GDDA707 – Advanced Data Engineering

16 August 2024

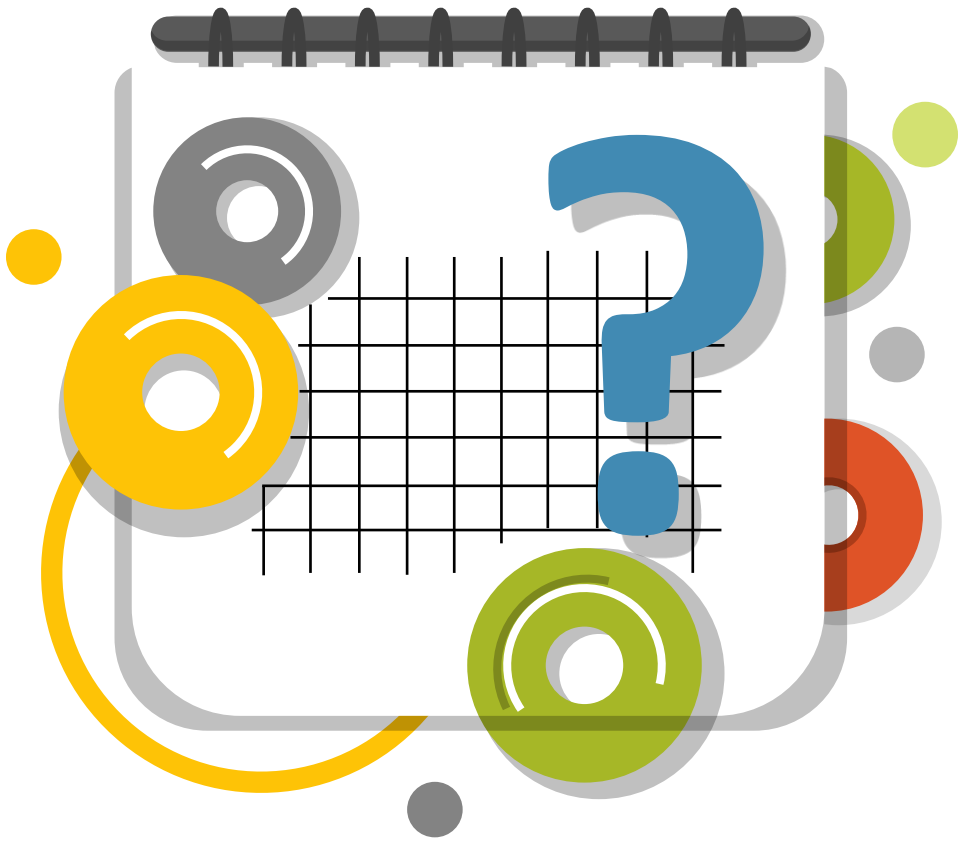


Table of Contents

1. Introduction

Overview

This assessment is designed to highlight students' competencies in performing Extract, Transform, Load (ETL) operations, integrating data for analysis, and applying advanced data engineering techniques on big data platforms. Students will utilize any cloud platform to handle and process large datasets, demonstrating their ability to manage and analyze big data effectively.

Participants will be evaluated on their proficiency in:

ETL Operations: Executing efficient ETL processes to extract data from various sources, transform it into a suitable format, and load it into a target system for analysis.

Data Integration: Combining data from different sources to create a unified dataset, ensuring data consistency, accuracy, and completeness.

Data Engineering on Big Data Platforms: Implementing data engineering techniques to manage, process, and analyze large volumes of data using cloud-based big data platforms.

The assessment will culminate in the creation of an e-portfolio and a comprehensive final report, where students will document their processes, methodologies, and findings. This format will allow students to showcase their practical skills and understanding of data engineering principles in a professional and organized manner.

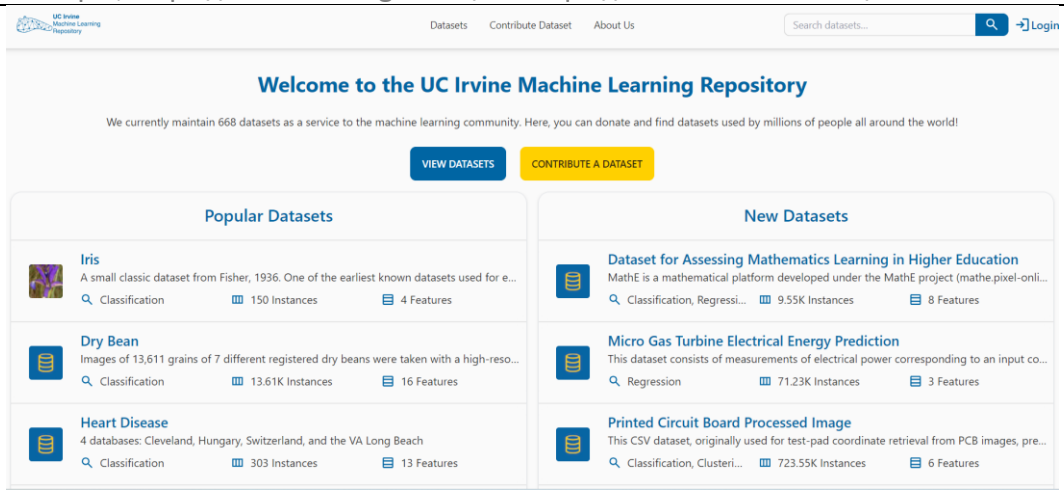
Purpose

The purpose of these assessments is to have students perform ETL operations to integrate data for analysis and apply data engineering techniques on big data platforms.

Part A: ETL Operations

Task 1: Selection of Dataset and Tolls

Choose or develop your own scenario and dataset (minimum two datasets) in your selected domain. Clarify and describe your chosen scenario and database. As an example, https://www.stats.govt.nz/ or https://archive.ics.uci.edu/



I found my data sets in this option https://archive.ics.uci.edu/

My data sets:

Customer.csv
Market.csv
Product.csv

Select appropriate tools or platforms for ETL operations and data engineering on big data. Justify your selection of these tools/platforms.

I decided to get Python with pandas I know I can get Apache Spark but I don't have a big data set because I cannot work too big dados on my computer, but I tried to get a data set in a scenario with more fields.

Task 2: Load and Pre-processing

Load data from selected datasets to process it for next step processing. Provide a clear and detailed explanation of the undertaken steps.

```
# # Read customer.csv, market.csv and products.csv
customer_df = pd.read_csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering
market_df = pd.read_csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\
product_df =  pd.read_csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering
```

```
.csv
esktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Assessment_2\\customer.csv")
ktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Assessment_2\\market.csv")
esktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Assessment_2\\product.csv")
```

Data sets loading customer, market, and product.

```
customer_df
```

	Customer_Name	Province	Region	Customer_Segment	Cust_id
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_1
1	BARRY FRENCH	NUNAVUT	NUNAVUT	CONSUMER	Cust_2
2	CLAY ROZENDAL	NUNAVUT	NUNAVUT	CORPORATE	Cust_3
3	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	Cust_4
4	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_5
...	...	...	...	...	...
1827	NICOLE BRENNAN	ALBERTA	WEST	CONSUMER	Cust_1828
1828	JASON FORTUNE	ALBERTA	WEST	CORPORATE	Cust_1829
1829	HARRY GREENE	ALBERTA	WEST	CORPORATE	Cust_1830
1830	GRANT DONATELLI	ALBERTA	WEST	CONSUMER	Cust_1831
1831	MICK BROWN	ALBERTA	WEST	CONSUMER	Cust_1832

First data set: **Customer.csv**

```
market_df
```

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51	3.60	0.56
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56	0.93	0.54
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90	2.50	0.59
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34	14.30	0.37
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87	26.30	0.38
...	...	...	...	...	...	...	...	...	...	...
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	7.69	0.59
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	6.92	0.37
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	5.35	0.40
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	30.00	0.62
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	4.86	0.38

