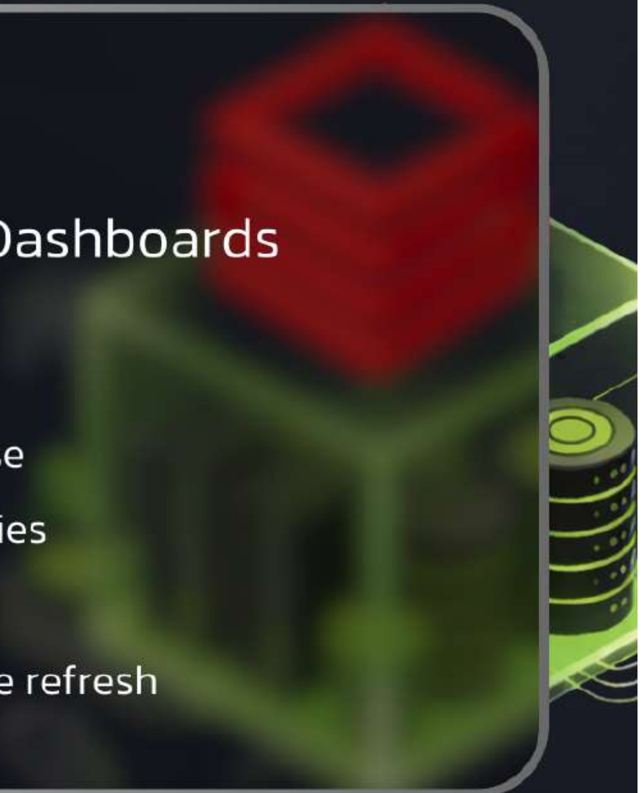
INDIAN DATA CLUB

CODE BASICS

databricks

14 DAYS AI CHALLENGE

DAY 09



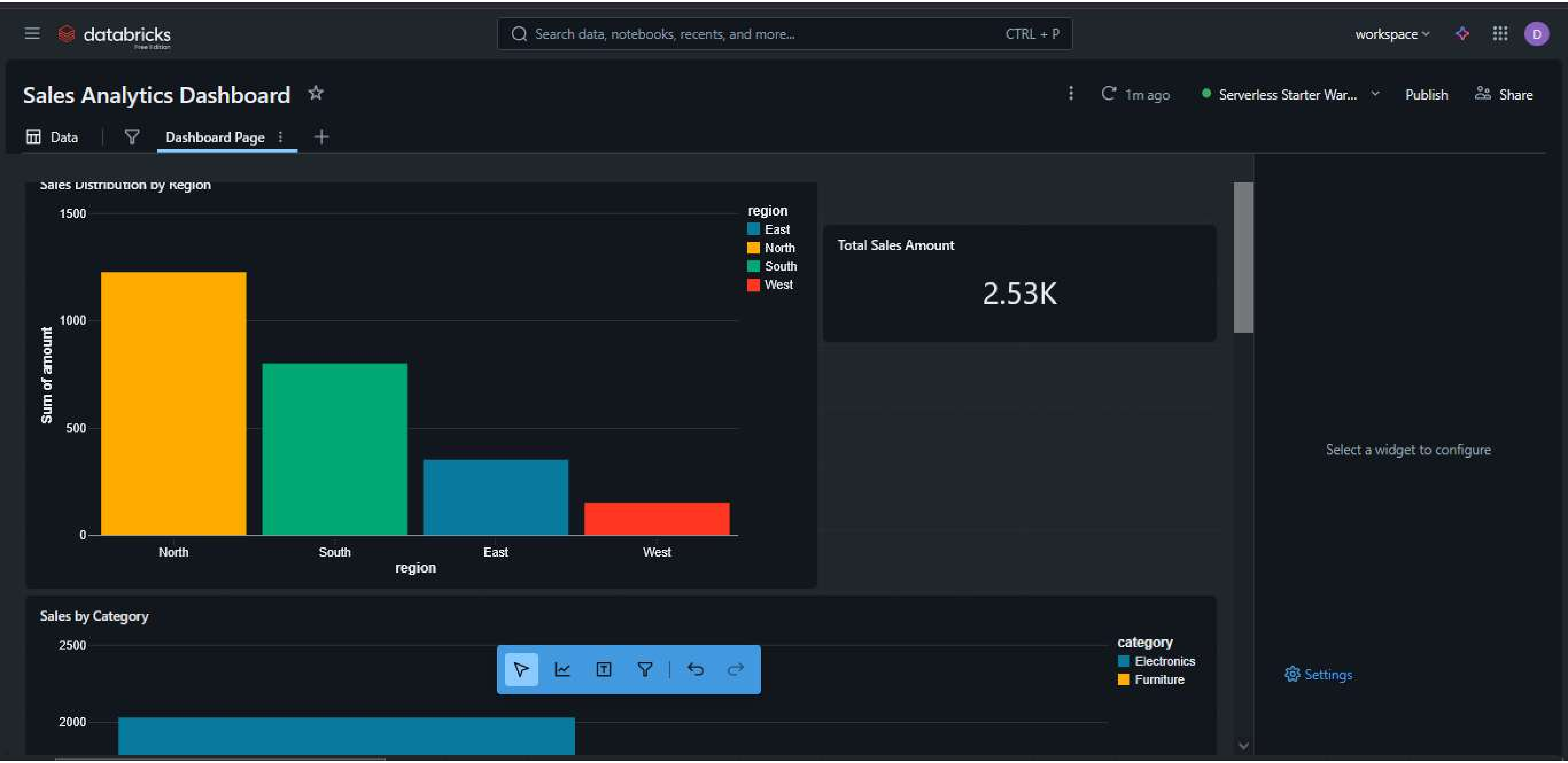
Topic:

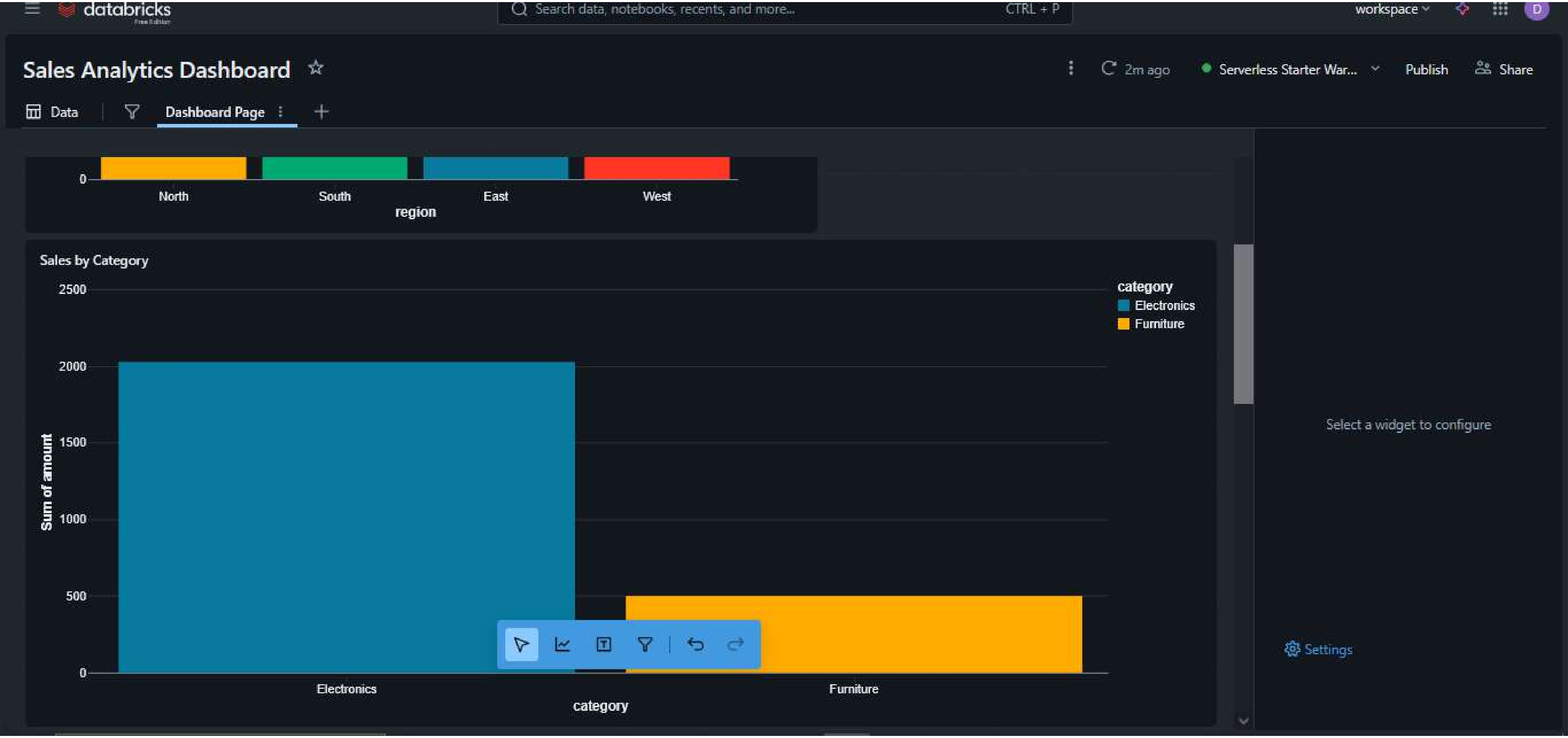
SQL Analytics & Dashboards

