**Assignment 7 - Pyspark DataFrame Sales Analysis**

**Dataset Description**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| ORDERNUMBER | Integer | A unique identifier for each order placed. Used for tracking and managing sales orders. |
| QUANTITYORDERED | Integer | The quantity of the product ordered in the specific transaction. |
| PRICEEACH | Float | The price per unit of the product in the order. |
| ORDERDATE | Date | The date when the order was placed. Used to track the timing of sales transactions. |
| STATUS | String | The status of the order (e.g., Shipped, Cancelled, In Process). |
| CUSTOMERNAME | String | The name of the customer placing the order. |
| PHONE | String | The phone number of the customer. |
| ADDRESSLINE1 | String | The primary address line of the customer. |
| ADDRESSLINE2 | String | The secondary address line of the customer (optional). |
| CITY | String | The city where the customer is located. |
| STATE | String | The state or region where the customer is located (if applicable). |
| POSTALCODE | String | The postal or ZIP code of the customer's address. |
| COUNTRY | String | The country where the customer is located. |
| SALES | Float | The total sales amount for the transaction (calculated as QUANTITYORDERED \* PRICEEACH). |
| DEALSIZE | String | The size of the deal based on total sales (e.g., Small, Medium, Large). |
| PRODUCTCODE | String | A unique identifier for the product sold. |
| PRODUCTLINE | String | The product line/category (e.g., Classic Cars, Motorcycles, Trucks) the product belongs to. |
| MSRP | Float | Manufacturer's Suggested Retail Price. The standard price for the product. |

**Questions:**

a) Create a new Spark Session with new SparkConfig

b) Create new instance of Spark SQL session and define new DataFrame using sales\_data\_sample.csv dataset.

c) Find the shape of DataFrame.

d) Find the Summary of DataFrame for all numerical data columns.

e) Identify and handle missing or null values in the columns.

f) Calculate the total revenue generated per country by combining the columns QUANTITYORDERED and PRICEEACH using Spark DataFrame operations?

g) Determine the top 5 products with the highest total sales revenue using Spark DataFrame?

h) Find the average order quantity for each product using groupBy and agg operations?

i) Using Spark DataFrame, filter orders where the SALES value exceeds $10,000 and sort the results by the ORDERDATE column?

j) Filter out rows where the STATUS is 'Cancelled' and calculate the total sales from the remaining orders?

k) Use Spark Data Frame transformations to derive the yearly sales for each customer (CUSTOMERNAME) based on the ORDERDATE column?

l) Add a new column to the DataFrame that categorizes orders as "High", "Medium", or "Low" sales based on the SALES value?

m) Assume, If you have another DataFrame with customer demographic data, how would you perform a join to compute the total sales per demographic group?

n) Can you implement a cumulative distribution function (CDF) over the SALES value for each CUSTOMERNAME? What insights can you gather from analyzing the CDF distribution for each customer?

o) Write spark dataframe code to rank products by total revenue within each country (COUNTRY)?

p) Calculate a running total of SALES for each customer and show the top 5 customers by this cumulative total?

q) Find and handle Invalid and Outliers values in entire DataFrame. [Check for only continuous dataset].

r) How would you cache a DataFrame containing sales data from the top 10 countries by sales to avoid recomputation in subsequent transformations? What persistence level (e.g. MEMORY\_ONLY, MEMORY\_AND\_DISK) would you choose and why?

s) How would you pivot the data to show PRODUCTLINE as columns and the total SALES for each ORDERDATE as the values? What are the implications of pivoting large datasets in Spark?

t) How would you calculate the percentage growth of total sales month over month for each PRODUCTLINE using Spark DataFrame?

u) How can you rebalance the data by portioning based on the COUNTRY column to ensure that large data partitions are avoided?

v) Suppose you have a smaller lookup table with customer details. How would you perform a broadcast join with the large sales\_data\_sample dataset to improve join performance? What are the key considerations when using broadcast joins?

w) Create a UDF that categorizes the sales values (SALES) into custom buckets like “Low”, “Medium”, “High”. Apply this UDF to the DataFrame and calculate the count of orders in each category per COUNTRY.

x) Create a Python UDF to calculate discounts for specific product lines. For example, give a 10% discount for Classic Cars and 5% for Motorcycles. Apply this UDF to derive new discounted sales values.

y) How would you set up an incremental loading mechanism for orders placed daily based on the ORDERDATE column? How can Spark checkpointing can be used with incremental load to ensure no data loss occurs during failures?

z) How do you implement a cumulative distribution function (CDF) over the SALES value for each CUSTOMERNAME? What insights can you gather from analyzing the CDF distribution for each customer?