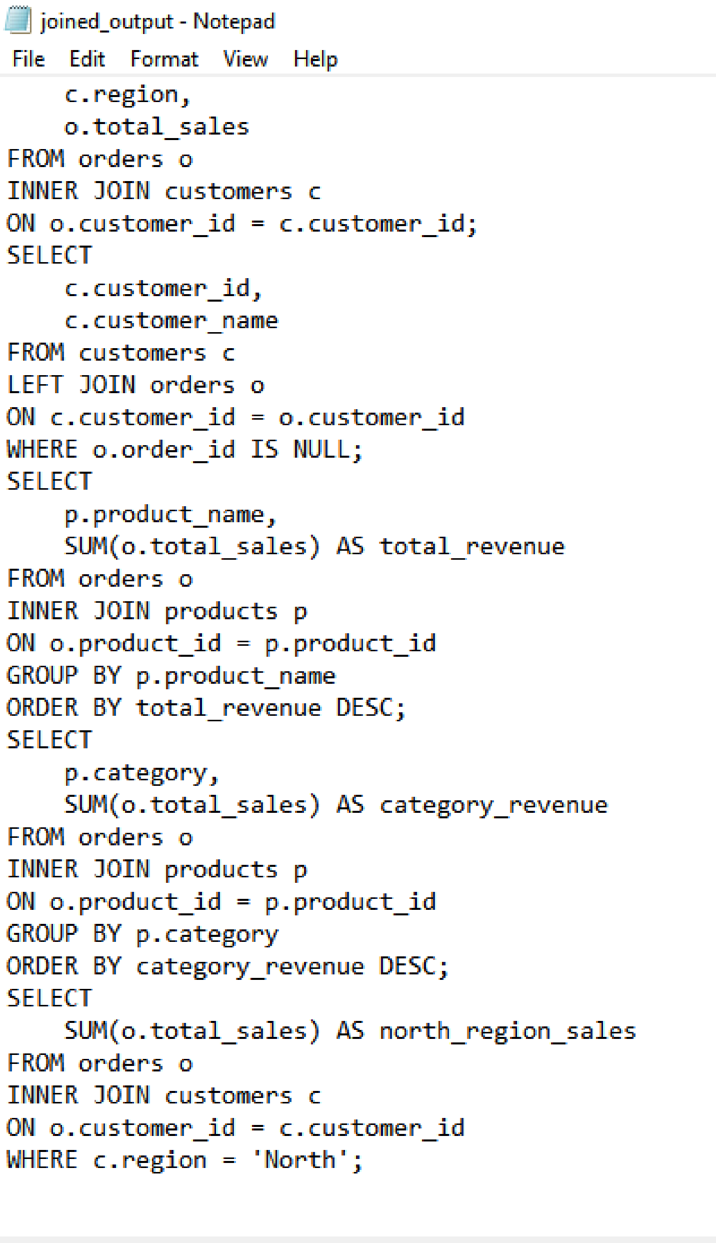
RAW-DATASET

Ecommerce-data

OUTLINE :



```
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
SELECT * FROM customers LIMIT 5;
SELECT * FROM products LIMIT 5;
SELECT * FROM orders LIMIT 5;
SELECT
    o.order_id,
    o.order_date,
    c.customer_name,
    c.region,
    o.total_sales
FROM orders o
INNER JOIN customers c
ON o.customer_id = c.customer_id;
SELECT
    c.customer_id,
    c.customer_name
FROM customers c
LEFT JOIN orders o
ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
SELECT
    p.product_name,
    SUM(o.total_sales) AS total_revenue
FROM orders o
INNER JOIN products p
ON o.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_revenue DESC;
SELECT
    p.category,
    SUM(o.total_sales) AS category_revenue
FROM orders o
INNER JOIN products p
ON o.product_id = p.product_id
GROUP BY p.category
```



```
    c.region,
    o.total_sales
FROM orders o
INNER JOIN customers c
ON o.customer_id = c.customer_id;
SELECT
    c.customer_id,
    c.customer_name
FROM customers c
LEFT JOIN orders o
ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
SELECT
    p.product_name,
    SUM(o.total_sales) AS total_revenue
FROM orders o
INNER JOIN products p
ON o.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_revenue DESC;
SELECT
    p.category,
    SUM(o.total_sales) AS category_revenue
FROM orders o
INNER JOIN products p
ON o.product_id = p.product_id
GROUP BY p.category
ORDER BY category_revenue DESC;
SELECT
    SUM(o.total_sales) AS north_region_sales
FROM orders o
INNER JOIN customers c
ON o.customer_id = c.customer_id
WHERE c.region = 'North';
```

Data validation :

Counted total records in the raw dataset to ensure successful import.

