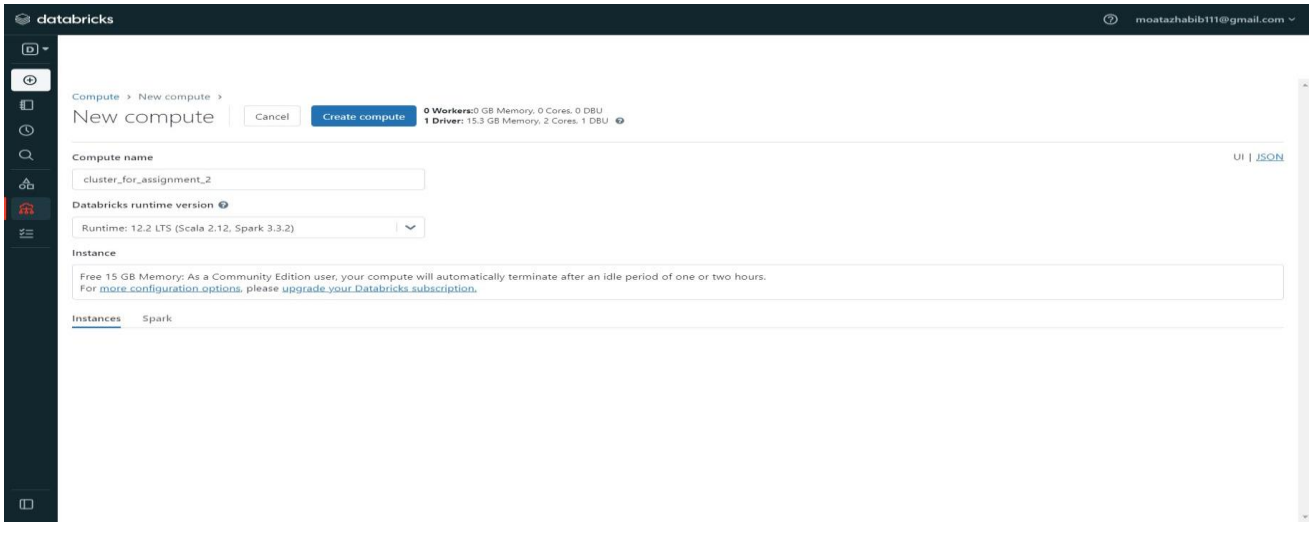
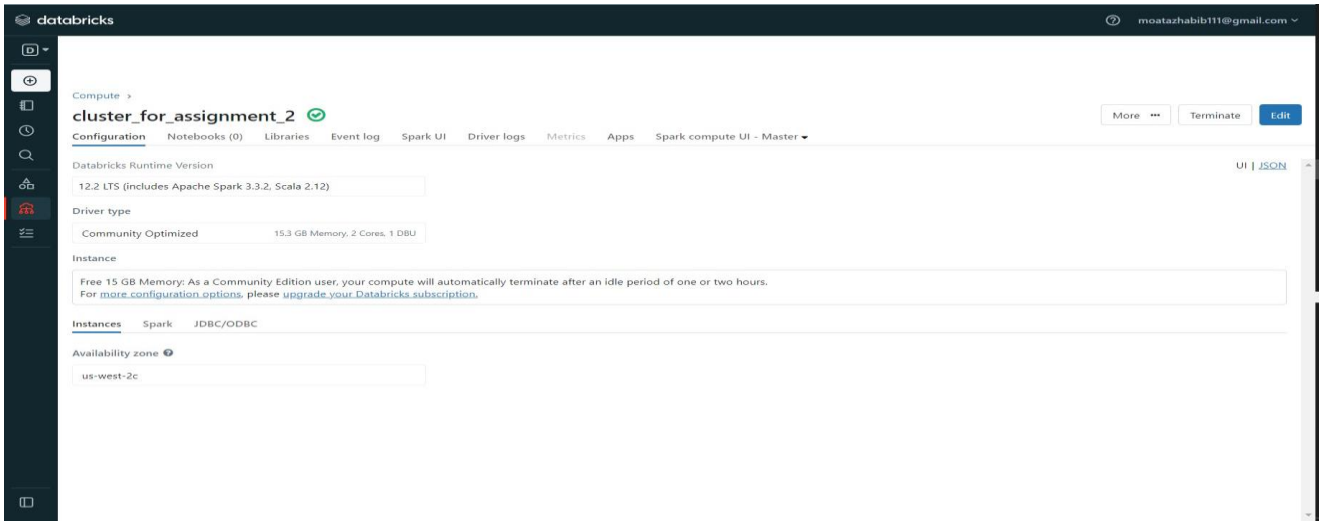
Part 2 – Spark Examples:

1. Data Transformation Pipelines:

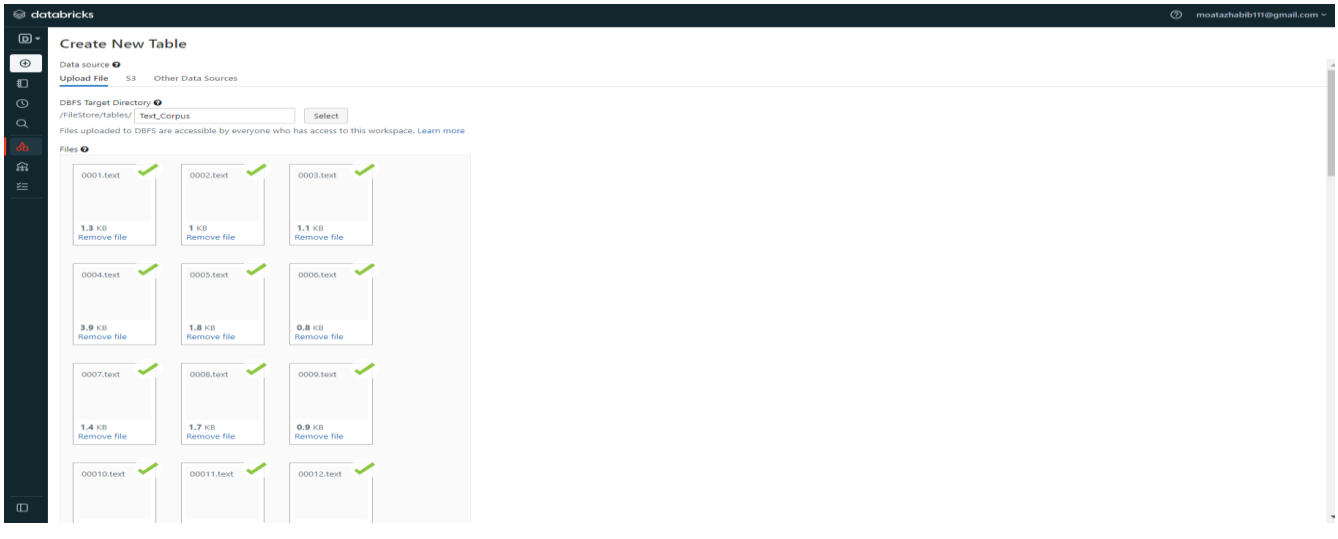
Cluster Setup:

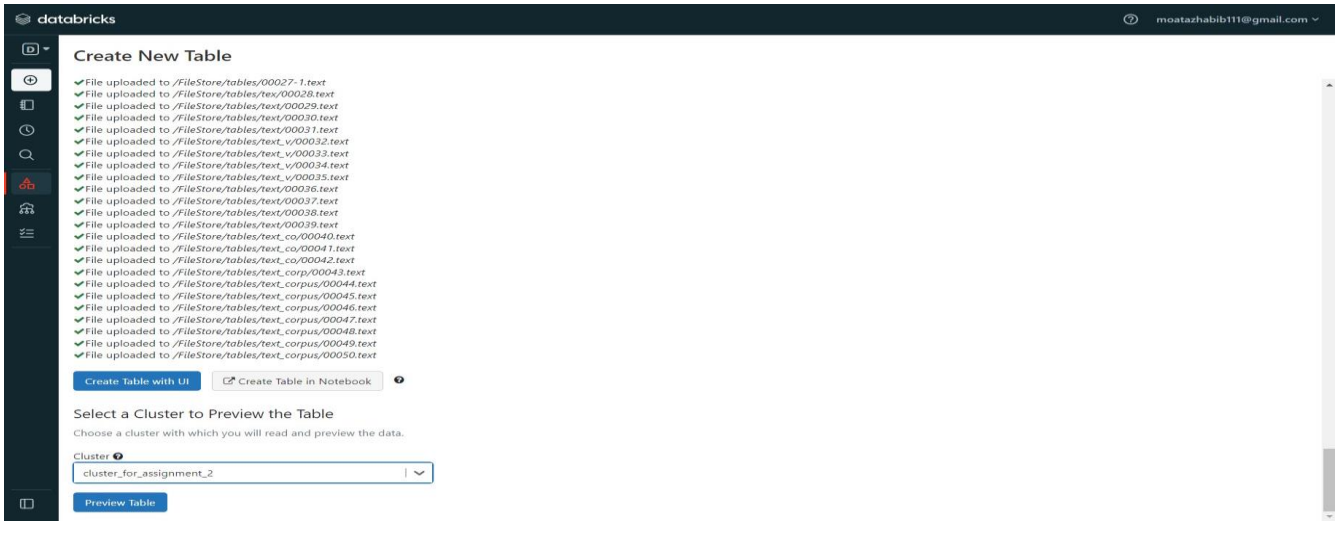


Cluster run Successfully



a) Uploaded the files to your DBFS table space.





