Case Study – Azure

Bavatharani S

Exploratory Data Analysis (EDA) on Retail Sales Data

Since Databricks Community Edition doesn't support direct Azure Data Lake Storage (ADLS) mounting, I simulated the ADLS step by manually uploading a dataset. The workflow demonstrates EDA and Delta table queries in Databricks.

Step 1: Prepare a dataset (CSV file)

sales.csv

```
order_id,product,quantity,price
101,Laptop,2,75000
102,Mobile,5,15000
103,Headphones,10,2500
104,Tablet,3,30000
```

Step 2: Upload dataset to Databricks Community Edition

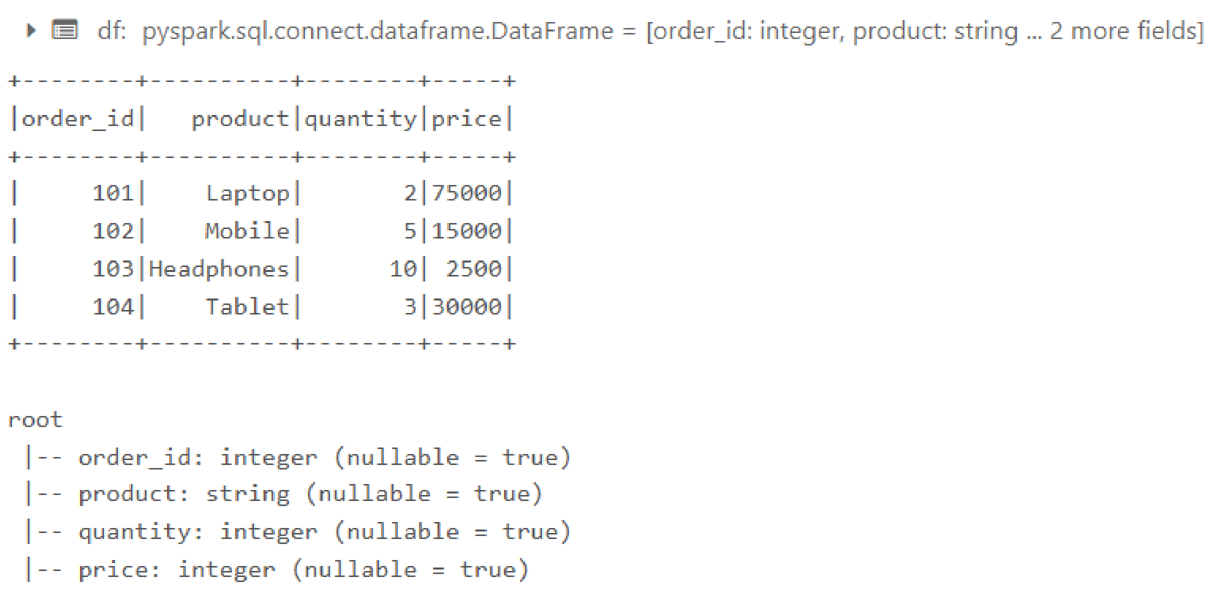
- Login: community.cloud.databricks.com
- Navigate → Data tab (left sidebar)
- Click Add Data → Upload File
- Select sales.csv and upload

Step 3: Create Notebook & Start Cluster

- Go to Workspace > User Folder
- Click Create > Notebook
- Name: Retail_EDA
- Language: Python
- Attach to a cluster

Step 4: Read CSV into Spark DataFrame

```
df = spark.read.format("csv").option("header", "true").load("/FileStore/tables/sales.csv")
df.show()
```



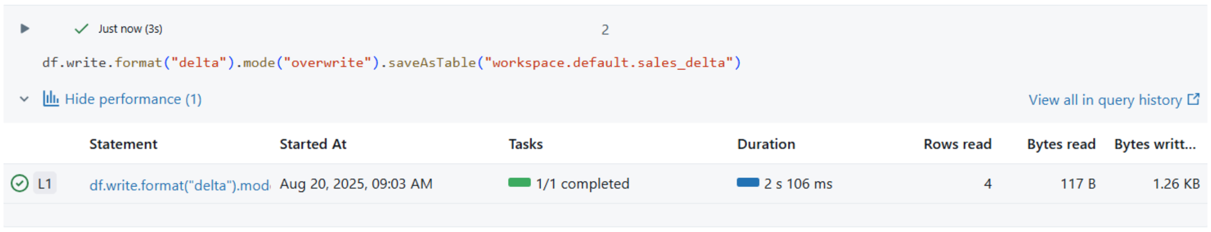
```
df:  pyspark.sql.connect.dataframe.DataFrame = [order_id: integer, product: string ... 2 more fields]
```

```
+--------+----------+--------+-----+
|order_id|   product|quantity|price|
+--------+----------+--------+-----+
|     101|    Laptop|       2|75000|
|     102|    Mobile|       5|15000|
|     103|Headphones|      10| 2500|
|     104|    Tablet|       3|30000|
+--------+----------+--------+-----+

root
 |-- order_id: integer (nullable = true)
 |-- product: string (nullable = true)
 |-- quantity: integer (nullable = true)
 |-- price: integer (nullable = true)
```

Step 5: Save DataFrame as Delta Table

df.write.format("delta").mode("overwrite").saveAsTable("sales_delta")