Table creation :

Created customers table using DISTINCT customer details.

Created products table with product and category information.

Created orders table containing transactional sales data.

Data verification :

Used COUNT and LIMIT queries to validate table creation and data accuracy.

INNER JOIN queries :

Joined orders with customers to analyze order details with customer information.

Joined orders with products to calculate total revenue per product.

LEFT JOIN query :

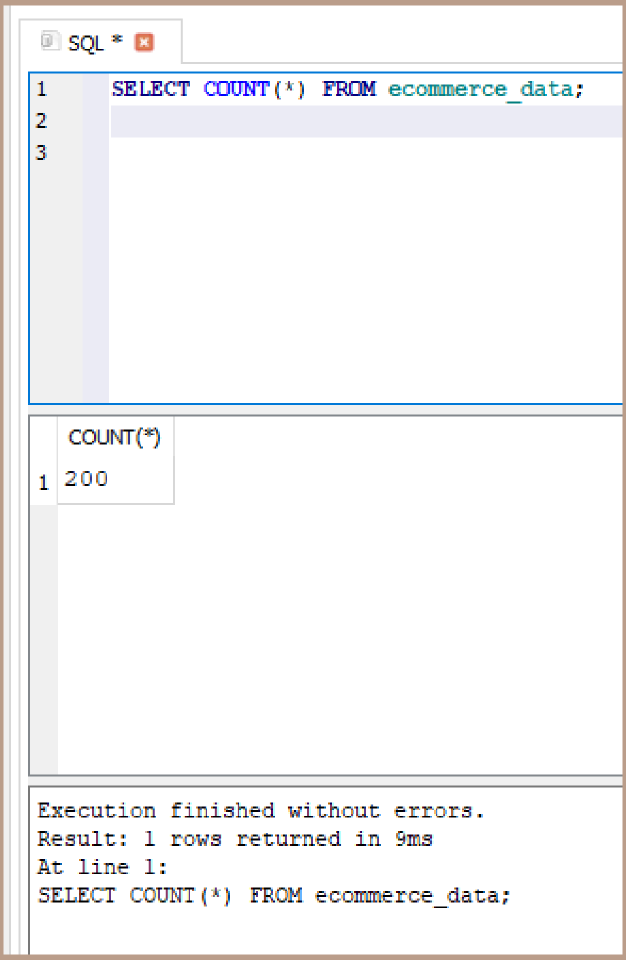
Identified customers who have never placed an order.

Aggregation queries :

Calculated category-wise revenue distribution.

Analyzed region-based sales using WHERE conditions.

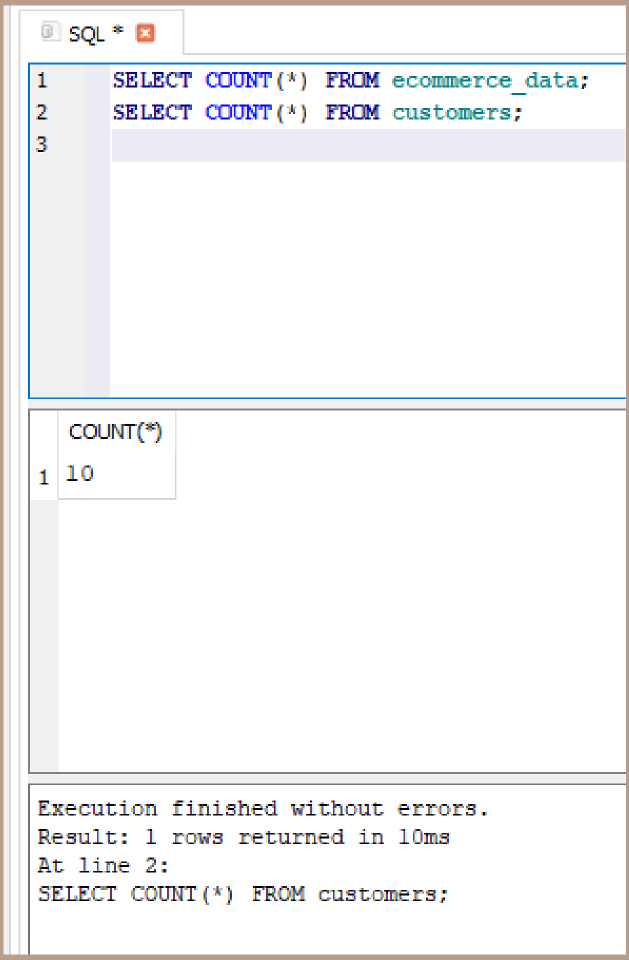
RESULTS :



```sql
SELECT COUNT(*) FROM ecommerce_data;
```