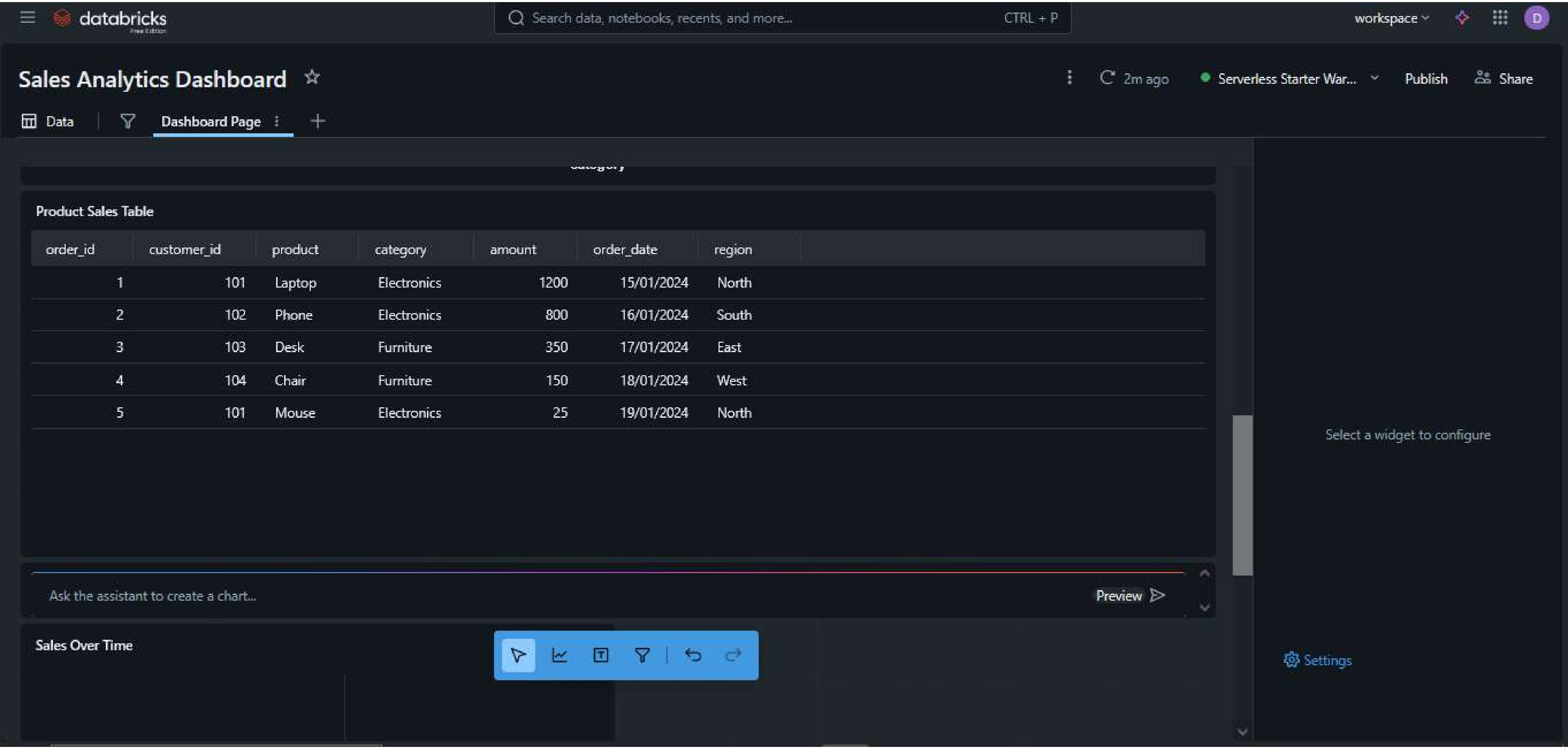
Challenge:

1. Create SQL warehouse
2. Write analytical queries
3. Build dashboard
4. Add filters & schedule refresh

#DatabricksWithIDC


databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
D
Sales Analytics Dashboard
1m ago
Serverless Starter War...
Publish
Share
Data
Dashboard Page
Sales Distribution by Region
1500
1000
500
0
Sum of amount
North
South
East
West
region
region
East
North
South
West
Total Sales Amount
2.53K
Select a widget to configure
Settings
Sales by Category
2500
2000
category
Electronics
Furniture

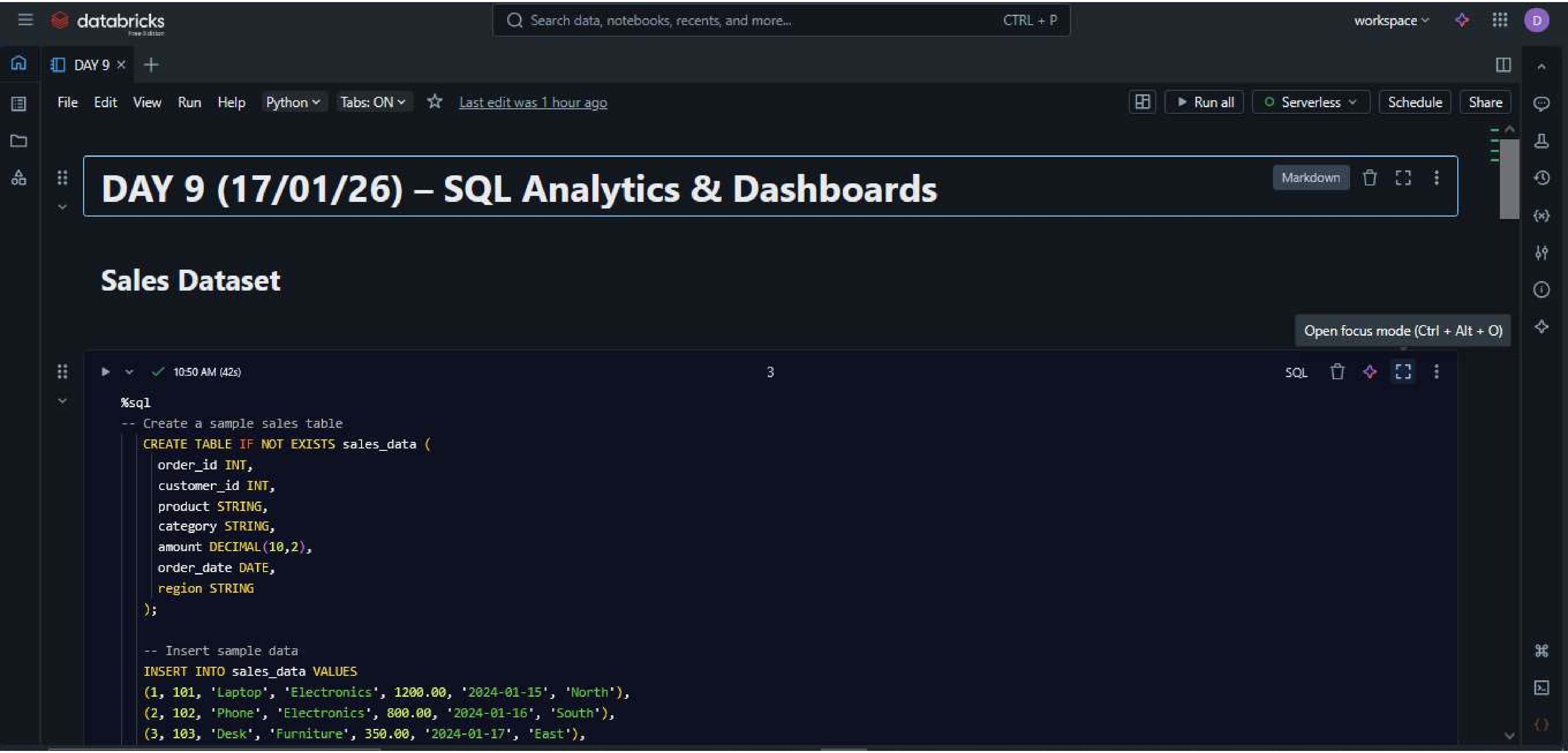

databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
D
Sales Analytics Dashboard
2m ago
Serverless Starter War...
Publish
Share
Data
Dashboard Page
0
North
South
East
West
region
Sales by Category
2500
2000
1500
1000
500
0
Sum of amount
Electronics
Furniture
category
category
Electronics
Furniture
Select a widget to configure
Settings



databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
D
Sales Analytics Dashboard
2m ago
Serverless Starter War...
Publish
Share
Data
Dashboard Page
category
Ask the assistant to create a chart...
Preview
Sales Over Time
Select a widget to configure
Settings