Just now (3s) 2

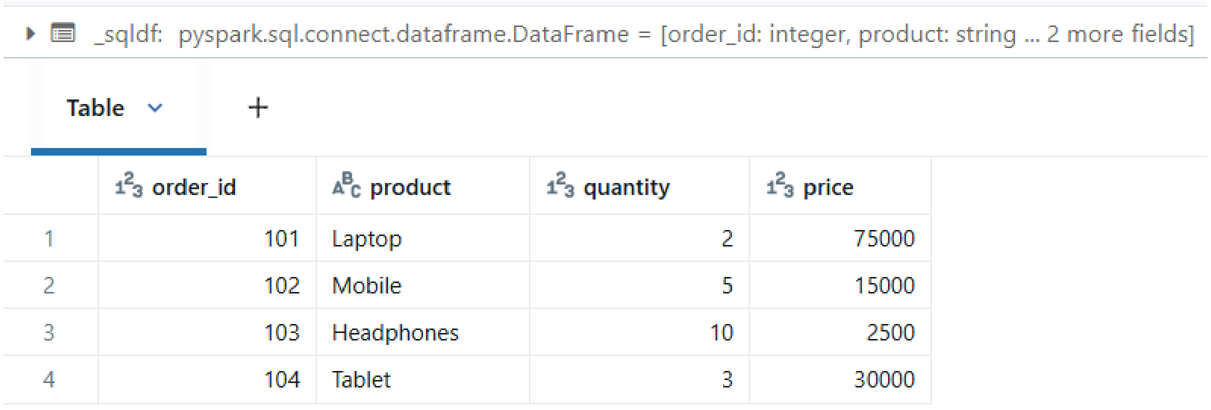
```
df.write.format("delta").mode("overwrite").saveAsTable("workspace.default.sales_delta")
```

	Statement	Started At	Tasks	Duration	Rows read	Bytes read	Bytes writt...
✓ L1	df.write.format("delta").mod	Aug 20, 2025, 09:03 AM	🟩 1/1 completed	🟦 2 s 106 ms	4	117 B	1.26 KB

Step 6: Run EDA Queries on Delta Table

-- Show all products

SELECT * FROM sales_delta;

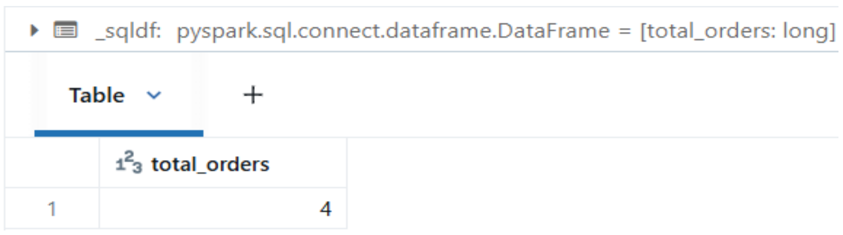


```
_sqldf:  pyspark.sql.connect.dataframe.DataFrame = [order_id: integer, product: string ... 2 more fields]
```

	order_id	product	quantity	price
1	101	Laptop	2	75000
2	102	Mobile	5	15000
3	103	Headphones	10	2500
4	104	Tablet	3	30000

-- Count total orders

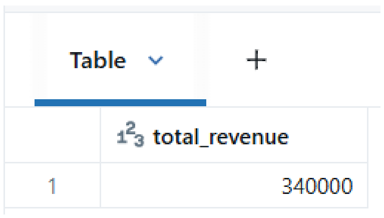
SELECT COUNT(*) AS total_orders FROM sales_delta;


_sqldf: pyspark.sql.connect.dataframe.DataFrame = [total_orders: long]

	total_orders
1	4

-- Find total revenue

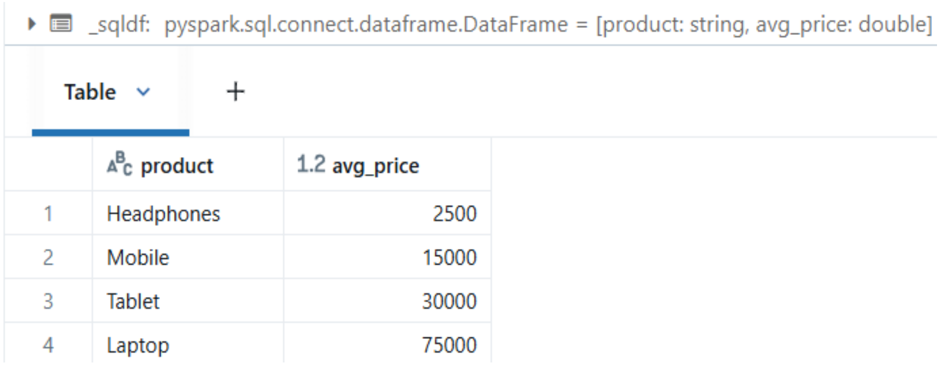
SELECT SUM(CAST(quantity AS INT) * CAST(price AS INT)) AS total_revenue FROM sales_delta;



	total_revenue
1	340000

-- Average price per product

SELECT product, AVG(CAST(price AS INT)) AS avg_price FROM sales_delta GROUP BY product;


_sqldf: pyspark.sql.connect.dataframe.DataFrame = [product: string, avg_price: double]

	product	avg_price
1	Headphones	2500
2	Mobile	15000
3	Tablet	30000
4	Laptop	75000

Conclusion

This case study showcased how to perform **EDA on a retail sales dataset** using Databricks Community Edition by simulating ADLS ingestion. The dataset was uploaded manually, processed into a Delta table, and analyzed using SQL queries. The analysis provided insights such as total revenue, product-level averages, and order counts. Even without direct ADLS integration, Databricks enabled efficient data exploration with Delta Lake.