

















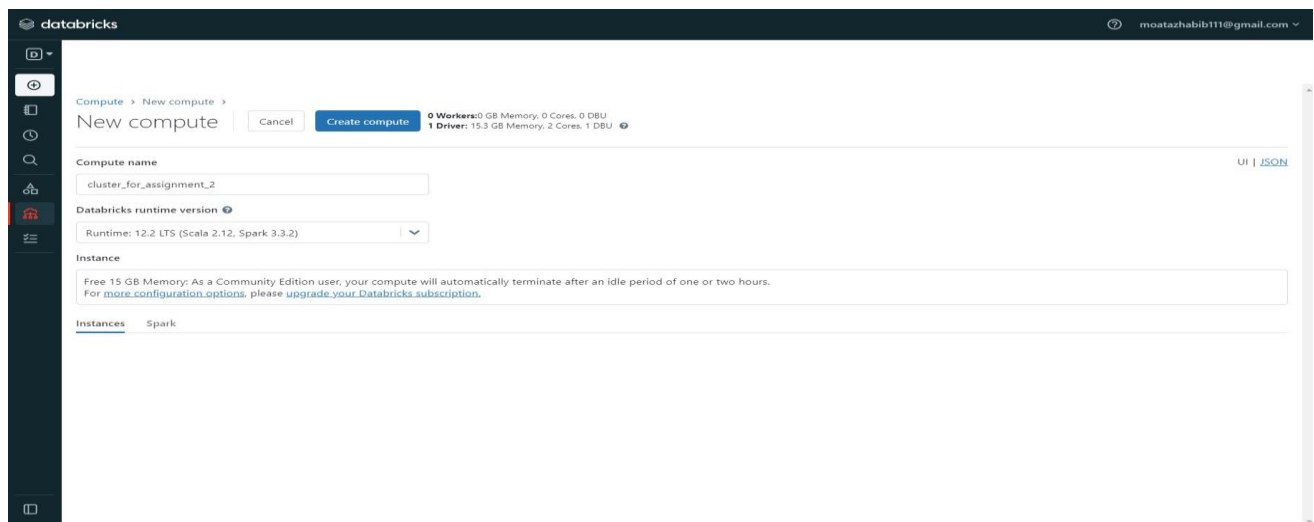




## Part 2 – Spark Examples:

### 1. Data Transformation Pipelines:

Cluster Setup:



The screenshot shows the Databricks web interface for creating a new compute cluster. The top navigation bar includes the Databricks logo and a user profile dropdown for 'moatazhabib111@gmail.com'. The left sidebar contains navigation icons for home, workspace, recent clusters, search, and a menu. The main content area is titled 'Compute > New compute' and features a 'Cancel' button and a blue 'Create compute' button. A status bar indicates '0 Workers: 0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU'. The 'Compute name' field is set to 'cluster\_for\_assignment\_2' with a 'UI | JSON' link. The 'Databricks runtime version' is set to 'Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2)'. An 'Instance' section contains a warning: 'Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options, please upgrade your Databricks subscription.' At the bottom, there are tabs for 'Instances' and 'Spark'.

## Cluster run Successfully

The screenshot shows the Databricks Compute page for a cluster named "cluster\_for\_assignment\_2". The cluster is in a "Running" state, indicated by a green checkmark. The configuration shows the Databricks Runtime Version as 12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12). The driver type is "Community Optimized" with 15.3 GB Memory, 2 Cores, and 1 DBU. The instance is "Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options, please upgrade your Databricks subscription." The availability zone is "us-west-2c". The cluster is configured with Spark and JDBC/ODBC drivers.

a) Uploaded the files to your DBFS table space.

