uOttawa

ELG5166 Cloud Analytics
"Definitions and Spark Examples"

Assignment 2

Instructor: Dr. Benjamin Eze

Group: 4

Student Name: Ali El-Sherif ID: 300327246

Student Name: Basma Abd-Elwahab ID: 300327209

Student name: Abdelrhman Rezkallah ID: 300327290

Student Name: Abdulrahman Ahmed ID: 300327218

Table of Contents

Personal Ethics & Academic Integrity Statement....3

Part 1: Definitions....4

1. Describe a distributed file system? Briefly describe with examples, any two implementations of a distributed file system.....4

2. Describe briefly 3 features of Apache Hadoop Map-Reduce and 3 limitations associated with it when compared to Apache Spark?.....4

3. Describe briefly the low-level and high-level APIs in Apache Spark. What differentiates them and when do you use one over the other.....5

4. Describe the following Apache Spark terms with examples.....6

a. Immutability in Spark.....6

b. Lazy Evaluation and its impact on Spark performance.....6

c. How is SparkSession different from SparkContext.....6

d. Spark MLlib Transformers, Estimators, and Evaluators.....6

5. Describe briefly – with examples - how Spark Streaming differs from Spark Structured Streaming?.....7

Part 2: Spark Examples.....8

1. Data Transformation Pipelines:.....8

A) Uploaded the files to your DBFS table space:.....9

B) Use Spark Scala to load your data into an RDD:.....10

C) Count the number of lines across all the files:.....10

D) Find the number of occurrences of the word “antibiotics”.....10

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.....11

2. Retail Data Analysis:.....11

A) Uploaded the files to your DBFS table space:.....12

B) Output the total number of transactions across all the files and the total value of the transactions:.....13

C) Output the 5 top-selling products:.....14

D) Output the 5 topmost valuable products:.....14

E) Output each country and the total value of their purchases:.....14

F) Use the dataset from step (c) to plot a line graph of the import process –showing two timelines – records imported and sale values:.....15

3. Structured Streaming:.....16

A) Create a new notebook:.....16

B) 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.....17

C) 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):..... 18

D) Use the dataset from step (c)to plot a line graph of the import process –showing two timelines –records imported and sale values: 19

References..... 20

Personal Ethics & Academic Integrity Statement

By typing in my name and student ID on this form and submitting it electronically, I am attesting to the fact that I have reviewed not only my work but the work of my team member, in its entirety.

I attest to the fact that my work in this project adheres to the fraud policies as outlined in the Academic Regulations in the University's Graduate Studies Calendar. I further attest that I have knowledge of and have respected the "Beware of Plagiarism" brochure for the university. To the best of my knowledge, I also believe that each of my group colleagues has also met the aforementioned requirements and regulations. I understand that if my group assignment is submitted without a completed copy of this Personal Work Statement from each group member, it will be interpreted by the school that the missing student(s) name is confirmation of non-participation of the aforementioned student(s) in the required work. We, by typing in our names and student IDs on this form and submitting it electronically,

• warrant that the work submitted herein is our own group members' work and not the work of others.

• acknowledge that we have read and understood the University Regulations on Academic Misconduct.

• acknowledge that it is a breach of University Regulations to give or receive unauthorized and/or unacknowledged assistance on a graded piece of work.

Part 1: Definitions

1. *Describe a distributed file system? Briefly describe with examples, any two implementations of a distributed file system.*

Distributed file system is a system which is allocated on many file servers or locations to allow users to share information. It has a lot of features such as: transparency (structure, access, naming and replication), user mobility, performance, simplicity and ease of use, high availability, scalability, high reliability, data integrity, security and heterogeneity [11]. There are a lot of applications for DFS and two of them are:

1. Network File System (NFS), which is a client-server architecture that allows users to view, store and update files remotely. It is one of many standards for Network-Attached Storage (NAS) [11].
2. Server Message Block (SMB), which was invented by IMB, is a protocol for file sharing used by computers to read and write files to a remote host over LAN. The directories on the remote host that can be accessed by SMB are called shares [11].
3. Common internet file system (CIFS), which is designed by Microsoft, is an application of SIMB protocol [11].

2. *Describe briefly 3 features of Apache Hadoop Map-Reduce and 3 limitations associated with it when compared to Apache Spark?*

	Apache Hadoop Map-reduced features		Limitations compared to Apache Spark
Flexibility [10]	It enables access to both structured and unstructured data and draws significant value from the many data sources. Plus, the MapReduce framework supports different languages and data from a variety of sources, including email, social media.	Batch Processing	Apache spark supports real-time processing, Hadoop MapReduce is good in patch processing but it doesn't have that feature.
Scalability [10]	Can store and distributes huge amount of data into a lot of servers, allowing to run programs over a huge number of nodes which is called horizontal scalability, which involves terabytes of data.	Performance [10]	Apache Spark is 100 times faster in-memory than Hadoop MapReduce which runs on the disk driver. This is because the data can be distributed all over the cluster in MapReduce and that reduces processing speed compared to Spark.

Data Locality [5]	Moving the processing unit to the data of the map-reduce framework, instead of moving the data to it.	Ease of Use [10]	While MapReduce is only limited to be written in java, Spark provides APIs for python, Java, SQL and Scala.

3. *Describe briefly the low-level and high-level APIs in Apache Spark. What differentiates them and when do you use one over the other.*

	Low- level API	High-Level API
Differences [1]	▪ Resilient Distributed Datasets (RDD) which is a fundamental part of data structure of the spark system that can works in parallel using spark parallelization. ▪ When applying multi transformation on it, there is a fault tolerance, if it fails, the data is recovered automatically by the RDD. ▪ Can store data across many nodes after partitioning. ▪ The user can access the RDDs directly whether using structured and unstructured data.	▪ Datasets and dataframes are much faster than RDDs. ▪ Dataframes organize the data as row/column format using schemas. ▪ During runtime, dataframes allow debugging. ▪ High level of abstraction and low level of details (user friendly). ▪ Less customizable than the low-level. ▪ MLlib can be easily used as a fixed API is used among ML algorithms in different languages.
Usage [1]	▪ With unstructured data. ▪ On dataset with low level of transformation. ▪ Using dataset as a functional programming.	▪ With structured data. ▪ If the processing requires high level expressions, maps, aggregation, SQL queries, columnar access and use of lambda functions on semi-structured data. ▪ In using domain specific APIs ▪ When higher level of type safety at compile time is needed. ▪ When a custom view into structured and semi structured data is needed.

4. Describe the following Apache Spark terms with examples

a. Immutability in Spark

Immutability means that data cannot be changed after it is created unless transformations are used to make these changes. This contributes significantly to the reduction of complexity of data synchronization.

If a dataframe is created after applying transformations to it and assigning it to a new variable, this will result no changes to the local data and the old instance will be removed. If the data is loaded again, no changes will be noticed, but only the instance being used is switched.

b. Lazy Evaluation and its impact on Spark performance

Lazy evaluation means that the code will not be executed unless an action function is triggered, nevertheless, an execution plan will be made for the list of transformations for analyzation and optimization [4]. The impact on Spark performance is that it enhances the power of Apache Spark by reducing the execution time of the RDD operations, and also, it maintains the lineage graph to remember the operations on RDD. As a result, it Optimizes the performance and achieves fault tolerance [9].

