ELG 5166 – Cloud Analytics

Assignment #2 - (Teamwork)

Due: Submit on MS Teams or email, no later than Midnight, November 06th, 2022.

This is a teamwork assignment (**Only one submission per team**).

Weight: 20% of the final mark**.**

Important Note: Read the following academic integrity statement, type in your full name and student ID, and include a copy in your submission. Submitting this form electronically by one of the team members is considered the same as signing the document by all members of the team.

Personal Ethics & Academic Integrity Statement

Student name: Sondos Mohammed Hussein Ali Student ID: 300327219

Student Name: Esraa Ahmed Abdel hakam Student ID: 300327225

Student Name: Abdallah Mohamed Mahmoud Mohamed Ragab Student ID: 300327288

Student Name: Hosam Mahmoud Ibrahim Mahmoud Student ID: 300327269

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 nonparticipation 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,

ELG 5166 – Cloud Analytics

- 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

ELG 5166 – Cloud Analytics

Part 1 Definitions (50 points)

1. ***Describe a distributed file system? Briefly describe with examples, any two implementations of a distributed file system. (6 pts)***

A Distributed File System (DFS), is a file system that is distributed on multiple file servers or multiple locations. It allows programs to access or store isolated files as they do with the local ones, allowing programmers to access files from any network or computer.

Implementations of a distributed file system:

a- Hadoop:

Hadoop is a group of open-source software services. It gives a software framework for distributed storage and operating of big data using the MapReduce programming model. The core of Hadoop contains a storage part, known as Hadoop Distributed File System (HDFS), and an operating part which is a MapReduce programming model.

b- NFS:

NFS stands for Network File System. It is a client-server architecture that allows a computer user to view, store, and update files remotely. The protocol of NFS is one of the several distributed file system standards for Network-Attached Storage (NAS).

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

2. ***Describe briefly 3 features of Apache Hadoop Map-Reduce and 3 limitations associated with it when compared to Apache Spark? (12 pts)***

Features:

- **Highly scalable**
  A framework with excellent scalability is Apache Hadoop MapReduce. This is because of its capacity for distributing and storing large amounts of data across numerous servers.
- **Simplicity:**
  Easy to compute the number of blocks required to store a file
- **Secure:**
  The MapReduce programming model uses the HBase and HDFS security approaches, and only authenticated users are permitted to view and manipulate the data.

Limitations:

- Fixed Map & Reduce slots:
  map-reduce performs the function of resource management and processing.
- Synchronization barriers:
  all map tasks must complete before the reducers are initiated.
- Single Job Tracker:
  a cluster is configured for a single job at a time

What Is MapReduce? Meaning, Working, Features, and Uses |. https://www.spiceworks.com/tech/big-data/articles/what-is-map-reduce/. Accessed 4 Nov. 2022

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 (12 pts)***

Description	low-level APIs in Apache Spark (RDDs) Is an immutable distributed collection of data elements partitioned across cluster nodes that can be operated in parallel with a low-level API that offers transformations and actions.	High-level APIs in Apache Spark (Data frames or Datasets) is an immutable distributed data collection. It provides a domain-specific language API for data distribution manipulation. It is based on RDDs and is compatible with a variety of programming languages, including R, Scala, and Python.
Differences:	• data is unstructured like, media streams or streams of text • make the cluster partitions visible.	• data is structured • provide a wide range of functionalities
when do you use one over the other	• When someone wants to manipulate data using functional programming constructs rather than domain-specific expressions. • When someone doesn't care about imposing a schema, like columnar format, while processing or accessing data attributes by name or column • When someone is willing to forego some of the optimization and performance benefits provided by Data Frames and Datasets for structured and semi-structured data	• When someone desires extensive semantics. he can use Data Frames or Datasets. • When someone's processing demands high-level expressions, filters, maps, aggregation, averages, sum, SQL queries, columnar access, and use of lambda functions on semi-structured data. He can use Data Frames or Datasets. • When someone wants to unify and simplify APIs across Spark Libraries. He can use Data Frames or Datasets. • When someone is R or Python user, He can use Data Frames first and then RDDs if he needs more control, so he can use Data Frames • if someone wants a higher level of type-safety at compile time, typed JVM objects, Catalyst optimization, and Tungsten's efficient code generation. He can Use Dataset

Databricks, 14 July 2016, https://www.databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html.

4. ***Describe the following Apache Spark terms with examples (10 pts each)***

   a. ***Immutability in Spark***

   immutability means that you can't change an object after its creation. In spark this means you can't change the data after the creation in other words when performing a transformation on the data you don't change the original data it creates a new vision of the data which is not stored and saves the list of a transformation like what is shown in the following figure

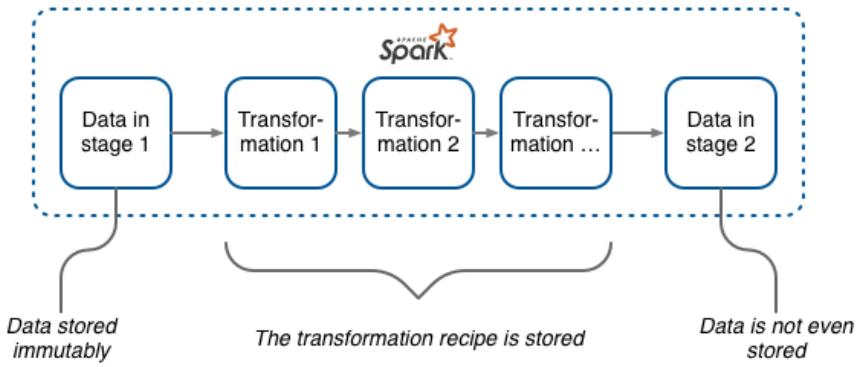


Figure 1. Immutability

RDD, Datasets, and dataframes are considered immutable storage.

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

First, there is 2 type of operation transformation and action, transformations are applied to all the dataset like mapping or grouping and more in the other hand actions tend to produce some results like aggregation operation like sum, min, max, and so on.