	COUNT(*)
1	200

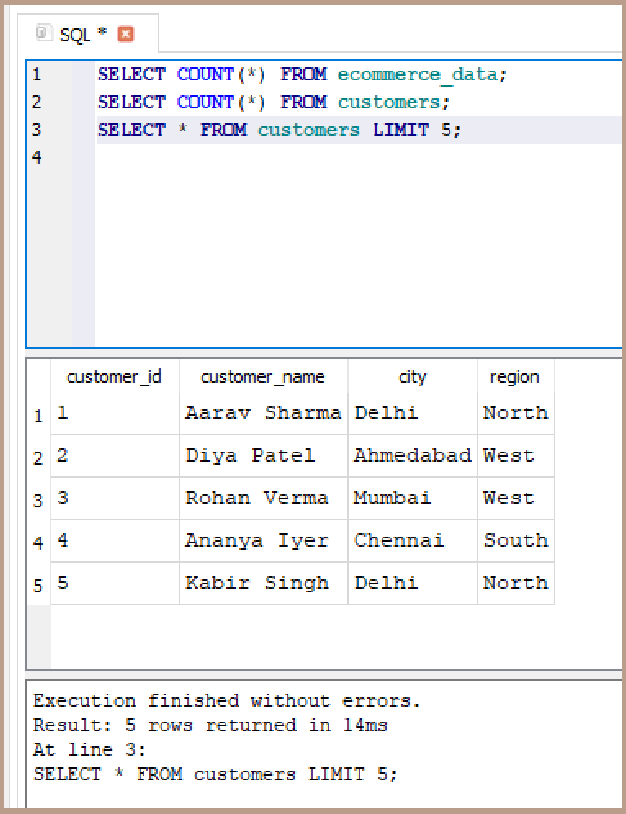
```
Execution finished without errors.
Result: 1 rows returned in 9ms
At line 1:
SELECT COUNT(*) FROM ecommerce_data;
```



```sql
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
```

	COUNT(*)
1	10

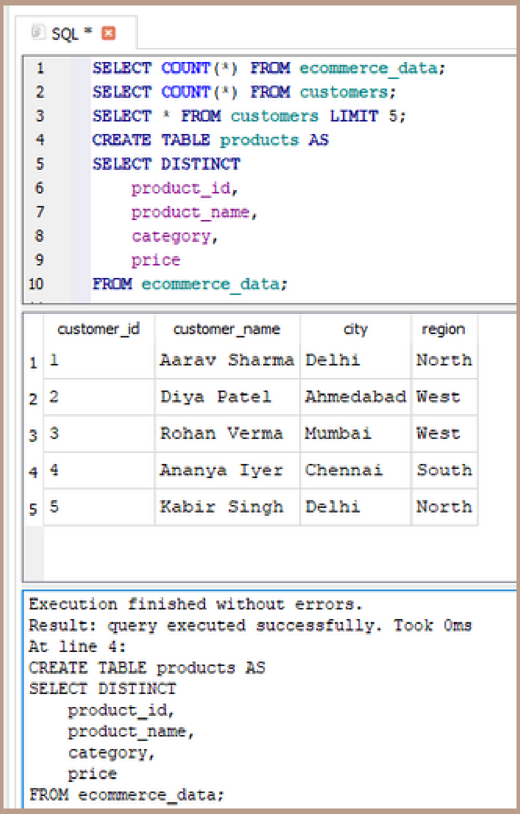
```
Execution finished without errors.
Result: 1 rows returned in 10ms
At line 2:
SELECT COUNT(*) FROM customers;
```



```sql
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
SELECT * FROM customers LIMIT 5;
```

	customer_id	customer_name	city	region
1	1	Aarav Sharma	Delhi	North
2	2	Diya Patel	Ahmedabad	West
3	3	Rohan Verma	Mumbai	West
4	4	Ananya Iyer	Chennai	South
5	5	Kabir Singh	Delhi	North