If two dataframes need to be joined and filtered by rows, Spark will make filtering at first then do the joining which is a lot faster.

c. How is SparkSession different from SparkContext

SparkSession has a common entry point for the spark applications and functionality. It performs all the functionalities of the SparkContext, and perform the user-defined computations throughout the cluster [8].

SparkContext such as SQL and Hive interactions, it's an old spark entry point. It must be generated before any spark interactions. It's required in the RDDs (Lower-level APIs) and contained within the SparkSession, that can create a new instance for each new action [2].

d. Spark MLlib Transformers, Estimators, and Evaluators

Transformers convert the DataFrame into another data forms, to be used in the machine learning models training. It implements a method transform(), that can attach one or more columns as the new features[6].

Examples:

- Tokenizer Transformer.
- StopWords Removal.
- SQL Transformer.

Estimators implement the fit() method, that can abstract the learning algorithm, fit or train different models on the data[6].

Example: the XGBOOST is an estimator, and calling the fit() method is to train the model.

Evaluators selects the best model, hyperparameters, metrices for evaluating the model and the performance of the model [6].

Examples:

- MultiClassClassificationEvaluator()
- BinaryClassificationEvaluator()
- ClusteringEvaluator()

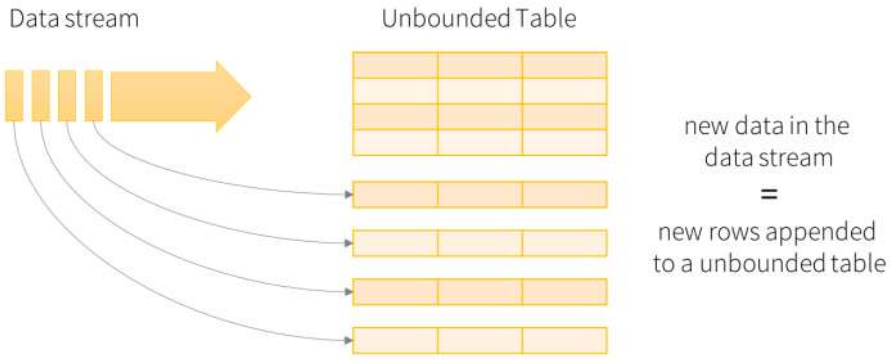
5. *Describe briefly – with examples - how Spark Streaming differs from Spark Structured Streaming?*

Spark streaming uses the concept of discretized streams, or DStreams API, that is powered by spark RDDs and use the micro-batch streaming.

Each DStream is a sequence of the RDDs, when spark receives the data it splits them into batches for the spark engine, and each incoming record is part of the RDDs (managed by the streaming context). After that the batches of the processed data will be sent to the destination. It reduces the latency, doesn't work on the concept of the event time and doesn't take it into consideration[7].

Structured streaming moves towards the real streaming concept. It doesn't work with batches. It uses the DataFrame or the table to store the data record by record and to perform the operations of the streaming.

When the data is received in a trigger, it appended to the continuously flowing data stream. After that, the processing is performed to each row of the data stream. The unbounded result table take the result and stores it. The result is depending on the operation's mode, for example you can see the latest updated result, the new results, or all of the results. It can be complete, update or append. [3]



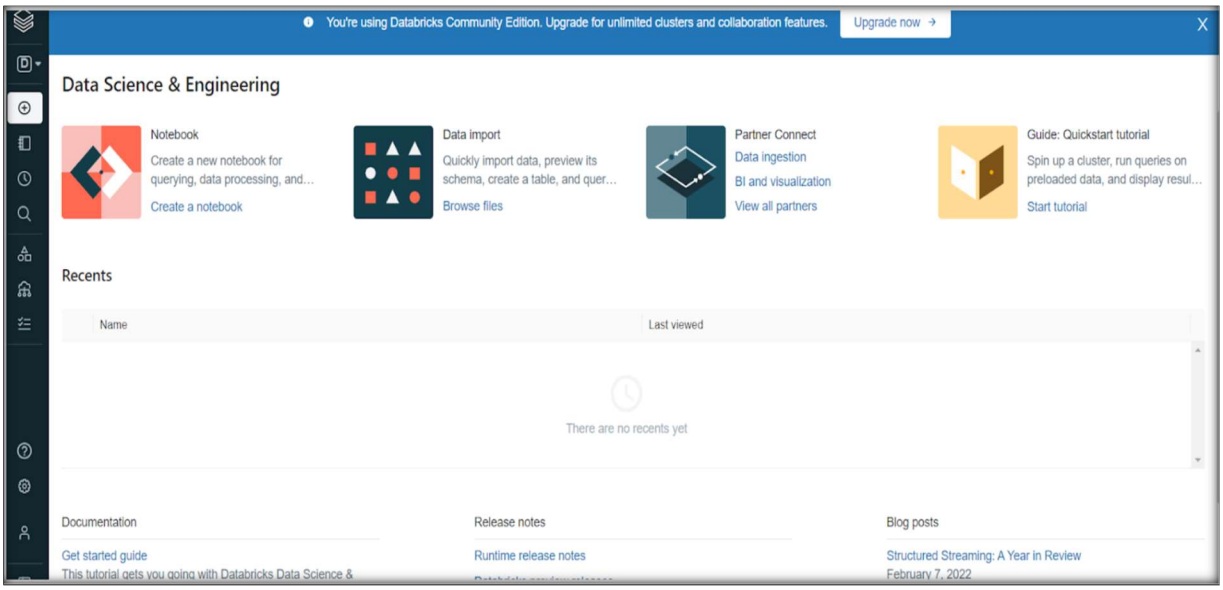
Data stream as an unbounded table

Part 2: Spark Examples

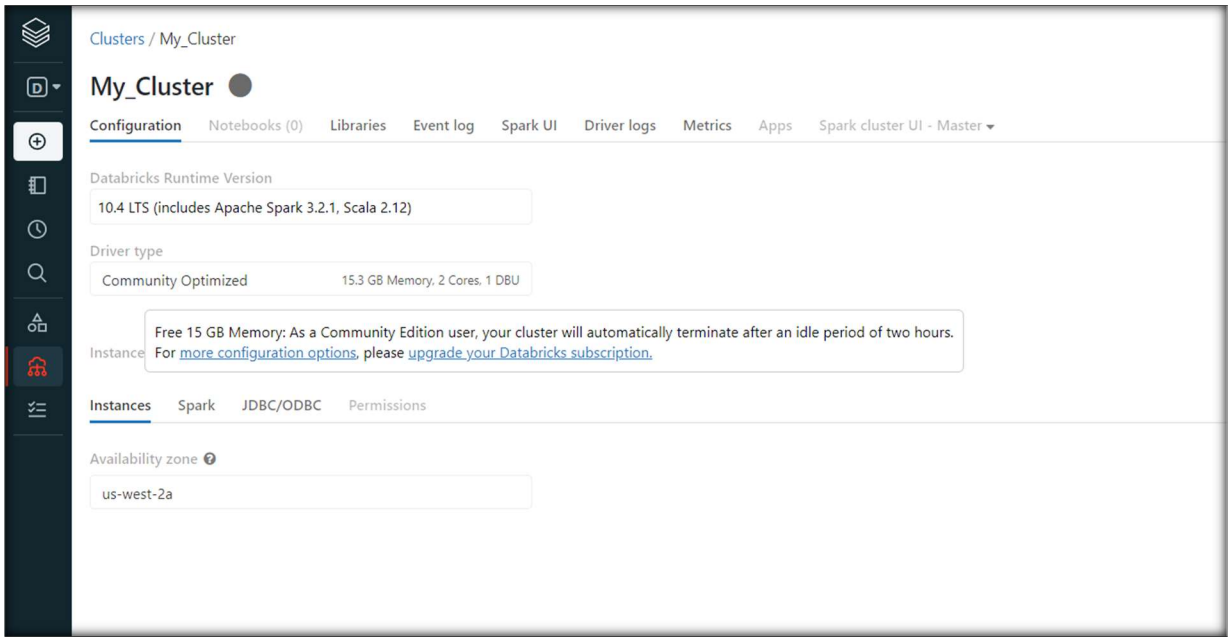
1. Data Transformation Pipelines:

- **Evidence of the setup:**

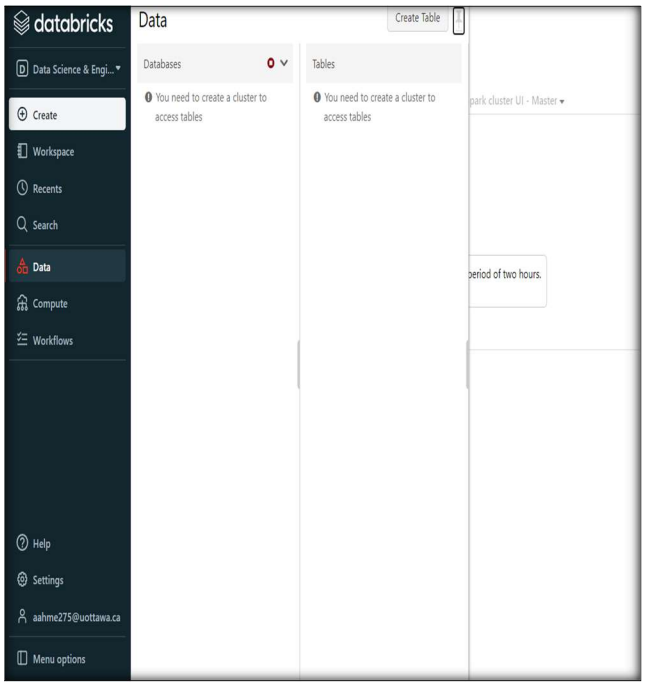
- Sign into the databricks:

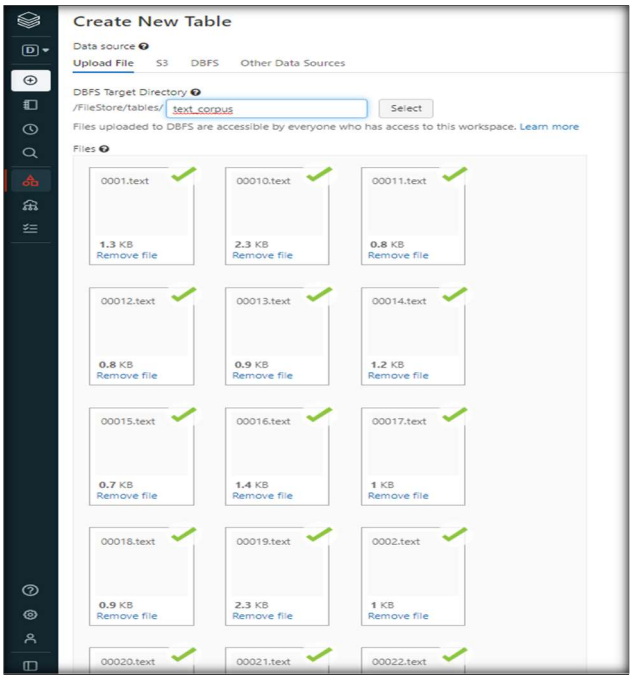


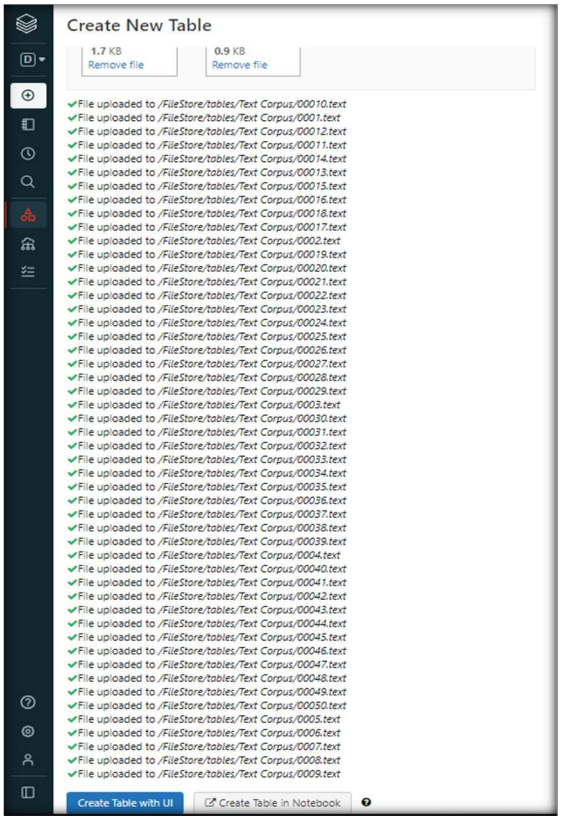
- Creating the cluster:

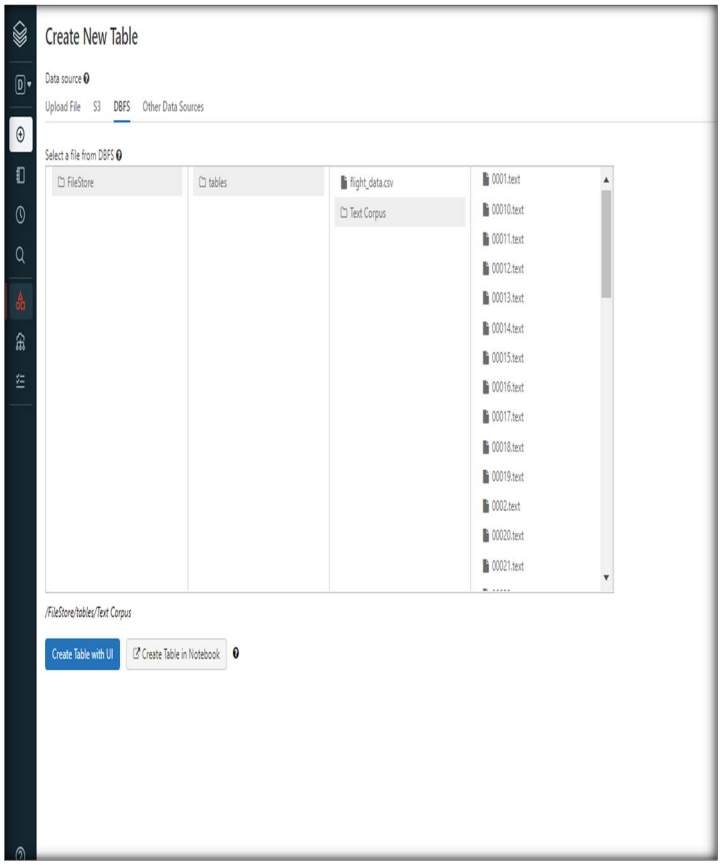


A) Uploaded the files to your DBFS table space:

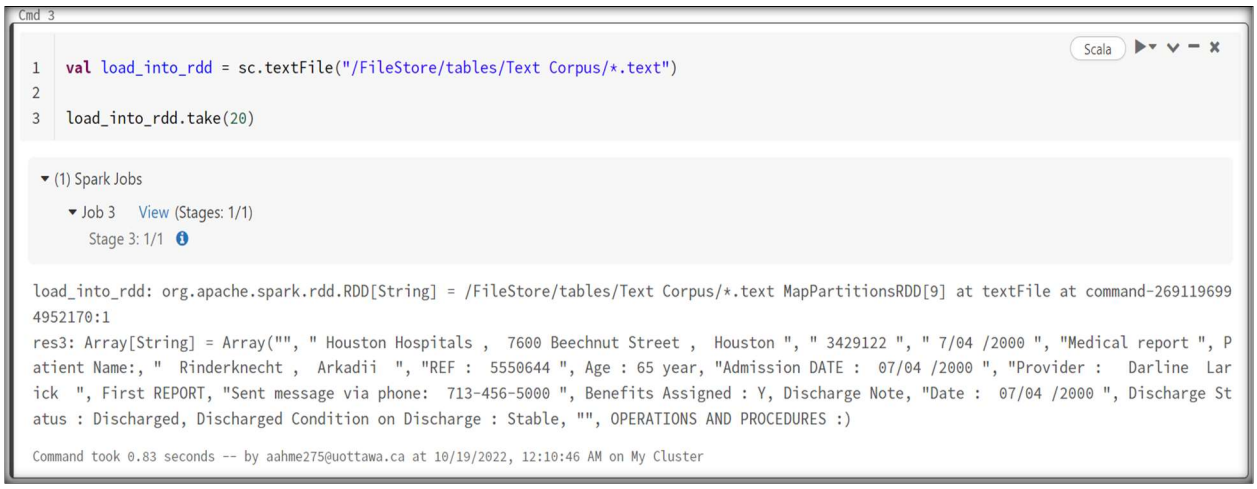








B) Use Spark Scala to load your data into an RDD:



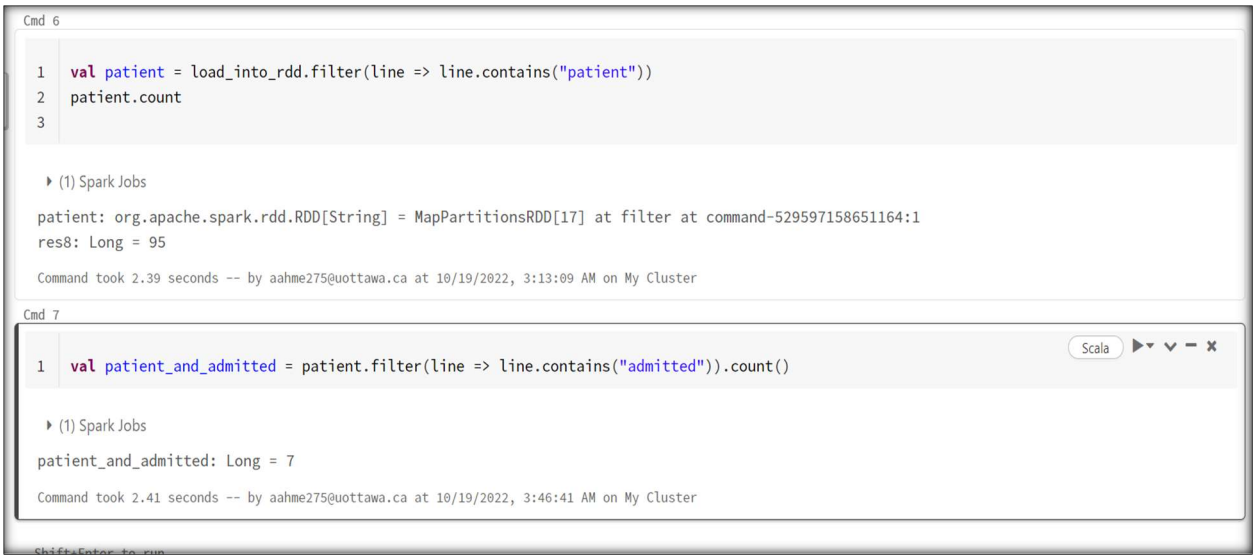
C) Count the number of lines across all the files:

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

The output :2

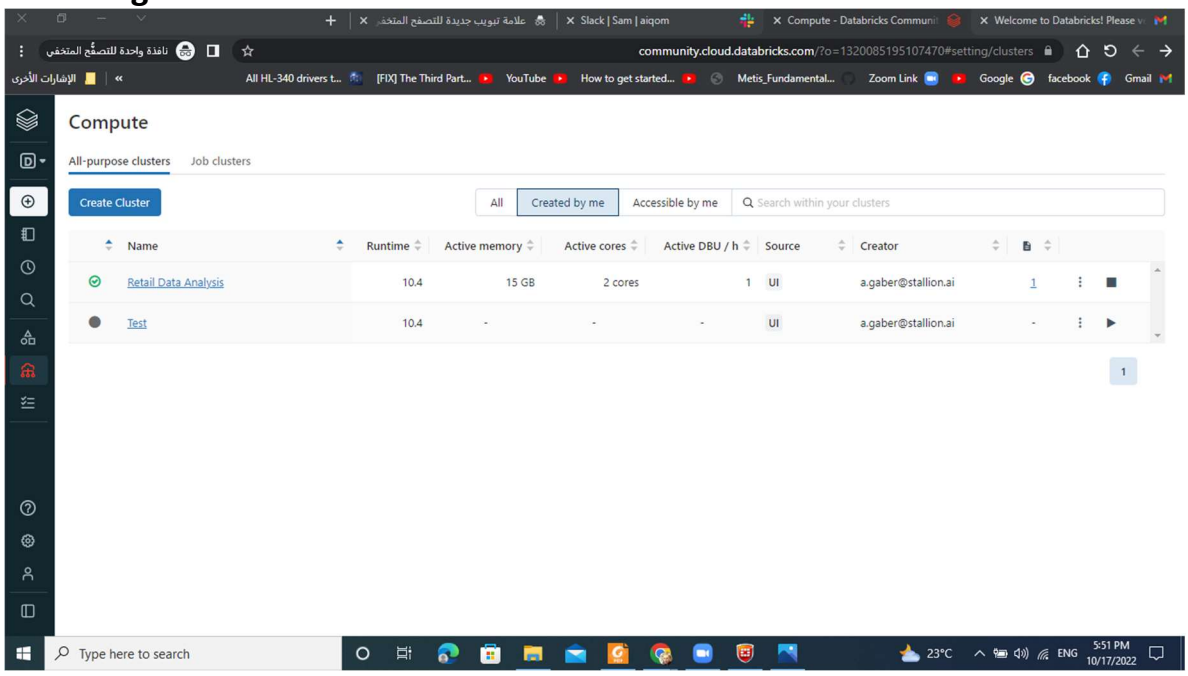
E) Count the occurrence of the word “patient” and “admitted” on the same line of text. Please ensure that your code contains at least 2 transformation functions in a pipeline