Lazy Evaluation means the transformation operation will not be executed until action is executed which means an execution plan will be produced for the list of the transformation operation which can be analyzed and optimized
For example, there is a need to join two dataframe and then perform an action like filtering the rows, spark will analyze the operation and might perform the filtering first then join the results which maybe be much faster.

impact on Spark performance:

- Saves Computation and increases Speed.
- Reduces time and space complexity.

c. How is SparkSession different from SparkContext

- SparkContext
  was the entry point for spark capabilities in spark 1.0 and to use other APIs like SQL, Hive, and streaming context you have to create the context separately.

- SparkSession
  In Spark 2.0 sprakSession provides a entry point for spark capability like SparkContext but father more no need to create a separate context for SQL, Hive, and streaming and it provides

d. Spark MLlib Transformers, Estimators, and Evaluators

- **Transformers**
  Consider functions that map the data from one form to another one Like StopWordsRemover or change changing the column data type.

- **Estimators**
  It means that any algorithm that trains on data by using the learned parameter to transform the data like StandardScaler which will learn mean and standard deviation using these parameters to scale the data.

- **Evaluators**
  It is a collection of functions to measure the performance of the model and return suite metric.
  There is many evaluators like BinaryClassificationEvaluator and ClusteringEvaluator

References:

https://freecontent.manning.com/the-majestic-role-of-the-dataframe-in-spark/]

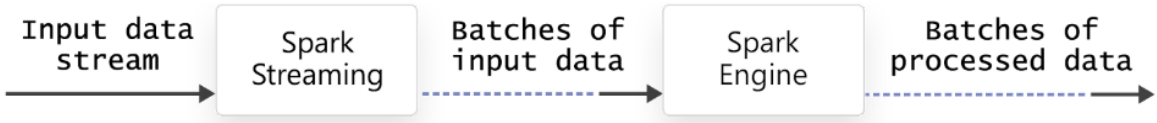
https://www.techtarget.com/searchdatamanagement/tip/Why-Spark-DataFrame-lazy-evaluation-models-outpace-MapReduce

https://towardsdatascience.com/sparksession-vs-sparkcontext-vs-sqlcontext-vs-hivecontext-741d50c9486a

5. ***Describe briefly – with examples - how Spark Streaming differs from Spark Structured Streaming? (10 pts)***

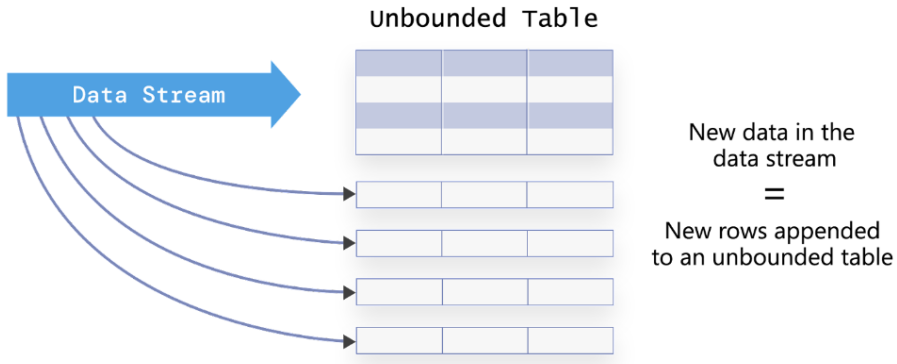
1- Spark Streaming

- The data is received by Spark Streaming, which then splits it into batches for the Spark Engine to process.
- Each incoming record belongs to a batch of DStream. Each batch represents an RDD, and as it's known the RDDs are slower than both the Dataframes and the Datasets.
- Spark Streaming only works with the timestamp when the data is received by the Spark. Based on the ingestion timestamp, Spark Streaming puts the data in a batch even if the event is generated early and belonged to the earlier batch, which may result in less accurate information as it is equal to the data loss.



2- Structured Streaming

- In Structured Streaming, there is no batch concept. The received data in a trigger is appended to the continuously flowing data stream. Each row of the data stream is processed and the result is updated into the unbounded result table.
- Structured Streaming uses DataFrame and Dataset APIs to perform streaming operations.
- Structured Streaming provides the functionality to process data on the basis of event-time when the timestamp of the event is included in the data received, with this feature Structured Streaming provides a different way of processing the data according to the time of data generation in the real world, which could handle data coming in late and get more accurate results.



Data stream as an unbounded table

References:
Spark Structured Streaming: Tutorial With Examples (macrometa.com)
Spark Streaming vs Structured Streaming. | by Prag Tyagi | towardsdataanalytics | Medium
Explain RDDs Datasets and Dataframes in Apache Spark (projectpro.io)

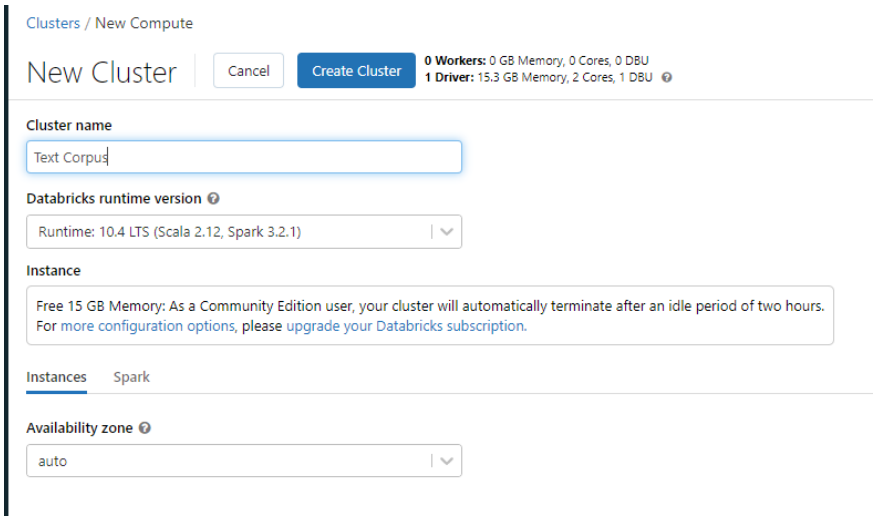
Part 2 – Spark Examples (50 points)

1) Data Transformation Pipelines (15 pts)

Please show evidence of your setup with screenshots.