b) Use Spark Scala to load your data into an RDD.

```
Cmd 1
Scala
val TextFile = spark.sparkContext.textFile("/FileStore/tables/Text_Corpus/*.text")

TextFile: org.apache.spark.rdd.RDD[String] = /FileStore/tables/Text_Corpus/*.text MapPartitionsRDD[1] at textFile at command-2483502180336441:1
Command took 2.44 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 2:14:24 PM on cluster_for_assignment_2
Cmd 2
```

c) Count the number of lines across all the files.

```
Cmd 2
TextFile.count()

(1) Spark Jobs
res5: Long = 1645
Command took 7.22 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 2:14:26 PM on cluster_for_assignment_2
```

d) Find the number of occurrences of the word “antibiotics”.

```
Scala
val antibioticsCount = TextFile.filter(line => line.contains("antibiotics")).count()

(1) Spark Jobs
antibioticsCount: Long = 2
Command took 3.51 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 2:15:00 PM on cluster_for_assignment_2
```

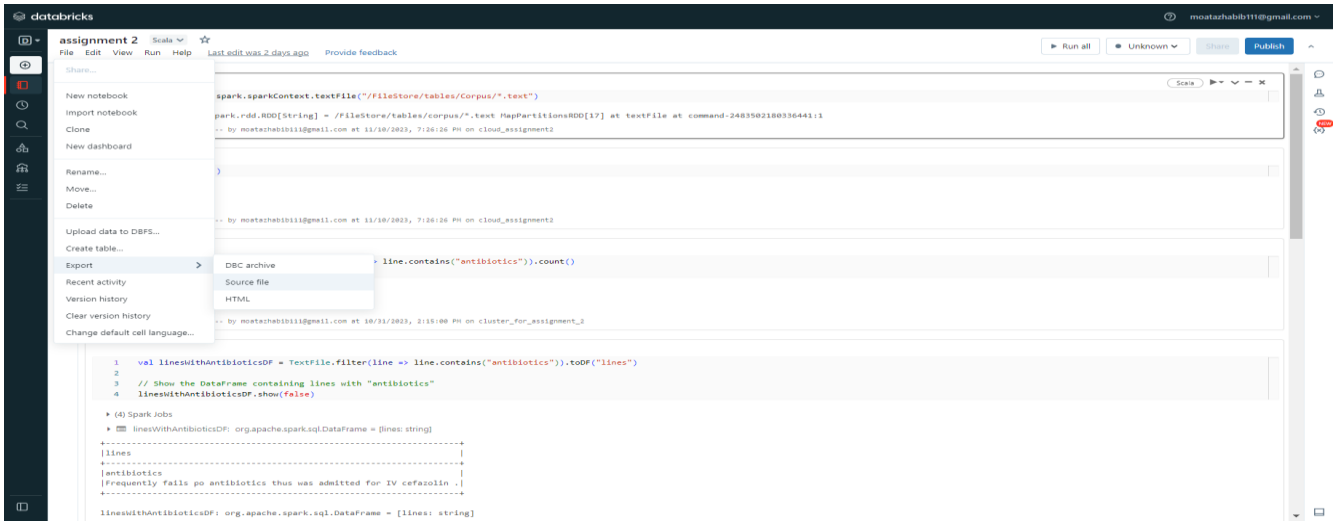
```
Cmd 4
Scala
val linesWithAntibioticsDF = TextFile.filter(line => line.contains("antibiotics")).toDF("lines")

// Show the DataFrame containing lines with "antibiotics"
linesWithAntibioticsDF.show(false)

(4) Spark Jobs
linesWithAntibioticsDF: org.apache.spark.sql.DataFrame = [lines: string]
+-----------------------------------------------------------------+
|lines                                                            |
+-----------------------------------------------------------------+
|antibiotics                                                      |
|Frequently fails po antibiotics thus was admitted for IV cefazolin .|
+-----------------------------------------------------------------+

linesWithAntibioticsDF: org.apache.spark.sql.DataFrame = [lines: string]
Command took 13.62 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 2:27:44 PM on cluster_for_assignment_2
```

e) Count the occurrence of the word “patient” and “admitted” on the same line of text. Please ensure that your code contains at least 2 transformation functions in a pipeline.

```
Cmd 5
Scala
val patientAdmittedCount = TextFile.filter(line => line.contains("patient"))
                                   .filter(line => line.contains("admitted"))
patientAdmittedCount.count()

(1) Spark Jobs
patientAdmittedCount: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[11] at filter at command-2483502180336446:2
res12: Long = 7
Command took 2.42 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 2:33:08 PM on cluster_for_assignment_2
```