Product Sales Table

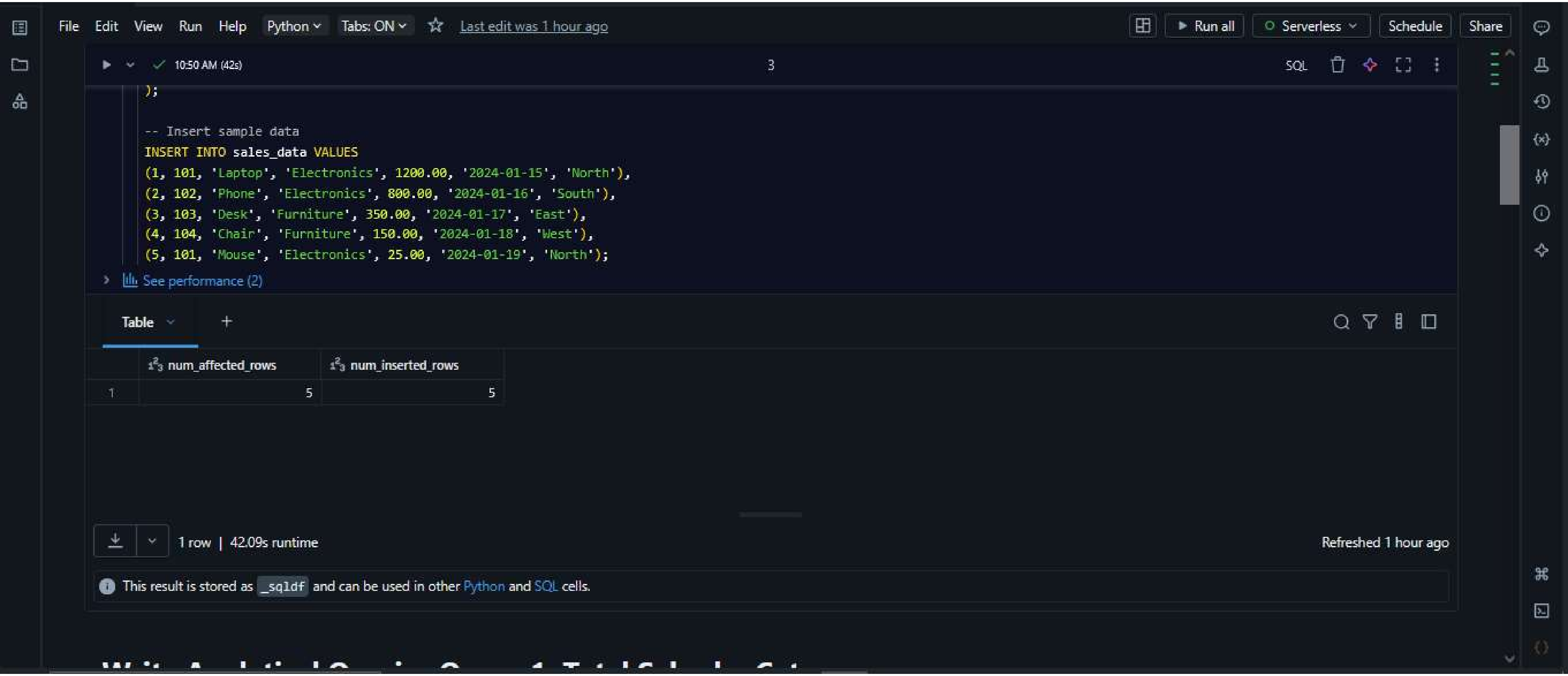
order_id	customer_id	product	category	amount	order_date	region
1	101	Laptop	Electronics	1200	15/01/2024	North
2	102	Phone	Electronics	800	16/01/2024	South
3	103	Desk	Furniture	350	17/01/2024	East
4	104	Chair	Furniture	150	18/01/2024	West
5	101	Mouse	Electronics	25	19/01/2024	North



DAY 9 (17/01/26) – SQL Analytics & Dashboards

Sales Dataset

```
%sql
-- Create a sample sales table
  CREATE TABLE IF NOT EXISTS sales_data (
    order_id INT,
    customer_id INT,
    product STRING,
    category STRING,
    amount DECIMAL(10,2),
    order_date DATE,
    region STRING
  );

  -- Insert sample data
  INSERT INTO sales_data VALUES
  (1, 101, 'Laptop', 'Electronics', 1200.00, '2024-01-15', 'North'),
  (2, 102, 'Phone', 'Electronics', 800.00, '2024-01-16', 'South'),
  (3, 103, 'Desk', 'Furniture', 350.00, '2024-01-17', 'East'),
```