- Creating an account via https://community.cloud.databricks.com/

- Creating cluster named: (Text corpus)

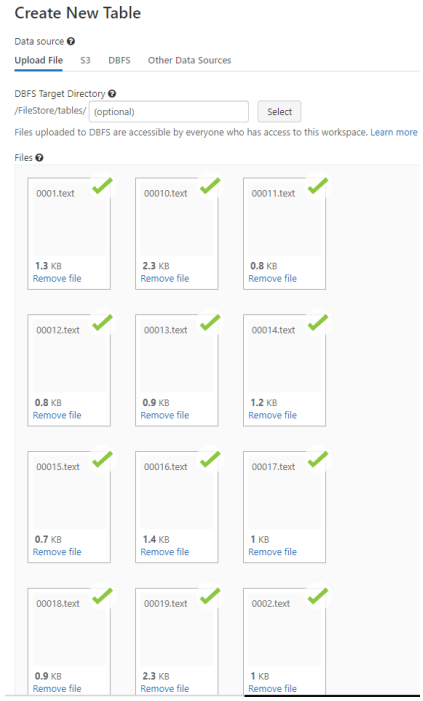


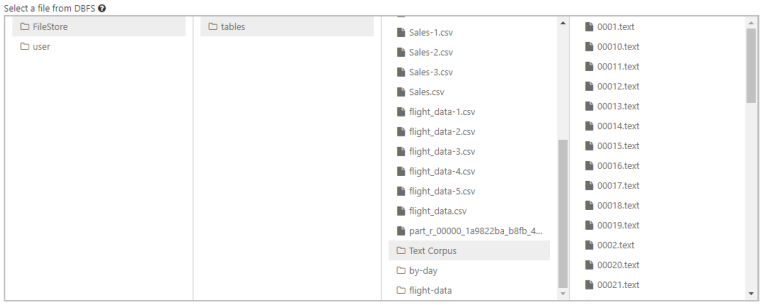
Download and use this text corpus for this question - https://1drv.ms/u/s!AkmeceHomH_DiLltIz-7D1VgatoUOQ?e=cqF3mA

Create a notebook in Scala on an active cluster, and use the entire directory as your text file data source. Use your notebook to answer the questions below. Provide **screenshots** of the following:

ELG 5166 – Cloud Analytics

a) Uploaded the files to your DBFS table space.





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

```
val textFile = spark.sparkContext.textFile("/FileStore/tables/Text Corpus")
// textFile.count()
display(dbutils.fs.ls("/FileStore/tables/Text Corpus"))
```

Table +

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/Text Corpus/0001.text	0001.text	1250	1667503047000
2	dbfs:/FileStore/tables/Text Corpus/00010.text	00010.text	2305	1667503047000
3	dbfs:/FileStore/tables/Text Corpus/00011.text	00011.text	788	1667503047000
4	dbfs:/FileStore/tables/Text Corpus/00012.text	00012.text	826	1667503047000
5	dbfs:/FileStore/tables/Text Corpus/00013.text	00013.text	945	1667503048000
6	dbfs:/FileStore/tables/Text Corpus/00014.text	00014.text	1155	1667503048000
7	dbfs:/FileStore/tables/Text Corpus/00015.text	00015.text	682	1667503049000

Showing all 50 rows. | 1.21 seconds runtime

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

```
println("the number of lines across all the files are " +textFile.count())
```

(1) Spark Jobs

```
the number of lines across all the files are 1645
```

Command took 2.25 seconds -- by sali207@uottawa.ca at 11/4/2022, 1:05:10 AM on Text Corpus

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

```
val word_count = textFile.flatMap(line => line.split(","))
                    .map(word => (word, 1))
                    .filter(word=> word._1.contains("antibiotics"))
                    .reduceByKey(_+_)

println("word_count : "+word_count.count())
```

```
▸ (1) Spark Jobs
word_count : 2
word_count: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[35] at reduceByKey at command-437863842162084:4
Command took 2.98 seconds -- by sali207@uottawa.ca at 11/3/2022, 10:38:41 PM on Text Corpus
```

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

```
val word_count = textFile.flatMap(line => line.split(","))
                    .map(word => (word, 2))
                    .filter(word=> word._1.contains("patient") &&  word._1.contains("admitted"))
println("word_count : "+ word_count.count())
```

```
▸ (1) Spark Jobs
word_count : 7
word_count: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[70] at filter at command-568687511816308:3
Command took 2.16 seconds -- by sali207@uottawa.ca at 11/4/2022, 1:18:09 AM on Text Corpus
```

f) Upload your exported (. scala format) notebook as part of your submission. Please include the data in your Brightspace submission.

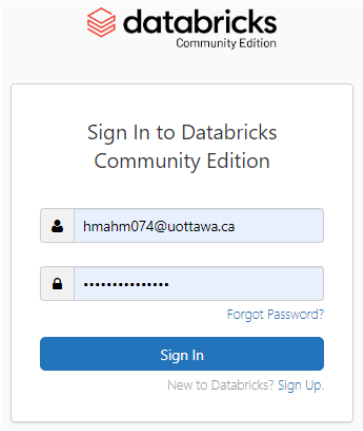
2) **Retail Data Analysis (15 pts)**

Download the retail-data - https://1drv.ms/u/s!AkmeceHomH_DiKwNUGQDcshwBb1Sw?e=w31bLw

Use a new Data Bricks notebook (in SQL, Python, or Scala) to answer the following questions. Provide **screenshots** of the following:

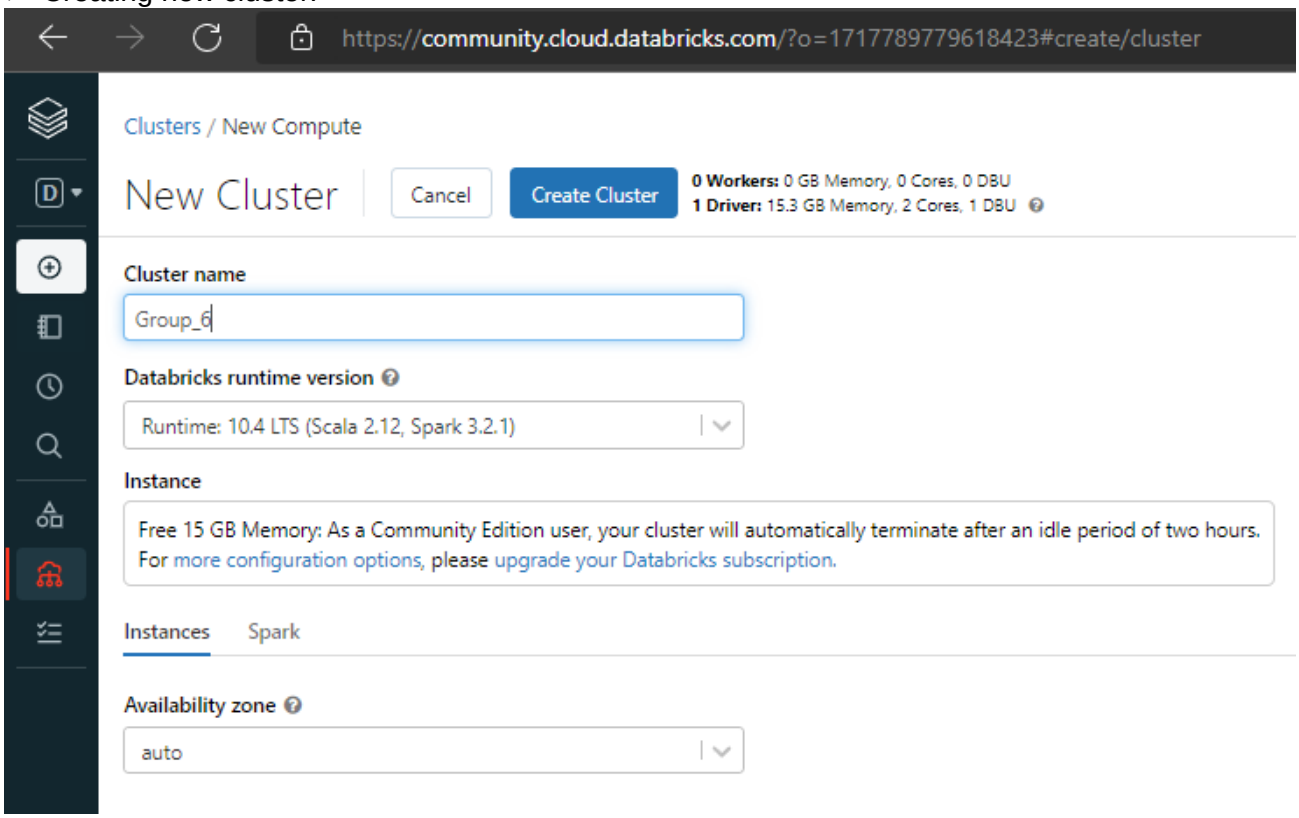
a) Uploaded the files to your DBFS table space.

- Login to databricks using my uOttawa Email.

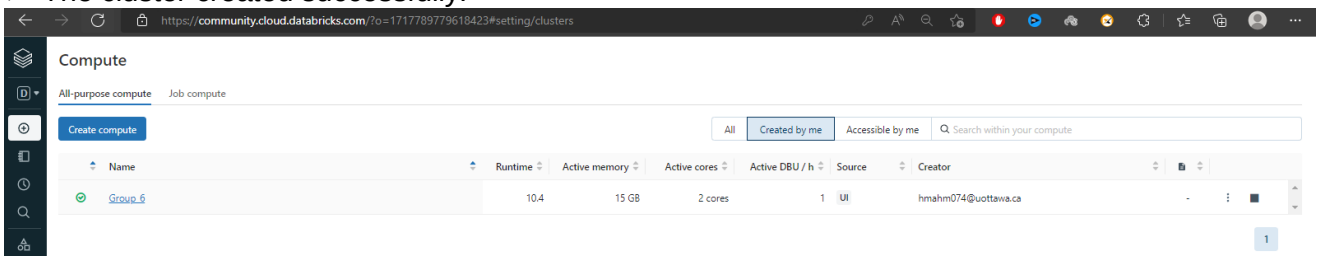


ELG 5166 – Cloud Analytics

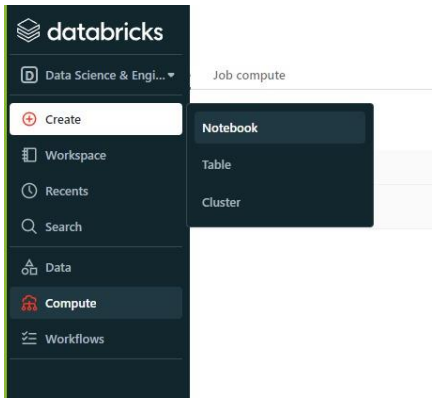
- Creating new cluster.



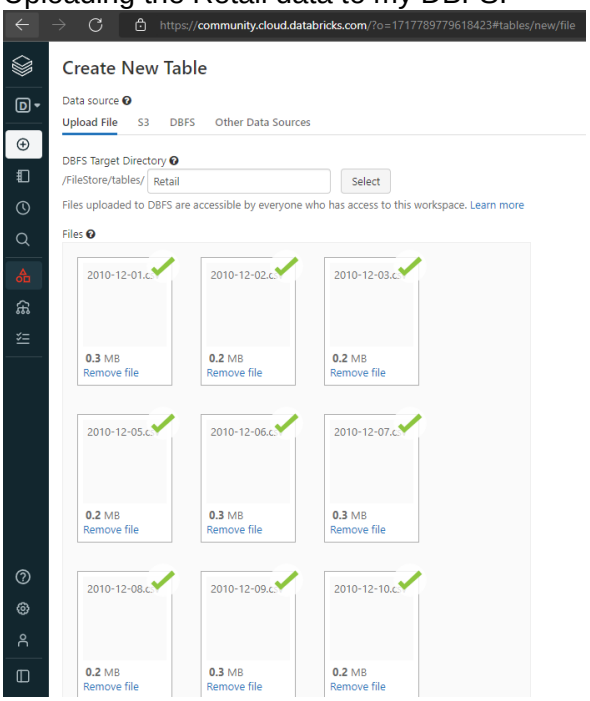
- The cluster created successfully.