The output: 7 patient and admitted

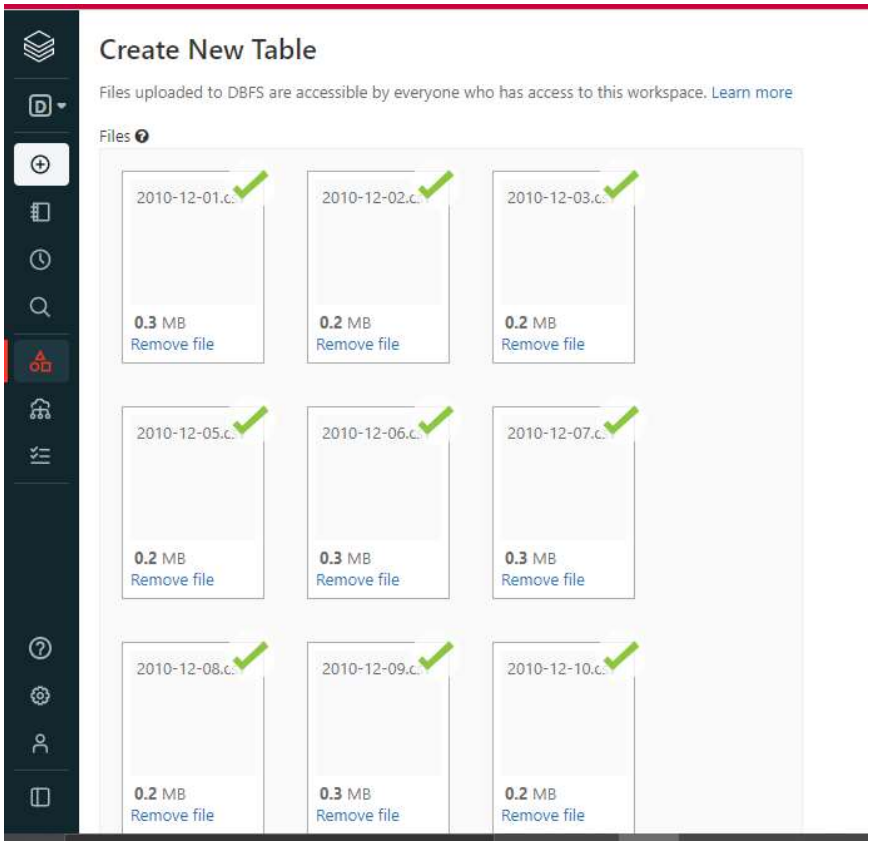


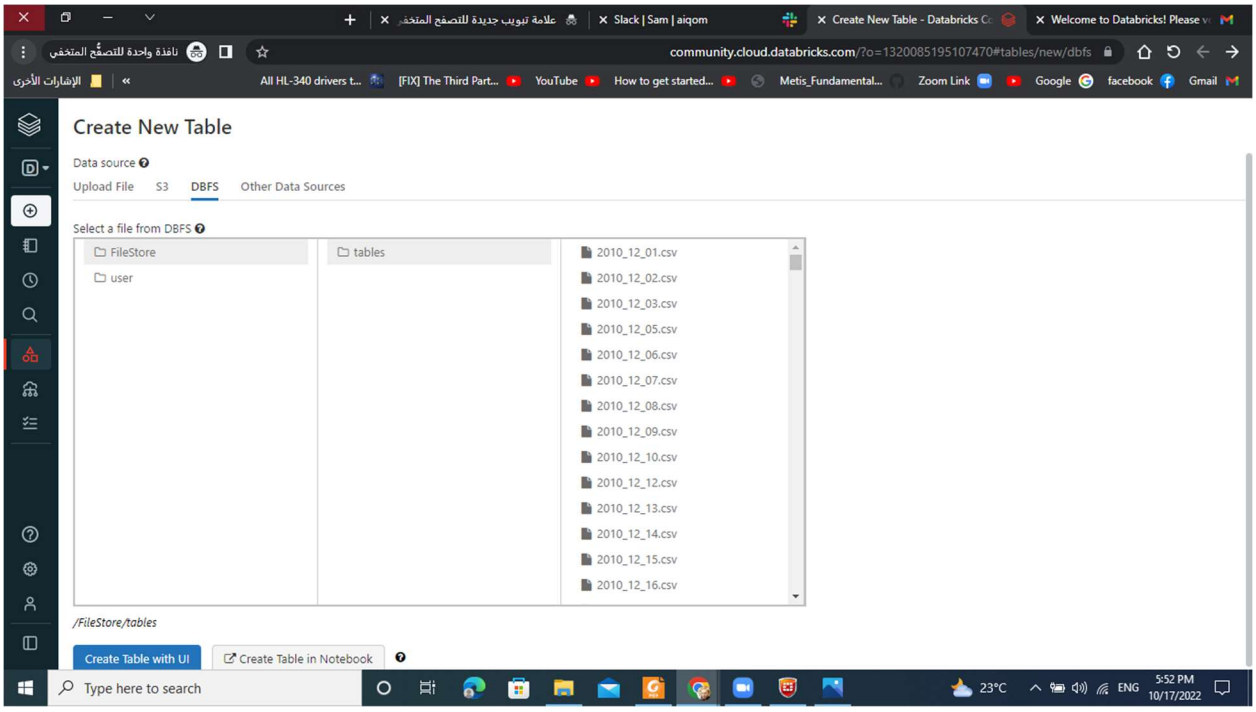
2. *Retail Data Analysis:*

Creating the cluster:



A) Uploaded the files to your DBFS table space:





```
schema = spark.read.csv(path = '/FileStore/tables/2010_12_01.csv',header=True,inferSchema=True)
retail_df = spark.read.csv(path = '/FileStore/tables',header=True,schema=schema.schema)
display(retail_df)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	580538	23084	RABBIT NIGHT LIGHT	48	2011-12-05T08:38:00.000+0000	1.79	14075	United Kingdom
2	580538	23077	DOUGHNUT LIP GLOSS	20	2011-12-05T08:38:00.000+0000	1.25	14075	United Kingdom
3	580538	22906	12 MESSAGE CARDS WITH ENVELOPES	24	2011-12-05T08:38:00.000+0000	1.65	14075	United Kingdom
4	580538	21914	BLUE HARMONICA IN BOX	24	2011-12-05T08:38:00.000+0000	1.25	14075	United Kingdom
5	580538	22467	GUMBALL COAT RACK	6	2011-12-05T08:38:00.000+0000	2.55	14075	United Kingdom
6	580538	21544	SKULLS WATER TRANSFER TATTOOS	48	2011-12-05T08:38:00.000+0000	0.85	14075	United Kingdom
7	580538	23126	FELTCRAFT GIRL AMELIE KIT	8	2011-12-05T08:38:00.000+0000	4.95	14075	United Kingdom
8	580538	21833	CAMOUFLAGE LED TORCH	24	2011-12-05T08:38:00.000+0000	1.69	14075	United Kingdom
9	580539	21479	WHITE SKULL HOT WATER BOTTLE	4	2011-12-05T08:39:00.000+0000	4.25	18180	United Kingdom
10	580539	84030E	ENGLISH ROSE HOT WATER BOTTLE	4	2011-12-05T08:39:00.000+0000	4.25	18180	United Kingdom

Truncated results, showing first 1000 rows.

B) Output the total number of transactions across all the files and the total value of the transactions:

Total number of rows:

```
retail_df.count()

Out[2]: 541909
```

Total number of transactions:

```
1  from pyspark.sql.functions import countDistinct
2  # b) detect the total number of transactions
3  total_number_of_transactions = retail_df.select(countDistinct('InvoiceNo'))
4
5  total_number_of_transactions.show()
```

▸ (3) Spark Jobs

```
+-------------------------+
|count(DISTINCT InvoiceNo)|
+-------------------------+
|                    25900|
+-------------------------+
```

Command took 9.68 seconds -- by a.gaber@stallion.ai at 17ء 5:33:15 2022/10/ on Retail Data Analysis

Total value of the transactions:

```
1  #b) total value of the transactions
2  import pyspark.sql.functions as f
3  value = value_of_the_transactions.select(f.sum('sales_value')).collect()[0][0]
4  print(f'the total sales of all transaction is {value} USD')
```

▸ (2) Spark Jobs

```
the total sales of all transaction is 9747747.933999462 USD
```

C) Output the 5 top-selling products:

```
#c) the 5 top-selling products
value_of_the_transactions.groupBy("StockCode").sum('Quantity').sort(f.desc('sum(Quantity)')).show(5)
```

(2) Spark Jobs

```
+---------+-------------+
|StockCode|sum(Quantity)|
+---------+-------------+
|    22197|        56450|
|    84077|        53847|
|   85099B|        47363|
|   85123A|        38830|
|    84879|        36221|
+---------+-------------+
```

D) Output the 5 topmost valuable products:

```
# d)the 5 topmost valuable products.
valuable_products = value_of_the_transactions.groupBy("StockCode").sum('sales_value').sort(f.desc('sum(sales_value)'))
valuable_products.show(5)
```

(2) Spark Jobs

```
+---------+------------------+
|StockCode| sum(sales_value)|
+---------+------------------+
|      DOT|         206245.48|
|    22423|         164762.19|
|    47566|          98302.98|
|   85123A|           97894.5|
|   85099B|92356.02999999994|
+---------+------------------+
only showing top 5 rows
```

E) Output each country and the total value of their purchases:

The countries also sorted here:

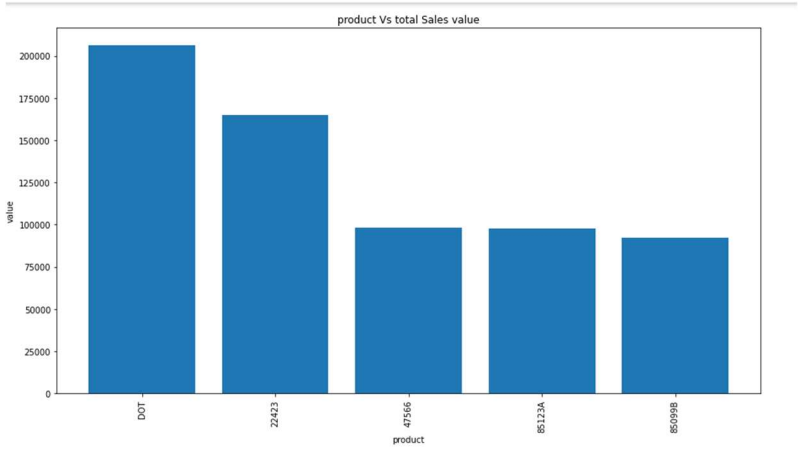
```
# e) the Output each country and the total value of their purchases
value_for_each_country = value_of_the_transactions.groupBy("Country").sum('sales_value').sort(f.desc('sum(sales_value)'))
value_for_each_country.show()
```

(2) Spark Jobs

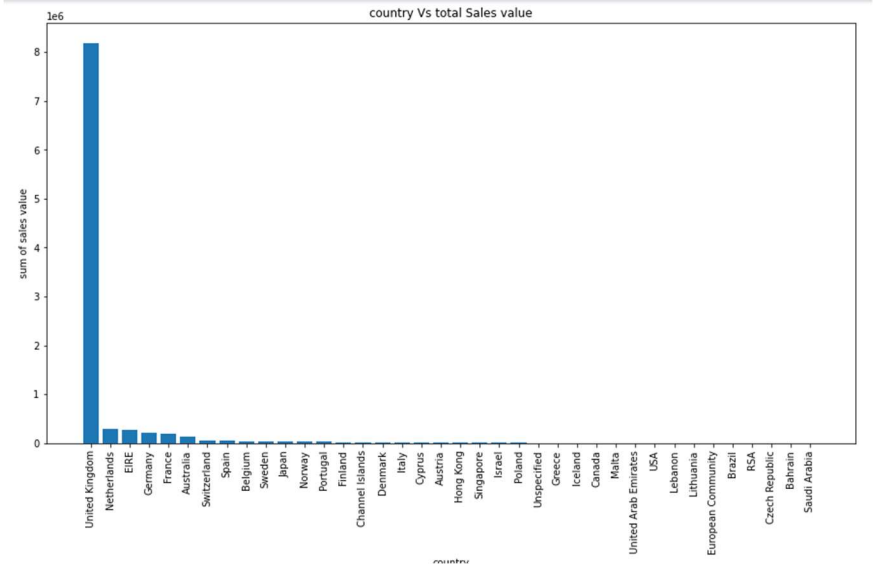
```
+--------------+------------------+
|       Country|  sum(sales_value)|
+--------------+------------------+
|United Kingdom|  8187806.36399976|
|   Netherlands| 284661.5399999999|
|          EIRE|263276.82000000024|
|       Germany|         221698.21|
|        France|          197403.9|
|     Australia|137077.26999999996|
|   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|
```

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