The screenshot shows the "Create New Table" page in Databricks. The "Data source" is set to "Upload File". The "DBFS Target Directory" is "/FileStore/tables/Text\_Corpus". A list of files is shown, each with a green checkmark indicating successful upload. The files are:

- 0001.text (1.3 KB)
- 0002.text (1 KB)
- 0003.text (1.1 KB)
- 0004.text (3.9 KB)
- 0005.text (1.8 KB)
- 0006.text (0.8 KB)
- 0007.text (1.4 KB)
- 0008.text (1.7 KB)
- 0009.text (0.9 KB)
- 00010.text
- 00011.text
- 00012.text

The screenshot shows the "Create New Table" page in Databricks. The "Data source" is set to "Upload File". The "DBFS Target Directory" is "/FileStore/tables/Text\_Corpus". A list of files is shown, each with a green checkmark indicating successful upload. The files are:

- File uploaded to /FileStore/tables/00027-1.text
- File uploaded to /FileStore/tables/text/00038.text
- File uploaded to /FileStore/tables/text/00029.text
- File uploaded to /FileStore/tables/text/00030.text
- File uploaded to /FileStore/tables/text/00031.text
- File uploaded to /FileStore/tables/text\_v/00032.text
- File uploaded to /FileStore/tables/text\_v/00033.text
- File uploaded to /FileStore/tables/text\_v/00034.text
- File uploaded to /FileStore/tables/text\_v/00035.text
- File uploaded to /FileStore/tables/text\_v/00036.text
- File uploaded to /FileStore/tables/text/00037.text
- File uploaded to /FileStore/tables/text/00038.text
- File uploaded to /FileStore/tables/text/00039.text
- File uploaded to /FileStore/tables/text\_co/00040.text
- File uploaded to /FileStore/tables/text\_co/00041.text
- File uploaded to /FileStore/tables/text\_co/00042.text
- File uploaded to /FileStore/tables/text\_corp/00043.text
- File uploaded to /FileStore/tables/text\_corp/00044.text
- File uploaded to /FileStore/tables/text\_corp/00045.text
- File uploaded to /FileStore/tables/text\_corp/00046.text
- File uploaded to /FileStore/tables/text\_corp/00047.text
- File uploaded to /FileStore/tables/text\_corp/00048.text
- File uploaded to /FileStore/tables/text\_corp/00049.text
- File uploaded to /FileStore/tables/text\_corp/00050.text

Buttons: "Create Table with UI", "Create Table in Notebook".

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

Cluster: cluster\_for\_assignment\_2

Preview Table

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

```
Cmd 1
1 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 moatazhabib11@gmail.com at 10/31/2023, 2:14:24 PM on cluster_for_assignment_2
```

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

```
Cmd 2
1 TextFile.count()
(1) Spark Jobs
res5: Long = 1645
Command took 7.22 seconds -- by moatazhabib11@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”.