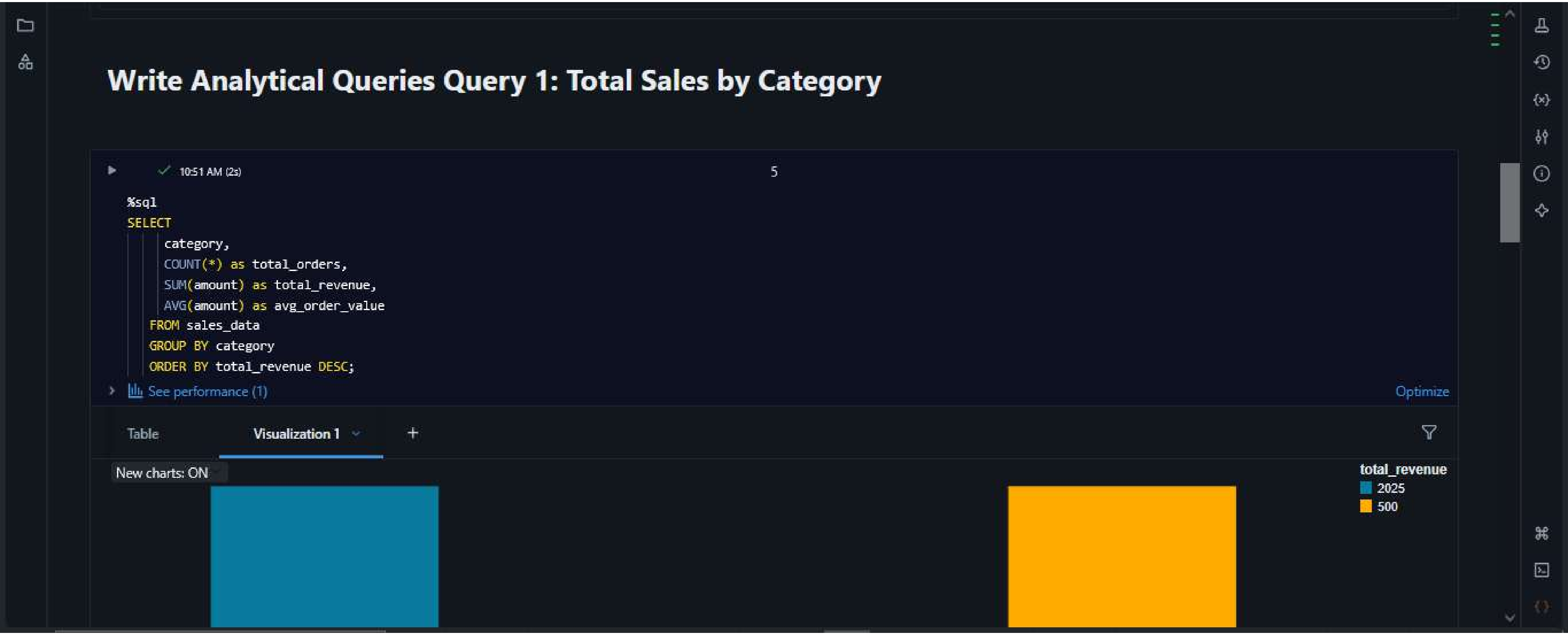
```
);

-- Insert sample data
INSERT INTO sales_data VALUES
(1, 101, 'Laptop', 'Electronics', 1200.00, '2024-01-15', 'North'),
(2, 102, 'Phone', 'Electronics', 800.00, '2024-01-16', 'South'),
(3, 103, 'Desk', 'Furniture', 350.00, '2024-01-17', 'East'),
(4, 104, 'Chair', 'Furniture', 150.00, '2024-01-18', 'West'),
(5, 101, 'Mouse', 'Electronics', 25.00, '2024-01-19', 'North');
```