```
# f) visualization
import matplotlib.pyplot as plt
product = [x["StockCode"] for x in valuable_products.rdd.collect()]
sales_value_p = [x["sum(sales_value)"] for x in valuable_products.rdd.collect()]
plt.figure(figsize=[15,8])
plt.bar(product[:5],sales_value_p[:5])
plt.xlabel('product')
plt.ylabel('value')
plt.xticks(rotation=90)
plt.title('product Vs total Sales value')
```

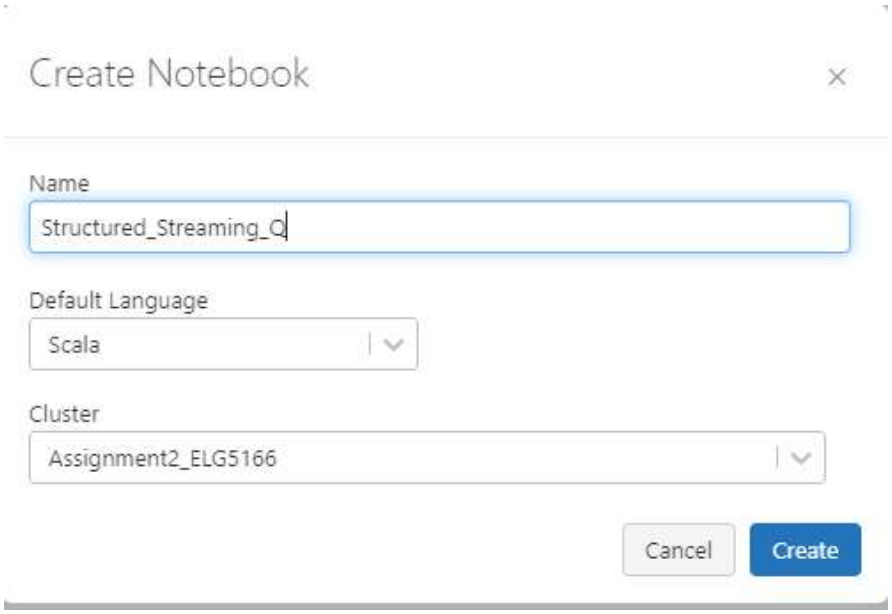



```
# f) visualization
country = [x["Country"] for x in value_for_each_country.rdd.collect()]
sales_value = [x["sum(sales_value)"] for x in value_for_each_country.rdd.collect()]
plt.figure(figsize=[15,8])
plt.bar(country,sales_value)
plt.xlabel('country')
plt.ylabel('sum of sales value')
plt.xticks(rotation=90)
plt.title('country Vs total Sales value')
```



3. Structured Streaming:

A) Create a new notebook:




```
1  /** Reading the dataset as a DataFrame **/
2  val Retail_data = spark.read.format("csv")
3                    .option("header", "true")
4                    .csv("/FileStore/tables/Retail_Data/*.csv")
5  val Retail_data_schema = Retail_data.schema
```

```
1  Retail_data.show()
```

▸ (1) Spark Jobs

```
+---------+---------+--------------------+--------+-------------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|        InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+-------------------+---------+----------+--------------+
|   580538|    23084|  RABBIT NIGHT LIGHT|      48|2011-12-05 08:38:00|     1.79|   14075.0|United Kingdom|
|   580538|    23077| DOUGHNUT LIP GLOSS |      20|2011-12-05 08:38:00|     1.25|   14075.0|United Kingdom|
|   580538|    22906|12 MESSAGE CARDS ...|      24|2011-12-05 08:38:00|     1.65|   14075.0|United Kingdom|
|   580538|    21914|BLUE HARMONICA IN...|      24|2011-12-05 08:38:00|     1.25|   14075.0|United Kingdom|
|   580538|    22467|   GUMBALL COAT RACK|       6|2011-12-05 08:38:00|     2.55|   14075.0|United Kingdom|
|   580538|    21544|SKULLS  WATER TRA...|      48|2011-12-05 08:38:00|     0.85|   14075.0|United Kingdom|
|   580538|    23126|FELTCRAFT GIRL AM...|       8|2011-12-05 08:38:00|     4.95|   14075.0|United Kingdom|
|   580538|    21833|CAMOUFLAGE LED TORCH|      24|2011-12-05 08:38:00|     1.69|   14075.0|United Kingdom|
|   580539|    21479|WHITE SKULL HOT W...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
|   580539|   84030E|ENGLISH ROSE HOT ...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
|   580539|    23355|HOT WATER BOTTLE ...|       4|2011-12-05 08:39:00|     4.95|   18180.0|United Kingdom|
|   580539|    22111|SCOTTIE DOG HOT W...|       3|2011-12-05 08:39:00|     4.95|   18180.0|United Kingdom|
|   580539|    21115|ROSE CARAVAN DOOR...|       8|2011-12-05 08:39:00|     1.95|   18180.0|United Kingdom|
|   580539|    21411|GINGHAM HEART  DO...|       8|2011-12-05 08:39:00|     1.95|   18180.0|United Kingdom|
|   580539|    23235|STORAGE TIN VINTA...|      12|2011-12-05 08:39:00|     1.25|   18180.0|United Kingdom|
|   580539|    23239|SET OF 4 KNICK KN...|       6|2011-12-05 08:39:00|     1.65|   18180.0|United Kingdom|
|   580539|    22197|      POPCORN HOLDER|      36|2011-12-05 08:39:00|     0.85|   18180.0|United Kingdom|
|   580539|    22693|GROW A FLYTRAP OR...|      24|2011-12-05 08:39:00|     1.25|   18180.0|United Kingdom|
```

Command took 1.80 seconds -- by babde014@uottawa.ca at 11/3/2022, 4:27:02 PM on Assignment2_ELG5166

B) 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.

```
/*Reads 20 file per trigger */
val Retail_Streaming_Data = spark
                  .readStream.format("csv")
                  .schema(Retail_data_schema)
                  .option("maxFilesPerTrigger", 20)
                  .option("header", "true")
                  .load("/FileStore/tables/Retail_Data/*.csv")
```

```
import org.apache.spark.sql.functions.{sum, col}

/*sum of the quantity column and unitprice column*/

val sum_of_values = Retail_Streaming_Data
              .withColumn("sum_values",col("Quantity")*col("UnitPrice"))
              .groupBy("CustomerID").agg(sum("sum_values").alias("total value"),sum("Quantity").alias("total stocks"))
```

```
val values = sum_of_values.writeStream.queryName("sum_values")
                  .format("memory").outputMode("complete")
                  .start()
```

```
for( i <- 1 to 5 ) {
    spark.sql("SELECT * FROM sum_values").show()
    Thread.sleep(2000)
}
```

▸ (5) Spark Jobs

```
+----------+------------------+------------+
|CustomerID|       total value|total stocks|
+----------+------------------+------------+
|   14349.0|133.50000000000006|        86.0|
|   17966.0|           1098.43|      2571.0|
|   13259.0| 292.3199999999999|       132.0|
|   17955.0|             557.3|       273.0|
|   17786.0|            278.74|       200.0|
|   13178.0| 5725.469999999999|      3570.0|
|   16982.0|            384.06|       182.0|
|   12891.0|             331.0|       950.0|
|   16553.0| 5664.570000000001|      4595.0|
|   12535.0| 716.3500000000001|       219.0|
|   13514.0|152.20000000000002|        40.0|
|   16557.0|            281.85|       111.0|
|   16917.0| 391.5200000000001|       232.0|
|   15396.0|288.17999999999995|       122.0|
|   15039.0|19786.440000000002|      9191.0|
|   14542.0|103.25000000000001|        19.0|
|   12985.0|           1215.62|      1413.0|
|   13067.0|115.46000000000002|        32.0|
```

C) 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):

```
val Trigger_Time = Retail_Streaming_Data
              .withColumn("sum_values",col("Quantity")*col("UnitPrice"))
              .withColumn("trigger_time", current_timestamp)
              .groupBy("trigger_time").agg(sum("sum_values").alias("total value"),count("sum_values").alias("Records Imported"))
```