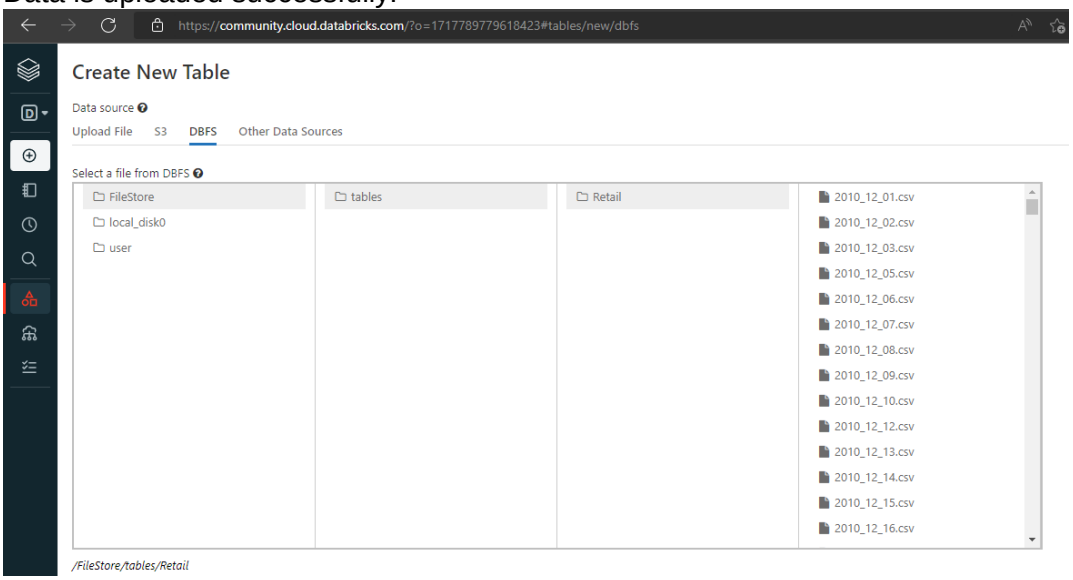
- Create New Notebook.



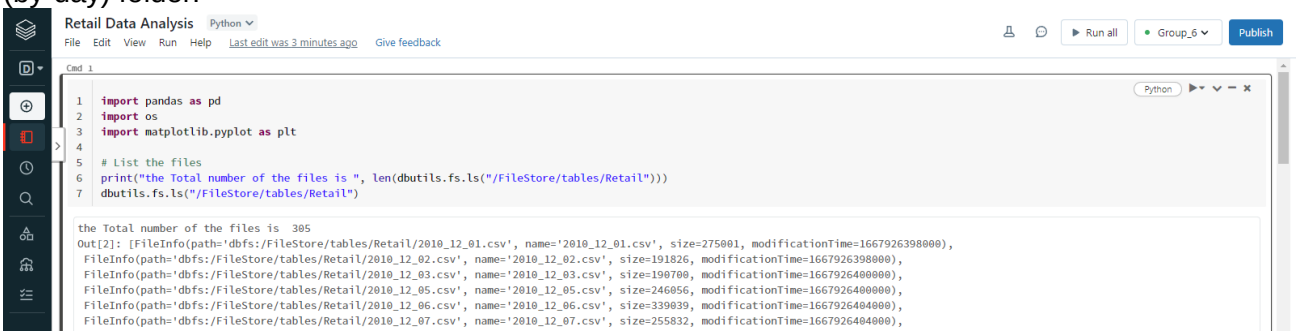
- Uploading the Retail data to my DBFS.



- Data is uploaded successfully.



- The number of the uploaded files is (305) which is the same number of files that in (by-day) folder.



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

- The Total number of transactions is 541909

Retail Data Analysis Python
File Edit View Run Help Last edit was 6 minutes ago Give feedback
Run all Group_6 Publish

Cmd 3

```python
# b) Output the total number of transactions across all the files and the total value of the transactions.
# Read all csv files and print the transactions
files = []
total_transactions = 0
for i in range(len(dbutils.fs.ls("/FileStore/tables/Retail"))):
    temp = r"/FileStore/tables/Retail/" + dbutils.fs.ls("/FileStore/tables/Retail")[i][1]
    files.append(spark.read.option("header",True).csv(temp).toPandas())
    print("number of transactions for this file", dbutils.fs.ls("/FileStore/tables/Retail")[i][1] ,"=", len(files[i]["InvoiceNo"]))
    total_transactions = total_transactions + len(files[i]["InvoiceNo"])

print("the total number of the transactions is :", total_transactions)
```

(100) Spark Jobs

```
number of transactions for this file 2011_11_17.csv = 3621
number of transactions for this file 2011_11_18.csv = 2920
number of transactions for this file 2011_11_20.csv = 3334
number of transactions for this file 2011_11_21.csv = 2930
number of transactions for this file 2011_11_22.csv = 3967
number of transactions for this file 2011_11_23.csv = 3619
number of transactions for this file 2011_11_24.csv = 3766
number of transactions for this file 2011_11_25.csv = 3125
number of transactions for this file 2011_11_27.csv = 2543
number of transactions for this file 2011_11_28.csv = 3330
number of transactions for this file 2011_11_29.csv = 4313
number of transactions for this file 2011_11_30.csv = 3454
number of transactions for this file 2011_12_01.csv = 2901
number of transactions for this file 2011_12_02.csv = 2880
number of transactions for this file 2011_12_04.csv = 2038
number of transactions for this file 2011_12_05.csv = 5331
number of transactions for this file 2011_12_06.csv = 3365
number of transactions for this file 2011_12_07.csv = 2438
number of transactions for this file 2011_12_08.csv = 4940
number of transactions for this file 2011_12_09.csv = 1632
the total number of the transactions is : 541909
```