```
1 val antibioticsCount = TextFile.filter(line => line.contains("antibiotics")).count()
2
(1) Spark Jobs
antibioticsCount: Long = 2
Command took 3.51 seconds -- by moatazhabib11@gmail.com at 10/31/2023, 2:15:00 PM on cluster_for_assignment_2
```

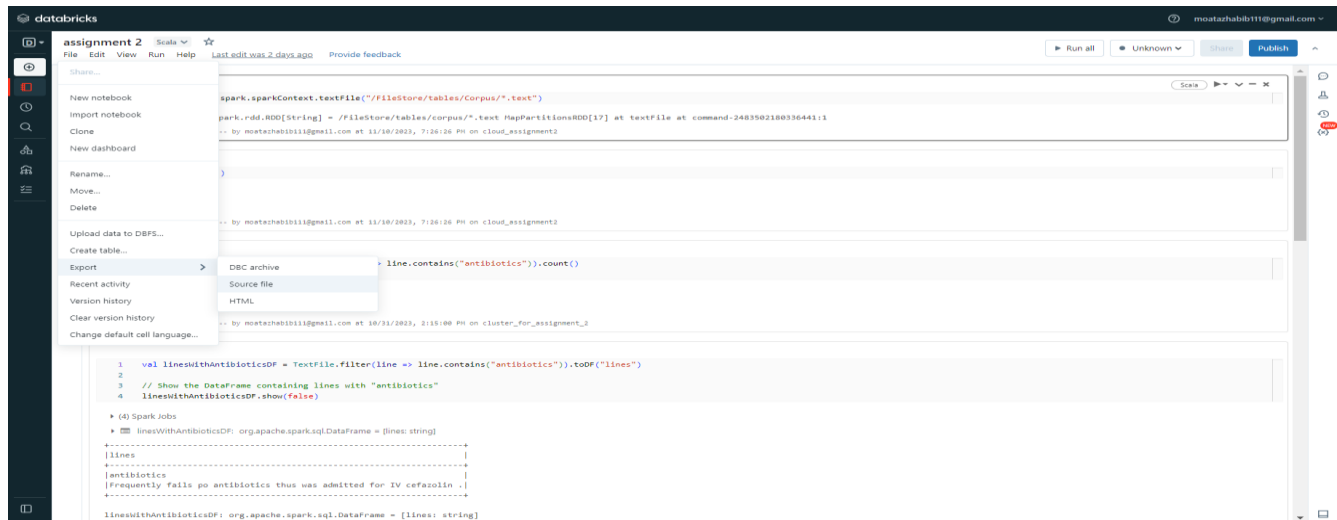
```
Cmd 4
1 val linesWithAntibioticsDF = TextFile.filter(line => line.contains("antibiotics")).toDF("lines")
2
3 // Show the DataFrame containing lines with "antibiotics"
4 linesWithAntibioticsDF.show(false)
(4) Spark Jobs
linesWithAntibioticsDF: org.apache.spark.sql.DataFrame = [lines: string]
