PYSPARK EXERCISES 2 (3.9.2024)

**Dataset Preparation\*\***

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

import pandas as pd

from datetime import datetime

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'.")

from pyspark.sql import SparkSession

**#1. Initialize SparkSession**

spark = SparkSession.builder \

    .appName("Sales Dataset Analysis") \

    .getOrCreate()

**# 2.Load the CSV file into a PySpark DataFrame**

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

**# Display the first few rows of the DataFrame**

print("Sales DataFrame:")

sales\_df.show()

**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.**

print("Schema of the DataFrame:")

sales\_df.printSchema()

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

**# Display the first 5 rows of the DataFrame.**

print("First 5 rows of the DataFrame:")

sales\_df.show(5)

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

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

print("Summary Statistics for Numeric Columns:")

sales\_df.describe("Quantity", "Price").show()

**4. Perform Data Transformations and Analysis\*\***

**Perform the following tasks to analyze the data:**

from pyspark.sql.functions import col, sum as spark\_sum

**#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", col("Quantity") \* col("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\_df = sales\_df.groupBy("ProductID") \

                                      .agg(spark\_sum("TotalSales").alias("TotalSales"))

print("Total Sales Per Product:")

total\_sales\_per\_product\_df.show()

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

**# Find the product that generated the highest total sales.**

top\_selling\_product\_df = total\_sales\_per\_product\_df.orderBy(col("TotalSales").desc()).limit(1)

print("Top-Selling Product:")

top\_selling\_product\_df.show()

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

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

total\_sales\_by\_date\_df = sales\_df.groupBy("Date") \

                                 .agg(spark\_sum("TotalSales").alias("TotalSales"))

print("Total Sales by Date:")

total\_sales\_by\_date\_df.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 = sales\_df.filter(col("TotalSales") > 500)

print("High-Value Transactions:")

high\_value\_transactions\_df.show()

**#Additional Exercises**

**#\*Identify Repeat Customers:\*\***

from pyspark.sql.functions import avg

**#Count how many times each customer has made a purchase and display the customers who have made more than one purchase**.

repeat\_customers\_df = sales\_df.groupBy("CustomerID") \

                              .count() \

                              .filter(col("count") > 1) \

                              .withColumnRenamed("count", "PurchaseCount")

print("Repeat Customers:")

repeat\_customers\_df.show()

**#Calculate the Average Sale Price Per Product:\*\***

**#Calculate the average price per unit for each product and display the results**.

sales\_df.groupBy("ProductID") \

        .agg(avg("Price").alias("AveragePrice")) \

        .show()