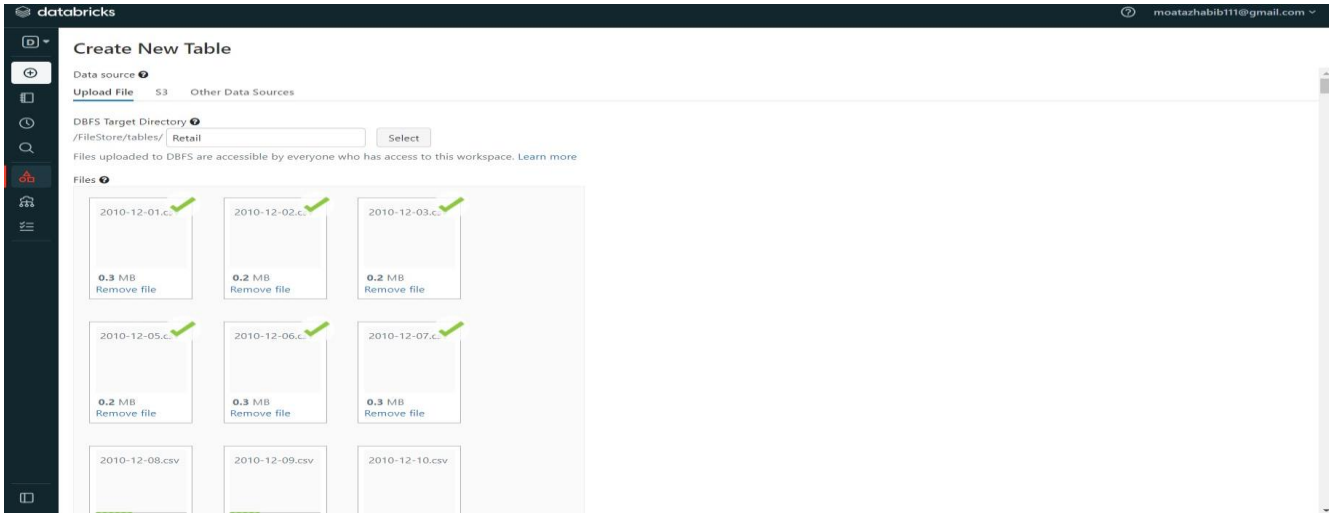
```
Scala
// Filter lines containing both "patient" and "admitted", and create a DataFrame
val linesWithPatientAdmittedDF = TextFile.filter(line => line.contains("patient"))
                                         .filter(line => line.contains("admitted")).toDF("lines")

// Show the DataFrame containing lines with both "patient" and "admitted"
linesWithPatientAdmittedDF.show(false)

(4) Spark Jobs
linesWithPatientAdmittedDF: org.apache.spark.sql.DataFrame = [lines: string]
+-------------------------------------------------------------------------------------------------------------------------------------+
|lines                                                                                                                                |
+-------------------------------------------------------------------------------------------------------------------------------------+
|Your patient was admitted under the care of   Swenk , Danyel A  with a preliminary diagnosis of L HIP FX .                          |
|The patient 's disposition at the end of the visit was admitted as an inpatient to  West Texas Va Health Care System  .             |
|The patient was admitted for the lap coli with an intraoperative cholangiogram by Dr.    Ellenburg   under general anesthesia .       |
|The patient was admitted to floor .                                                                                                  |
|The patient was admitted to the Medical Service for management of her congestive heart failure .                                    |
|DISCHARGE DATE : The patient was admitted to the hospital on  July 15th  for a chole with a principle diagnosis of recurrent biliary colic .|
|The patient was admitted in  December  of 1999 , at that time with anasarca and congestive heart failure , responsive to diuretics and ACE inhibitors .|
+-------------------------------------------------------------------------------------------------------------------------------------+

linesWithPatientAdmittedDF: org.apache.spark.sql.DataFrame = [lines: string]
Command took 4.75 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 3:12:32 PM on cluster_for_assignment_2
```