8399 rows × 10 columns

Second data set: **Market.csv**

```
product_df
```

	Product_Category	Product_Sub_Category	Prod_id
0	OFFICE SUPPLIES	STORAGE & ORGANIZATION	Prod_1
1	OFFICE SUPPLIES	APPLIANCES	Prod_2
2	OFFICE SUPPLIES	BINDERS AND BINDER ACCESSORIES	Prod_3
3	TECHNOLOGY	TELEPHONES AND COMMUNICATION	Prod_4
4	FURNITURE	OFFICE FURNISHINGS	Prod_5
5	OFFICE SUPPLIES	PAPER	Prod_6
6	OFFICE SUPPLIES	RUBBER BANDS	Prod_7
7	TECHNOLOGY	COMPUTER PERIPHERALS	Prod_8
8	OFFICE SUPPLIES	ENVELOPES	Prod_9
9	FURNITURE	BOOKCASES	Prod_10
10	FURNITURE	TABLES	Prod_11
11	OFFICE SUPPLIES	LABELS	Prod_12
12	OFFICE SUPPLIES	PENS & ART SUPPLIES	Prod_13
13	TECHNOLOGY	COPIERS AND FAX	Prod_14
14	FURNITURE	CHAIRS & CHAIRMATS	Prod_15
15	OFFICE SUPPLIES	SCISSORS, RULERS AND TRIMMERS	Prod_16
16	TECHNOLOGY	OFFICE MACHINES	Prod_17

Third data set: **Product.csv**

```
merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8399 entries, 0 to 8398
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Customer_Name         8399 non-null   object
 1   Province              8399 non-null   object
 2   Region                8399 non-null   object
 3   Customer_Segment      8399 non-null   object
 4   Sales                 8399 non-null   float64
 5   Discount              8399 non-null   float64
 6   Order_Quantity        8399 non-null   int64
 7   Profit                8399 non-null   float64
 8   Shipping_Cost         8399 non-null   float64
 9   Product_Base_Margin   8336 non-null   float64
 10  Product_Category      8399 non-null   object
 11  Product_Sub_Category  8399 non-null   object
dtypes: float64(5), int64(1), object(6)
memory usage: 787.5+ KB
```

Information about data sets all columns, types, and ranges.

```
# Display summary statistics of the dataset
print("\nSummary Statistics:")
print(merge_df.describe())
```

```
Summary Statistics:
              Sales     Discount  Order_Quantity        Profit  Shipping_Cost  \
count   8399.000000  8399.000000     8399.000000   8399.000000    8399.000000
mean    1775.878179     0.049671       25.571735    181.184424      12.838557
std     3585.050525     0.031823       14.481071   1196.653371      17.264052
min        2.240000     0.000000        1.000000 -14140.700000       0.490000
25%      143.195000     0.020000       13.000000    -83.315000       3.300000
50%      449.420000     0.050000       26.000000     -1.500000       6.070000
75%     1709.320000     0.080000       38.000000    162.750000      13.990000
max    89061.050000     0.250000       50.000000  27220.690000     164.730000

       Product_Base_Margin
count          8336.000000
mean              0.512513
std               0.135589
min               0.350000
25%               0.380000
50%               0.520000
75%               0.590000
max               0.850000
```

Summary statistics about all columns we can get more information count, mean, std, min, max etc.

Cleanse, format, and transform data as necessary for further operations. Provide a clear and detailed explanation of the undertaken steps.

```
merge_df.drop(columns= ['Cust_id', 'Ord_id', 'Prod_id', 'Ship_id'], inplace = True)
merge_df
```

	Customer_Name	Province	Region	Customer_Segment	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin	Pr
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	261.54	0.04	6	-213.25	35.00	0.80	
1	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	370.48	0.04	28	-5.45	4.51	0.59	
2	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	905.08	0.09	22	127.70	6.22	NaN	
3	MONICA FEDERLE	NUNAVUT	NUNAVUT	CORPORATE	2781.82	0.07	21	-695.26	35.00	NaN	
4	NICOLE HANSEN	NUNAVUT	NUNAVUT	SMALL BUSINESS	3338.98	0.00	40	-846.73	35.00	0.81	
...	...	...	...	...	...	...	...	...	...	...	
8394	SHUI TOM	ALBERTA	WEST	HOME OFFICE	5793.46	0.01	38	2477.77	13.99	0.38	
8395	FRANK HAWLEY	ALBERTA	WEST	HOME OFFICE	12007.05	0.09	16	2713.95	55.30	0.40	
8396	AARON BERGMAN	ALBERTA	WEST	CORPORATE	4233.15	0.08	35	1219.87	26.30	0.38	
8397	ADRIAN SHAMI	ALBERTA	WEST	CONSUMER	754.92	0.00	7	-129.57	19.99	0.52	
8398	HARRY GREENE	ALBERTA	WEST	CORPORATE	10071.09	0.10	41	1977.69	17.86	0.58	

Profit	Shipping_Cost	Product_Base_Margin	Product_Category	Product_Sub_Category
-213.25	35.00	0.80	OFFICE SUPPLIES	STORAGE & ORGANIZATION
-5.45	4.51	0.59	OFFICE SUPPLIES	STORAGE & ORGANIZATION
127.70	6.22	NaN	OFFICE SUPPLIES	STORAGE & ORGANIZATION
-695.26	35.00	NaN	OFFICE SUPPLIES	STORAGE & ORGANIZATION
-846.73	35.00	0.81	OFFICE SUPPLIES	STORAGE & ORGANIZATION
...	...	...	...	...
2477.77	13.99	0.38	TECHNOLOGY	OFFICE MACHINES
2713.95	55.30	0.40	TECHNOLOGY	OFFICE MACHINES
1219.87	26.30	0.38	TECHNOLOGY	OFFICE MACHINES
-129.57	19.99	0.52	TECHNOLOGY	OFFICE MACHINES
1977.69	17.86	0.58	TECHNOLOGY	OFFICE MACHINES

Dropped columns are Cust_id', 'Ord_id', 'Prod_id', and 'Ship_id.

```
# Cheking missing values
merge_df.isnull().sum()
```

```
Customer_Name            0
Province                 0
Region                   0
Customer_Segment         0
Sales                    0
Discount                 0
Order_Quantity           0
Profit                   0
Shipping_Cost            0
Product_Base_Margin     63
Product_Category         0
Product_Sub_Category     0
dtype: int64
```

```
3]: # Fill missing salaries with the median product base margin
    merge_df['Product_Base_Margin'].fillna(merge_df['Product_Base_Margin'].median(), inplace=True)
```

Missing values and I got a median about product base margin.

```
# New colunm total sales
merge_df['TotalSales'] = merge_df['Sales'] * merge_df['Order_Quantity'] + merge_df['Shipping_Cost']
print(merge_df['TotalSales'])
```

```
0         1604.24
1        10377.95
2        19917.98
3        58453.22
4       133594.20
          ...
8394    220165.47
8395    192168.10
8396    148186.55
8397      5304.43
8398    412932.55
Name: TotalSales, Length: 8399, dtype: float64
```

Sum = sales + order quantity + shipping cost shows us the total payer per order.

```
# Perform analysis: Province sold by Region
sales_by_province_region = merge_df.groupby(['Province', 'Region', ])['Sales'].sum().reset_index()
print("\nProvince sold by Region:")
print(sales_by_province_region)
```

```
Province sold by Region:
                  Province                 Region         Sales
0                  ALBERTA                   WEST  1.704791e+06
1         BRITISH COLUMBIA                   WEST  1.892758e+06
2                 MANITOBA                 PRARIE  1.372849e+06
3            NEW BRUNSWICK               ATLANTIC  6.842115e+05
4              NEWFOUNDLAND               ATLANTIC  1.029241e+05
5   NORTHWEST TERRITORIES  NORTHWEST TERRITORIES  8.008473e+05
6              NOVA SCOTIA               ATLANTIC  8.177294e+05
7                  NUNAVUT                 NUNAVUT  1.163765e+05
8                  ONTARIO                 ONTARIO  3.063212e+06
9    PRINCE EDWARD ISLAND               ATLANTIC  4.093832e+05
10                 QUEBEC                   QUEBEC  1.510195e+06
11             SASKACHEWAN                 PRARIE  1.464456e+06
12                   YUKON                   YUKON  9.758674e+05
```

Group by province, region sum sales. After the result shows us the total per region and province.

