In [263…
```
sc.stop()
```

a) Create a new Spark Session with new SparkConfig

In [264…
```
from pyspark import SparkConf, SparkContext
```

In [265…
```
config = SparkConf().setMaster("local[2]").setAppName("AssignmentSession")
sc = SparkContext(conf = config)
```

In [266…
```
sc
```

Out[266… **SparkContext**

Spark UI

Version	`v2.4.8`
Master	`local[2]`
AppName	`AssignmentSession`

b) Create new instance of Spark SQL session and define new DataFrame using sales_data_sample.csv dataset.

In [267…
```
from pyspark.sql import SparkSession
```

In [268…
```
spark = SparkSession.builder.appName("SQLSession").getOrCreate()
```

In [269…
```
sales_df=spark.read.csv("file:///home/hadoop/Downloads/sales_data_sample.csv",header=True, inferSchema = True)
```

In [270…
```
sales_df.show(2)
```

```
+-----------+---------------+---------+---------------+------+-------------+-------+------+--------+-------+-----
------+----+-----------+------------------+----------+--------------------+------------+-----+-----+----------+---
----+---------+---------------+----------------+--------+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER| SALES|     ORDERDATE| STATUS|QTR_ID|MONTH_ID|YEAR_ID|PRODU
CTLINE|MSRP|PRODUCTCODE|      CUSTOMERNAME|     PHONE|        ADDRESSLINE1|ADDRESSLINE2|  CITY|STATE|POSTALCODE|COU
NTRY|TERRITORY|CONTACTLASTNAME|CONTACTFIRSTNAME|DEALSIZE|
+-----------+---------------+---------+---------------+------+-------------+-------+------+--------+-------+-----
------+----+-----------+------------------+----------+--------------------+------------+-----+-----+----------+---
----+---------+---------------+----------------+--------+
|      10107|             30|     95.7|              2|2871.0|2/24/2003 0:00|Shipped|     1|       2|   2003|Motor
cycles|  95|   S10_1678| Land of Toys Inc.|2125557818|897 Long Airport ...|        null|  NYC|   NY|     10022|
USA|       NA|             Yu|            Kwai|   Small|
|      10121|             34|    81.35|              5|2765.9| 5/7/2003 0:00|Shipped|     2|       5|   2003|Motor
cycles|  95|   S10_1678|Reims Collectables|26.47.1555|  59 rue de l'Abbaye|        null|Reims| null|     51100| Fr
ance|     EMEA|        Henriot|            Paul|   Small|
+-----------+---------------+---------+---------------+------+-------------+-------+------+--------+-------+-----
------+----+-----------+------------------+----------+--------------------+------------+-----+-----+----------+---
----+---------+---------------+----------------+--------+
only showing top 2 rows
```

In [271…
```
sales_df.printSchema()
```

```
root
 |-- ORDERNUMBER: integer (nullable = true)
 |-- QUANTITYORDERED: integer (nullable = true)
 |-- PRICEEACH: double (nullable = true)
 |-- ORDERLINENUMBER: integer (nullable = true)
 |-- SALES: double (nullable = true)
 |-- ORDERDATE: string (nullable = true)
 |-- STATUS: string (nullable = true)
 |-- QTR_ID: integer (nullable = true)
 |-- MONTH_ID: integer (nullable = true)
 |-- YEAR_ID: integer (nullable = true)
 |-- PRODUCTLINE: string (nullable = true)
 |-- MSRP: integer (nullable = true)
 |-- PRODUCTCODE: string (nullable = true)
 |-- CUSTOMERNAME: string (nullable = true)
```

```
 |-- PHONE: string (nullable = true)
 |-- ADDRESSLINE1: string (nullable = true)
 |-- ADDRESSLINE2: string (nullable = true)
 |-- CITY: string (nullable = true)
 |-- STATE: string (nullable = true)
 |-- POSTALCODE: string (nullable = true)
 |-- COUNTRY: string (nullable = true)
 |-- TERRITORY: string (nullable = true)
 |-- CONTACTLASTNAME: string (nullable = true)
 |-- CONTACTFIRSTNAME: string (nullable = true)
 |-- DEALSIZE: string (nullable = true)
```

c) Find the shape of DataFrame.

In [272…

```
num_rows = sales_df.count()

num_cols = len(sales_df.columns)

print(f"The shape of the DataFrame is: ({num_rows}, {num_cols})")
```

```
The shape of the DataFrame is: (2823, 25)
```

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

In [273…

```
from pyspark.sql.types import NumericType
from pyspark.sql.functions import *

numerical_columns = [field.name for field in sales_df.schema.fields if isinstance(field.dataType, NumericType)]

sales_df.select(numerical_columns).summary().show()
```

```
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+------------
------+-----------------+-----------------+-----------------+
|summary|       ORDERNUMBER|  QUANTITYORDERED|         PRICEEACH|  ORDERLINENUMBER|            SALES|
QTR_ID|          MONTH_ID|          YEAR_ID|             MSRP|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+------------
------+-----------------+-----------------+-----------------+
|  count|             2823|             2823|             2823|             2823|             2823|
2823|             2823|             2823|             2823|
|   mean|10258.725115125753|35.09280906836698| 83.65854410201929|6.466170740347148|  3553.88907190932|2.7176762309
599716|7.0924548352816155|2003.8150903294368|100.71555083244775|
| stddev|  92.0854775957196| 9.74144273706958|20.174276527840536| 4.22584096469094|1841.8651057401842| 1.203878088
001756| 3.656633307661765|0.6996701541300869| 40.18791167720266|
|    min|            10100|                6|            26.88|                1|            482.13|
1|                1|             2003|               33|
|    25%|            10180|               27|             68.8|                3|           2203.11|
2|                4|             2003|               68|
|    50%|            10262|               35|             95.7|                6|            3184.8|
3|                8|             2004|               99|
|    75%|            10334|               43|            100.0|                9|            4508.0|
4|               11|             2004|              124|
|    max|            10425|               97|            100.0|               18|           14082.8|
4|               12|             2005|              214|
+-------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+------------
------+-----------------+-----------------+-----------------+
```

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

In [274…

```
sales_df.select([sum(col(column).isNull().cast('int')).alias(column) for column in sales_df.columns]).show()
```

```
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+------------
---+---------------+--------+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER|SALES|ORDERDATE|STATUS|QTR_ID|MONTH_ID|YEAR_ID|PRODUCTLINE|
MSRP|PRODUCTCODE|CUSTOMERNAME|PHONE|ADDRESSLINE1|ADDRESSLINE2|CITY|STATE|POSTALCODE|COUNTRY|TERRITORY|CONTACTLASTN
AME|CONTACTFIRSTNAME|DEALSIZE|
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+------------
---+---------------+--------+
|          0|             0|        0|             0|    0|        0|     0|     0|       0|      0|          0|
0|          0|           0|    0|           0|        2521|   0| 1486|        76|      0|        0|
0|              0|       0|
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+------------
---+---------------+--------+
```

In [275…
```
sales_df1 =  sales_df.fillna({'ADDRESSLINE2':"N/A", 'STATE':"N/A", 'POSTALCODE':"N/A"})
```

In [276…
```
sales_df1.select([sum(col(column).isNull().cast('int')).alias(column) for column in sales_df1.columns]).show()
```

```
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+-----------
---+---------------+--------+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER|SALES|ORDERDATE|STATUS|QTR_ID|MONTH_ID|YEAR_ID|PRODUCTLINE|
MSRP|PRODUCTCODE|CUSTOMERNAME|PHONE|ADDRESSLINE1|ADDRESSLINE2|CITY|STATE|POSTALCODE|COUNTRY|TERRITORY|CONTACTLASTN
AME|CONTACTFIRSTNAME|DEALSIZE|
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+-----------
---+---------------+--------+
|          0|             0|        0|             0|    0|        0|     0|     0|       0|      0|          0|
0|          0|           0|    0|           0|           0|   0|    0|         0|      0|        0|
0|               0|       0|
+-----------+--------------+---------+--------------+-----+---------+------+------+--------+-------+-----------+
----+-----------+------------+-----+------------+------------+----+-----+----------+-------+---------+-----------
---+---------------+--------+
```

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

In [404…
```
sales_df1 = sales_df1.withColumn("REVENUE", col("QUANTITYORDERED") * col("PRICEEACH"))


sales_df1.groupBy("COUNTRY").agg(sum("REVENUE").alias("TOTAL_REVENUE")).show()
```

```
+-----------+-----------------+
|    COUNTRY|     TOTAL_REVENUE|
+-----------+-----------------+
|     Sweden|174264.10000000006|
|Philippines| 80291.16999999998|
|  Singapore| 227985.5000000001|
|    Germany|         178689.08|
|     France| 919257.8499999997|
|    Belgium|          94528.88|
|    Finland|268714.70000000007|
|      Italy| 309402.8699999999|
|     Norway| 246115.8000000001|
|      Spain|1021705.9700000002|
|    Denmark|         192747.63|
|    Ireland|          43237.24|
|        USA|2986425.2099999995|
|         UK|413203.33999999997|
|Switzerland| 93344.90999999999|
|     Canada|193504.34000000003|
|      Japan|153076.68999999994|
|  Australia|521598.45999999985|
|    Austria|172793.05000000002|
+-----------+-----------------+
```