+-----+
|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 moatazhabib11@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
1 val patientAdmittedCount = TextFile.filter(line => line.contains("patient"))
2 |> .filter(line => line.contains("admitted"))
3 patientAdmittedCount.count()
4
(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 moatazhabib11@gmail.com at 10/31/2023, 2:33:08 PM on cluster_for_assignment_2
```

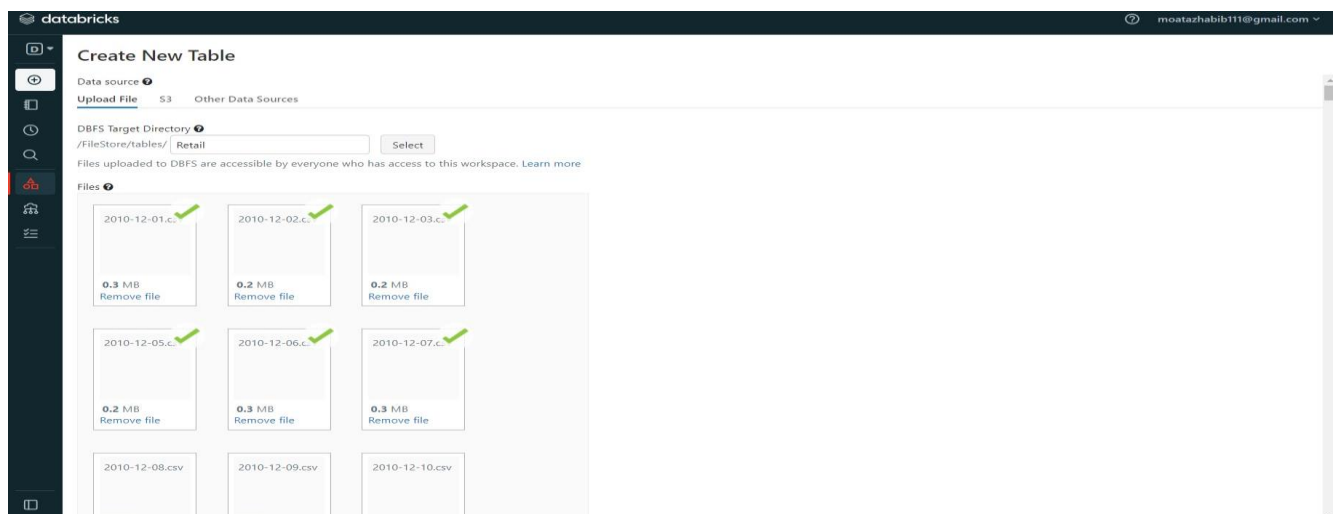
```
1 // Filter lines containing both "patient" and "admitted", and create a DataFrame