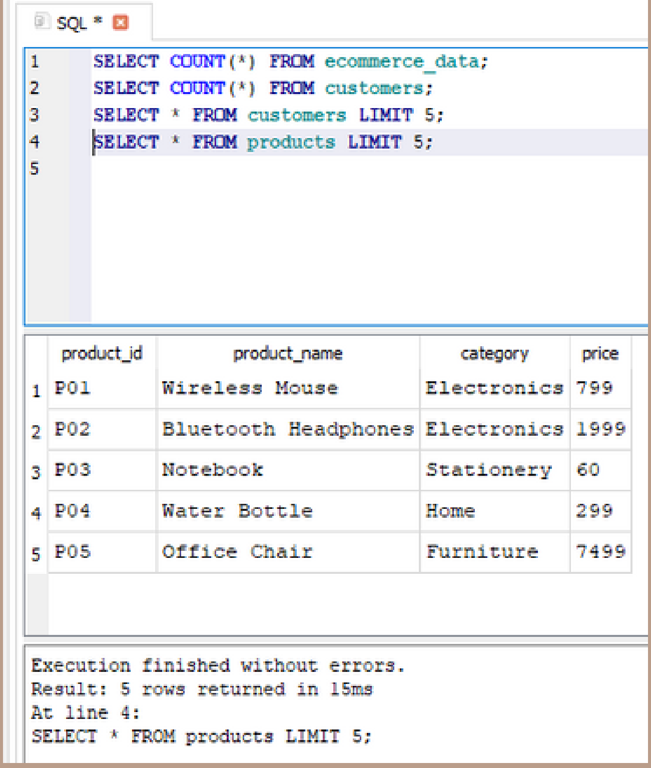
```
Execution finished without errors.
Result: 5 rows returned in 14ms
At line 3:
SELECT * FROM customers LIMIT 5;
```



```sql
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
SELECT * FROM customers LIMIT 5;
CREATE TABLE products AS
SELECT DISTINCT
    product_id,
    product_name,
    category,
    price
FROM ecommerce_data;
```

	customer_id	customer_name	city	region
1	1	Aarav Sharma	Delhi	North
2	2	Diya Patel	Ahmedabad	West
3	3	Rohan Verma	Mumbai	West
4	4	Ananya Iyer	Chennai	South
5	5	Kabir Singh	Delhi	North

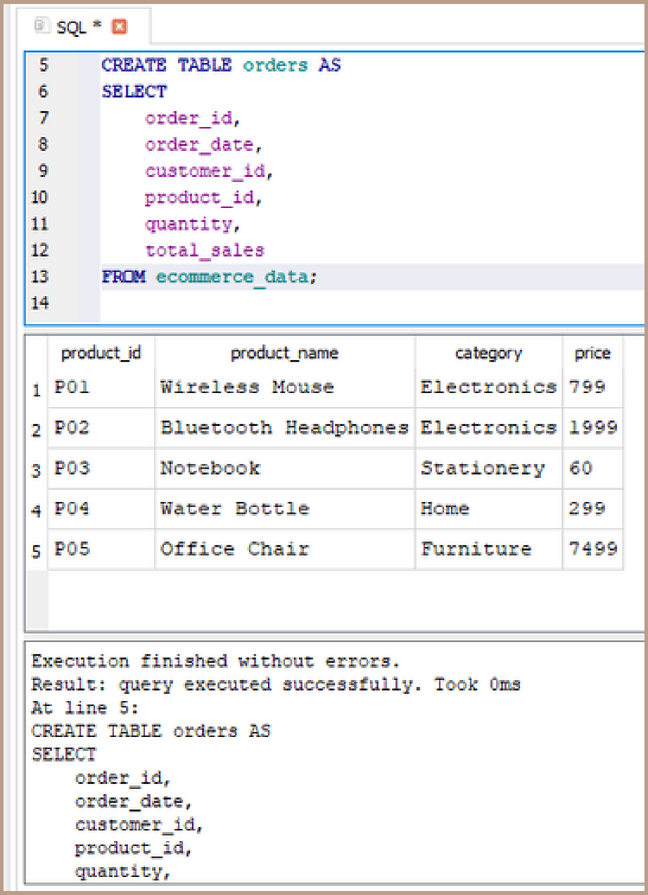
```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 4:
CREATE TABLE products AS
SELECT DISTINCT
    product_id,
    product_name,
    category,
    price
FROM ecommerce_data;
```



```
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
SELECT * FROM customers LIMIT 5;
SELECT * FROM products LIMIT 5;
```