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

In [405…
```
# revenue_df = df.withColumn("REVENUE", col("QUANTITYORDERED") * col("PRICEEACH"))

total_revenue_per_product = sales_df1.groupBy("PRODUCTLINE").agg(
    sum("SALES").alias("TOTAL_SALES_REVENUE")
)


total_revenue_per_product.orderBy(col("TOTAL_SALES_REVENUE"),ascending = False).limit(5).show()
```

```
+---------------+------------------+
|     PRODUCTLINE|TOTAL_SALES_REVENUE|
+---------------+------------------+
|    Classic Cars|  3919615.659999997|
|    Vintage Cars| 1903150.8399999992|
|     Motorcycles| 1166388.3400000003|
|Trucks and Buses| 1127789.8399999996|
|          Planes|  975003.5700000001|
+---------------+------------------+
```

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

In [406…

```
sales_df1.groupBy("PRODUCTCODE").agg(
    avg("QUANTITYORDERED").alias("AVG_QUANTITY")
).show()
```

```
+-----------+-----------------+
|PRODUCTCODE|     AVG_QUANTITY|
+-----------+-----------------+
|   S18_4600| 38.18518518518518|
|   S18_1749| 36.45454545454545|
|   S12_3891| 35.42307692307692|
|   S18_2248| 33.77272727272727|
|  S700_1138| 34.69230769230769|
|   S32_1268|32.333333333333336|
|   S12_1099|             33.52|
|   S18_2795|30.346153846153847|
|   S24_1937|             33.76|
|   S32_3522| 35.44444444444444|
|   S18_1097| 35.67857142857143|
|   S18_1662| 36.15384615384615|
|   S12_1666|34.714285714285715|
|   S24_3969| 33.86363636363637|
|   S24_1578| 35.80769230769231|
|   S24_4048| 32.46153846153846|
|   S18_3320| 34.96153846153846|
|   S24_3816| 33.46153846153846|
|   S18_3136|32.333333333333336|
|   S32_2509|34.107142857142854|
+-----------+-----------------+
only showing top 20 rows
```

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

In [407…

```
sales_df1 = sales_df1.withColumn(
    "ORDERDATE",
    to_timestamp(col("ORDERDATE"), "M/d/yyyy H:mm")
)

filtered_orders = sales_df1.filter(col("SALES") > 10000)

sorted_orders = filtered_orders.orderBy(col("ORDERDATE"))

sorted_orders.select(['PRODUCTCODE', 'SALES', 'ORDERDATE']).show()
```

```
+-----------+-------+-------------------+
|PRODUCTCODE|  SALES|          ORDERDATE|
+-----------+-------+-------------------+
|   S12_1108|11279.2|2003-06-03 00:00:00|
|   S10_1949|10993.5|2003-09-19 00:00:00|
|   S12_1108|10606.2|2004-05-05 00:00:00|
|   S10_1949|10172.7|2004-10-11 00:00:00|
|   S10_1949|11623.7|2004-10-21 00:00:00|
|   S18_2325|12536.5|2004-11-04 00:00:00|
|   S18_3320|11336.7|2004-11-18 00:00:00|
|   S24_3151|10758.0|2004-11-23 00:00:00|
|   S24_4278|10039.6|2005-02-03 00:00:00|
|  S700_1691|10066.6|2005-03-03 00:00:00|
|   S10_4698|11886.6|2005-04-08 00:00:00|
|   S24_3856|11739.7|2005-04-14 00:00:00|
|   S18_3685|10468.9|2005-04-15 00:00:00|
|   S18_1749|14082.8|2005-04-22 00:00:00|
|   S18_3232|11887.8|2005-05-03 00:00:00|
|   S10_1949|12001.0|2005-05-31 00:00:00|
+-----------+-------+-------------------+
```

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

In [394…

```
#calculate the total sales from the remaining orders where the status is not cancelled
filtered_df1 = sales_df1.filter(col('STATUS')!= 'Cancelled')
filtered_df1.agg(sum('SALES').alias('TOTAL_SALES')).show()
```

```
+-----------------+
|      TOTAL_SALES|
+-----------------+
|9838141.370000018|
+-----------------+
```

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

In [408…

```
sales_df1 = sales_df1.withColumn("YEAR", year(col("ORDERDATE")))


sales_df1.groupBy("CUSTOMERNAME", "YEAR").agg(
    sum("SALES").alias("TOTAL_SALES")
).show()
```

```
+--------------------+----+------------------+
|        CUSTOMERNAME|YEAR|       TOTAL_SALES|
+--------------------+----+------------------+
|  Baane Mini Imports|2003|56176.659999999996|
|Stylish Desk Deco...|2004|13739.900000000001|
|Marseille Mini Autos|2003|52481.840000000004|
|Danish Wholesale ...|2004|          60157.62|
|Toms Spezialitten...|2003|          31363.18|
|Australian Collec...|2004|140859.56999999998|
|Dragon Souveniers...|2004|           3127.88|
|    Super Scale Inc.|2003|          42498.76|
|Collectables For ...|2004|15110.800000000001|
|Royal Canadian Co...|2004| 74634.84999999999|
|Online Diecast Cr...|2003|           76114.7|
|   Gifts4AllAges.com|2005|          48316.89|
|        Herkku Gifts|2004|           16363.1|
|Diecast Classics ...|2004|115971.34000000001|
|Motor Mint Distri...|2003|          27398.82|
|Daedalus Designs ...|2003|48874.280000000006|
|Stylish Desk Deco...|2003|           75064.6|
|         Mini Caravy|2005|          35680.35|
|     Mini Wheels Co.|2004|          30348.72|
|Scandinavian Gift...|2005|          31606.72|
+--------------------+----+------------------+
only showing top 20 rows
```

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

In [395…

```
quantiles = sales_df1.approxQuantile("Sales", [0.33, 0.66], 0.01)

q1 = quantiles[0]
q2 = quantiles[1]

sales_df1 = sales_df.withColumn(
    "Category",
    when(col("Sales") < q1, "Low")
    .when(col("Sales") < q2, "Medium")
    .otherwise("High")
)


sales_df1.select(['Sales','Category']).show()
```

```
+-------+--------+
|  Sales|Category|
+-------+--------+
| 2871.0|  Medium|
| 2765.9|  Medium|
|3884.34|  Medium|
| 3746.7|  Medium|
|5205.27|    High|
|3479.76|  Medium|
|2497.77|  Medium|
|5512.32|    High|
|2168.54|     Low|
|4708.44|    High|
|3965.66|    High|
|2333.12|     Low|
|3188.64|  Medium|
|3676.76|  Medium|
|4177.35|    High|
|4099.68|    High|
|2597.39|  Medium|
|4394.38|    High|
|4358.04|    High|
|4396.14|    High|
```

```
+-------+--------+
only showing top 20 rows
```

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?

In [284…

```
demographics_df = spark.read.csv("file:///home/hadoop/Downloads/tempdata.csv", header=True, inferSchema=True)


joined_df = sales_df1.join(demographics_df, on="CUSTOMERNAME", how="inner")


total_sales_per_demographic = joined_df.groupBy("REGION") \
                                       .agg(sum("SALES").alias("TOTAL_SALES"))


total_sales_per_demographic.show()
```

```
+-------------+-----------------+
|       REGION|      TOTAL_SALES|
+-------------+-----------------+
|       Europe| 784200.7199999997|
|North America|1112155.9099999997|
+-------------+-----------------+
```

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?