```
val Trigger_Time_Query = Trigger_Time.writeStream.queryName("Trigger_Time")
                  .format("memory").outputMode("complete")
                  .start()
```

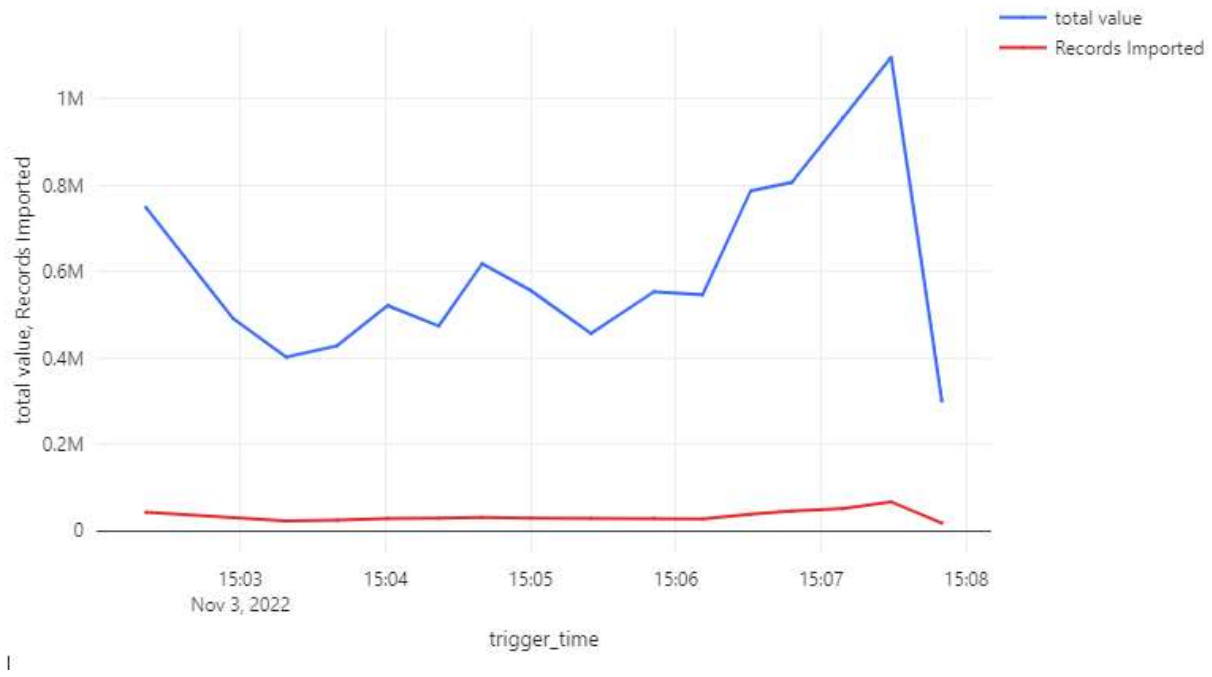
```
for( i <- 1 to 5 ) {
    spark.sql("SELECT * FROM Trigger_Time").show()
    Thread.sleep(2000)
}
```

▸ (15) Spark Jobs

```
+--------------------+------------------+----------------+
|        trigger_time|       total value|Records Imported|
+--------------------+------------------+----------------+
|2022-11-03 15:06:...| 546813.7599999994|           27235|
|2022-11-03 15:07:...| 300605.2199999994|           17706|
|2022-11-03 15:05:...| 553666.8210000007|           29065|
|2022-11-03 15:07:...| 1096823.310000005|           67085|
|2022-11-03 15:05:...|457003.47999999946|           28830|
|2022-11-03 15:04:...|474026.51099999883|           28782|
|2022-11-03 15:04:...| 618570.0800000019|           30849|
|2022-11-03 15:03:...|402141.14999999804|           22240|
|2022-11-03 15:02:...|         491519.26|           30333|
|2022-11-03 15:03:...| 428129.9299999994|           24503|
|2022-11-03 15:02:...| 748957.0200000004|           42481|
|2022-11-03 15:06:...|  806761.440999999|           45765|
|2022-11-03 15:07:...| 957173.6099999974|           51145|
|2022-11-03 15:05:...| 556884.2399999996|           29232|
|2022-11-03 15:06:...| 787156.3709999995|           38098|
|2022-11-03 15:04:...|  521515.729999999|           28560|
+--------------------+------------------+----------------+
```

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

```
for (i <- 1 to 5 ){
  display(spark.sql("SELECT * FROM Trigger_Time"))
  Thread.sleep(2000)
}
```



References

[1] A Tale of Three Apache Spark APIs: RDDs vs DataFrames and Datasets. (2016, July 14). Databricks. https://www.databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

[2] Difference Between SparkSession , SparkContext , SQLContext & HiveContext -. (2019, September 9). Gankrin. https://gankrin.org/sparksession-vs-sparkcontext-vs-sqlcontext-vs-hivecontext/

[3] Explain RDDs Datasets and Dataframes in Apache Spark -. (n.d.). ProjectPro. Retrieved November 9, 2022, from https://www.projectpro.io/recipes/what-is-difference-between-rdds-datasets-and-dataframes-apache-spark#:~:text=The%20Dataframes%20provide%20API%20quickly

[4] Hasni, A. (2022, September 27). 3 Reasons Why Spark's Lazy Evaluation is Useful. *Medium*. https://towardsdatascience.com/3-reasons-why-sparks-lazy-evaluation-is-useful-ed06e27360c4#:~:text=In%20Spark%2C%20Lazy%20Evaluation%20means

[5] MapReduce Tutorial | Mapreduce Example in Apache Hadoop. (2016, November 15). Edureka. https://www.edureka.co/blog/mapreduce-tutorial/

[6] Overview: estimators, transformers and pipelines - spark.ml - Spark 2.0.0 Documentation. (n.d.). Spark.apache.org. Retrieved November 9, 2022, from https://spark.apache.org/docs/2.0.0-preview/ml-guide.html

[7] Spark Streaming vs. Structured Streaming - DZone Big Data. (n.d.). Dzone.com. Retrieved November 9, 2022, from https://dzone.com/articles/spark-streaming-vs-structured-streaming

[8] Srivastava, R., & Bhatt, V. (2022, March 16). *SparkSession Vs SparkContext - What Are The Differences?* Ksolves Blog. https://www.ksolves.com/blog/big-data/spark/sparksession-vs-sparkcontext-what-are-the-

differences#:~:text=From%20Spark%202.0%2C%20SparkSession%20provides%20a%20common%20entry

[9] Team, D. (2017, April 1). *Lazy Evaluation in Apache Spark – A Quick guide*. DataFlair. https://data-flair.training/blogs/apache-spark-lazy-evaluation/#:~:text=Hence%2C%20Lazy%20evaluation%20enhances%20the,performance%20and%20achieves%20fault%20tolerance.

[10] Tobin, D. (n.d.). *Spark vs Hadoop MapReduce: 5 Key Differences*. Integrate.io. Retrieved November 8, 2022, from https://www.integrate.io/blog/apache-spark-vs-hadoop-mapreduce/#:~:text=Spark%20vs%20MapReduce%3A%20Performance

[11] What is DFS(Distributed File System)? (2020, July 5). GeeksforGeeks. https://www.geeksforgeeks.org/what-is-dfsdistributed-file-system/