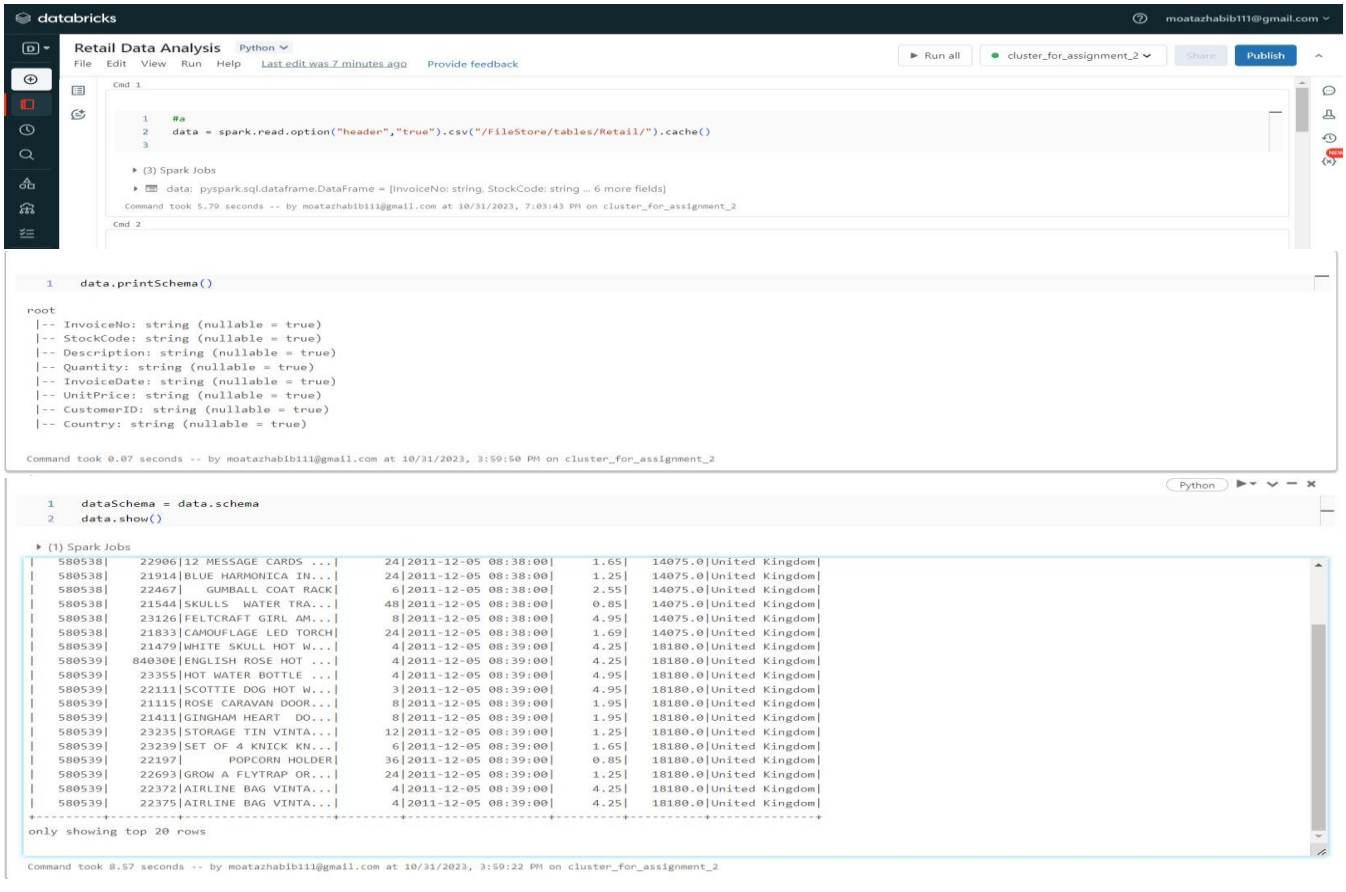
f) exported (. scala format) notebook



2) Retail Data Analysis

a) Uploaded the files to your DBFS table space.





b) Output the total number of transactions across all the files and the total value of the transactions.

Total number of rows= 541909

Total number of unique InvoiceNo= 25900

Total Number of transactions = Sum(Quantity)= 5176450

```python
#b
# Number of rows
num_rows = data.count()

# Number of distinct InvoiceNo
num_distinct_invoice = data.select("InvoiceNo").distinct().count()

print("Number of rows:", num_rows)
print("Number of distinct InvoiceNo:", num_distinct_invoice)
```

(5) Spark Jobs

```
Number of rows: 541909
Number of distinct InvoiceNo: 25900
```

Command took 2.58 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:04:24 PM on cluster_for_assignment_2

Cmd 5

```python
# number of transactions
data.createOrReplaceTempView("Retail")
q="SELECT SUM (Quantity) as TotalTransactions FROM Retail;"
sqlDF = spark.sql(q)
sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [TotalTransactions: double]

```
+-----------------+
|TotalTransactions|
+-----------------+
|        5176450.0|
+-----------------+
```

The total value of the transactions = sum((Quantity * UnitPrice)) = 9747747.93999462

Cmd 6

```python
# Total value of transactions as DataFrame
total_value_df = data.withColumn("TotalValue", (col("Quantity") * col("UnitPrice"))).groupBy().agg(sum("TotalValue").alias("TotalValue"))
total_value_df.show()
```

(2) Spark Jobs

total_value_df: pyspark.sql.dataframe.DataFrame = [TotalValue: double]

```
+-----------------+
|       TotalValue|
+-----------------+
|9747747.933999462|
+-----------------+
```

Command took 1.13 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 6:09:50 PM on cluster_for_assignment_2

Cmd 7

c) Output the 5 top-selling products.

```python
# c
data.createOrReplaceTempView("Retail")
q="SELECT StockCode, SUM (Quantity) as TotalQuantity FROM Retail GROUP BY StockCode ORDER BY TotalQuantity DESC LIMIT 5;"
sqlDF = spark.sql(q)
sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [StockCode: string, TotalQuantity: double]

```
+---------+-------------+
|StockCode|TotalQuantity|
+---------+-------------+
|    22197|      56450.0|
|    84077|      53847.0|
|   85099B|      47363.0|
|   85123A|      38830.0|
|    84879|      36221.0|
+---------+-------------+
```

Command took 3.42 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 4:17:51 PM on cluster_for_assignment_2

d) Output the 5 topmost valuable products.

```python
# d
data.createOrReplaceTempView("Retail")

query = """
    SELECT StockCode, SUM (Quantity*UnitPrice) AS TotalPrice
    FROM Retail
    GROUP BY StockCode
    ORDER BY TotalPrice DESC
    LIMIT 5
"""

sqlDF = spark.sql(query)

sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [StockCode: string, TotalPrice: double]

```
+---------+-----------------+
|StockCode|       TotalPrice|
+---------+-----------------+
|      DOT|        206245.48|
|    22423|        164762.19|
|    47566|         98302.98|
|   85123A|          97894.5|
|   85099B|92356.02999999994|
+---------+-----------------+
```

Command took 1.63 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:16:45 PM on cluster_for_assignment_2

e) Output each country and the total value of their purchases.

