
Exercise: E-Commerce Analytics with PySpark in Azure Databricks

---

Objective

You are a Data Engineer at an online retail company. You've been asked to analyze orders, products, and customer data to generate insights for the business.

You'll work entirely in **Azure Databricks (PySpark)**.

Step 1 – Setup Your Notebook

Create a new **Notebook** in Databricks → Select **Python** → Attach to a running cluster.

At the top of the notebook, set up the Spark Session (Databricks provides this by default, but keep it for consistency):

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("ECommerceAnalysis").getOrCreate()
```

Step 2 – Create and Upload Data

a) Create three CSV files on your local machine (or directly in Databricks > DBFS > FileStore > tables):

customers.csv

```
customer_id,name,city,age
1,Rahul Sharma,Bangalore,28
2,Priya Singh,Delhi,32
3,Aman Kumar,Hyderabad,25
4,Sneha Reddy,Chennai,35
5,Arjun Mehta,Mumbai,30
6,Divya Nair,Delhi,29
```

products.csv

```
product_id,product_name,category,price
101,Laptop,Electronics,55000
102,Mobile,Electronics,25000
103,Headphones,Electronics,3000
104,Chair,Furniture,5000
105,Book,Stationery,700
106,Shoes,Fashion,2500
```

orders.csv

```
order_id,customer_id,product_id,quantity,order_date
1001,1,101,1,2024-01-10
1002,2,102,2,2024-01-12
1003,1,103,3,2024-02-05
1004,3,104,1,2024-02-08
1005,5,105,5,2024-03-01
1006,6,106,2,2024-03-15
1007,7,101,1,2024-03-20
```

Step 3 – Load the Files into DataFrames

```
customers_df = spark.read.csv("/FileStore/tables/customers.csv", header=True,
inferSchema=True)
products_df  = spark.read.csv("/FileStore/tables/products.csv",  header=True,
inferSchema=True)
orders_df    = spark.read.csv("/FileStore/tables/orders.csv",    header=True,
inferSchema=True)
```

Check them:

```
display(customers_df)
display(products_df)
display(orders_df)
```

Step 4 – Perform Transformations

1. **Add Total Price column** to orders → `quantity * price`

   ```
   joined_df = orders_df.join(products_df, "product_id")
   orders_total_df = joined_df.withColumn("total_price", joined_df.quantity *
   joined_df.price)
   display(orders_total_df)
   ```

2. **Filter orders** above 40 000.
3. **Extract month** from `order_date`.
4. **Sort by total_price** descending.

Step 5 – Aggregations

1. Find **total revenue per city**. *(Hint – join orders → customers → products)*
2. Find **average age of customers per city**.
3. Find **total revenue per category**.
4. Find the **top 3 customers** by total spending.

Step 6 – Joins

1. Perform:
    - Inner Join (orders + customers)
    - Left Join (customers + orders)
    - Right Join (orders + products)
2. Identify:
    - Customers with **no orders**
    - Products that were **never ordered**

Step 7 – Use PySpark SQL

```
customers_df.createOrReplaceTempView("customers")
products_df.createOrReplaceTempView("products")
orders_total_df.createOrReplaceTempView("orders")
```

Now run:

1. **Top 2 cities by total revenue**
2. **Most popular category** by revenue
3. **Customers who spent > 50 000**
4. **Monthly sales trend** (using `month(order_date)`)

Step 8 – File Operations

1. Save `orders_total_df` to DBFS:

    ```
    orders_total_df.write.mode("overwrite").csv("/FileStore/tables/orders_summary")
    ```

2. Verify by listing the folder:

    ```
    display(dbutils.fs.ls("/FileStore/tables/orders_summary"))
    ```

3. Read back into a new DataFrame:

    ```
    summary_df = spark.read.csv("/FileStore/tables/orders_summary", header=True,
    inferSchema=True)
    display(summary_df)
    ```

Step 9 – Visualization (Optional)

Convert to Pandas and plot using Matplotlib/Seaborn.

```
import pandas as pd
import matplotlib.pyplot as plt

region_df = orders_total_df.groupBy("category").sum("total_price").toPandas()
```

```
plt.bar(region_df["category"], region_df["sum(total_price)"], color="skyblue")
plt.title("Total Sales by Category")
plt.xlabel("Category")
plt.ylabel("Revenue (□)")
plt.show()
```