- The Total value of transactions is 9747747.9

Retail Data Analysis Python
File Edit View Run Help Last edit was 7 minutes ago Give feedback
Run all Group_6 Publish

Cmd 4

```python
# Create total DataFrame
totalDF = pd.DataFrame()

for i in range(len(files)):
    totalDF = pd.concat((totalDF, files[i]),axis = 0)

totalDF["UnitPrice"] = totalDF["UnitPrice"].astype(float)
totalDF["Quantity"] = totalDF["Quantity"].astype(float)

totalDF['total_purchases'] = totalDF['Quantity'] * totalDF['UnitPrice']

totalvalue = totalDF['total_purchases'].sum()
print("the total value of the transactions is :", totalvalue)
```

```
the total value of the transactions is : 9747747.933999998
```

Command took 6.46 seconds -- by hmahm074@uottawa.ca at 11/8/2022, 11:19:55 PM on Group_6

c) Output the 5 top-selling products.

Retail Data Analysis Python
File Edit View Run Help Last edit was 3 minutes ago Give feedback
Run all Group_6 Publish

Cmd 5

```python
# Output the 5 top-selling products
print("the 5 top-selling products")
totalDF["Quantity"] = totalDF["Quantity"].astype(float)
top_selling_products = totalDF.copy().groupby(["StockCode"]).sum().sort_values(by=['Quantity'], ascending=False)["Quantity"].reset_index().iloc[0:5]
top_selling_products.head(5)
```

```
the 5 top-selling products
```

	StockCode	Quantity
0	22197	56450.0
1	84077	53847.0
2	85099B	47363.0
3	85123A	38830.0
4	84879	36221.0

Command took 0.38 seconds -- by hmahm074@uottawa.ca at 11/8/2022, 11:12:54 PM on Group_6

d) Output the 5 topmost valuable products.

- We have solved the questions in 2 ways:
  - The first way is we have sorted the unit price column and take the highest five products and we removed the duplicates products (we only toke the highest unit price for each product).

Retail Data Analysis Python
File Edit View Run Help Last edit was 14 minutes ago Give feedback
Run all Group_6 Publish

Cmd 6

```python
# Output the 5 topmost valuable products (5 top most higher unit prices)
print("the 5 topmost valuable products (5 top most higher unit prices)")
totalDF["UnitPrice"] = totalDF["UnitPrice"].astype(float)
topmost_valuable_products_unit_price = totalDF.sort_values('UnitPrice', ascending=False).drop_duplicates(['StockCode'])
topmost_valuable_products_unit_price.head(5)
```

```
the 5 topmost valuable products (5 top most higher unit prices)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	total_purchases
857	C556445	M	Manual	-1.0	2011-06-10 15:31:00	38970.00	15098.0	United Kingdom	-38970.00
399	C580605	AMAZONFEE	AMAZON FEE	-1.0	2011-12-05 11:36:00	17836.46	None	United Kingdom	-17836.46
628	A563185	B	Adjust bad debt	1.0	2011-08-12 14:50:00	11062.06	None	United Kingdom	11062.06
826	C551685	POST	POSTAGE	-1.0	2011-05-03 12:51:00	8142.75	16029.0	United Kingdom	-8142.75
353	562955	DOT	DOTCOM POSTAGE	1.0	2011-08-11 10:14:00	4505.17	None	United Kingdom	4505.17

Command took 0.32 seconds -- by hmahm074@uottawa.ca at 11/8/2022, 11:36:34 PM on Group_6

- The second way is we have calculated the value by multiplying quantities * unit prices and we aggregated by the sum of this value for each product.

Retail Data Analysis Python
File Edit View Run Help Last edit was 15 minutes ago Give feedback
Run all Group_6 Publish

Cmd 7

```python
# Output the 5 topmost valuable products (5 top most higher values (group by stock id) (Quantity * unit price))
print("the 5 topmost valuable products (5 top most higher values (group by stock id) (Quantity * unit price)")
topmost_valuable_products_value = totalDF.copy().groupby(["StockCode"]).sum().sort_values(by=['total_purchases'], ascending=False)
topmost_valuable_products_value = topmost_valuable_products_value.iloc[:, [2]].reset_index()
topmost_valuable_products_value.head(5)
```

```
the 5 topmost valuable products (5 top most higher values (group by stock id) (Quantity * unit price)
```

	StockCode	total_purchases
0	DOT	206245.48
1	22423	164762.19
2	47566	98302.98
3	85123A	97894.50
4	85099B	92356.03

Command took 0.18 seconds -- by hmahm074@uottawa.ca at 11/8/2022, 11:55:17 PM on Group_6

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

Retail Data Analysis Python
File Edit View Run Help Last edit was 16 minutes ago Give feedback
Run all Group_6 Publish

Cmd 8

```python
# Output each country and the total value of their purchases.
print("total value of purchases for each country")
pd.options.display.float_format = '{:.2f}'.format
country_purchases = totalDF.groupby(["Country"]).sum().sort_values(by=['total_purchases'], ascending=False)["total_purchases"].reset_index()
totalDF['total_purchases'] = totalDF['total_purchases'].astype(float)
country_purchases.head(len(totalDF["Country"].unique().tolist()))
```

