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(df)

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

**#1. \*\*Initialize the SparkSession:\*\*-Create a Spark session named `"Sales Dataset Analysis"`.**

from pyspark.sql import SparkSession

# Initialize SparkSession

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.**

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

sales\_df = spark.read.csv('sales\_data.csv', header=True, inferSchema=True)

sales\_df.show(5)

**#### \*\*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.**

sales\_df.printSchema()

**#2. \*\*Show the First Few Rows:\*\* - Display the first 5 rows of the DataFrame**.

sales\_df.show(5)

**#3. \*\*Get Summary Statistics:\*\*- Get summary statistics for numeric columns (`Quantity` and `Price`).**

sales\_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`.**

sales\_df = sales\_df.withColumn("TotalSales", sales\_df["Quantity"] \* sales\_df["Price"])

sales\_df.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 = sales\_df.groupBy("ProductID").sum("TotalSales")

total\_sales\_per\_product.show()

**#3. \*\*Identify the Top-Selling Product:\*\*- Find the product that generated the highest total sales.**

top\_selling\_product = total\_sales\_per\_product.orderBy("sum(TotalSales)", ascending=False).first()

print(top\_selling\_product)

**#4. \*\*Calculate the Total Sales by Date:\*\*- Group the data by `Date` and calculate the total sales for each day.**

total\_sales\_per\_date = sales\_df.groupBy("Date").sum("TotalSales")

total\_sales\_per\_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 = sales\_df.filter(sales\_df["TotalSales"] > 500)

high\_value\_transactions.show()

**### \*\*Additional Challenge (Optional):\*\***

**#1. \*\*Identify Repeat Customers:\*\*- Count how many times each customer has made a purchase and display the customers who have made more than one purchase.**

repeat\_customers = sales\_df.groupBy("CustomerID").count().filter("count > 1")

repeat\_customers.show()

**#2. \*\*Calculate the Average Sale Price Per Product:\*\*- Calculate the average price per unit for each product and display the results.**

avg\_price\_per\_product = sales\_df.groupBy("ProductID").avg("Price")

avg\_price\_per\_product.show()