Task 3: ETL Operations and Integration

Perform ETL operations to integrate data from different datasets into a unified dataset. Provide a clear and detailed explanation of the undertaken steps.

```
# Assuming 'Cust_id' is the common key, merge both DataFrames
merge_df = pd.merge(customer_df, market_df, on='Cust_id')

# Assuming 'Prod_id' is the common key, merge another DataFrame
merge_df = pd.merge(merge_df, product_df, on='Prod_id')
```

I made a merged customer, market, and product.

```
2]: # Show data set after merge
    merge_df
```

2]:

	Customer_Name	Province	Region	Customer_Segment	Cust_id	Ord_id	Prod_id	Ship_id	Sales	Discount	Order_Quantity	Profit	Shipp
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_1	Ord_1	Prod_1	SHP_1	261.54	0.04	6	-213.25	
1	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	Cust_4	Ord_20	Prod_1	SHP_27	370.48	0.04	28	-5.45	
2	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_5	Ord_5	Prod_1	SHP_7	905.08	0.09	22	127.70	
3	MONICA FEDERLE	NUNAVUT	NUNAVUT	CORPORATE	Cust_6	Ord_6	Prod_1	SHP_8	2781.82	0.07	21	-695.26	
4	NICOLE HANSEN	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_24	Ord_55	Prod_1	SHP_72	3338.98	0.00	40	-846.73	
...	...	...	...	...	...	...	...	...	...	...	...	...	
8394	SHUI TOM	ALBERTA	WEST	HOME OFFICE	Cust_1816	Ord_5400	Prod_17	SHP_7540	5793.46	0.01	38	2477.77	
8395	FRANK HAWLEY	ALBERTA	WEST	HOME OFFICE	Cust_1817	Ord_5450	Prod_17	SHP_7690	12007.05	0.09	16	2713.95	
8396	AARON BERGMAN	ALBERTA	WEST	CORPORATE	Cust_1818	Ord_5485	Prod_17	SHP_7664	4233.15	0.08	35	1219.87	
8397	ADRIAN SHAMI	ALBERTA	WEST	CONSUMER	Cust_1820	Ord_5489	Prod_17	SHP_7671	754.92	0.00	7	-129.57	
8398	HARRY GREENE	ALBERTA	WEST	CORPORATE	Cust_1830	Ord_5504	Prod_17	SHP_7696	10071.09	0.10	41	1977.69	

8399 rows × 16 columns

Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin	Product_Category	Product_Sub_Category
0.04	6	-213.25	35.00	0.80	OFFICE SUPPLIES	STORAGE & ORGANIZATION
0.04	28	-5.45	4.51	0.59	OFFICE SUPPLIES	STORAGE & ORGANIZATION
0.09	22	127.70	6.22	NaN	OFFICE SUPPLIES	STORAGE & ORGANIZATION
0.07	21	-695.26	35.00	NaN	OFFICE SUPPLIES	STORAGE & ORGANIZATION
0.00	40	-846.73	35.00	0.81	OFFICE SUPPLIES	STORAGE & ORGANIZATION
...	...	...	...	...	...	...
0.01	38	2477.77	13.99	0.38	TECHNOLOGY	OFFICE MACHINES
0.09	16	2713.95	55.30	0.40	TECHNOLOGY	OFFICE MACHINES
0.08	35	1219.87	26.30	0.38	TECHNOLOGY	OFFICE MACHINES
0.00	7	-129.57	19.99	0.52	TECHNOLOGY	OFFICE MACHINES
0.10	41	1977.69	17.86	0.58	TECHNOLOGY	OFFICE MACHINES

Result after merging my dataset.

Use chosen tools/platforms to effectively manage and process large volumes of data.

We had some tools in our assessment **Hadoop** together HDFS (Hadoop Distributed File System) for distributed data storage and processing, **Apache Spark** good for the process of quick information, and **Apache Kafka** for real-time data ingestion and transmission example in my task I took to get information in Facebook. I'm writing more details in Project Report Requirements.

Document challenges faced during ETL and integration, along with solutions implemented.

I have a lot of challenges in this assessment. First, in my country, ETL is a tool after I checked with my Lecturer Dra Sara, I could get this process and I can work with pandas. I got some error in my computer configuration.

```
Py4JJavaError: An error occurred while calling None.org.apache.spark.api.java.JavaSparkContext.
: org.apache.spark.SparkException: Invalid Spark URL: spark://HeartbeatReceiver@Pati_Rangel:59983
        at org.apache.spark.rpc.RpcEndpointAddress$.apply(RpcEndpointAddress.scala:66)
        at org.apache.spark.rpc.netty.NettyRpcEnv.asyncSetupEndpointRefByURI(NettyRpcEnv.scala:140)
        at org.apache.spark.rpc.RpcEnv.setupEndpointRefByURI(RpcEnv.scala:102)
        at org.apache.spark.rpc.RpcEnv.setupEndpointRef(RpcEnv.scala:110)
        at org.apache.spark.util.RpcUtils$.makeDriverRef(RpcUtils.scala:36)
        at org.apache.spark.executor.Executor.<init>(Executor.scala:301)
        at org.apache.spark.scheduler.local.LocalEndpoint.<init>(LocalSchedulerBackend.scala:64)
        at org.apache.spark.scheduler.local.LocalSchedulerBackend.start(LocalSchedulerBackend.scala:132)
        at org.apache.spark.scheduler.TaskSchedulerImpl.start(TaskSchedulerImpl.scala:235)
        at org.apache.spark.SparkContext.<init>(SparkContext.scala:604)
        at org.apache.spark.api.java.JavaSparkContext.<init>(JavaSparkContext.scala:58)
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:45)
```

Problems when I tried to start the spark session in Python.

```
# Read customer.csv
customer_df = spark.read.csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Assessment_2\\
```

```
Py4JJavaError                             Traceback (most recent call last)
Cell In[7], line 2
      1 # Read customer.csv
----> 2 customer_df = spark.read.csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Asse
ssment_2\\customer.csv",header=True, inferSchema=True)

File ~\anaconda3\Lib\site-packages\pyspark\sql\session.py:1706, in SparkSession.read(self)
   1669 @property
   1670 def read(self) -> DataFrameReader:
   1671     """
   1672     Returns a :class:`DataFrameReader` that can be used to read data
   1673     in as a :class:`DataFrame`.
   (...)
   1704     +----+----------+
   1705     """
-> 1706     return DataFrameReader(self)

File ~\anaconda3\Lib\site-packages\pyspark\sql\readwriter.py:70, in DataFrameReader.__init__(self, spark)
     69 def __init__(self, spark: "SparkSession"):
---> 70     self._jreader = spark._jsparkSession.read()
     71     self._spark = spark

File ~\anaconda3\Lib\site-packages\py4j\java_gateway.py:1322, in JavaMember.__call__(self, *args)
   1316 command = proto.CALL_COMMAND_NAME +\
   1317     self.command_header +\
   1318     args_command +\
   1319     proto.END_COMMAND_PART
   1321 answer = self.gateway_client.send_command(command)
-> 1322 return_value = get_return_value(
   1323     answer, self.gateway_client, self.target_id, self.name)
   1325 for temp_arg in temp_args:
   1326     if hasattr(temp_arg, "_detach"):

File ~\anaconda3\Lib\site-packages\pyspark\errors\exceptions\captured.py:179, in capture_sql_exception.<locals>.deco(*a, **kw)
```

Loading CSV with spark I got this answer.

I had other problems on my computer with sections Part A – Task 1, and Part B – Task 3 and my docker one container didn't work well after checking in the internet forum and with my colleagues. I fixed the problem, and I could do my tasks. If I catch all problems I will have one long list because I have a problems in Apache Spark, Docker Hadoop, Kafka, and Facebook. I forgot to get print about all the trouble I need to be calm and keep going.

Part B: Big Data Analysis and Application of Engineering Techniques

Task 1

Develop a data ingestion pipeline to load customer transaction data from various sources into HDFS, ensuring fault tolerance and data integrity. Describe the process and provide the required evidence (screenshot, diagram, etc.).

```
C:\Users\patir\docker-hadoop>docker exec -it namenode /bin/bash
root@24e12b5bdd4a:/# hdfs dfs -mkdir/transaction
-mkdir/transaction: Unknown command
root@24e12b5bdd4a:/# hdfs dfs -mkdir /transaction
root@24e12b5bdd4a:/#
```

I made this folder transaction in hdfs in docker-Hadoop.