```
total value of purchases for each country
```

	Country	total_purchases
0	United Kingdom	8187806.36
1	Netherlands	284661.54
2	EIRE	263276.82
3	Germany	221698.21
4	France	197403.90
5	Australia	137077.27
6	Switzerland	56385.35
7	Spain	54774.58
8	Belgium	40910.96
9	Sweden	36595.91
10	Japan	35340.62

11	Norway	35163.46
12	Portugal	29367.02
13	Finland	22326.74
14	Channel Islands	20086.29
15	Denmark	18768.14
16	Italy	16890.51
17	Cyprus	12946.29
18	Austria	10154.32
19	Hong Kong	10117.04
20	Singapore	9120.39
21	Israel	7907.82
22	Poland	7213.14
23	Unspecified	4749.79
24	Greece	4710.52
25	Iceland	4310.00
26	Canada	3666.38
27	Malta	2505.47
28	United Arab Emirates	1902.28
29	USA	1730.92
30	Lebanon	1693.88
31	Lithuania	1661.06
32	European Community	1291.75
33	Brazil	1143.60
34	RSA	1002.31
35	Czech Republic	707.72
36	Bahrain	548.40
37	Saudi Arabia	131.17

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

- The graph of the first way (max 5 elements of unit prices)

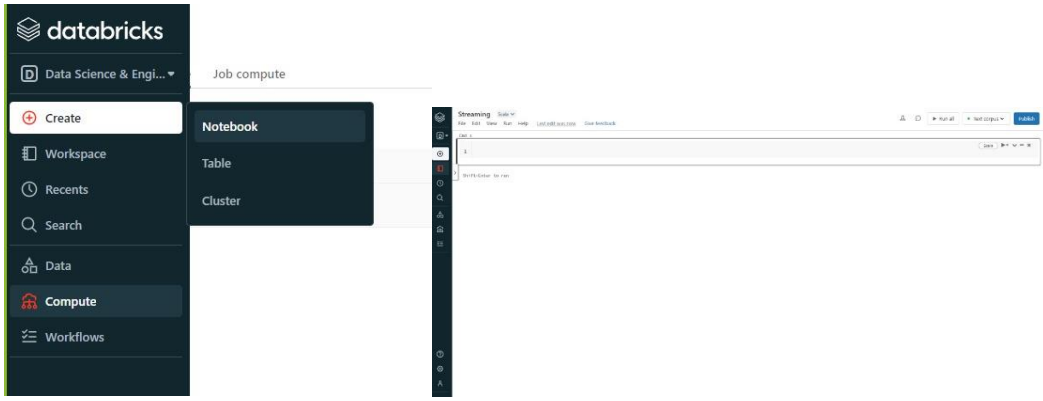
- The graph of the second way (max 5 of (quantities * unit prices))

g) Upload your notebook as part of your submission. Please do not upload the data on Brightspace.

3) **Structured Streaming (20 pts)**

Use the retail data from question Part 2. Provide **screenshots** of the following:

a) Create a new notebook.



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.

- Read the schema from the data

Cmd 1

```
val Static_Data = spark
  .read
  .option("inferSchema", "true")
  .option("header", "true")
  .csv("/FileStore/tables/retail-data/by-day/*.csv")


val dataSchema = Static_Data.schema
```

(5) Spark Jobs

Static_Data: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
Static_Data: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6
dataSchema: org.apache.spark.sql.types.StructType = StructType(StructField(InvoiceNo,Str
Field(InvoiceDate,TimestampType,true),StructField(UnitPrice,DoubleType,true),StructField
```

Command took 11.15 seconds -- by ragababdallah1589@gmail.com at 11/8/2022, 8:24:32 PM on Text corpus

- Read the stream with previous schema

```
val streaming = spark
                .readStream.schema(dataSchema)
                .option("maxFilesPerTrigger", 20)
                .csv("/FileStore/tables/retail-data/by-day/*.csv")
```

(1) Spark Jobs

streaming: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
streaming: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
```

Command took 1.92 seconds -- by ragababdallah1589@gmail.com at 11/8/2022, 8:24:33 PM on Text corpus

ELG 5166 – Cloud Analytics

- Create Query

```
import org.apache.spark.sql.functions._

val activityCounts = streaming.withColumn("Stockesvalue",col("UnitPrice")*col("Quantity")).groupBy("CustomerID").agg(
      sum("Quantity").as("total stocks"),
      sum("Stockesvalue").as("total value"))
```

activityCounts: org.apache.spark.sql.DataFrame = [CustomerID: double, total stocks: long ... 1 more field]

```
import org.apache.spark.sql.functions._
activityCounts: org.apache.spark.sql.DataFrame = [CustomerID: double, total stocks: bigint ... 1 more field]
```

Command took 0.91 seconds -- by ragababdallah1589@gmail.com at 11/8/2022, 8:24:33 PM on Text corpus

```
val Customer_Stock_Query = activityCounts.writeStream.queryName("Stock_Query")
  .format("memory").outputMode("complete")
  .start()
```

Cancel

(1) Spark Jobs

Stock_Query (id: fdd0635b-0938-406c-a9cd-346bb5754323) *Last updated: 10 seconds ago*

```
Customer_Stock_Query: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@45c8a932
```

- Print streamed data this screen shows just the first batch

```
spark.streams.active

Thread.sleep(16000)
```

Command took 16.46 seconds -- by ragababdallah1589@gmail.com at 11/8/2022, 8:25:13 PM on Text corpus

```
for( i <- 1 to 10 ) {
    spark.sql("SELECT * FROM Stock_Query").show()
    Thread.sleep(1000)
}
```

(10) Spark Jobs

```
+----------+------------+------------------+
|CustomerID|total stocks|       total value|
+----------+------------+------------------+
|   16916.0|         103|198.07999999999996|
|   16596.0|          47|            120.65|
|   17884.0|         236|            324.62|
|   13094.0|         192|            223.68|
|   17633.0|         -10|             -67.5|
|   13649.0|          67|             197.8|
|   16656.0|         396|            428.76|
|   16858.0|          83|345.89000000000004|
|   15160.0|          84|158.16000000000003|
|   17392.0|         221|            312.41|
|   15311.0|        1145|           2609.21|
|   16353.0|         120|             510.0|
|   17062.0|         334|431.83000000000015|
|   12967.0|         438|            1660.9|
|   15750.0|         301|326.27000000000004|
|   15898.0|         230|            521.69|
|   17659.0|          85|359.95000000000005|
|   13846.0|         234|411.40000000000015|
```

Command took 1.08 minutes -- by ragababdallah1589@gmail.com at 11/8/2022, 8:25:13 PM on Text corpus

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)

- Create Query