In [285…

```
window_spec = Window.partitionBy("CUSTOMERNAME").orderBy("SALES")

df_with_cdf = sales_df1.withColumn("Rank", percent_rank().over(window_spec)) \
                .withColumn("CDF", col("Rank"))

df_with_cdf_sorted = df_with_cdf.orderBy(col("CUSTOMERNAME"), col("SALES").desc())


df_with_cdf_sorted.select(['CUSTOMERNAME', 'SALES', 'CDF']).show(100)
```

```
+--------------------+-------+------------------+
|        CUSTOMERNAME|  SALES|               CDF|
+--------------------+-------+------------------+
|      AV Stores, Co.| 7310.0|               1.0|
|      AV Stores, Co.|6570.76|              0.98|
|      AV Stores, Co.| 6069.0|              0.96|
|      AV Stores, Co.|5942.28|              0.94|
|      AV Stores, Co.|5433.08|              0.92|
|      AV Stores, Co.|5074.39|               0.9|
|      AV Stores, Co.| 4982.7|              0.88|
|      AV Stores, Co.|4375.98|              0.86|
|      AV Stores, Co.|3859.68|              0.84|
|      AV Stores, Co.|3759.04|              0.82|
|      AV Stores, Co.|3724.42|               0.8|
|      AV Stores, Co.|3685.95|              0.78|
|      AV Stores, Co.| 3670.4|              0.76|
|      AV Stores, Co.| 3633.4|              0.74|
|      AV Stores, Co.|3600.65|              0.72|
|      AV Stores, Co.| 3525.6|               0.7|
|      AV Stores, Co.| 3515.7|              0.68|
|      AV Stores, Co.|3488.78|              0.66|
|      AV Stores, Co.|3472.98|              0.64|
|      AV Stores, Co.|3382.08|              0.62|
|      AV Stores, Co.|3360.45|               0.6|
|      AV Stores, Co.|3232.31|              0.58|
|      AV Stores, Co.| 3207.4|              0.56|
|      AV Stores, Co.| 3201.5|              0.54|
|      AV Stores, Co.|3166.84|              0.52|
|      AV Stores, Co.| 3069.0|               0.5|
|      AV Stores, Co.|2979.08|              0.48|
|      AV Stores, Co.| 2884.8|              0.46|
|      AV Stores, Co.| 2700.0|              0.44|
|      AV Stores, Co.|2606.48|              0.42|
|      AV Stores, Co.|2574.18|               0.4|
|      AV Stores, Co.|2499.56|              0.38|
|      AV Stores, Co.|2427.03|              0.36|
|      AV Stores, Co.|2315.18|              0.34|
|      AV Stores, Co.|2264.15|              0.32|
|      AV Stores, Co.|2223.52|               0.3|
```

```
|      AV Stores, Co.|2082.85|               0.28|
|      AV Stores, Co.|2051.56|               0.26|
|      AV Stores, Co.|1987.74|               0.24|
|      AV Stores, Co.|1859.44|               0.22|
|      AV Stores, Co.| 1759.2|                0.2|
|      AV Stores, Co.|1729.65|               0.18|
|      AV Stores, Co.|1721.73|               0.16|
|      AV Stores, Co.|1654.56|               0.14|
|      AV Stores, Co.| 1608.0|               0.12|
|      AV Stores, Co.| 1592.0|                0.1|
|      AV Stores, Co.|1538.55|               0.08|
|      AV Stores, Co.|1307.32|               0.06|
|      AV Stores, Co.|1264.08|               0.04|
|      AV Stores, Co.|1152.58|               0.02|
|      AV Stores, Co.|  710.2|                0.0|
|        Alpha Cognac|8331.61|                1.0|
|        Alpha Cognac|6490.68| 0.9473684210526315|
|        Alpha Cognac|5386.56| 0.8947368421052632|
|        Alpha Cognac|5274.72| 0.8421052631578947|
|        Alpha Cognac|5192.64| 0.7894736842105263|
|        Alpha Cognac|3789.72| 0.7368421052631579|
|        Alpha Cognac|3644.75| 0.6842105263157895|
|        Alpha Cognac|3492.48|  0.631578947368421|
|        Alpha Cognac|3431.25| 0.5789473684210527|
|        Alpha Cognac|3350.52| 0.5263157894736842|
|        Alpha Cognac|3293.24|0.47368421052631576|
|        Alpha Cognac|2748.56|0.42105263157894735|
|        Alpha Cognac| 2577.6| 0.3684210526315789|
|        Alpha Cognac|2301.75| 0.3157894736842105|
|        Alpha Cognac| 2169.9| 0.2631578947368421|
|        Alpha Cognac|2116.16|0.21052631578947367|
|        Alpha Cognac|1988.28|0.15789473684210525|
|        Alpha Cognac| 1930.5|0.10526315789473684|
|        Alpha Cognac|1514.52|0.05263157894736842|
|        Alpha Cognac| 1463.0|                0.0|
|  Amica Models & Co.|8411.56|                1.0|
|  Amica Models & Co.| 8253.0|               0.96|
|  Amica Models & Co.|8014.82|               0.92|
|  Amica Models & Co.| 5239.5|               0.88|
|  Amica Models & Co.|5126.24|               0.84|
|  Amica Models & Co.|4946.06|                0.8|
|  Amica Models & Co.| 4750.8|               0.76|
|  Amica Models & Co.| 4455.0|               0.72|
|  Amica Models & Co.|4242.24|               0.68|
|  Amica Models & Co.|3704.05|               0.64|
|  Amica Models & Co.| 3668.6|                0.6|
|  Amica Models & Co.|3474.46|               0.56|
|  Amica Models & Co.|3006.43|               0.52|
|  Amica Models & Co.|2954.53|               0.48|
|  Amica Models & Co.|2941.89|               0.44|
|  Amica Models & Co.|2819.28|                0.4|
|  Amica Models & Co.|2800.08|               0.36|
|  Amica Models & Co.|2418.24|               0.32|
|  Amica Models & Co.|2137.05|               0.28|
|  Amica Models & Co.|2084.81|               0.24|
|  Amica Models & Co.|1921.92|                0.2|
|  Amica Models & Co.|1656.69|               0.16|
|  Amica Models & Co.| 1574.0|               0.12|
|  Amica Models & Co.|1557.36|               0.08|
|  Amica Models & Co.|1381.05|               0.04|
|  Amica Models & Co.|  577.6|                0.0|
|Anna's Decoration...|8470.14|                1.0|
|Anna's Decoration...|8344.71| 0.9777777777777777|
|Anna's Decoration...|5848.68| 0.9555555555555556|
+--------------------+-------+-------------------+
only showing top 100 rows
```

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

In [398…

```
from pyspark.sql.window import Window
from pyspark.sql.functions import rank

revenue_df = sales_df1.groupBy("COUNTRY", "PRODUCTCODE").agg(sum("REVENUE").alias("TOTAL_REVENUE"))


windowSpec = Window.partitionBy("COUNTRY").orderBy(col("TOTAL_REVENUE").desc())


ranked_df = revenue_df.withColumn("RANK", rank().over(windowSpec))


ranked_df.select("COUNTRY", "PRODUCTCODE", "TOTAL_REVENUE", "RANK").show()
```

```
+-------+-----------+------------------+----+
|COUNTRY|PRODUCTCODE|     TOTAL_REVENUE|RANK|
+-------+-----------+------------------+----+
| Sweden|   S18_4600|            9700.0|   1|
| Sweden|   S24_2300|            9000.0|   2|
| Sweden|   S24_2011|            7400.0|   3|
| Sweden|   S18_2949|            7000.0|   4|
| Sweden|   S10_1949|            6600.0|   5|
| Sweden|   S12_1099|           5675.04|   6|
| Sweden|   S10_4962|            5600.0|   7|
| Sweden|  S700_1138| 5579.620000000001|   8|
| Sweden|   S12_3990|           5319.32|   9|
| Sweden|   S12_3380|            5309.5|  10|
| Sweden|   S24_3151| 5113.049999999999|  11|
| Sweden|   S12_4675|            4700.0|  12|
| Sweden|   S18_2319|            4600.0|  13|
| Sweden|   S24_1578|            4500.0|  14|
| Sweden|   S10_4757|            4400.0|  15|
| Sweden|   S18_4522|            4300.5|  16|
| Sweden|   S18_1662|            4300.0|  17|
| Sweden|   S24_3816|4276.9400000000005|  18|
| Sweden|   S12_1666|            4100.0|  19|
| Sweden|   S18_1097|            4100.0|  19|
+-------+-----------+------------------+----+
only showing top 20 rows
```

Insight

- Sweden has the produced the highest total revenue
- In each country the classic cars has more demand and produces more revenue

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

In [399…

```
windowSpec = Window.partitionBy('CUSTOMERNAME').orderBy('SALES').rowsBetween(Window.unboundedPreceding,
                                                                             Window.currentRow)

#this will provide the running total for each customer, To find the top 5 , the running SALES is aggregated(max)
# using CUSTOMERNAME and sorted in desc order and took top 5
df_with_running_total = sales_df.withColumn('RunningTotal', sum('SALES').over(windowSpec))

# df_with_running_total.select(['CUSTOMERNAME', "SALES", 'RunningTotal']).show()

total_sales_df = df_with_running_total.groupBy('CUSTOMERNAME').agg(max('RunningTotal').alias('TotalSales'))

top_customers_df = total_sales_df.orderBy(col('TotalSales').desc()).limit(5)

top_customers_df.show()
```

```
+--------------------+-----------------+
|        CUSTOMERNAME|       TotalSales|
+--------------------+-----------------+
|Euro Shopping Cha...| 912294.1100000002|
|Mini Gifts Distri...| 654858.0600000002|
|Australian Collec...|200995.40999999995|
|  Muscle Machine Inc|197736.94000000003|
|   La Rochelle Gifts|180124.90000000008|
+--------------------+-----------------+
```

q) Identify and handle Outliers in DataFrame. [Check for only continuous dataset].