	product_id	product_name	category	price
1	P01	Wireless Mouse	Electronics	799
2	P02	Bluetooth Headphones	Electronics	1999
3	P03	Notebook	Stationery	60
4	P04	Water Bottle	Home	299
5	P05	Office Chair	Furniture	7499

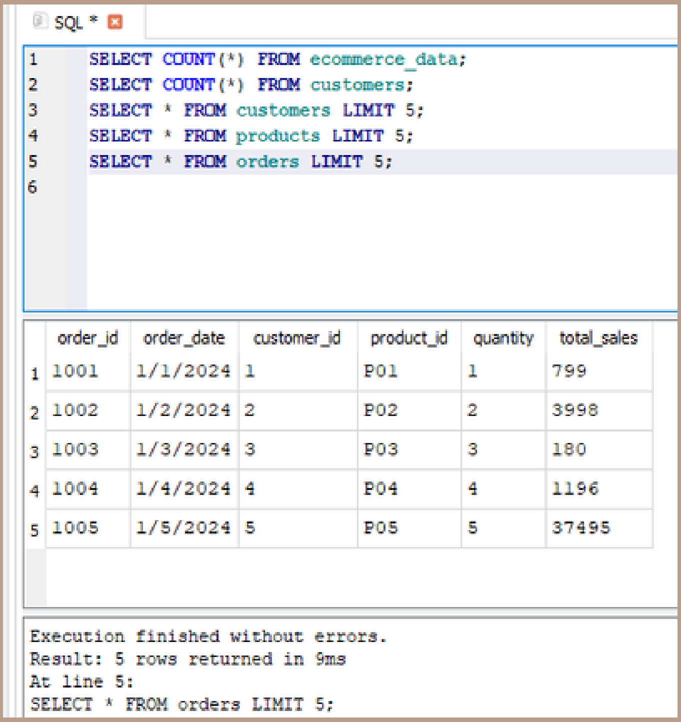
```
Execution finished without errors.
Result: 5 rows returned in 15ms
At line 4:
SELECT * FROM products LIMIT 5;
```



```
CREATE TABLE orders AS
SELECT
    order_id,
    order_date,
    customer_id,
    product_id,
    quantity,
    total_sales
FROM ecommerce_data;
```

	product_id	product_name	category	price
1	P01	Wireless Mouse	Electronics	799
2	P02	Bluetooth Headphones	Electronics	1999
3	P03	Notebook	Stationery	60
4	P04	Water Bottle	Home	299
5	P05	Office Chair	Furniture	7499