```
val Stream_C = streaming
                            .withColumn("Sales_Value", col("Quantity")*col("UnitPrice"))
                            .withColumn("Records_Imported", lit(1))
                            .withColumn("Trigger_Time", current_timestamp())
                            .groupBy("Trigger_Time")
                            .agg(count("Records_Imported").as("RecordsImported"), sum("Sales_Value").as("SaleValue"))
```

Stream_C: org.apache.spark.sql.DataFrame = [Trigger_Time: timestamp, RecordsImported: long ... 1 more field]

```
Stream_C: org.apache.spark.sql.DataFrame = [Trigger_Time: timestamp, RecordsImported: bigint ... 1 more field]
```

Command took 8.00 seconds -- by ragababdallah1589@gmail.com at 11/8/2022, 8:28:47 PM on Text corpus

```
val stream_c_query = Stream_C.writeStream.queryName("stream_c")
  .format("memory").outputMode("complete")
  .start()
```

Cancel

(1) Spark Jobs

stream_c (id: b1cd449b-e855-4495-bf5c-6fe589de696b) *Last updated: 15 seconds ago*

```
stream_c_query: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@6817b256
```

- Print streamed data this screen shows just a few batches

```
spark.streams.active

Thread.sleep(40000)
```

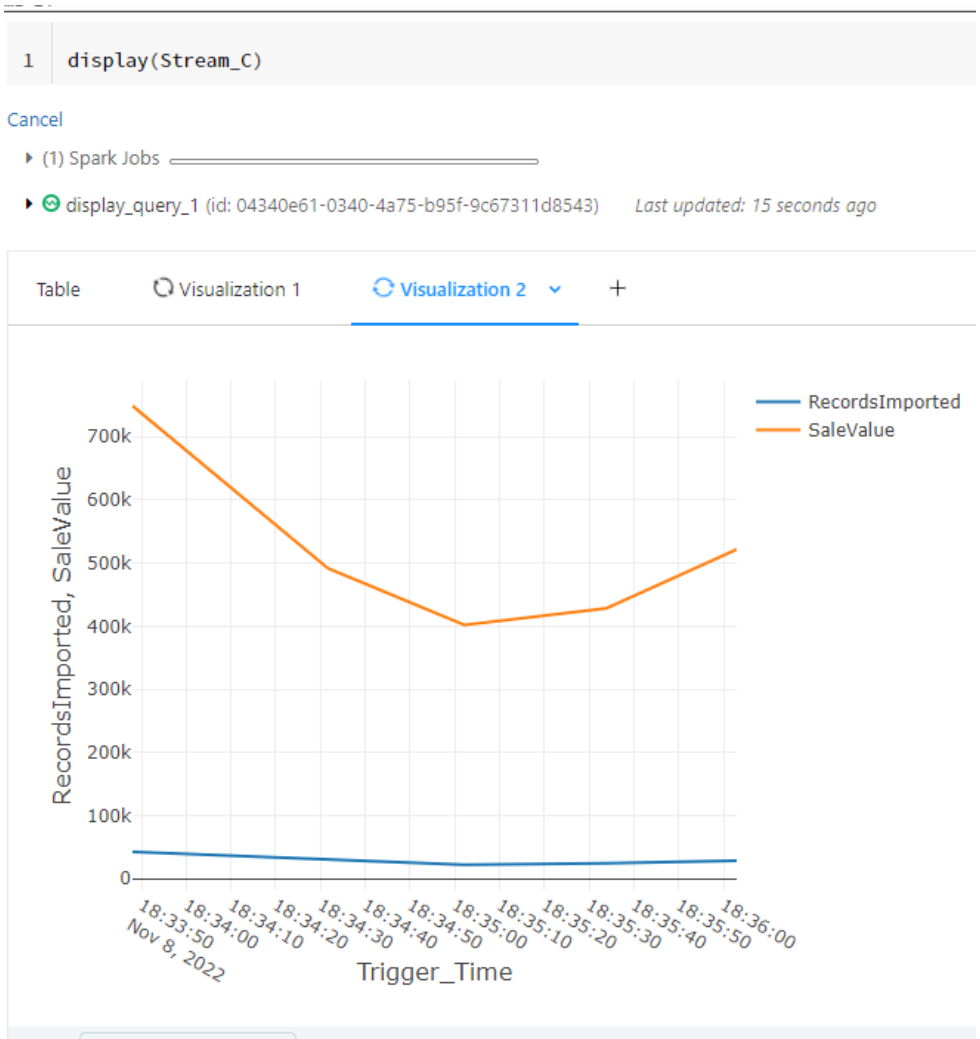
```
for( i <- 1 to 10 ) {
    spark.sql("SELECT * FROM stream_c").show()
    Thread.sleep(2000)
}
```

(18) Spark Jobs

```
+--------------------+---------------+-----------------+
|        Trigger_Time|RecordsImported|        SaleValue|
+--------------------+---------------+-----------------+
|2022-11-08 20:46:...|          42501|748957.0200000004|
+--------------------+---------------+-----------------+

+--------------------+---------------+-----------------+
|        Trigger_Time|RecordsImported|        SaleValue|
+--------------------+---------------+-----------------+
|2022-11-08 20:46:...|          42501|748957.0200000004|
+--------------------+---------------+-----------------+

+--------------------+---------------+-----------------+
|        Trigger_Time|RecordsImported|        SaleValue|
+--------------------+---------------+-----------------+
|2022-11-08 20:46:...|          42501|748957.0200000004|
|2022-11-08 20:47:...|          30353|        491519.26|
+--------------------+---------------+-----------------+

+--------------------+---------------+-----------------+
|        Trigger_Time|RecordsImported|        SaleValue|
```

Command took 3.11 minutes -- by ragababdallah1589@gmail.com at 11/8/2022, 10:46:58 PM on Text corpus

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



e) Upload your notebook as part of your submission. Please do not upload the data on Brightspace.

Note: Sale value = Unit Price * Quantity