```python
data.createOrReplaceTempView("Retail")
query = """
    SELECT Country, SUM (Quantity*UnitPrice) AS TotalValue
    FROM Retail
    GROUP BY Country
    ORDER BY TotalValue DESC
"""

sqlDF = spark.sql(query)
sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [Country: string, TotalValue: double]

```
+---------------+------------------+
|        Country|        TotalValue|
+---------------+------------------+
| United Kingdom|  8187806.36399976|
|    Netherlands| 284661.5399999999|
|           EIRE|263276.82000000024|
|        Germany|         221698.21|
|         France|          197403.9|
|      Australia|137077.26999999996|
|    Switzerland| 56385.34999999997|
|          Spain| 54774.57999999999|
|        Belgium|          40910.96|
|         Sweden|          36595.91|
|          Japan|35340.619999999995|
|         Norway| 35163.45999999999|
|       Portugal|29367.020000000004|
|        Finland|          22326.74|
|Channel Islands|20086.290000000005|
|        Denmark|18768.139999999996|
|          Italy|          16890.51|
|         Cyprus|12946.289999999997|
```

Command took 1.90 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:19:55 PM on cluster_for_assignment_2

most performing countries in purchases

```python
# e
data.createOrReplaceTempView("Retail")

query = """
    SELECT Country, SUM (Quantity*UnitPrice) AS TotalValue
    FROM Retail
    GROUP BY Country
"""
sqlDF = spark.sql(query)
sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [Country: string, TotalValue: double]

```
+---------------+------------------+
|        Country|        TotalValue|
+---------------+------------------+
|         Sweden|          36595.91|
|        Germany|         221698.21|
|         France|          197403.9|
|         Greece|           4710.52|
|        Belgium|          40910.96|
|        Finland|          22326.74|
|          Malta|2505.470000000003|
|   Unspecified|           4749.79|
|          Italy|          16890.51|
|           EIRE|263276.82000000024|
|         Norway| 35163.45999999999|
|          Spain| 54774.57999999999|
|        Denmark|18768.139999999996|
|      Hong Kong|          10117.04|
|        Iceland|           4310.0|
|Channel Islands|20086.290000000005|
|            USA|1730.919999999992|
|    Switzerland| 56385.34999999997|
```

f) Use a graphical representation to describe the result from step (d)

```python
# f
data.createOrReplaceTempView("Retail")
query = """
    SELECT StockCode, SUM (Quantity*UnitPrice) AS TotalPrice
    FROM Retail
    GROUP BY StockCode
    ORDER BY TotalPrice DESC
    LIMIT 5
"""
sqlDF = spark.sql(query)
sqlDF.show()
```

(2) Spark Jobs

sqlDF: pyspark.sql.dataframe.DataFrame = [StockCode: string, TotalPrice: double]

```
+---------+------------------+
|StockCode|        TotalPrice|
+---------+------------------+
|      DOT|         206245.48|
|    22423|         164762.19|
|   47566|          98302.98|
|  85123A|           97894.5|
|  85099B|92356.02999999994|
+---------+------------------+
```

Command took 2.93 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:26:28 PM on cluster_for_assignment_2

```python
display(sqlDF)
```

(2) Spark Jobs

Table

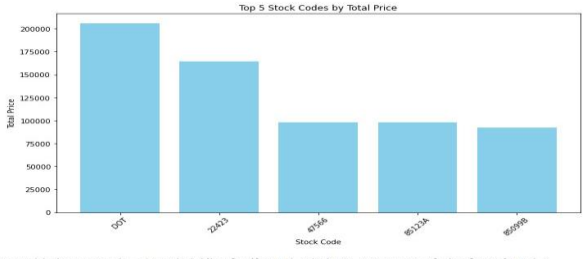
	StockCode	TotalPrice
1	DOT	206245.48
2	22423	164762.19
3	47566	98302.98
4	85123A	97894.5
5	85099B	92356.02999999994

5 rows | 2.21 seconds runtime

Refreshed 1 minute ago

Command took 2.21 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:40:21 PM on cluster_for_assignment_2

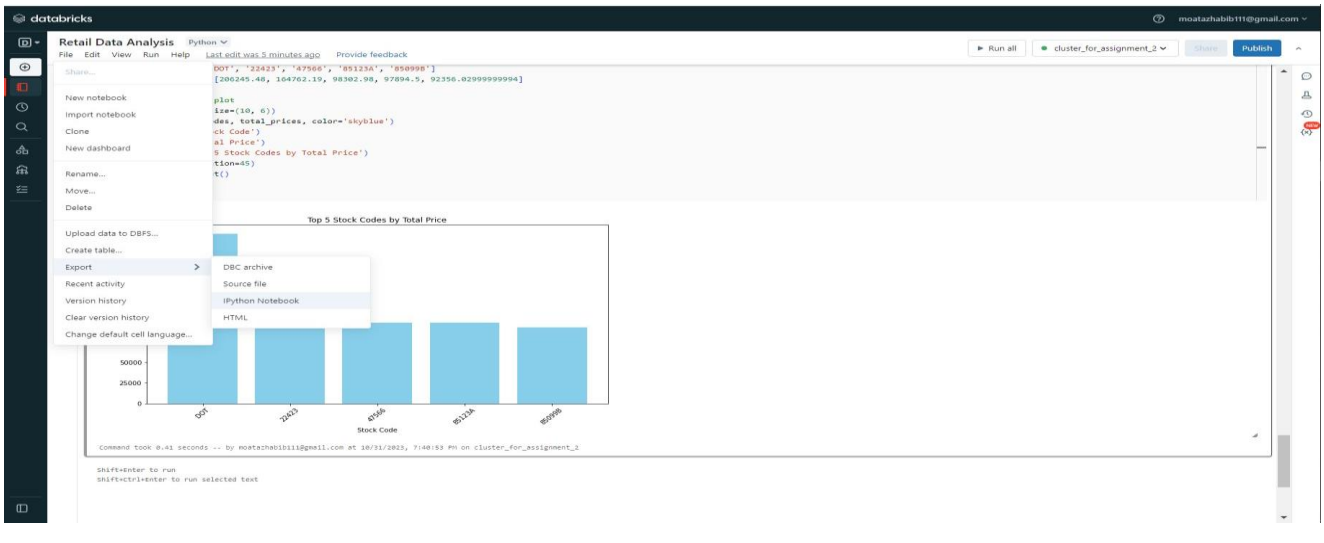
```python
import matplotlib.pyplot as plt