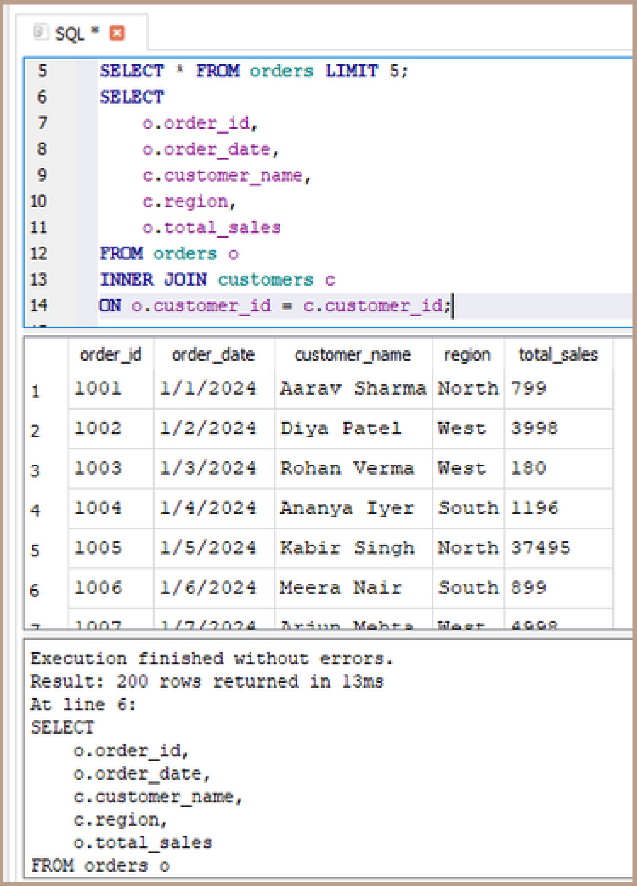
```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 5:
CREATE TABLE orders AS
SELECT
    order_id,
    order_date,
    customer_id,
    product_id,
    quantity,
```



```
SELECT COUNT(*) FROM ecommerce_data;
SELECT COUNT(*) FROM customers;
SELECT * FROM customers LIMIT 5;
SELECT * FROM products LIMIT 5;
SELECT * FROM orders LIMIT 5;
```

	order_id	order_date	customer_id	product_id	quantity	total_sales
1	1001	1/1/2024	1	P01	1	799
2	1002	1/2/2024	2	P02	2	3998
3	1003	1/3/2024	3	P03	3	180
4	1004	1/4/2024	4	P04	4	1196
5	1005	1/5/2024	5	P05	5	37495

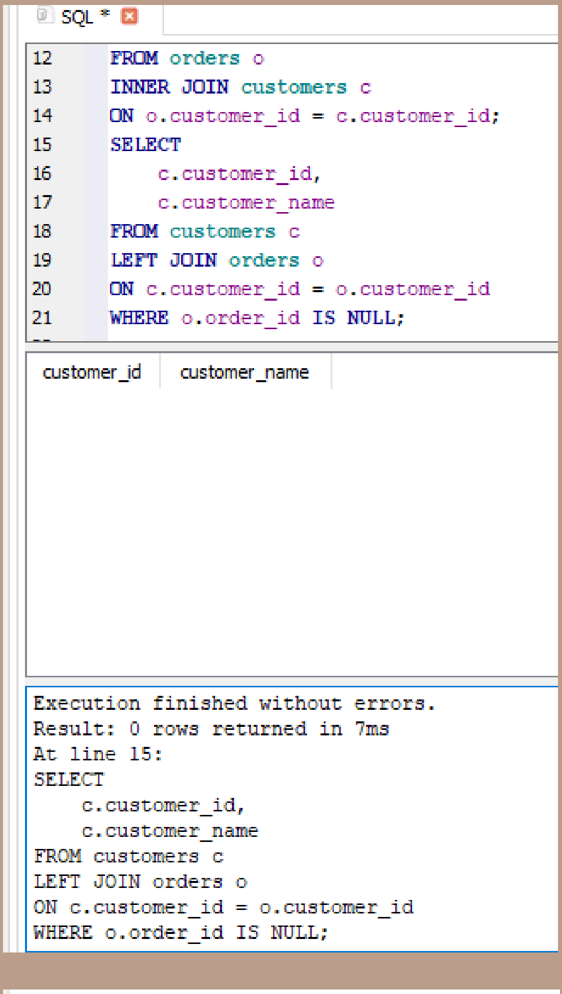
```
Execution finished without errors.
Result: 5 rows returned in 9ms
At line 5:
SELECT * FROM orders LIMIT 5;
```



```
SELECT * FROM orders LIMIT 5;
SELECT
    o.order_id,
    o.order_date,
    c.customer_name,
    c.region,
    o.total_sales
FROM orders o
INNER JOIN customers c
ON o.customer_id = c.customer_id;
```

	order_id	order_date	customer_name	region	total_sales
1	1001	1/1/2024	Aarav Sharma	North	799
2	1002	1/2/2024	Diya Patel	West	3998
3	1003	1/3/2024	Rohan Verma	West	180
4	1004	1/4/2024	Ananya Iyer	South	1196
5	1005	1/5/2024	Kabir Singh	North	37495
6	1006	1/6/2024	Meera Nair	South	899
7	1007	1/7/2024	Arjun Mehra	West	4998