In [336…

```
numerical_columns = [field.name for field in sales_df1.schema.fields if isinstance(field.dataType, NumericType)]

for column in numerical_columns:
    quantiles = sales_df1.approxQuantile(column, [0.25, 0.5, 0.75], 0.01)
    q1, q2, q3 = quantiles[0], quantiles[1], quantiles[2]
    iqr = q3 - q1

    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    print("Outliers")
    sales_df1.fselect(['PRODUCTCODE', column]).filter(
        (col(column) < lower_bound) | (col(column) > upper_bound)
    ).show()


    filtered_df = sales_df1.filter(
        (col(column) >= lower_bound) & (col(column) <= upper_bound)
    )

    print("\nCleaned DataFrame")

    filtered_df.select(['PRODUCTCODE', 'ORDERNUMBER']).show()
```

```
Outliers
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
+-----------+-----------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+---------------+
|PRODUCTCODE|QUANTITYORDERED|
+-----------+---------------+
|   S12_4675|             97|
|   S18_1749|             76|
|   S24_2300|             70|
|   S24_2766|             76|
|   S24_3856|             76|
|   S24_3856|             70|
|  S700_2466|             85|
|  S700_3167|             77|
+-----------+---------------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
```

```
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+---------+
|PRODUCTCODE|PRICEEACH|
+-----------+---------+
+-----------+---------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+---------------+
|PRODUCTCODE|ORDERLINENUMBER|
+-----------+---------------+
+-----------+---------------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
```

```
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+-------+
|PRODUCTCODE|  SALES|
+-----------+-------+
|   S10_1949|10993.5|
|   S10_1949|8014.82|
|   S10_1949|9064.89|
|   S10_1949|8014.82|
|   S10_1949|10172.7|
|   S10_1949|11623.7|
|   S10_1949| 8254.8|
|   S10_1949|12001.0|
|   S10_4698|9264.86|
|   S10_4698| 8892.9|
|   S10_4698| 8714.7|
|   S10_4698|8065.89|
|   S10_4698|9774.03|
|   S10_4698|8336.94|
|   S10_4698|11886.6|
|   S10_4698|9218.16|
|   S10_4757|9661.44|
|   S12_1099|8008.56|
|   S12_1099|9245.76|
|   S12_1099|8296.35|
+-----------+-------+
only showing top 20 rows


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+------+
|PRODUCTCODE|QTR_ID|
+-----------+------+
+-----------+------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
```

```
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+--------+
|PRODUCTCODE|MONTH_ID|
+-----------+--------+
+-----------+--------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+-------+
|PRODUCTCODE|YEAR_ID|
+-----------+-------+
+-----------+-------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+----+
|PRODUCTCODE|MSRP|
+-----------+----+
|   S10_1949| 214|
|   S10_1949| 214|
```

```
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
|   S10_1949| 214|
+-----------+----+
only showing top 20 rows


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+-------+
|PRODUCTCODE|REVENUE|
+-----------+-------+
|   S10_1678| 6600.0|
|   S10_4698| 6600.0|
|   S10_4757| 6400.0|
|   S12_2823| 6600.0|
|   S12_4675|9048.16|
|   S18_1749| 7600.0|
|   S18_3685| 6500.0|
|   S18_4409| 6134.7|
|   S24_2300| 7000.0|
|   S24_2766| 7182.0|
|   S24_3856| 7600.0|
|   S24_3856| 7000.0|
|  S700_2466|7543.75|
|  S700_3167| 7084.0|
+-----------+-------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S10_1678|      10107|
|   S10_1678|      10121|
|   S10_1678|      10134|
|   S10_1678|      10145|
|   S10_1678|      10159|
|   S10_1678|      10168|
|   S10_1678|      10180|
|   S10_1678|      10188|
|   S10_1678|      10201|
```

```
|   S10_1678|      10211|
|   S10_1678|      10223|
|   S10_1678|      10237|
|   S10_1678|      10251|
|   S10_1678|      10263|
|   S10_1678|      10275|
|   S10_1678|      10285|
|   S10_1678|      10299|
|   S10_1678|      10309|
|   S10_1678|      10318|
|   S10_1678|      10329|
+-----------+-----------+
only showing top 20 rows

Outliers
+-----------+---------+
|PRODUCTCODE|cum_sales|
+-----------+---------+
+-----------+---------+


Cleaned DataFrame
+-----------+-----------+
|PRODUCTCODE|ORDERNUMBER|
+-----------+-----------+
|   S12_1108|      10306|
|   S12_3148|      10306|
|   S12_3891|      10306|
|   S18_1342|      10332|
|   S18_1367|      10332|
|   S18_1589|      10110|
|   S18_1749|      10110|
|   S18_2248|      10110|
|   S18_2248|      10332|
|   S18_2325|      10110|
|   S18_2325|      10332|
|   S18_2795|      10110|
|   S18_2795|      10332|
|   S18_2957|      10332|
|   S18_3136|      10332|
|   S18_3140|      10306|
|   S18_3259|      10306|
|   S18_4027|      10306|
|   S18_4409|      10110|
|   S18 4409|      10332|
```

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?

I will choose MEMORY_ONLY because since the data is restricted to a count of 10, Caching small dataset on MEMORY avoiding unnecessary DISK I/O

In [389…

```
from pyspark import StorageLevel

top_10_countries_df = sales_df.orderBy(col("Sales").desc()).limit(10)

top_10_countries_df.persist(StorageLevel.MEMORY_ONLY)

top_10_countries_df.count()

top_10_countries_df.show()

top_10_countries_df.unpersist()
```

```
+-----------+--------------+---------+--------------+-------+--------------+----------+------+--------+-------+
------------+----+-----------+--------------------+-------------+--------------------+------------+-------------+
-----+----------+---------+---------+--------------+---------+--------+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER|  SALES|     ORDERDATE|    STATUS|QTR_ID|MONTH_ID|YEAR_ID|
PRODUCTLINE|MSRP|PRODUCTCODE|        CUSTOMERNAME|         PHONE|        ADDRESSLINE1|ADDRESSLINE2|         CITY|S
TATE|POSTALCODE|  COUNTRY|TERRITORY|CONTACTLASTNAME|CONTACTFIRSTNAME|DEALSIZE|
+-----------+--------------+---------+--------------+-------+--------------+----------+------+--------+-------+
------------+----+-----------+--------------------+-------------+--------------------+------------+-------------+
-----+----------+---------+---------+--------------+---------+--------+
|      10407|            76|    100.0|             2|14082.8| 4/22/2005 0:00|   On Hold|     2|       4|   2005|
Vintage Cars| 170|   S18_1749|The Sharp Gifts W...|    4085553659|      3086 Ingle Ln.|        null|     San Jose|
CA|     94217|      USA|       NA|          Frick|             Sue|   Large|
|      10322|            50|    100.0|             6|12536.5| 11/4/2004 0:00|   Shipped|     4|      11|   2004|
Vintage Cars| 127|   S18_2325|Online Diecast Cr...|    6035558647|2304 Long Airport...|        null|       Nashua|
NH|     62005|      USA|       NA|          Young|         Valarie|   Large|
|      10424|            50|    100.0|             6|12001.0| 5/31/2005 0:00|In Process|     2|       5|   2005|
```

```
Classic Cars| 214|   S10_1949|Euro Shopping Cha...|(91) 555 94 44|  C/ Moralzarzal, 86|       null|       Madrid|
null|     28034|    Spain|     EMEA|         Freyre|          Diego|   Large|
|      10412|            60|    100.0|             9|11887.8|  5/3/2005 0:00|   Shipped|     2|       5|   2005|
Classic Cars| 169|   S18_3232|Euro Shopping Cha...|(91) 555 94 44|  C/ Moralzarzal, 86|       null|       Madrid|
null|     28034|    Spain|     EMEA|         Freyre|          Diego|   Large|
|      10403|            66|    100.0|             9|11886.6|  4/8/2005 0:00|   Shipped|     2|       4|   2005|
Motorcycles| 193|   S10_4698|UK Collectables, ...|(171) 555-2282|Berkeley Gardens ...|       null|    Liverpool|
null|   WX1 6LT|       UK|     EMEA|          Devon|       Elizabeth|   Large|
|      10405|            76|    100.0|             3|11739.7| 4/14/2005 0:00|   Shipped|     2|       4|   2005|
Classic Cars| 140|   S24_3856|         Mini Caravy|    88.60.1555|    24, place Kluber|       null|   Strasbourg|
null|     67000|   France|     EMEA|        Citeaux|      Frederique|   Large|
|      10312|            48|    100.0|             3|11623.7|10/21/2004 0:00|   Shipped|     4|      10|   2004|
Classic Cars| 214|   S10_1949|Mini Gifts Distri...|    4155551450|     5677 Strong St.|       null|   San Rafael|
CA|     97562|      USA|       NA|         Nelson|         Valarie|   Large|
|      10333|            46|    100.0|             2|11336.7|11/18/2004 0:00|   Shipped|     4|      11|   2004|
Vintage Cars|  99|   S18_3320|     Mini Wheels Co.|    6505555787|5557 North Pendal...|       null|San Francisco|
CA|      null|      USA|       NA|         Murphy|           Julie|   Large|
|      10127|            46|    100.0|             2|11279.2|  6/3/2003 0:00|   Shipped|     2|       6|   2003|
Classic Cars| 207|   S12_1108|  Muscle Machine Inc|    2125557413|   4092 Furth Circle|   Suite 400|          NYC|
NY|     10022|      USA|       NA|          Young|            Jeff|   Large|
|      10150|            45|    100.0|             8|10993.5| 9/19/2003 0:00|   Shipped|     3|       9|   2003|
Classic Cars| 214|   S10_1949|Dragon Souveniers...|  +65 221 7555|Bronz Sok., Bronz...|       null|    Singapore|
null|     79903|Singapore|    Japan|      Natividad|            Eric|   Large|
+-----------+--------------+---------+--------------+-------+--------------+----------+------+--------+-------+
------------+----+-----------+--------------------+-------------+--------------------+-----------+-------------+
-----+----------+---------+---------+--------------+----------------+--------+
```

Out[389…

```
DataFrame[ORDERNUMBER: int, QUANTITYORDERED: int, PRICEEACH: double, ORDERLINENUMBER: int, SALES: double, ORDERDAT
E: string, STATUS: string, QTR_ID: int, MONTH_ID: int, YEAR_ID: int, PRODUCTLINE: string, MSRP: int, PRODUCTCODE:
string, CUSTOMERNAME: string, PHONE: string, ADDRESSLINE1: string, ADDRESSLINE2: string, CITY: string, STATE: stri
ng, POSTALCODE: string, COUNTRY: string, TERRITORY: string, CONTACTLASTNAME: string, CONTACTFIRSTNAME: string, DEA
LSIZE: string]
```

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?