	num_affected_rows	num_inserted_rows
1	5	5

1 row | 42.09s runtime

Refreshed 1 hour ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

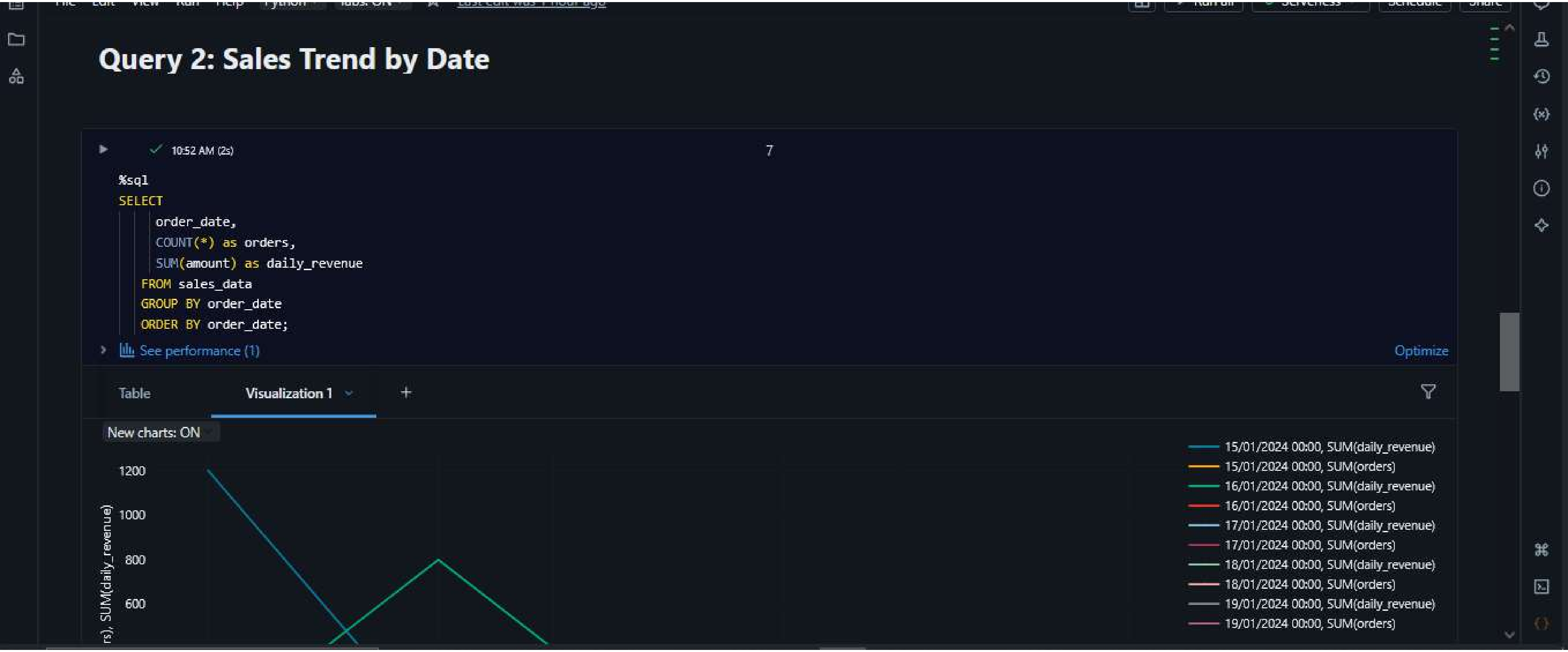


Write Analytical Queries Query 1: Total Sales by Category

10:51 AM (2s)

5

```
%sql
SELECT
    category,
    COUNT(*) as total_orders,
    SUM(amount) as total_revenue,
    AVG(amount) as avg_order_value
  FROM sales_data
  GROUP BY category
  ORDER BY total_revenue DESC;
```



File Edit View Run Help Python Tabs: ON
Last edit was 1 hour ago
Run all
Serverless
Schedule
Share

Query 2: Sales Trend by Date

10:52 AM (2s)

7

```
%sql
SELECT
    order_date,
    COUNT(*) as orders,
    SUM(amount) as daily_revenue
  FROM sales_data
  GROUP BY order_date
  ORDER BY order_date;
```

See performance (1)
Optimize
Table
Visualization 1
+
New charts: ON
1200
1000
800
600
rs), SUM(daily_revenue)
15/01/2024 00:00, SUM(daily_revenue)
15/01/2024 00:00, SUM(orders)
16/01/2024 00:00, SUM(daily_revenue)
16/01/2024 00:00, SUM(orders)
17/01/2024 00:00, SUM(daily_revenue)
17/01/2024 00:00, SUM(orders)
18/01/2024 00:00, SUM(daily_revenue)
18/01/2024 00:00, SUM(orders)
19/01/2024 00:00, SUM(daily_revenue)
19/01/2024 00:00, SUM(orders)