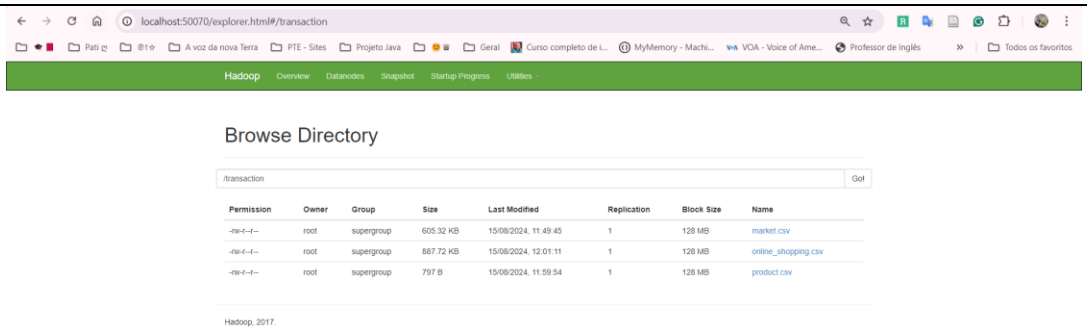
```
PS C:\Users\patir\docker-hadoop> ls


    Directory: C:\Users\patir\docker-hadoop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/08/2024     11:25           2322 docker-compose.yml
-a----        14/08/2024     18:40            407 docker-compose_spark.yml
-a----        09/07/2024     20:07         619850 market.csv
-a----        31/07/2024     18:06         909024 online_shopping.csv
-a----        09/07/2024     20:07            797 product.csv


PS C:\Users\patir\docker-hadoop> # Copie o arquivo para o contêiner
PS C:\Users\patir\docker-hadoop> docker cp C:/Users/patir/docker-hadoop/market.csv namenode:/tmp/market.csv
Successfully copied 622kB to namenode:/tmp/market.csv
PS C:\Users\patir\docker-hadoop>
```

Copy all my CSV's (market, product, and online shopping) to the main direct Docker, and then I send a copy to tmp inside the container.

docker cp C:/Users/patir/docker-hadoop/market.csv namenode:/tmp/market.csv. After I made a copy inside the directory you can check below.

```
root@5b57735ac4de:/# ls -l /tmp
total 616
drwxr-xr-x 3 root root   4096 Aug 14 23:25 Jetty_0_0_0_0_50070_hdfs____w2cu08
drwxr-xr-x 1 root root   4096 Aug 14 23:47 hsperfdata_root
-rwxr-xr-x 1 root root 619850 Jul  9 08:07 market.csv
root@5b57735ac4de:/#
```

```
drwxr-xr-x 3 root root   4096 Aug 14 23:25 Jetty_0_0_0_0_50070_hdfs____w2cu08
drwxr-xr-x 1 root root   4096 Aug 14 23:47 hsperfdata_root
-rwxr-xr-x 1 root root 619850 Jul  9 08:07 market.csv
root@5b57735ac4de:/# hdfs dfs -mkdir -p /transaction
root@5b57735ac4de:/# hdfs dfs -put /tmp/market.csv /transaction/
root@5b57735ac4de:/#
```



Check all files after copying in the container docker-Hadoop.

Task 2

Implement a data storage and querying solution for any application using any one of the data storages or a combination of Hive, Cassandra, and MongoDB. Discuss the rationale behind selecting the database for different types of data.

```
df = pd.read_csv("C:\\Users\\patir\\project_jupyter\\online_retail_II.csv")
```

```
df.head()
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

Loading data set online retail II and show columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067371 entries, 0 to 1067370
Data columns (total 10 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   _id          1067371 non-null  object
 1   index        1067371 non-null  int64
 2   Invoice      1067371 non-null  object
 3   StockCode    1067371 non-null  object
 4   Description  1062989 non-null  object
 5   Quantity     1067371 non-null  int64
 6   InvoiceDate  1067371 non-null  object
 7   Price        1067371 non-null  float64
 8   Customer ID  824364 non-null   float64
 9   Country      1067371 non-null  object
dtypes: float64(2), int64(2), object(6)
memory usage: 81.4+ MB
```

Information about columns, types, and ranges.

```
#Connect to MongoDB

#url = "mongodb+srv://patirangelsantos:aUdT7Jxk%40yUx2hu@cluster0.pnlxrbn.mongodb.net/"
url = "mongodb://localhost:27017/"

#client = MongoClient("mongodb+srv://patirangelsantos:aUdT7Jxk%40yUx2hu@cluster0.pnlxrbn.mongodb.net/")
client = MongoClient("mongodb://localhost:27017/")

try:
    # Create MongoClien objective
    db = client['gdda707_MongoDB']
    # Trying to connect to the server
    client.admin.command('ping')
    # Access collection
    collection = db['gdda707_MongoDB_collection_v3']
    print("successfu lconnection")
except ConnectionFailure:
    print("connection fail")
```

```
successfu lconnection
```

Connection Mongo DB and show us a 'successful connection'.

```
collection
```

```
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'gdda707_MongoD
B'), 'gdda707_MongoDB_collection_v3')
```

Print collection I could get database information.

```
df.reset_index(inplace=True)
df_dict = df.to_dict('records')
df_dict
```

```
[{'index': 0,
  'Invoice': '489434',
  'StockCode': '85048',
  'Description': '15CM CHRISTMAS GLASS BALL 20 LIGHTS',
  'Quantity': 12,
  'InvoiceDate': '2009-12-01 07:45:00',
  'Price': 6.95,
  'Customer ID': 13085.0,
  'Country': 'United Kingdom'},
 {'index': 1,
  'Invoice': '489434',
  'StockCode': '79323P',
  'Description': 'PINK CHERRY LIGHTS',
  'Quantity': 12,
  'InvoiceDate': '2009-12-01 07:45:00',
  'Price': 6.75,
  'Customer ID': 13085.0,
  'Country': 'United Kingdom'},
 {'index': 2,
```

Print on register ready to save.

```
collection.insert_many(df_dict)
```

InsertManyResult([ObjectId('66b96282b3de891b5fa79418'), ObjectId('66b96282b3de891b5fa79419'), ObjectId('66b96282b3de891b5fa7941a'), ObjectId('66b96282b3de891b5fa7941b'), ObjectId('66b96282b3de891b5fa7941c'), ObjectId('66b96282b3de891b5fa7941d'), ObjectId('66b96282b3de891b5fa7941e'), ObjectId('66b96282b3de891b5fa7941f'), ObjectId('66b96282b3de891b5fa79420'), ObjectId('66b96282b3de891b5fa79421'), ObjectId('66b96282b3de891b5fa79422'), ObjectId('66b96282b3de891b5fa79423'), ObjectId('66b96282b3de891b5fa79424'), ObjectId('66b96282b3de891b5fa79425'), ObjectId('66b96282b3de891b5fa79426'), ObjectId('66b96282b3de891b5fa79427'), ObjectId('66b96282b3de891b5fa79428'), ObjectId('66b96282b3de891b5fa79429'), ObjectId('66b96282b3de891b5fa7942a'), ObjectId('66b96282b3de891b5fa7942b'), ObjectId('66b96282b3de891b5fa7942c'), ObjectId('66b96282b3de891b5fa7942d'), ObjectId('66b96282b3de891b5fa7942e'), ObjectId('66b96282b3de891b5fa7942f'), ObjectId('66b96282b3de891b5fa79430'), ObjectId('66b96282b3de891b5fa79431'), ObjectId('66b96282b3de891b5fa79432'), ObjectId('66b96282b3de891b5fa79433'), ObjectId('66b96282b3de891b5fa79434'), ObjectId('66b96282b3de891b5fa79435'), ObjectId('66b96282b3de891b5fa79436'), ObjectId('66b96282b3de891b5fa79437'), ObjectId('66b96282b3de891b5fa79438'), ObjectId('66b96282b3de891b5fa79439'), ObjectId('66b96282b3de891b5fa7943a'), ObjectId('66b96282b3de891b5fa7943b'), ObjectId('66b96282b3de891b5fa7943c'), ObjectId('66b96282b3de891b5fa7943d'), ObjectId('66b96282b3de891b5fa7943e'), ObjectId('66b96282b3de891b5fa7943f'), ObjectId('66b96282b3de891b5fa79440'), ObjectId('66b96282b3de891b5fa79441'), ObjectId('66b96282b3de891b5fa79442'), ObjectId('66b96282b3de891b5fa79443'), ObjectId('66b96282b3de891b5fa79444'), ObjectId('66b96282b3de891b5fa79445'), ObjectId('66b96282b3de891b5fa79446'), ObjectId('66b96282b3de891b5fa79447'), ObjectId('66b96282b3de891b5fa79448'), ObjectId('66b96282b3de891b5fa79449'), ObjectId('66b96282b3de891b5fa7944a'), ObjectId('66b96282b3de891b5fa7944b'), ObjectId('66b96282b3de891b5fa7944c'), ObjectId('66b96282b3de891b5fa7944d'), ObjectId('66b96282b3de891b5fa7944e'), ObjectId('66b96282b3de891b5fa7944f'), ObjectId('66b96282b3de891b5fa79450'), ObjectId('66b96282b3de891b5fa79451'), ObjectId('66b96282b3de891b5fa79452'), ObjectId('66b96282b3de891b5fa79453'), ObjectId('66b96282b3de891b5fa79454'), ObjectId('66b96282b3de891b5fa79455'), O