In [390…

```
pivot_df = sales_df1.groupBy("ORDERDATE") \
    .pivot("PRODUCTLINE") \
    .agg(sum("SALES"))



pivot_df.show()
```

```
+-------------------+------------------+------------------+------------------+------------------+-------+---------
-------+------------+
|          ORDERDATE|      Classic Cars|       Motorcycles|            Planes|             Ships| Trains|Trucks an
d Buses|Vintage Cars|
+-------------------+------------------+------------------+------------------+------------------+-------+---------
-------+------------+
|2005-03-02 00:00:00|              null|            4175.6|              null|              null|   null|
null|        null|
|2004-11-09 00:00:00|              null|              null|              null|           6673.29|3807.68|
null|     9665.35|
|2005-05-03 00:00:00|25040.629999999997|              null|              null|              null|   null|        2
7247.11|        null|
|2003-09-11 00:00:00| 43593.54000000001|              null|              null|              null|   null|
null|     3598.22|
|2005-04-01 00:00:00|           9661.44|              null|           9036.06|            6284.0|   null|
null|    12545.34|
|2005-05-10 00:00:00|              null|            7567.8|30429.010000000002|              null|   null|
null|     3513.13|
|2004-04-26 00:00:00|              null|              null|              null|              null|   null|
null|      7129.0|
|2003-10-17 00:00:00| 40321.60999999999|              null|              null|              null|   null|
null|        null|
|2004-09-27 00:00:00|              null|5307.9800000000005|              null|              null|   null|
null|        null|
|2003-10-20 00:00:00|            4860.24|              null|              null|              null|   null|
null|    20424.51|
|2004-04-09 00:00:00|           31329.56|              null|              null|              null|   null|
null|        null|
|2005-05-06 00:00:00|            2764.88|              null|              null|23664.609999999997|5808.48|
null|    16078.92|
|2003-02-17 00:00:00|              null|              null|39205.310000000005|           6598.34|   null|
null|    10377.67|
|2004-12-07 00:00:00|           19489.57|10394.560000000001|              null|              null|   null|
null|        null|
|2005-04-22 00:00:00|           40207.85|              null|              null|              null|   null|
null|    18229.19|
|2003-12-01 00:00:00|              null|25431.879999999997|            7120.96|              null|   null|
null|      1113.6|
```

```
|2003-01-06 00:00:00|              null|              null|              null|              null|   null|
null|    12133.25|
|2004-05-18 00:00:00|              null|          27987.07|              null|              null|   null|
null|        null|
|2005-01-05 00:00:00|              null|          13529.57|              null|              null|   null|
null|        null|
|2004-11-17 00:00:00| 47423.57000000001|12786.920000000002|              null|              null|   null|
null|    37747.89|
+-------------------+------------------+------------------+------------------+------------------+-------+---------
-------+------------+
only showing top 20 rows
```

Implications

Pivoting large datasets in Spark can lead to high memory use, slow performance due to data shuffling, and potential data skew. It may also cause disk I/O issues and make the DataFrame more complex to handle.

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

In [410…

```
sales_df1 = sales_df1.withColumn("MONTH", month(col("ORDERDATE")))


monthly_sales_df = sales_df1.groupBy("YEAR", "MONTH", "PRODUCTLINE", 'ORDERDATE') \
                     .agg(sum("SALES").alias("TOTAL_SALES"))


window_spec = Window.partitionBy("PRODUCTLINE").orderBy("YEAR", "MONTH")


monthly_sales_df = monthly_sales_df.withColumn("PREV_MONTH_SALES", lag("TOTAL_SALES").over(window_spec))

monthly_sales_df = monthly_sales_df.withColumn(
    "PERCENTAGE_GROWTH",
    (col("TOTAL_SALES") - col("PREV_MONTH_SALES")) / col("PREV_MONTH_SALES") *100
)


monthly_sales_df.orderBy(col('PRODUCTLINE'),col('PERCENTAGE_GROWTH').desc()).show()
```

```
+----+-----+------------+-------------------+------------------+------------------+------------------+
|YEAR|MONTH| PRODUCTLINE|          ORDERDATE|       TOTAL_SALES|  PREV_MONTH_SALES| PERCENTAGE_GROWTH|
+----+-----+------------+-------------------+------------------+------------------+------------------+
|2003|   11|Classic Cars|2003-11-12 00:00:00|          81754.22|            1906.5| 4188.183582480986|
|2005|    5|Classic Cars|2005-05-17 00:00:00|59475.100000000006|           2764.88|2051.0915482769597|
|2004|   12|Classic Cars|2004-12-01 00:00:00|29714.590000000004|           1962.22|1414.3352936979545|
|2005|    1|Classic Cars|2005-01-06 00:00:00|          35716.64|            2611.8|1267.5105291369935|
|2004|    7|Classic Cars|2004-07-20 00:00:00|42464.450000000004|            3508.8|1110.2271431828544|
|2005|    2|Classic Cars|2005-02-10 00:00:00|          37136.78|            3420.0|  985.870760233918|
|2003|   11|Classic Cars|2003-11-20 00:00:00|          92216.81|           10806.7|  753.329971221557|
|2003|   11|Classic Cars|2003-11-26 00:00:00|          29788.46|           3492.48|  752.931441268096|
|2004|    2|Classic Cars|2004-02-22 00:00:00|          50432.55|            5997.6| 740.8788515406162|
|2004|    6|Classic Cars|2004-06-15 00:00:00|27052.409999999996|           3378.24| 700.7841361182154|
|2004|    9|Classic Cars|2004-09-01 00:00:00|          26193.68|            3427.2| 664.2880485527544|
|2003|   12|Classic Cars|2003-12-02 00:00:00|          94914.25|12936.779999999999| 633.6775457262163|
|2005|    3|Classic Cars|2005-03-04 00:00:00|41151.979999999996|           5651.52|  628.157734556367|
|2003|   10|Classic Cars|2003-10-17 00:00:00| 40321.60999999999|           6732.09|498.94638960560525|
|2004|   11|Classic Cars|2004-11-05 00:00:00|          40174.03|           6880.61|483.87308683387084|
|2004|   11|Classic Cars|2004-11-29 00:00:00|19162.010000000002|           3375.55|467.67074995185976|
|2003|   10|Classic Cars|2003-10-04 00:00:00|          27257.79|           4860.24|  460.832181126858|
|2004|    1|Classic Cars|2004-01-15 00:00:00|26618.280000000002|            4875.2|445.99360026255334|
|2005|    2|Classic Cars|2005-02-09 00:00:00|          26422.82|            4846.7|445.17135370458243|
|2003|   11|Classic Cars|2003-11-04 00:00:00|          37852.99|           7557.04|  400.897044345405|
+----+-----+------------+-------------------+------------------+------------------+------------------+
only showing top 20 rows
```

Insight: Through this we can find out the year and respective month with maximum growth and less growth for each productLine

eg: Classic cars highest growth on 2003 on month 11

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

df.repartition(num_partitions, "COUNTRY") repartitions the DataFrame based on the COUNTRY column. This distributes the rows more evenly across the specified number of partitions.