# Data from the SQL result
stockcodes = ['DOT', '22423', '47566', '85123A', '85099B']
total_prices = [206245.48, 164762.19, 98302.98, 97894.5, 92356.02999999994]

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(stockcodes, total_prices, color='skyblue')
plt.xlabel('Stock Code')
plt.ylabel('Total Price')
plt.title('Top 5 Stock Codes by Total Price')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Command took 0.41 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 7:40:53 PM on cluster_for_assignment_2

'DOT' StockCode is remarkably the most valuable product

g) save the notebook .



3. Structured Streaming:

Create a notebook

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import sum, current_timestamp
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, TimestampType

spark = SparkSession.builder.appName("RetailDataStream").getOrCreate()
```

Command took 0.11 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:14:00 PM on assignment 2

Load the retail data as a stream, at 20 files per trigger. For each batch pulled, capture the customer stock aggregates – total stocks, total value.

```python
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType, DoubleType
data_schema = StructType([
    StructField("InvoiceNo", IntegerType(), True),
    StructField("StockCode", StringType(), True),
    StructField("Description", StringType(), True),
    StructField("Quantity", IntegerType(), True),
    StructField("InvoiceDate", TimestampType(), True),
    StructField("UnitPrice", DoubleType(), True),
    StructField("CustomerID", StringType(), True),
    StructField("Country", StringType(), True),
])
```

Command took 0.13 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:14:02 PM on assignment 2

```python
streaming_data = (
    spark
    .readStream
    .format("csv")
    .schema(data_schema)
    .option("maxFilesPerTrigger",20)
    .option("header","true")
    .csv("/FileStore/tables/retail")
)
```

streaming_data: pyspark.sql.dataframe.DataFrame = [InvoiceNo: integer, StockCode: string ... 6 more fields]

Command took 0.46 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:14:24 PM on assignment 2

For each batch of the input stream, create a new stream that populates another dataframe or dataset with progress for each loaded set of data. This data set should have the columns – TriggerTime (Date/Time), Records Imported, Sale value (Total value of transactions)

```python

from pyspark.sql.functions import sum, col
customer_aggregates = (
    streaming_data
    .groupBy("CustomerID")
    .agg(
        sum(col("Quantity")).alias("TotalStocks"),
         sum(col("Quantity") * col("UnitPrice")).alias("TotalValue")
         )
    )
```

customer_aggregates: pyspark.sql.dataframe.DataFrame = [CustomerID: string, TotalStocks: long ... 1 more field]

Command took 0.15 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:14:35 PM on assignment 2

```python
customer_aggregates_result = (
                                customer_aggregates
                                .writeStream
                                .outputMode("complete")
                                .format("memory")
                                .queryName("customer_aggregates")
                                .start()
)
```

Cancel

(1) Spark Jobs

customer_aggregates (id: 5b4cff39-8665-446d-b3f8-77b39c181a6b) *Last updated: 20 seconds ago*

```python
spark.sql("SELECT * FROM customer_aggregates").show()
```

Cmd 7

```python
# create a new stream that populates another dataset with progress for each loaded set of data
from pyspark.sql.functions import current_timestamp,count
progress_data = (
    customer_aggregates
    .select(current_timestamp().alias("TriggerTime"),
    count("*").alias("RecordsImported"),
    sum("TotalValue").alias("SaleValue")
    )
)
progress_data_query = (
    progress_data
    .writeStream
    .outputMode("update")
    .format("memory")
    .queryName("progress_data")
    .start()
)
```

Python