2 val linesWithPatientAdmittedDF = TextFile.filter(line => line.contains("patient"))
3 |> .filter(line => line.contains("admitted")).toDF("lines")
4
5 // Show the DataFrame containing lines with both "patient" and "admitted"
6 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 moatazhabib11@gmail.com at 10/31/2023, 3:12:32 PM on cluster_for_assignment_2
```

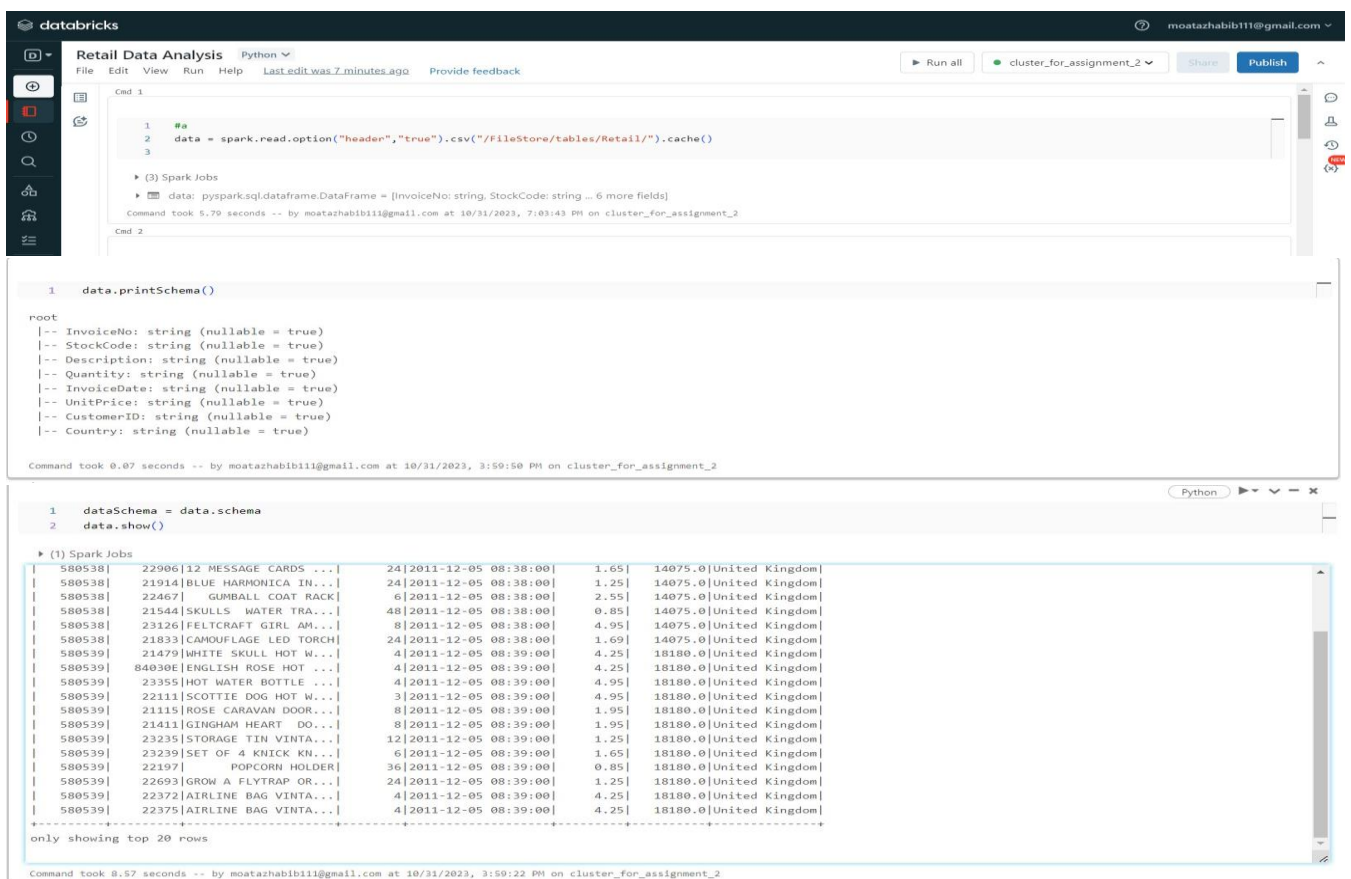
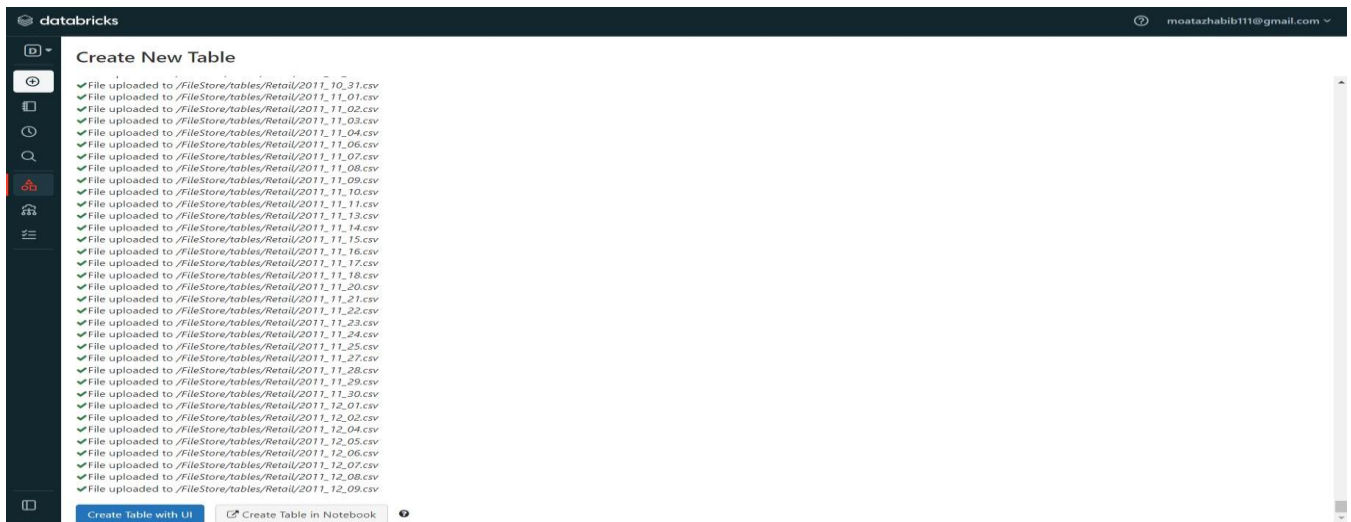
f) exported (. scala format) notebook



## 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