In [371…

```
num_partitions = 4
df_repartitioned = sales_df1.repartition(num_partitions, "COUNTRY")

print(f"Number of partitions: {df_repartitioned.rdd.getNumPartitions()}")

df_repartitioned.show(10)
```

```
Number of partitions: 4
+-----------+--------------+---------+--------------+-------+------------------+-------+------+--------+-------
+------------+----+-----------+--------------------+--------------+--------------------+------------+------------
--+----------+----------+---------+---------+--------------+----------------+--------+--------+-----+----+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER|  SALES|         ORDERDATE| STATUS|QTR_ID|MONTH_ID|YEAR_I
D| PRODUCTLINE|MSRP|PRODUCTCODE|        CUSTOMERNAME|         PHONE|       ADDRESSLINE1|ADDRESSLINE2|          C
ITY|     STATE|POSTALCODE|  COUNTRY|TERRITORY|CONTACTLASTNAME|CONTACTFIRSTNAME|DEALSIZE|Category|MONTH|YEAR|
+-----------+--------------+---------+--------------+-------+------------------+-------+------+--------+-------
+------------+----+-----------+--------------------+--------------+--------------------+------------+------------
--+----------+----------+---------+---------+--------------+----------------+--------+--------+-----+----+
|      10188|            48|    100.0|             1|5512.32|2003-11-18 00:00:00|Shipped|     4|      11|   200
3| Motorcycles|  95|   S10_1678|        Herkku Gifts|  +47 2267 3215|Drammen 121, PR 7...|        null|        Ber
gen|      null|    N 5804|   Norway|     EMEA|         Oeztan|          Veysel|  Medium|    High|   11|2003|
|      10223|            37|    100.0|             1|3965.66|2004-02-20 00:00:00|Shipped|     1|       2|   200
4| Motorcycles|  95|   S10_1678|Australian Collec...|   03 9520 4555|   636 St Kilda Road|     Level 3|     Melbou
rne|  Victoria|      3004|Australia|     APAC|       Ferguson|           Peter|  Medium|    High|    2|2004|
|      10309|            41|    100.0|             5|4394.38|2004-10-15 00:00:00|Shipped|     4|      10|   200
4| Motorcycles|  95|   S10_1678|  Baane Mini Imports|     07-98 9555|Erling Skakkes ga...|        null|       Stav
ern|      null|      4110|   Norway|     EMEA|     Bergulfsen|           Jonas|  Medium|    High|   10|2004|
|      10341|            41|    100.0|             9|7737.93|2004-11-24 00:00:00|Shipped|     4|      11|   200
4| Motorcycles|  95|   S10_1678|Salzburg Collecta...|      6562-9555|         Geislweg 14|        null|      Salzb
urg|      null|      5020|  Austria|     EMEA|          Pipps|           Georg|   Large|    High|   11|2004|
|      10361|            20|    72.55|            13| 1451.0|2004-12-17 00:00:00|Shipped|     4|      12|   200
4| Motorcycles|  95|   S10_1678|Souveniers And Th...|+61 2 9495 8555|Monitor Money Bui...|     Level 6|     Chatsw
ood|       NSW|      2067|Australia|     APAC|         Huxley|          Adrian|   Small|     Low|   12|2004|
|      10403|            24|    100.0|             7|2434.56|2005-04-08 00:00:00|Shipped|     2|       4|   200
5| Motorcycles|  95|   S10_1678|UK Collectables, ...| (171) 555-2282|Berkeley Gardens ...|        null|     Liverp
ool|      null|   WX1 6LT|       UK|     EMEA|          Devon|       Elizabeth|   Small|  Medium|    4|2005|
|      10103|            26|    100.0|            11|5404.62|2003-01-29 00:00:00|Shipped|     1|       1|   200
3|Classic Cars| 214|   S10_1949|  Baane Mini Imports|     07-98 9555|Erling Skakkes ga...|        null|       Stav
ern|      null|      4110|   Norway|     EMEA|     Bergulfsen|           Jonas|  Medium|    High|    1|2003|
|      10174|            34|    100.0|             4|8014.82|2003-11-06 00:00:00|Shipped|     4|      11|   200
3|Classic Cars| 214|   S10_1949|Australian Gift N...| 61-7-3844-6555|31 Duncan St. Wes...|        null|South Brisb
ane|Queensland|      4101|Australia|     APAC|       Calaghan|            Tony|   Large|    High|   11|2003|
|      10270|            21|    100.0|             9|4905.39|2004-07-19 00:00:00|Shipped|     3|       7|   200
4|Classic Cars| 214|   S10_1949|Souveniers And Th...|+61 2 9495 8555|Monitor Money Bui...|     Level 6|     Chatsw
ood|       NSW|      2067|Australia|     APAC|         Huxley|          Adrian|  Medium|    High|    7|2004|
|      10347|            30|    100.0|             1| 3944.7|2004-11-29 00:00:00|Shipped|     4|      11|   200
4|Classic Cars| 214|   S10_1949|Australian Collec...|   03 9520 4555|   636 St Kilda Road|     Level 3|     Melbou
rne|  Victoria|      3004|Australia|     APAC|       Ferguson|           Peter|  Medium|    High|   11|2004|
+-----------+--------------+---------+--------------+-------+------------------+-------+------+--------+-------
+------------+----+-----------+--------------------+--------------+--------------------+------------+------------
--+----------+----------+---------+---------+--------------+----------------+--------+--------+-----+----+
only showing top 10 rows
```

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?

Performing a broadcast join in Spark is a technique used to improve join performance when one of the tables is relatively small and fits into memory. By broadcasting the smaller table across all nodes, Spark can avoid shuffling the larger table, leading to more efficient joins.