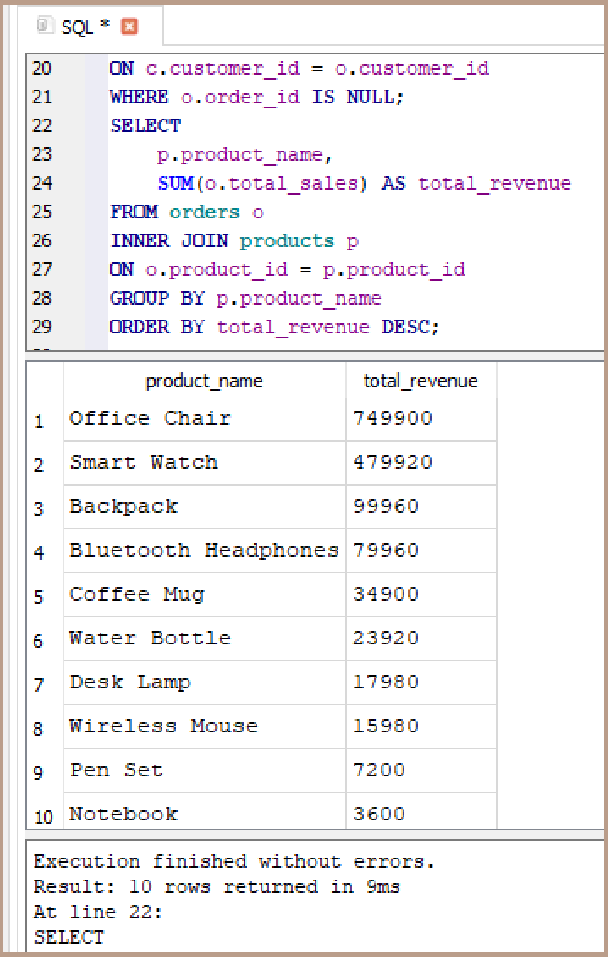
```
Execution finished without errors.
Result: 200 rows returned in 13ms
At line 6:
SELECT
    o.order_id,
    o.order_date,
    c.customer_name,
    c.region,
    o.total_sales
FROM orders o
```



```
12    FROM orders o
13    INNER JOIN customers c
14    ON o.customer_id = c.customer_id;
15    SELECT
16        c.customer_id,
17        c.customer_name
18    FROM customers c
19    LEFT JOIN orders o
20    ON c.customer_id = o.customer_id
21    WHERE o.order_id IS NULL;
```

customer_id	customer_name

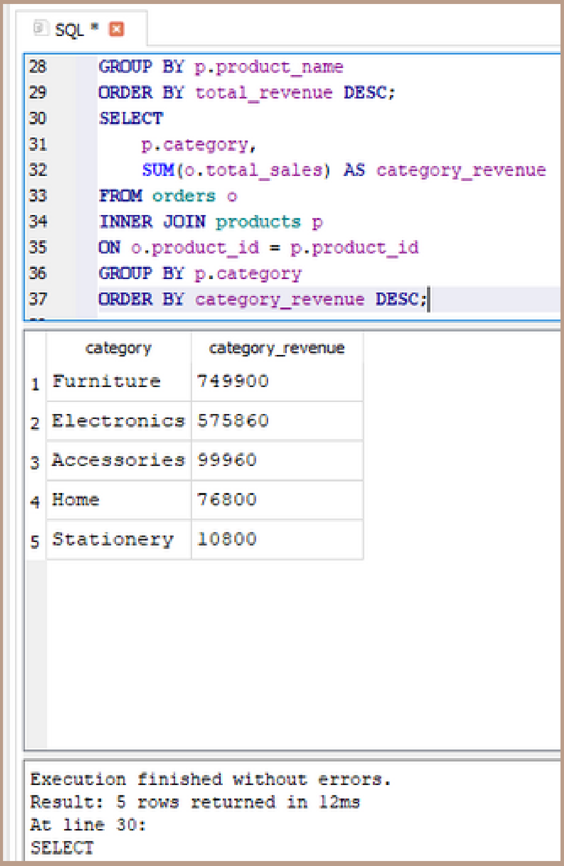
```
Execution finished without errors.
Result: 0 rows returned in 7ms
At line 15:
SELECT
    c.customer_id,
    c.customer_name
FROM customers c
LEFT JOIN orders o
ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
```



```
20    ON c.customer_id = o.customer_id
21    WHERE o.order_id IS NULL;
22    SELECT
23        p.product_name,
24        SUM(o.total_sales) AS total_revenue
25    FROM orders o
26    INNER JOIN products p
27    ON o.product_id = p.product_id
28    GROUP BY p.product_name
29    ORDER BY total_revenue DESC;
```