```
1 #b
2 # Number of rows
3 num_rows = data.count()
4
5 # Number of distinct InvoiceNo
6 num_distinct_invoice = data.select("InvoiceNo").distinct().count()
7
8 print("Number of rows:", num_rows)
9 print("Number of distinct InvoiceNo:", num_distinct_invoice)
10
```

▶ (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

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

▶ (2) Spark Jobs

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

TotalTransactions
5176450.0

The total value of the transactions =  $\text{sum}((\text{Quantity} * \text{UnitPrice})) = 9747747.93999462$

Cmd 6

```
1 # Total value of transactions as DataFrame
2 total_value_df = data.withColumn("TotalValue", (col("Quantity") * col("UnitPrice"))).groupBy().agg(sum("TotalValue").alias("TotalValue"))
3 total_value_df.show()
4
```

▶ (2) Spark Jobs

total\_value\_df: pyspark.sql.dataframe.DataFrame = [TotalValue: double]

TotalValue
9747747.93999462

Command took 1.13 seconds -- by moatazhabib111@gmail.com at 10/31/2023, 6:00:50 PM on cluster\_for\_assignment\_2

Cmd 7

c) Output the 5 top-selling products.

```
1 # c
2 data.createOrReplaceTempView("Retail")
3 q="SELECT StockCode, SUM (Quantity) as TotalQuantity FROM Retail GROUP BY StockCode ORDER BY TotalQuantity DESC LIMIT 5;"
4 sqlDF = spark.sql(q)
5 sqlDF.show()
6
```

▶ (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.

```
1 # d
2 data.createOrReplaceTempView("Retail")
3
4 query = """
5     SELECT StockCode, SUM (Quantity*UnitPrice) AS TotalPrice
6     FROM Retail
7     GROUP BY StockCode
8     ORDER BY TotalPrice DESC
9     LIMIT 5
10 """
11
12 sqlDF = spark.sql(query)
13
14 sqlDF.show()
15
```

▶ (2) Spark Jobs

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

StockCode	TotalPrice
DOT	206245.48
22423	164762.19
47566	98302.98
85123A	97804.5
85099B	92356.029999999994

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.

```
1 data.createOrReplaceTempView("Retail")
2
3 query = """
4     SELECT Country, SUM (Quantity*UnitPrice) AS TotalValue
5     FROM Retail
6     GROUP BY Country
7     ORDER BY TotalValue DESC
8 """
9 sqlDF = spark.sql(query)
10 sqlDF.show()
11
```

▶ (2) Spark Jobs

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

Country	TotalValue
United Kingdom	8187806.363999976
Netherlands	284661.53999999999
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 moatasshaib11@gmail.com at 10/31/2023, 7:19:55 PM on cluster\_for\_assignment\_2

most performing countries in purchases

```
1 # e
2 data.createOrReplaceTempView("Retail")
3
4 query = """
5     SELECT Country, SUM (Quantity*UnitPrice) AS TotalValue
6     FROM Retail
7     GROUP BY Country
8 """
9 sqlDF = spark.sql(query)
10 sqlDF.show()
11
12
```

▶ (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	2505.4700000000003
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	1730.9199999999992
Switzerland	56385.34999999997

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



```

1 # f
2 data.createOrReplaceTempView("Retail")
3 query = """
4     SELECT StockCode, SUM (Quantity*UnitPrice) AS TotalPrice
5     FROM Retail
6     GROUP BY StockCode
7     ORDER BY TotalPrice DESC
8     LIMIT 5
9 """
10 sqlDF = spark.sql(query)
11 sqlDF.show()
12

```

▶ (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 moatazhabib11@gmail.com at 10/31/2023, 7:26:28 PM on cluster\_for\_assignment\_2

```

1 display(sqlDF)
2

```

▶ (2) Spark Jobs

	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 moatazhabib11@gmail.com at 10/31/2023, 7:40:21 PM on cluster\_for\_assignment\_2