In [370…

```
customer_data = [
    ('C001', 'John Doe', 'USA','10347'),
    ('C002', 'Jane Smith', 'UK','10174'),
    ('C003', 'Tom Brown', 'Canada','10103')
]

customer_columns = ["CUSTOMER_ID", "NAME", "COUNTRY","ORDERNUMBER"]

customer_details = spark.createDataFrame(customer_data, customer_columns)


broadcast_customer_details = broadcast(customer_details)


joined_df = sales_df1.join(
    broadcast_customer_details,
    sales_df1.ORDERNUMBER == broadcast_customer_details.ORDERNUMBER,
    how='left'
)


joined_df.show()
```

```
+-----------+--------------+---------+--------------+-------+------------------+-------+------+--------+-------
+----------+----+-----------+--------------------+----------------+--------------------+------------+-----------
-+--------+----------+---------+---------+--------------+----------------+--------+--------+-----+----+----------
-+----+-------+-----------+
|ORDERNUMBER|QUANTITYORDERED|PRICEEACH|ORDERLINENUMBER|  SALES|          ORDERDATE| STATUS|QTR_ID|MONTH_ID|YEAR_I
D|PRODUCTLINE|MSRP|PRODUCTCODE|        CUSTOMERNAME|           PHONE|        ADDRESSLINE1|ADDRESSLINE2|         CI
TY|   STATE|POSTALCODE|  COUNTRY|TERRITORY|CONTACTLASTNAME|CONTACTFIRSTNAME|DEALSIZE|Category|MONTH|YEAR|CUSTOMER_
ID|NAME|COUNTRY|ORDERNUMBER|
+-----------+--------------+---------+--------------+-------+------------------+-------+------+--------+-------
+----------+----+-----------+--------------------+----------------+--------------------+------------+-----------
-+--------+----------+---------+---------+--------------+----------------+--------+--------+-----+----+----------
-+----+-------+-----------+
|      10107|             30|     95.7|             2| 2871.0|2003-02-24 00:00:00|Shipped|     1|       2|   200
3|Motorcycles|  95|   S10_1678|   Land of Toys Inc.|      2125557818|897 Long Airport ...|        null|          N
YC|      NY|     10022|      USA|       NA|             Yu|            Kwai|   Small|  Medium|    2|2003|       nu
ll|null|   null|       null|
|      10121|             34|    81.35|             5| 2765.9|2003-05-07 00:00:00|Shipped|     2|       5|   200
3|Motorcycles|  95|   S10_1678|  Reims Collectables|      26.47.1555|  59 rue de l'Abbaye|        null|        Rei
ms|     null|     51100|   France|     EMEA|        Henriot|            Paul|   Small|  Medium|    5|2003|       nu
ll|null|   null|       null|
|      10134|             41|    94.74|             2|3884.34|2003-07-01 00:00:00|Shipped|     3|       7|   200
3|Motorcycles|  95|   S10_1678|     Lyon Souveniers|+33 1 46 62 7555|27 rue du Colonel...|        null|        Par
is|     null|     75508|   France|     EMEA|       Da Cunha|          Daniel|  Medium|    High|    7|2003|       nu
ll|null|   null|       null|
|      10145|             45|    83.26|             6| 3746.7|2003-08-25 00:00:00|Shipped|     3|       8|   200
3|Motorcycles|  95|   S10_1678|   Toys4GrownUps.com|      6265557265|  78934 Hillside Dr.|        null|     Pasade
na|      CA|     90003|      USA|       NA|          Young|           Julie|  Medium|    High|    8|2003|       nu
ll|null|   null|       null|
|      10159|             49|    100.0|            14|5205.27|2003-10-10 00:00:00|Shipped|     4|      10|   200
3|Motorcycles|  95|   S10_1678|Corporate Gift Id...|      6505551386|     7734 Strong St.|        null|San Francis
co|      CA|      null|      USA|       NA|          Brown|           Julie|  Medium|    High|   10|2003|       nu
ll|null|   null|       null|
|      10168|             36|    96.66|             1|3479.76|2003-10-28 00:00:00|Shipped|     4|      10|   200
3|Motorcycles|  95|   S10_1678|Technics Stores Inc.|      6505556809|   9408 Furth Circle|        null|   Burlinga
me|      CA|     94217|      USA|       NA|         Hirano|            Juri|  Medium|    High|   10|2003|       nu
ll|null|   null|       null|
|      10180|             29|    86.13|             9|2497.77|2003-11-11 00:00:00|Shipped|     4|      11|   200
3|Motorcycles|  95|   S10_1678|Daedalus Designs ...|      20.16.1555|184, chausse de T...|        null|        Lil
le|     null|     59000|   France|     EMEA|          Rance|         Martine|   Small|  Medium|   11|2003|       nu
ll|null|   null|       null|
|      10188|             48|    100.0|             1|5512.32|2003-11-18 00:00:00|Shipped|     4|      11|   200
3|Motorcycles|  95|   S10_1678|        Herkku Gifts|   +47 2267 3215|Drammen 121, PR 7...|        null|       Berg
en|     null|    N 5804|   Norway|     EMEA|         Oeztan|          Veysel|  Medium|    High|   11|2003|       nu
ll|null|   null|       null|
|      10201|             22|    98.57|             2|2168.54|2003-12-01 00:00:00|Shipped|     4|      12|   200
3|Motorcycles|  95|   S10_1678|     Mini Wheels Co.|      6505555787|5557 North Pendal...|        null|San Francis
co|      CA|      null|      USA|       NA|         Murphy|           Julie|   Small|     Low|   12|2003|       nu
ll|null|   null|       null|
|      10211|             41|    100.0|            14|4708.44|2004-01-15 00:00:00|Shipped|     1|       1|   200
4|Motorcycles|  95|   S10_1678|    Auto Canal Petit|  (1) 47.55.6555|   25, rue Lauriston|        null|        Par
is|     null|     75016|   France|     EMEA|        Perrier|       Dominique|  Medium|    High|    1|2004|       nu
ll|null|   null|       null|
|      10223|             37|    100.0|             1|3965.66|2004-02-20 00:00:00|Shipped|     1|       2|   200
4|Motorcycles|  95|   S10_1678|Australian Collec...|    03 9520 4555|   636 St Kilda Road|     Level 3|    Melbour
ne|Victoria|      3004|Australia|     APAC|       Ferguson|           Peter|  Medium|    High|    2|2004|       nu
ll|null|   null|       null|
|      10237|             23|    100.0|             7|2333.12|2004-04-05 00:00:00|Shipped|     2|       4|   200
4|Motorcycles|  95|   S10_1678|     Vitachrome Inc.|      2125551500|   2678 Kingston Rd.|   Suite 101|          N
```

```
YC|      NY|     10022|      USA|       NA|          Frick|         Michael|   Small|  Medium|    4|2004|       nu
ll|null|   null|       null|
|      10251|            28|    100.0|             2|3188.64|2004-05-18 00:00:00|Shipped|     2|       5|   200
4|Motorcycles|  95|   S10_1678|Tekni Collectable...|      2015559350|       7476 Moss Rd.|        null|       Newa
rk|      NJ|     94019|      USA|       NA|          Brown|         William|  Medium|    High|    5|2004|       nu
ll|null|   null|       null|
|      10263|            34|    100.0|             2|3676.76|2004-06-28 00:00:00|Shipped|     2|       6|   200
4|Motorcycles|  95|   S10_1678|     Gift Depot Inc.|      2035552570| 25593 South Bay Ln.|        null|  Bridgewat
er|      CT|     97562|      USA|       NA|           King|           Julie|  Medium|    High|    6|2004|       nu
ll|null|   null|       null|
|      10275|            45|    92.83|             1|4177.35|2004-07-23 00:00:00|Shipped|     3|       7|   200
4|Motorcycles|  95|   S10_1678|   La Rochelle Gifts|      40.67.8555|67, rue des Cinqu...|        null|       Nant
es|     null|     44000|   France|     EMEA|        Labrune|          Janine|  Medium|    High|    7|2004|       nu
ll|null|   null|       null|
|      10285|            36|    100.0|             6|4099.68|2004-08-27 00:00:00|Shipped|     3|       8|   200
4|Motorcycles|  95|   S10_1678|Marta's Replicas Co.|      6175558555| 39323 Spinnaker Dr.|        null|    Cambrid
ge|      MA|     51247|      USA|       NA|      Hernandez|           Marta|  Medium|    High|    8|2004|       nu
ll|null|   null|       null|
|      10299|            23|    100.0|             9|2597.39|2004-09-30 00:00:00|Shipped|     3|       9|   200
4|Motorcycles|  95|   S10_1678|Toys of Finland, Co.|     90-224 8555|       Keskuskatu 45|        null|     Helsin
ki|     null|     21240|  Finland|     EMEA|      Karttunen|           Matti|   Small|  Medium|    9|2004|       nu
ll|null|   null|       null|
|      10309|            41|    100.0|             5|4394.38|2004-10-15 00:00:00|Shipped|     4|      10|   200
4|Motorcycles|  95|   S10_1678|  Baane Mini Imports|      07-98 9555|Erling Skakkes ga...|        null|      Stave
rn|     null|      4110|   Norway|     EMEA|     Bergulfsen|           Jonas|  Medium|    High|   10|2004|       nu
ll|null|   null|       null|
|      10318|            46|    94.74|             1|4358.04|2004-11-02 00:00:00|Shipped|     4|      11|   200
4|Motorcycles|  95|   S10_1678|Diecast Classics ...|      2155551555|    7586 Pompton St.|        null|    Allento
wn|      PA|     70267|      USA|       NA|             Yu|           Kyung|  Medium|    High|   11|2004|       nu
ll|null|   null|       null|
|      10329|            42|    100.0|             1|4396.14|2004-11-15 00:00:00|Shipped|     4|      11|   200
4|Motorcycles|  95|   S10_1678|   Land of Toys Inc.|      2125557818|897 Long Airport ...|        null|          N
YC|      NY|     10022|      USA|       NA|             Yu|            Kwai|  Medium|    High|   11|2004|       nu
ll|null|   null|       null|
+-----------+---------------+---------+---------------+-------+------------------+-------+------+--------+-------
+-----------+----+-----------+-------------------+---------------+--------------------+-----------+-----------
-+--------+----------+---------+---------+---------------+---------------+--------+--------+-----+----+----------
-+----+-------+-----------+
only showing top 20 rows
```

Key Considerations

- The smaller table should be small enough to fit in memory on all nodes. Typically, this means it should be under a few GBs.
- Broadcasting large tables can use a lot of memory on each node
- Spark has a limit on how big the broadcast table can be.

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.

In [372…

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf
from pyspark.sql.types import *

average_sales = sales_df1.select(avg("SALES")).collect()[0][0]

def categorize_sales_by_avg(sales):
    if sales < average_sales:
        return 'Low'
    elif sales == average_sales:
        return 'Medium'
    else:
        return 'High'

categorize_sales_by_avg_udf = udf(categorize_sales_by_avg, StringType())

df = sales_df1.withColumn("CATEGORY", categorize_sales_by_avg_udf(col("SALES")))

category_counts = df.groupBy("COUNTRY", "CATEGORY").count()

category_counts.orderBy(col('count').desc()).show()
```

```
+---------+--------+-----+
|  COUNTRY|CATEGORY|count|
+---------+--------+-----+
|      USA|     Low|  581|
|      USA|    High|  423|
|    Spain|     Low|  202|
```

```
|   France|     Low|  183|
|    Spain|    High|  140|
|   France|    High|  131|
|Australia|     Low|  110|
|       UK|     Low|   87|
|Australia|    High|   75|
|    Italy|     Low|   74|
|       UK|    High|   57|
|  Finland|     Low|   53|
|   Canada|     Low|   48|
|   Norway|     Low|   46|
|Singapore|     Low|   45|
|  Denmark|     Low|   39|
|  Finland|    High|   39|
|   Norway|    High|   39|
|    Italy|    High|   39|
|  Germany|     Low|   35|
+---------+--------+-----+
only showing top 20 rows
```

Insights: USA has the higher no.of orders

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.

In [373…

```
def apply_discount(productline, sales):
    if productline == 'Classic Cars':
        return sales * 0.90
    elif productline == 'Motorcycles':
        return sales * 0.95
    else:
        return sales

apply_discount_udf = udf(apply_discount, DoubleType())

df = sales_df1.withColumn("DISCOUNTED_SALES", apply_discount_udf(col("PRODUCTLINE"), col("SALES")))

df.select(['PRODUCTLINE','SALES','DISCOUNTED_SALES']).show()
```

```
+-----------+-------+------------------+
|PRODUCTLINE|  SALES|  DISCOUNTED_SALES|
+-----------+-------+------------------+
|Motorcycles| 2871.0|           2727.45|
|Motorcycles| 2765.9|          2627.605|
|Motorcycles|3884.34|          3690.123|
|Motorcycles| 3746.7|          3559.365|
|Motorcycles|5205.27|         4945.0065|
|Motorcycles|3479.76|          3305.772|
|Motorcycles|2497.77|         2372.8815|
|Motorcycles|5512.32|          5236.704|
|Motorcycles|2168.54|          2060.113|
|Motorcycles|4708.44| 4473.017999999999|
|Motorcycles|3965.66|3767.3769999999995|
|Motorcycles|2333.12|          2216.464|
|Motorcycles|3188.64|3029.2079999999996|
|Motorcycles|3676.76|          3492.922|
|Motorcycles|4177.35|         3968.4825|
|Motorcycles|4099.68|          3894.696|
|Motorcycles|2597.39|2467.5204999999996|
|Motorcycles|4394.38|          4174.661|
|Motorcycles|4358.04|          4140.138|
|Motorcycles|4396.14|4176.3330000000005|
+-----------+-------+------------------+
only showing top 20 rows
```

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?

In [400…