```python
spark.sql("SELECT * FROM customer_aggregates").show()
```

(1) Spark Jobs

CustomerID	TotalStocks	TotalValue
15039.0	491	1317.7600000000004
16553.0	1067	1605.1900000000003
13178.0	388	709.9999999999999
17812.0	144	217.55000000000004
16083.0	384	337.98
14532.0	28	222.6
15002.0	279	631.6499999999999
18085.0	78	303.90000000000003
15070.0	36	106.2
17905.0	172	394.15
17377.0	254	610.0800000000002
13165.0	496	354.64
16718.0	413	623.7500000000002
12913.0	146	417.62
17450.0	786	2028.84
15329.0	-6	-17.7
15005.0	266	470.4100000000001
14491.0	40	127.2

Command took 6.38 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:53:38 PM on assignment 2

```python
spark.sql("SELECT * FROM progress_data").show()
```

(3) Spark Jobs

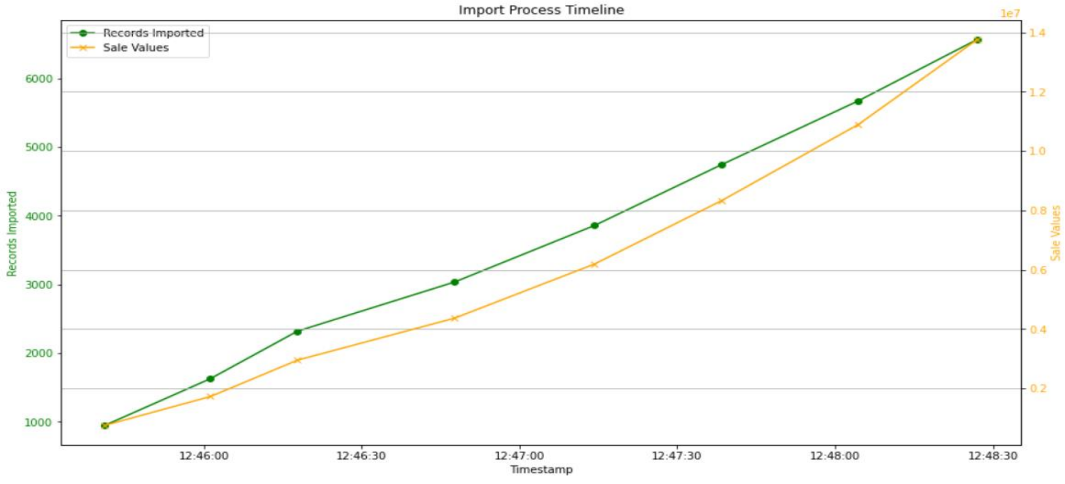
TriggerTime	RecordsImported	SaleValue
2023-11-12 12:45:...	949	748957.0200000012
2023-11-12 12:46:...	1629	1722763.4700000016
2023-11-12 12:46:...	2320	2948230.1800000025
2023-11-12 12:46:...	3036	4366905.1800000025
2023-11-12 12:47:...	3859	6192471.520000002
2023-11-12 12:47:...	4741	8321909.331000003
2023-11-12 12:48:...	5673	1.0900742242000002E7
2023-11-12 12:48:...	6568	1.3771209212000005E7
2023-11-12 12:48:...	7338	1.6760316212000007E7
2023-11-12 12:49:...	8167	2.0212073353000008E7
2023-11-12 12:49:...	8987	2.3835029493000001E7
2023-11-12 12:50:...	10009	2.8111866425000001E7
2023-11-12 12:50:...	11194	3.3117919076000001E7
2023-11-12 12:50:...	12504	3.8894408117000001E7
2023-11-12 12:51:...	13933	4.5635978987000002E7
2023-11-12 12:51:...	14393	5.0073378167000002E7
2023-11-12 12:52:...	14393	5.0073378167000002E7
2023-11-12 12:52:...	14393	5.0073378167000002E7

Command took 3.92 seconds -- by shahdmohamed67777@gmail.com at 11/12/2023, 2:54:05 PM on assignment 2

Use the dataset from step (c) to plot a line graph of the import process – showing two timelines – records imported and sale values.

```python
import matplotlib.pyplot as plt
import pandas as pd
progress_data_df = spark.table("progress_data").toPandas()

fig, ax1 = plt.subplots(figsize=(15, 8))

ax1.plot(progress_data_df["TriggerTime"], progress_data_df["RecordsImported"], label="Records Imported", color='green', marker='o')
ax1.set_xlabel("Timestamp")
ax1.set_ylabel("Records Imported", color='green')
ax1.tick_params(axis='y', labelcolor='green')

ax2 = ax1.twinx()

ax2.plot(progress_data_df["TriggerTime"], progress_data_df["SaleValue"], label="Sale Values", color='orange', marker='x')
ax2.set_ylabel("Sale Values", color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines + lines2, labels + labels2, loc='upper left')

plt.grid(True)
plt.title("Import Process Timeline")
plt.show()
```

▸ (1) Spark Jobs



▸ (1) Spark Jobs