```

1 import matplotlib.pyplot as plt
2
3 # Data from the SQL result
4 stockcodes = ['DOT', '22423', '47566', '85123A', '85099B']
5 total_prices = [206245.48, 164762.19, 98302.98, 97894.5, 92356.02999999994]
6
7 # Create a bar plot
8 plt.figure(figsize=(10, 6))
9 plt.bar(stockcodes, total_prices, color='skyblue')
10 plt.xlabel('Stock Code')
11 plt.ylabel('Total Price')
12 plt.title('Top 5 Stock Codes by Total Price')
13 plt.xticks(rotation=45)
14 plt.tight_layout()
15 plt.show()
16

```



Command took 0.41 seconds -- by moatazhabib11@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 .

**databricks** Retail Data Analysis Python

File Edit View Run Help Last edit was 5 minutes ago Provide feedback

▶ Run all cluster\_for\_assignment\_2 Share Publish

Share... [DOT, '22423', '47566', '85123A', '85099B'] [206245.48, 164762.19, 98302.98, 97894.5, 92356.02999999994]

New notebook  
Import notebook  
Clone  
New dashboard  
Rename...  
Move...  
Delete  
Upload data to DBFS...  
Create table...  
Export  
Recent activity  
Version history  
Clear version history  
Change default cell language...

Export  
DBC archive  
Source file  
Jupyter Notebook  
HTML

plot  
figsize=(10, 6)  
des, total\_prices, color='skyblue'  
ck Code'  
al Price')  
5 Stock Codes by Total Price')  
tion=45)  
t()

Top 5 Stock Codes by Total Price

50000  
25000  
0

DOT 22423 47566 85123A 85099B

Stock Code

Command took 0.41 seconds -- by moatazhabib11@gmail.com at 10/31/2023, 7:40:53 PM on cluster\_for\_assignment\_2

Shift+Enter to run  
Shift+Ctrl+Enter to run selected text

### 3. Structured Streaming:

#### Create a notebook

```
assignment 2 Python ☆
File Edit View Run Help Last edit was 1 minute ago Provide feedback

cmd 1

1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import sum, current_timestamp
3 from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, TimestampType
4
5 spark = SparkSession.builder.appName("RetailDataStream").getOrCreate()
6
7
```

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 ▶ ▼ - ✕

1 from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType, DoubleType
2 data_schema = StructType([
3     StructField("InvoiceNo", IntegerType(), True),
4     StructField("StockCode", StringType(), True),
5     StructField("Description", StringType(), True),
6     StructField("Quantity", IntegerType(), True),
7     StructField("InvoiceDate", TimestampType(), True),
8     StructField("UnitPrice", DoubleType(), True),
9     StructField("CustomerID", StringType(), True),
10    StructField("Country", StringType(), True),
11])

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

```
1 streaming_data = (
2     spark
3     .readStream
4     .format("csv")
5     .schema(data_schema)
6     .option("maxFilesPerTrigger", 20)
7     .option("header", "true")
8     .csv("/FileStore/tables/retail")
9 )

▶ 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)

```
1
2 from pyspark.sql.functions import sum, col
3 customer_aggregates = (
4     streaming_data
5     .groupBy("CustomerID")
6     .agg(
7         sum(col("Quantity")).alias("TotalStocks"),
8         sum(col("Quantity") * col("UnitPrice")).alias("TotalValue")
9     )
10 )

▶ 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 ▶ ▼ - ✕

1 customer_aggregates_result = (
2     customer_aggregates
3     .writeStream
4     .outputMode("complete")
5     .format("memory")
6     .queryName("customer_aggregates")
7     .start()
8 )

Cancel ==

▶ (1) Spark Jobs
▶ customer_aggregates (id: 5b4cff39-8665-446d-b3f8-77b39c181a6b) Last updated: 20 seconds ago
```