```
from datetime import datetime

last_processed_date = '2024-09-15'

spark.sparkContext.setCheckpointDir("file:///home/hadoop/Downloads/incremental/checkpoint")

new_df=spark.read.csv("file:///home/hadoop/Downloads/incremental/sales_data_sample.csv",header=True,
                      inferSchema = True).filter(col("ORDERDATE") > last_processed_date)

new_df = new_df.checkpoint()

new_df.write.format("parquet").mode("append").save("file:///home/hadoop/Downloads/incremental/")

new_last_processed_date = datetime.now().strftime('%Y-%m-%d')
print(f"Updated last processed date to {new_last_processed_date}")
```

```
Updated last processed date to 2024-09-18
```

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?

In [415…

```
window_spec = Window.partitionBy("CUSTOMERNAME").orderBy("SALES")

df_with_cdf = sales_df1.withColumn("Rank", percent_rank().over(window_spec)) \
                .withColumn("CDF", col("Rank"))

df_with_cdf_sorted = df_with_cdf.orderBy(col("CUSTOMERNAME"),
                                         col("CDF"))


df_with_cdf_sorted.select(['CUSTOMERNAME','YEAR','MONTH', 'SALES', 'CDF']).show(100)
```

```
+--------------------+----+-----+-------+--------------------+
|        CUSTOMERNAME|YEAR|MONTH|  SALES|                 CDF|
+--------------------+----+-----+-------+--------------------+
|      AV Stores, Co.|2003|    3|  710.2|                 0.0|
|      AV Stores, Co.|2004|   11|1152.58|                0.02|
|      AV Stores, Co.|2004|   11|1264.08|                0.04|
|      AV Stores, Co.|2004|   11|1307.32|                0.06|
|      AV Stores, Co.|2004|   11|1538.55|                0.08|
|      AV Stores, Co.|2004|   11| 1592.0|                 0.1|
|      AV Stores, Co.|2003|    3| 1608.0|                0.12|
|      AV Stores, Co.|2003|    3|1654.56|                0.14|
|      AV Stores, Co.|2003|    3|1721.73|                0.16|
|      AV Stores, Co.|2003|    3|1729.65|                0.18|
|      AV Stores, Co.|2004|   11| 1759.2|                 0.2|
|      AV Stores, Co.|2004|   11|1859.44|                0.22|
|      AV Stores, Co.|2003|    3|1987.74|                0.24|
|      AV Stores, Co.|2004|   10|2051.56|                0.26|
|      AV Stores, Co.|2004|   10|2082.85|                0.28|
|      AV Stores, Co.|2004|   11|2223.52|                 0.3|
|      AV Stores, Co.|2004|   11|2264.15|                0.32|
|      AV Stores, Co.|2004|   10|2315.18|                0.34|
|      AV Stores, Co.|2004|   11|2427.03|                0.36|
|      AV Stores, Co.|2003|    3|2499.56|                0.38|
|      AV Stores, Co.|2003|    3|2574.18|                 0.4|
|      AV Stores, Co.|2004|   10|2606.48|                0.42|
|      AV Stores, Co.|2004|   10| 2700.0|                0.44|
|      AV Stores, Co.|2004|   10| 2884.8|                0.46|
|      AV Stores, Co.|2004|   11|2979.08|                0.48|
|      AV Stores, Co.|2003|    3| 3069.0|                 0.5|
|      AV Stores, Co.|2004|   11|3166.84|                0.52|
|      AV Stores, Co.|2004|   11| 3201.5|                0.54|
|      AV Stores, Co.|2004|   10| 3207.4|                0.56|
|      AV Stores, Co.|2004|   10|3232.31|                0.58|
|      AV Stores, Co.|2003|    3|3360.45|                 0.6|
|      AV Stores, Co.|2004|   11|3382.08|                0.62|
|      AV Stores, Co.|2004|   11|3472.98|                0.64|
|      AV Stores, Co.|2004|   10|3488.78|                0.66|
|      AV Stores, Co.|2004|   10| 3515.7|                0.68|
|      AV Stores, Co.|2004|   10| 3525.6|                 0.7|
|      AV Stores, Co.|2004|   10|3600.65|                0.72|
|      AV Stores, Co.|2004|   10| 3633.4|                0.74|
|      AV Stores, Co.|2004|   10| 3670.4|                0.76|
|      AV Stores, Co.|2004|   11|3685.95|                0.78|
|      AV Stores, Co.|2003|    3|3724.42|                 0.8|
|      AV Stores, Co.|2004|   10|3759.04|                0.82|
|      AV Stores, Co.|2003|    3|3859.68|                0.84|
|      AV Stores, Co.|2004|   11|4375.98|                0.86|
```

```
|      AV Stores, Co.|2004|   10| 4982.7|                0.88|
|      AV Stores, Co.|2003|    3|5074.39|                 0.9|
|      AV Stores, Co.|2003|    3|5433.08|                0.92|
|      AV Stores, Co.|2003|    3|5942.28|                0.94|
|      AV Stores, Co.|2003|    3| 6069.0|                0.96|
|      AV Stores, Co.|2004|   10|6570.76|                0.98|
|      AV Stores, Co.|2004|   11| 7310.0|                 1.0|
|        Alpha Cognac|2005|    3| 1463.0|                 0.0|
|        Alpha Cognac|2003|   11|1514.52| 0.05263157894736842|
|        Alpha Cognac|2003|   11| 1930.5| 0.10526315789473684|
|        Alpha Cognac|2003|   11|1988.28| 0.15789473684210525|
|        Alpha Cognac|2005|    3|2116.16| 0.21052631578947367|
|        Alpha Cognac|2003|   11| 2169.9|  0.2631578947368421|
|        Alpha Cognac|2003|   11|2301.75|  0.3157894736842105|
|        Alpha Cognac|2005|    3| 2577.6|  0.3684210526315789|
|        Alpha Cognac|2003|   11|2748.56| 0.42105263157894735|
|        Alpha Cognac|2003|   11|3293.24| 0.47368421052631576|
|        Alpha Cognac|2003|   11|3350.52|  0.5263157894736842|
|        Alpha Cognac|2003|   11|3431.25|  0.5789473684210527|
|        Alpha Cognac|2003|   11|3492.48|   0.631578947368421|
|        Alpha Cognac|2003|    7|3644.75|  0.6842105263157895|
|        Alpha Cognac|2005|    3|3789.72|  0.7368421052631579|
|        Alpha Cognac|2005|    3|5192.64|  0.7894736842105263|
|        Alpha Cognac|2003|    7|5274.72|  0.8421052631578947|
|        Alpha Cognac|2003|   11|5386.56|  0.8947368421052632|
|        Alpha Cognac|2003|   11|6490.68|  0.9473684210526315|
|        Alpha Cognac|2003|    7|8331.61|                 1.0|
|  Amica Models & Co.|2004|    8|  577.6|                 0.0|
|  Amica Models & Co.|2004|    8|1381.05|                0.04|
|  Amica Models & Co.|2004|    8|1557.36|                0.08|
|  Amica Models & Co.|2004|    8| 1574.0|                0.12|
|  Amica Models & Co.|2004|    8|1656.69|                0.16|
|  Amica Models & Co.|2004|    9|1921.92|                 0.2|
|  Amica Models & Co.|2004|    9|2084.81|                0.24|
|  Amica Models & Co.|2004|    8|2137.05|                0.28|
|  Amica Models & Co.|2004|    9|2418.24|                0.32|
|  Amica Models & Co.|2004|    8|2800.08|                0.36|
|  Amica Models & Co.|2004|    9|2819.28|                 0.4|
|  Amica Models & Co.|2004|    9|2941.89|                0.44|
|  Amica Models & Co.|2004|    8|2954.53|                0.48|
|  Amica Models & Co.|2004|    8|3006.43|                0.52|
|  Amica Models & Co.|2004|    8|3474.46|                0.56|
|  Amica Models & Co.|2004|    8| 3668.6|                 0.6|
|  Amica Models & Co.|2004|    8|3704.05|                0.64|
|  Amica Models & Co.|2004|    9|4242.24|                0.68|
|  Amica Models & Co.|2004|    8| 4455.0|                0.72|
|  Amica Models & Co.|2004|    8| 4750.8|                0.76|
|  Amica Models & Co.|2004|    9|4946.06|                 0.8|
|  Amica Models & Co.|2004|    8|5126.24|                0.84|
|  Amica Models & Co.|2004|    8| 5239.5|                0.88|
|  Amica Models & Co.|2004|    8|8014.82|                0.92|
|  Amica Models & Co.|2004|    9| 8253.0|                0.96|
|  Amica Models & Co.|2004|    9|8411.56|                 1.0|
|Anna's Decoration...|2003|   11|1035.58|                 0.0|
|Anna's Decoration...|2005|    3| 1448.0|0.022222222222222223|
|Anna's Decoration...|2003|    9| 1506.5|0.044444444444444446|
+--------------------+----+-----+-------+--------------------+
only showing top 100 rows
```

The data provides the CDF over sales for each customername, Here we can get an insight of the highest sales and lowest sales of each customername

Insights:

AV Stores, Co. has a high sales figure in November 2004, indicating a peak period

Alpha Cognac shows a significant drop in sales in 2005 compared to 2003