	product_name	total_revenue
1	Office Chair	749900
2	Smart Watch	479920
3	Backpack	99960
4	Bluetooth Headphones	79960
5	Coffee Mug	34900
6	Water Bottle	23920
7	Desk Lamp	17980
8	Wireless Mouse	15980
9	Pen Set	7200
10	Notebook	3600

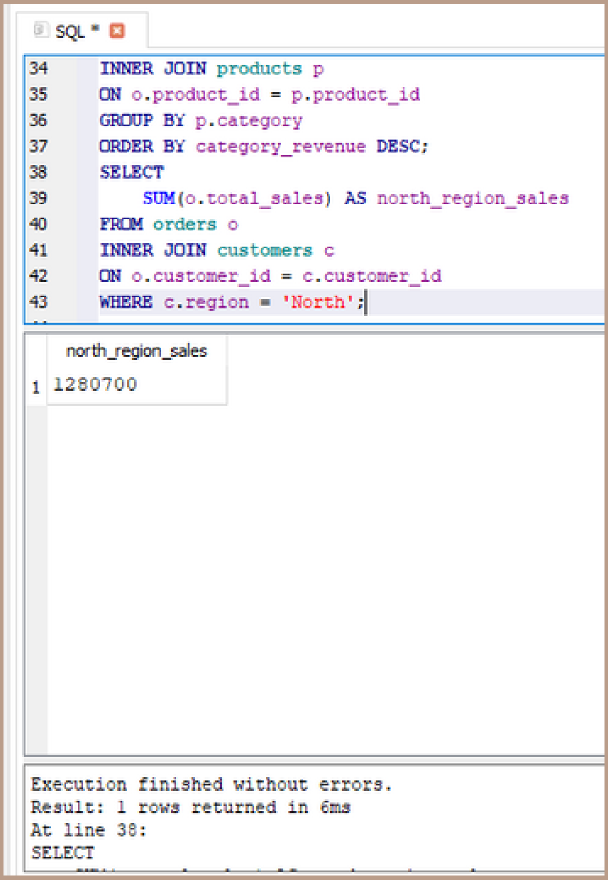
```
Execution finished without errors.
Result: 10 rows returned in 9ms
At line 22:
SELECT
```



```
28    GROUP BY p.product_name
29    ORDER BY total_revenue DESC;
30    SELECT
31        p.category,
32        SUM(o.total_sales) AS category_revenue
33    FROM orders o
34    INNER JOIN products p
35    ON o.product_id = p.product_id
36    GROUP BY p.category
37    ORDER BY category_revenue DESC;
```

	category	category_revenue
1	Furniture	749900
2	Electronics	575860
3	Accessories	99960
4	Home	76800
5	Stationery	10800

```
Execution finished without errors.
Result: 5 rows returned in 12ms
At line 30:
SELECT
```



```
34    INNER JOIN products p
35    ON o.product_id = p.product_id
36    GROUP BY p.category
37    ORDER BY category_revenue DESC;
38    SELECT
39        SUM(o.total_sales) AS north_region_sales
40    FROM orders o
41    INNER JOIN customers c
42    ON o.customer_id = c.customer_id
43    WHERE c.region = 'North';
```

	north_region_sales
1	1280700

```
Execution finished without errors.
Result: 1 rows returned in 6ms
At line 38:
SELECT
```