### \*\*Exercise: Analyzing a Sample Sales Dataset Using PySpark\*\*

In this exercise, you'll work with a simulated sales dataset and perform various data transformations and analyses using PySpark. The dataset includes fields like `TransactionID`, `CustomerID`, `ProductID`, `Quantity`, `Price`, and `Date`. Your task is to generate the dataset, load it into PySpark, and answer specific questions by performing data operations.

### \*\*Part 1: Dataset Preparation\*\*

#### \*\*Step 1: Generate the Sample Sales Dataset\*\*

Before starting the analysis, you'll need to create the sample sales dataset. Use the following Python code to generate the dataset and save it as a CSV file.

1. \*\*Run the Dataset Preparation Script:\*\*

```python

import pandas as pd

from datetime import datetime

# Sample sales data

data = {

"TransactionID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

"CustomerID": [101, 102, 103, 101, 104, 102, 103, 104, 101, 105],

"ProductID": [501, 502, 501, 503, 504, 502, 503, 504, 501, 505],

"Quantity": [2, 1, 4, 3, 1, 2, 5, 1, 2, 1],

"Price": [150.0, 250.0, 150.0, 300.0, 450.0, 250.0, 300.0, 450.0, 150.0, 550.0],

"Date": [

datetime(2024, 9, 1),

datetime(2024, 9, 1),

datetime(2024, 9, 2),

datetime(2024, 9, 2),

datetime(2024, 9, 3),

datetime(2024, 9, 3),

datetime(2024, 9, 4),

datetime(2024, 9, 4),

datetime(2024, 9, 5),

datetime(2024, 9, 5)

]

}

# Create a DataFrame

df = pd.DataFrame(data)

# Save the DataFrame to a CSV file

df.to\_csv('sales\_data.csv', index=False)

print("Sample sales dataset has been created and saved as 'sales\_data.csv'.")

```

2. \*\*Verify the Dataset:\*\*

- After running the script, ensure that the file `sales\_data.csv` has been created in your working directory.

---

### \*\*Part 2: Load and Analyze the Dataset Using PySpark\*\*

Now that you have the dataset, your task is to load it into PySpark and perform the following analysis tasks.

#### \*\*Step 2: Load the Dataset into PySpark\*\*

1. \*\*Initialize the SparkSession:\*\*

- Create a Spark session named `"Sales Dataset Analysis"`.

from pyspark.sql import SparkSession

from pyspark.sql.functions import \*

spark=SparkSession.builder\

.appName("Sales DataSet Analysis")\

.getOrCreate()

2. \*\*Load the CSV File into a PySpark DataFrame:\*\*

- Load the `sales\_data.csv` file into a PySpark DataFrame.

df=spark.read.csv("sales\_data.csv",header=True,inferSchema=True)

- Display the first few rows of the DataFrame to verify that the data is loaded correctly.

df.show()

#### \*\*Step 3: Explore the Data\*\*

Explore the data to understand its structure.

1. \*\*Print the Schema:\*\*

- Display the schema of the DataFrame to understand the data types.

df.printSchema()

2. \*\*Show the First Few Rows:\*\*

- Display the first 5 rows of the DataFrame.

df.show(5)

3. \*\*Get Summary Statistics:\*\*

- Get summary statistics for numeric columns (`Quantity` and `Price`).

df.describe("Quantity","Price").show()

#### \*\*Step 4: Perform Data Transformations and Analysis\*\*

Perform the following tasks to analyze the data:

1. \*\*Calculate the Total Sales Value for Each Transaction:\*\*

- Add a new column called `TotalSales`, calculated by multiplying `Quantity` by `Price`.

Total\_sales=df.withColumn("Total Cost",col("Quantity")\*col("Price"))

Total\_sales.show()

2. \*\*Group By ProductID and Calculate Total Sales Per Product:\*\*

- Group the data by `ProductID` and calculate the total sales for each product.

total\_sales\_per\_product = Total\_sales.groupBy("ProductID").sum("Total Cost").withColumnRenamed("sum(Total Cost)", "Total Cost")

total\_sales\_per\_product.show()

3. \*\*Identify the Top-Selling Product:\*\*

- Find the product that generated the highest total sales.

best\_seller=total\_sales\_per\_product.orderBy(desc("Total Cost")).first()

print(best\_seller)

4. \*\*Calculate the Total Sales by Date:\*\*

- Group the data by `Date` and calculate the total sales for each day.

total\_sales\_by\_date=Total\_sales.groupBy("Date").sum("Total Cost").withColumnRenamed("sum(Total Cost)","Total Cost")

total\_sales\_by\_date.show()

5. \*\*Filter High-Value Transactions:\*\*

- Filter the transactions to show only those where the total sales value is greater than ₹500.

high\_value\_transactions = df\_with\_total.filter(col("TotalSales") > 500)

high\_value\_transactions.show()