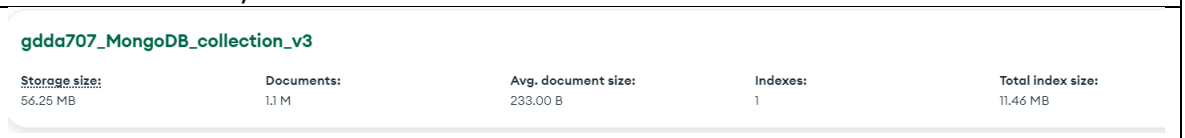
I can get all object Id after saving my collection in Mongo DB.

```
df =  pd.DataFrame(list(collection.find()))
df.head(n =5)
```

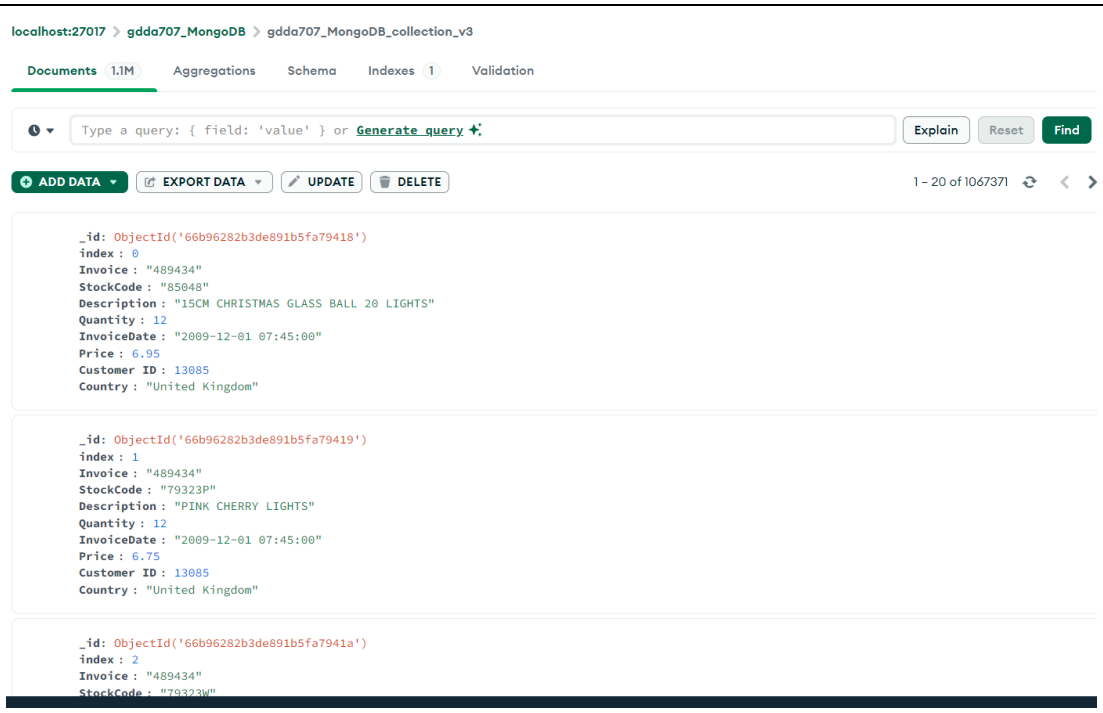
	_id	index	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	66b96282b3de891b5fa79418	0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	66b96282b3de891b5fa79419	1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	66b96282b3de891b5fa7941a	2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	66b96282b3de891b5fa7941b	3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	66b96282b3de891b5fa7941c	4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

```
client.close()
```

I found the list in my collection that shows in the data frame.



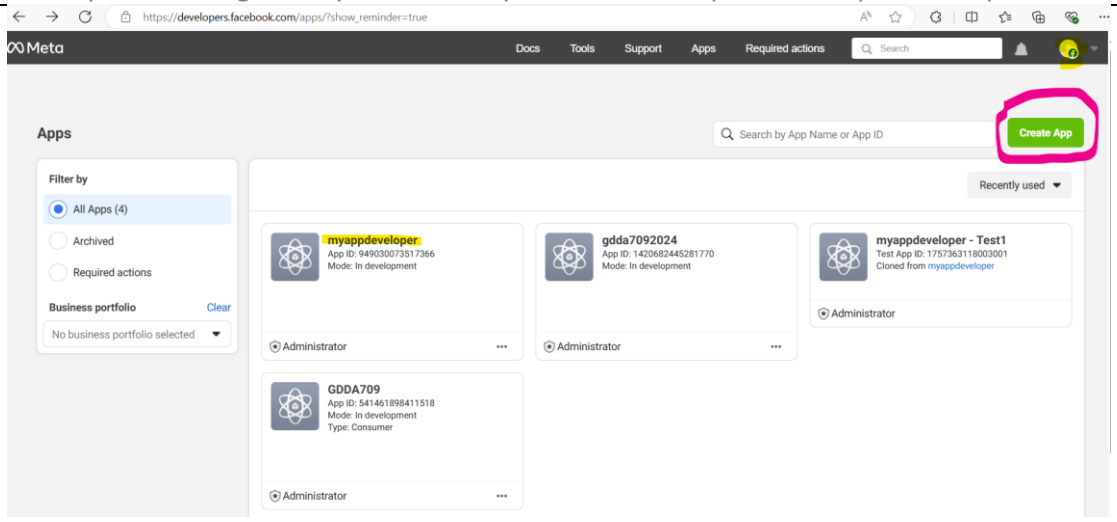
Database in Mongo DB after creating database gdda707_MongoDB and collection gdda707_MongoDb_Collection_V3.



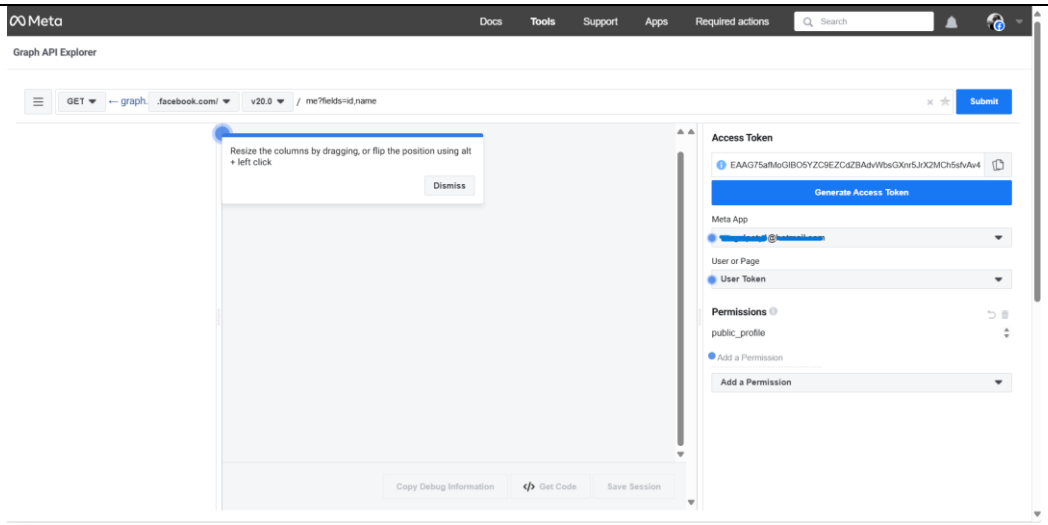
I found Object ID in my Mongo DB desktop after executing jupyter.

