ASSINGMENT -6

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
~~~~

Program:

~~~~
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)
~~~~

```
      ]
   }
   df = pd.DataFrame(data)
   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**

Program:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("Sales Dataset Analysis") \
    .getOrCreate()
```

2.1. **Initialize the SparkSession:**

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

```
  # Load CSV into DataFrame
```

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.

Program:

```
sales_df = spark.read.csv('sales_data.csv', header=True, inferSchema=True)
sales_df.show()
sales_df.printSchema()
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.

Program:

```
sales_df.printSchema()
```

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

   - Display the first 5 rows of the DataFrame.

Program:

```
sales_df.show(5)
```

3. **Get Summary Statistics:**

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

Program:

```
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`.

Program:

```
from pyspark.sql.functions import col
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.

Program:

```
sales_per_product = sales_df.groupBy('ProductID').sum('TotalSales')
sales_per_product.show()
```

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

   - Find the product that generated the highest total sales.

Program:

```
top_product = sales_per_product.orderBy(col('sum(TotalSales)').desc()).limit(1)
top_product.show()
```

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

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

Program:

```
sales_by_date = sales_df.groupBy('Date').sum('TotalSales')
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.

Program:

```
high_value_transactions = sales_df.filter(col('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.

Program:

```
repeat_customers = sales_df.groupBy('CustomerID').count().filter(col('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.

Program:

```
avg_price_per_product = sales_df.groupBy('ProductID').avg('Price')
avg_price_per_product.show()
```