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

Cmd 7

```
1 # create a new stream that populates another dataset with progress for each loaded set of data
2 from pyspark.sql.functions import current_timestamp,count
3 progress_data = (
4     customer_aggregates
5     .select(current_timestamp().alias("TriggerTime"),
6     count("*").alias("RecordsImported"),
7     sum("TotalValue").alias("SaleValue")
8     )
9 )
10 progress_data_query = (
11     progress_data
12     .writeStream
13     .outputMode("update")
14     .format("memory")
15     .queryName("progress_data")
16     .start()
17 )
18
```

```
1 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
18089.0	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 shahdmohamed6777@gmail.com at 11/12/2023, 2:53:38 PM on assignment 2

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

▶ (3) Spark Jobs

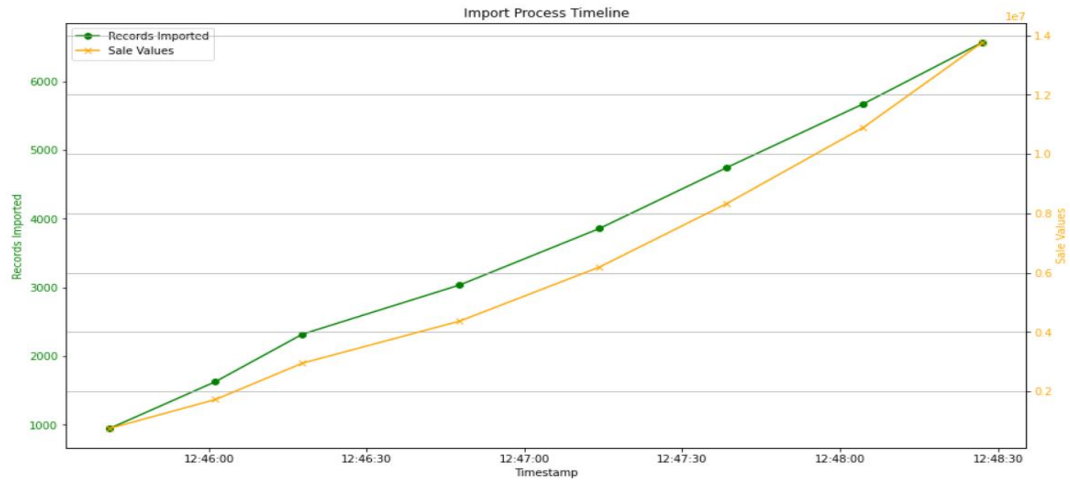
TriggerTime	RecordsImported	SaleValue
2023-11-12 12:45:...	949	748957.0200000012
2023-11-12 12:46:...	1629	1722763.4700000016
2023-11-12 12:46:...	2320	2948230.1800000025
2023-11-12 12:46:...	3036	4366905.1800000025
2023-11-12 12:47:...	3859	6192471.520000002
2023-11-12 12:47:...	4741	8321909.331000003
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	2.383502949300001E7
2023-11-12 12:50:...	10009	2.81186642500001E7
2023-11-12 12:50:...	11194	3.311791907600001E7
2023-11-12 12:50:...	12504	3.889440811700001E7
2023-11-12 12:51:...	13933	4.563597898700002E7
2023-11-12 12:51:...	14393	5.007337816700002E7
2023-11-12 12:52:...	14393	5.007337816700002E7
2023-11-12 12:52:...	14393	5.007337816700002E7

Command took 3.92 seconds -- by shahdmohamed6777@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.

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 progress_data_df = spark.table("progress_data").toPandas()
4
5 fig, ax1 = plt.subplots(figsize=(15, 8))
6
7 ax1.plot(progress_data_df["TriggerTime"], progress_data_df["RecordsImported"], label="Records Imported", color='green', marker='o')
8 ax1.set_xlabel("Timestamp")
9 ax1.set_ylabel("Records Imported", color='green')
10 ax1.tick_params(axis='y', labelcolor='green')
11
12 ax2 = ax1.twinx()
13
14 ax2.plot(progress_data_df["TriggerTime"], progress_data_df["SaleValue"], label="Sale Values", color='orange', marker='x')
15 ax2.set_ylabel("Sale Values", color='orange')
16 ax2.tick_params(axis='y', labelcolor='orange')
17
18 lines, labels = ax1.get_legend_handles_labels()
19 lines2, labels2 = ax2.get_legend_handles_labels()
20 ax1.legend(lines + lines2, labels + labels2, loc='upper left')
21
22 plt.grid(True)
23 plt.title("Import Process Timeline")
24 plt.show()
```

► (1) Spark Jobs



► (1) Spark Jobs