Task 3

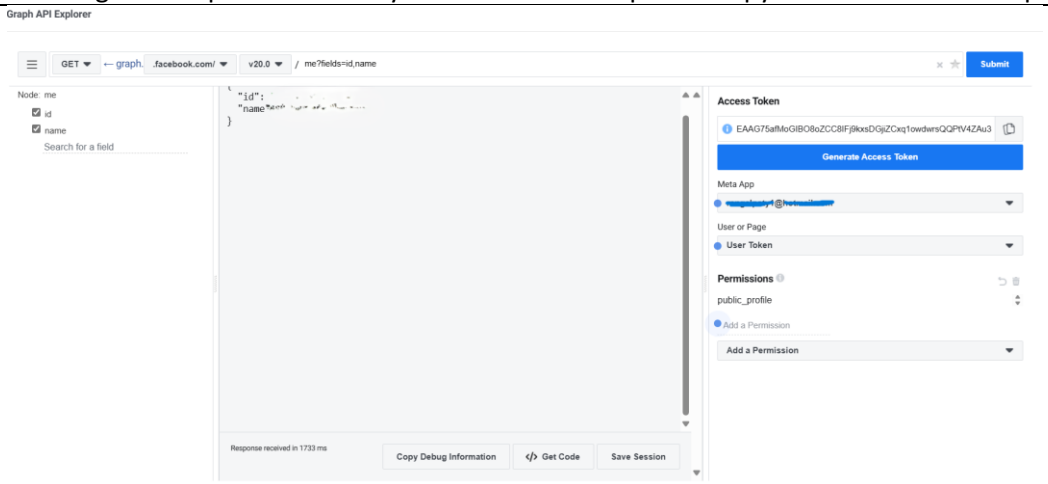
Build a real-time data clustering system using Apache Spark to stream and integrate data from social media platforms to get ready for data analysis. Describe the process and provide required evidence.



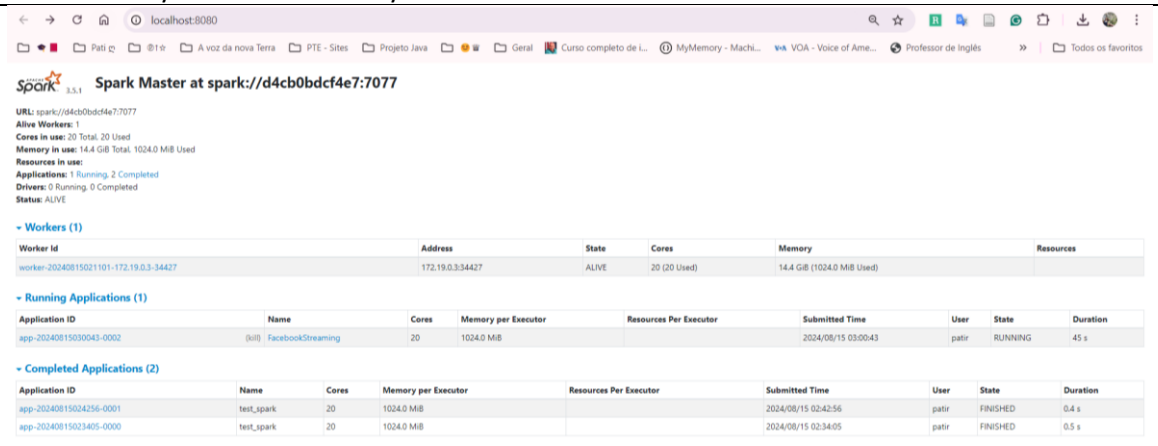
Login with facebook in this app



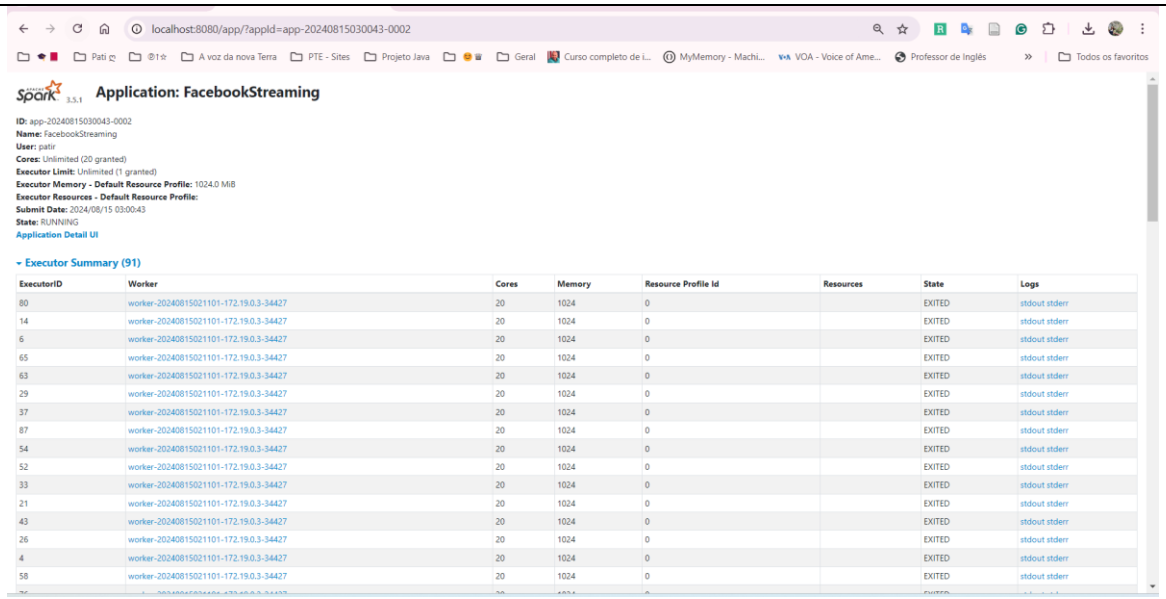
After login API Explorer create my access token and I put it in a python file to connect Apache Spark.



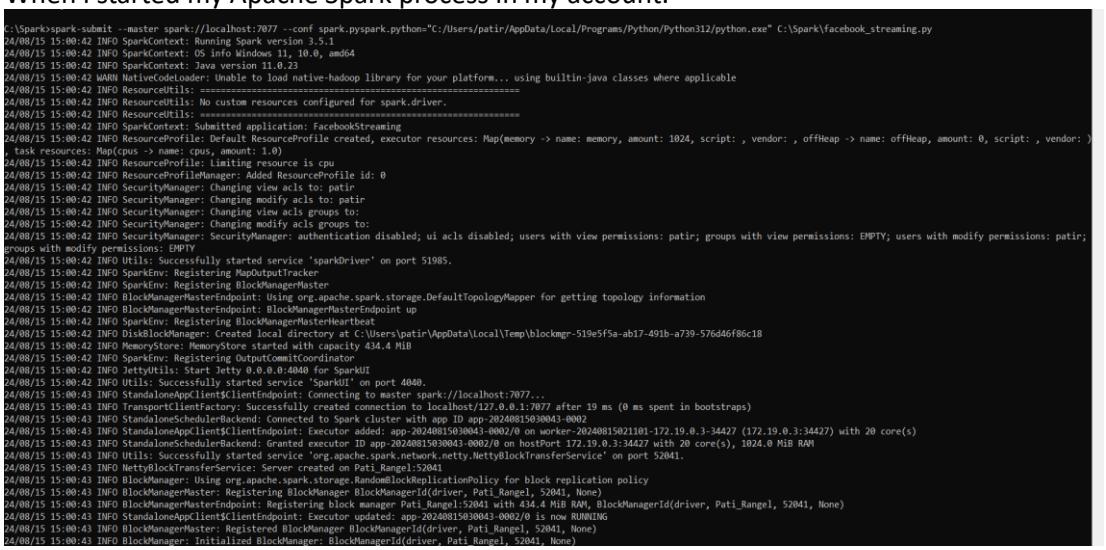
Shows me my ID and name in my account on Facebook.