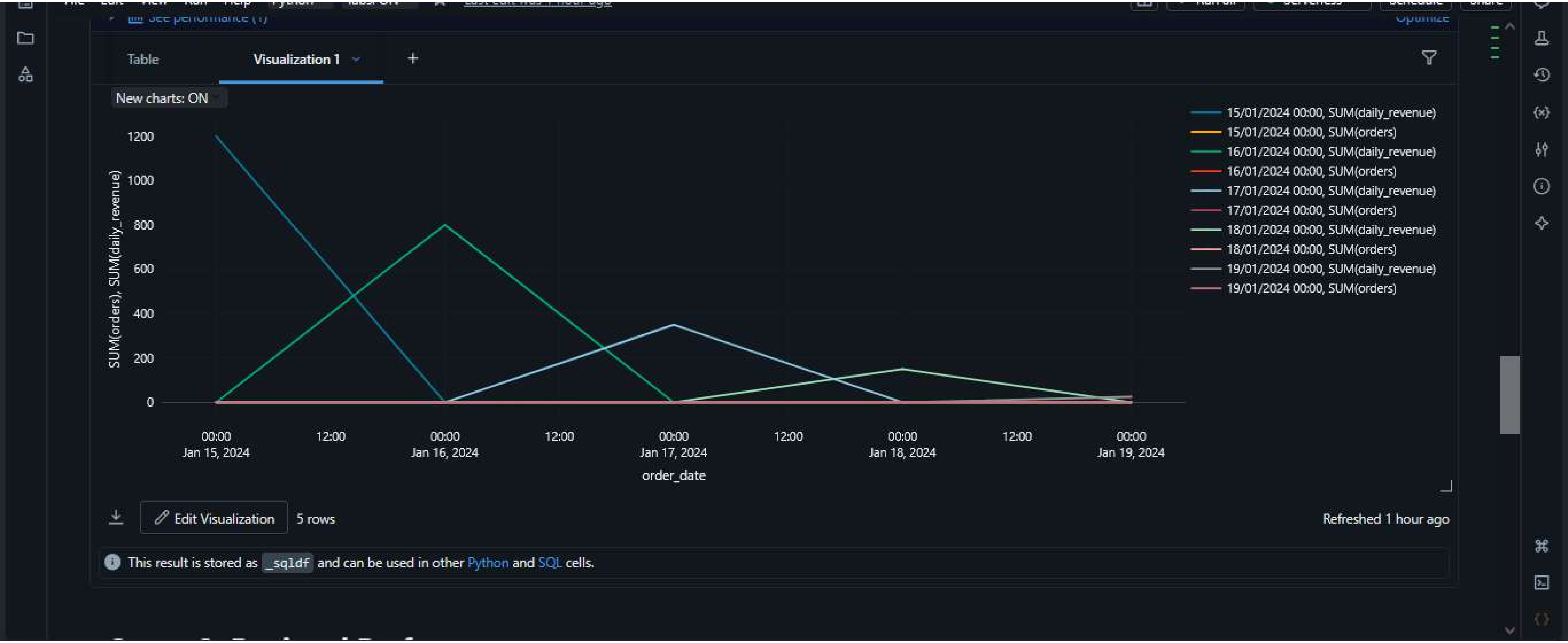


Table
Visualization 1
+
New charts: ON
1200
1000
800
600
400
200
0
SUM(orders), SUM(daily_revenue)
00:00 Jan 15, 2024
12:00
00:00 Jan 16, 2024
12:00
00:00 Jan 17, 2024
12:00
00:00 Jan 18, 2024
12:00
00:00 Jan 19, 2024
order_date
15/01/2024 00:00, SUM(daily_revenue)
15/01/2024 00:00, SUM(orders)
16/01/2024 00:00, SUM(daily_revenue)
16/01/2024 00:00, SUM(orders)
17/01/2024 00:00, SUM(daily_revenue)
17/01/2024 00:00, SUM(orders)
18/01/2024 00:00, SUM(daily_revenue)
18/01/2024 00:00, SUM(orders)
19/01/2024 00:00, SUM(daily_revenue)
19/01/2024 00:00, SUM(orders)
Edit Visualization

5 rows

Refreshed 1 hour ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

Day 9: SQL Analytics & Dashboards Challenge

1. Create SQL warehouse

Set up the foundational compute resource for running SQL queries.

2. Write analytical queries

Develop SQL queries to extract insights from your data.

3. Build dashboard

Visualize the results of your queries in a new dashboard.

4. Add filters & schedule refresh

Enhance interactivity and ensure your data stays current.

NotebookLM