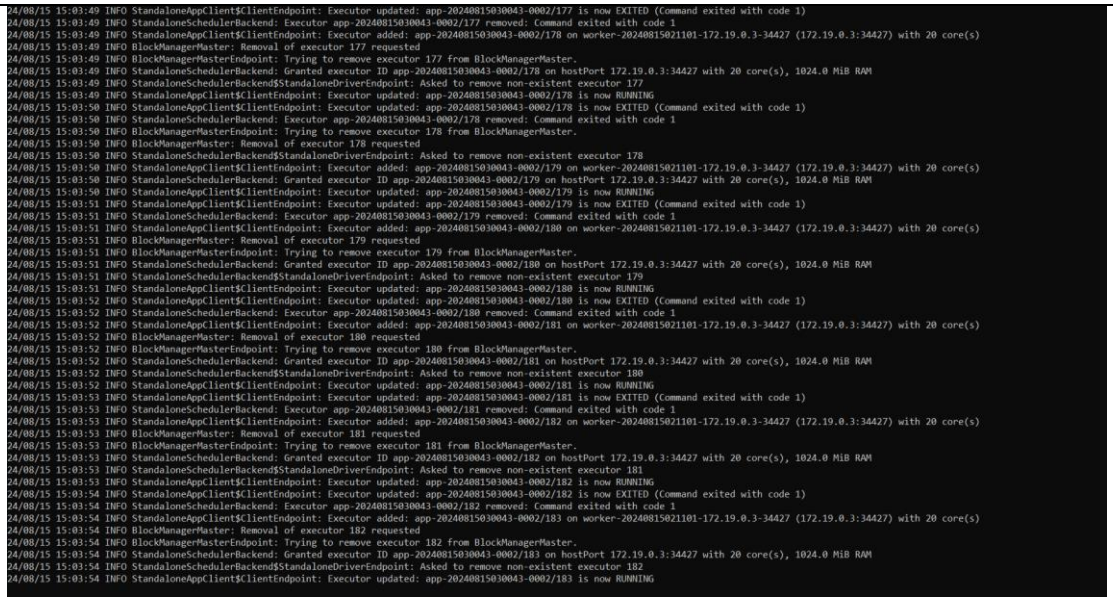
In docker create container Apache Spark and start localhost with configuration master and worker.



When I started my Apache Spark process in my account.



Start Apache Spark in my computer.



Running Apache Spark.

Task 4

Set up a data stream pipeline to integrate data for data analysis. Discuss how Kafka's reliability features ensure data consistency and fault tolerance in real-time processing.

```
@echo off
cd C:\kafka
call .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

Zookeeper.properties

```
@echo off
cd C:\kafka
call .\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic kafkaLab --from-beginning
```

Start_consumer.bat

```
@echo off
cd C:\kafka
call .\bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --topic kafkaLab
```

Start_kafka.bat

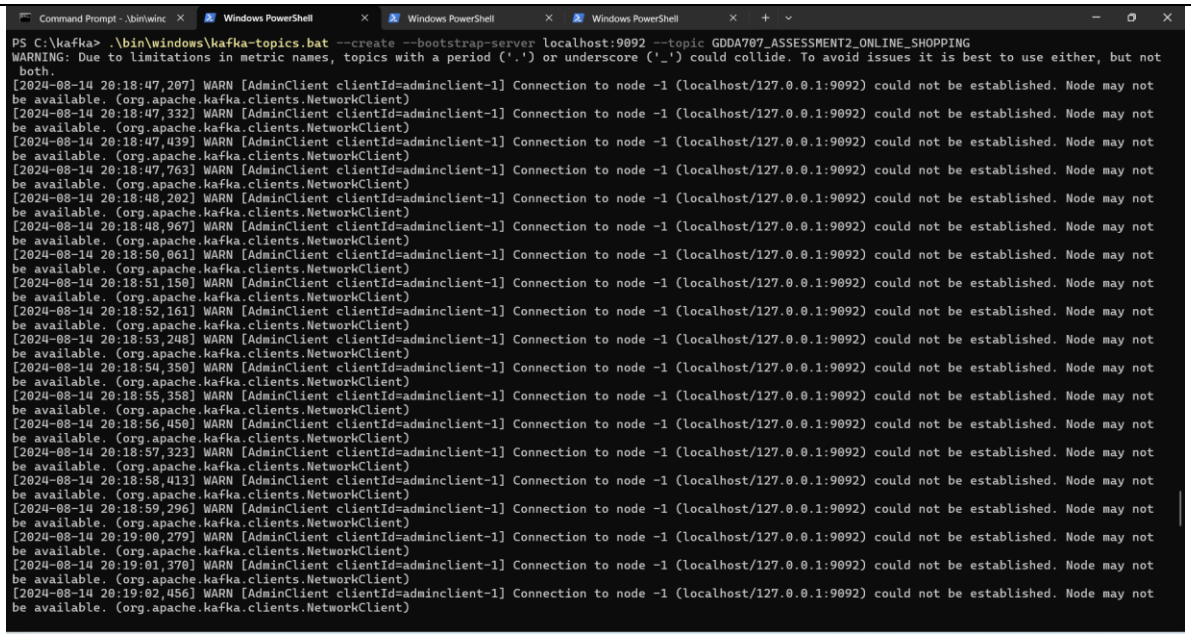
```
[2024-08-14 19:42:55,996] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2024-08-14 19:42:56,013] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMai
n)
[2024-08-14 19:42:56,014] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-08-14 19:42:56,015] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,015] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2024-08-14 19:42:56,016] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-08-14 19:42:56,025] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@2e222612 (org.apache.zookeep
er.server.ServerMetrics)
[2024-08-14 19:42:56,027] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-08-14 19:42:56,027] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-08-14 19:42:56,029] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-08-14 19:42:56,038] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO   ______                  _                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  |___  /                 | |                                         (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO     / /    ___     ___    | |__     ___     ___    _ __    ___   _ __  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO    / /    / _ \   / _ \   | |/ /   / _ \   / _ \  | '_ \  / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO   / /__  | (_) | | (_) |  |   <   |  __/ |  __/  | |_) | |  __/ | |    (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  /_____|  \___/   \___/   |_|\_\   \___|  \___|  | .__/   \___| |_|  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO                                                  | |                    (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO                                                  |_|                    (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apach
e.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:host.name=Pati_Rangel.nzse.com (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.version=11.0.23 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.home=C:\Program Files\Java\jdk-11 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.class.path=C:\kafka\libs\activation-1.1.1.jar;C:\kafka\libs\aopalliance-repackaged-2.6.1.jar;C:\kafka
```

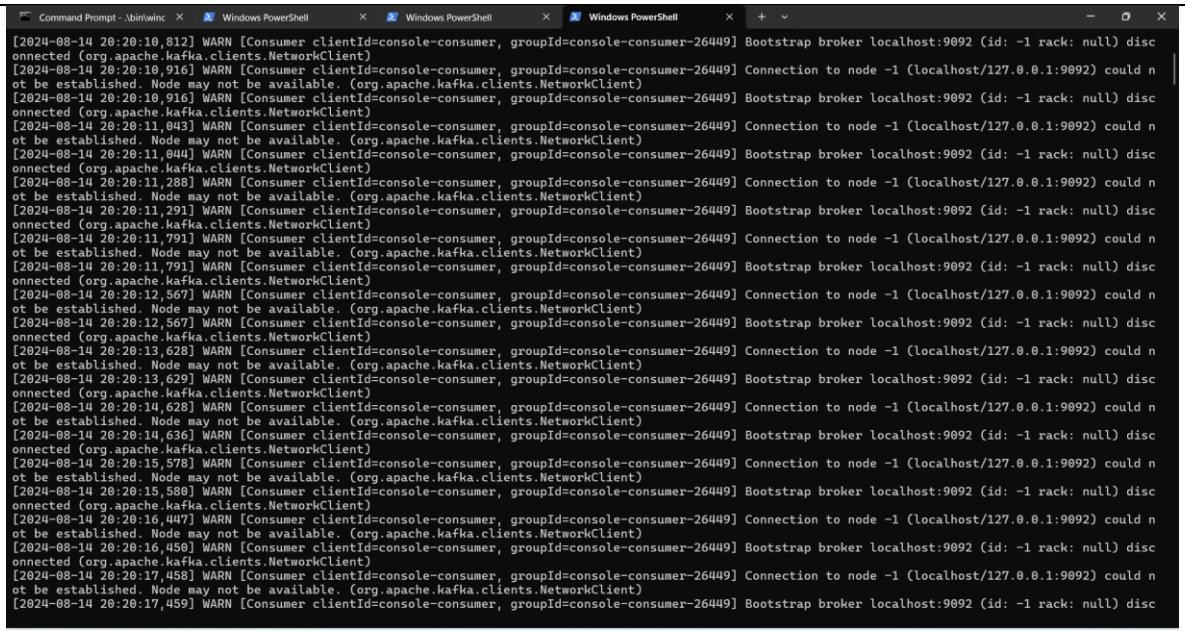
Start kafka.

```
Command Prompt - .\bin\winc   Windows PowerShell   Windows PowerShell   Windows PowerShell

be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:29,698] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:30,679] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:31,767] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:32,743] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:33,729] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:34,815] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:35,685] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:36,666] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:37,541] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:38,535] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:39,418] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:40,519] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:41,609] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:42,704] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:43,793] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:44,883] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:45,752] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
[2024-08-14 20:19:46,841] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not
be available. (org.apache.kafka.clients.NetworkClient)
Error while executing topic command : Timed out waiting for a node assignment. Call: createTopics
[2024-08-14 20:19:47,238] ERROR org.apache.kafka.common.errors.TimeoutException: Timed out waiting for a node assignment. Call: createTopics
 (org.apache.kafka.tools.TopicCommand)
PS C:\kafka> ^P
```

```
2024-08-14 19:42:56,042] INFO zookeeper.maxWriteQueuePollTime = 0 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,042] INFO zookeeper.maxBatchSize=1000 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,042] INFO zookeeper.intBufferStartingSizeBytes = 1024 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,043] INFO Weighed connection throttling is disabled (org.apache.zookeeper.server.BlueThrottle)
2024-08-14 19:42:56,044] INFO minSessionTimeout set to 6000 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,044] INFO maxSessionTimeout set to 60000 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,044] INFO getData response cache size is initialized with value 400. (org.apache.zookeeper.server.ResponseCache)
2024-08-14 19:42:56,044] INFO getChildren response cache size is initialized with value 400. (org.apache.zookeeper.server.ResponseCache)
2024-08-14 19:42:56,050] INFO zookeeper.pathStats.slotCapacity = 60 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,051] INFO zookeeper.pathStats.slotDuration = 15 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,051] INFO zookeeper.pathStats.maxDepth = 6 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,051] INFO zookeeper.pathStats.initialDelay = 5 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,051] INFO zookeeper.pathStats.delay = 5 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,051] INFO zookeeper.pathStats.enabled = false (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
2024-08-14 19:42:56,054] INFO The max bytes for all large requests are set to 104857600 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,054] INFO The large request threshold is set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,054] INFO zookeeper.enforce.auth.enabled = false (org.apache.zookeeper.server.AuthenticationHelper)
2024-08-14 19:42:56,055] INFO zookeeper.enforce.auth.schemes = [] (org.apache.zookeeper.server.AuthenticationHelper)
2024-08-14 19:42:56,055] INFO Created server with tickTime 3000 ms minSessionTimeout 6000 ms maxSessionTimeout 60000 ms clientPortListenBacklog -1 datadir
:\kafka\zooper-data\version-2 snapdir C:\kafka\zooper-data\version-2 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,059] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFa
tory)
2024-08-14 19:42:56,060] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)
2024-08-14 19:42:56,061] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 3 selector thread(s), 40 worker threads, and 64 k
 direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
2024-08-14 19:42:56,068] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
2024-08-14 19:42:56,081] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
2024-08-14 19:42:56,081] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
2024-08-14 19:42:56,082] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
2024-08-14 19:42:56,082] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
2024-08-14 19:42:56,086] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
2024-08-14 19:42:56,095] INFO Reading snapshot C:\kafka\zooper-data\version-2\snapshot.0 (org.apache.zookeeper.server.persistence.FileSnap)
2024-08-14 19:42:56,098] INFO The digest value is empty in snapshot (org.apache.zookeeper.server.DataTree)
2024-08-14 19:42:56,100] INFO Snapshot loaded in 17 ms, highest zxid is 0x0, digest is 1371985504 (org.apache.zookeeper.server.ZKDatabase)
2024-08-14 19:42:56,102] INFO Snapshotting: 0x0 to C:\kafka\zooper-data\version-2\snapshot.0 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
2024-08-14 19:42:56,103] INFO Snapshot taken in 1 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56,112] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.server.RequestThrottler)
2024-08-14 19:42:56,111] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
2024-08-14 19:42:56,122] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
2024-08-14 19:42:56,123] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
```

```
Command Prompt - .\bin\win  ×  Windows PowerShell  ×  Windows PowerShell  ×  Windows PowerShell  ×  +

\libs\argparse4j-0.7.0.jar;C:\kafka\libs\audience-annotations-0.12.0.jar;C:\kafka\libs\caffeine-2.9.3.jar;C:\kafka\libs\checker-qual-3.19.0.jar;C:\kafka\lib
s\commons-beanutils-1.9.4.jar;C:\kafka\libs\commons-cli-1.4.jar;C:\kafka\libs\commons-collections-3.2.2.jar;C:\kafka\libs\commons-digester-2.1.jar;C:\kafka\
libs\commons-io-2.11.0.jar;C:\kafka\libs\commons-lang3-3.8.1.jar;C:\kafka\libs\commons-logging-1.2.jar;C:\kafka\libs\commons-validator-1.7.jar;C:\kafka\libs
\connect-api-3.7.0.jar;C:\kafka\libs\connect-basic-auth-extension-3.7.0.jar;C:\kafka\libs\connect-file-3.7.0.jar;C:\kafka\libs\connect-json-3.7.0.jar;C:\kaf
ka\libs\connect-mirror-3.7.0.jar;C:\kafka\libs\connect-mirror-client-3.7.0.jar;C:\kafka\libs\connect-runtime-3.7.0.jar;C:\kafka\libs\connect-transforms-3.7.
0.jar;C:\kafka\libs\error_prone_annotations-2.10.0.jar;C:\kafka\libs\hk2-api-2.6.1.jar;C:\kafka\libs\hk2-locator-2.6.1.jar;C:\kafka\libs\hk2-utils-2.6.1.jar
;C:\kafka\libs\jackson-annotations-2.16.0.jar;C:\kafka\libs\jackson-core-2.16.0.jar;C:\kafka\libs\jackson-databind-2.16.0.jar;C:\kafka\libs\jackson-dataform
at-csv-2.16.0.jar;C:\kafka\libs\jackson-datatype-jdk8-2.16.0.jar;C:\kafka\libs\jackson-jaxrs-base-2.16.0.jar;C:\kafka\libs\jackson-jaxrs-json-provider-2.16.
0.jar;C:\kafka\libs\jackson-module-jaxb-annotations-2.16.0.jar;C:\kafka\libs\jackson-module-scala_2.13-2.16.0.jar;C:\kafka\libs\jakarta.activation-api-1.2.2
.jar;C:\kafka\libs\jakarta.annotation-api-1.3.5.jar;C:\kafka\libs\jakarta.inject-2.6.1.jar;C:\kafka\libs\jakarta.validation-api-2.0.2.jar;C:\kafka\libs\jaka
rta.ws.rs-api-2.1.6.jar;C:\kafka\libs\jakarta.xml.bind-api-2.3.3.jar;C:\kafka\libs\javassist-3.29.2-GA.jar;C:\kafka\libs\javax.activation-api-1.2.0.jar;C:\k
afka\libs\javax.annotation-api-1.3.2.jar;C:\kafka\libs\javax.servlet-api-3.1.0.jar;C:\kafka\libs\javax.ws.rs-api-2.1.1.jar;C:\kafka\libs\jaxb-api-2.3.1.jar;
C:\kafka\libs\jersey-client-2.39.1.jar;C:\kafka\libs\jersey-common-2.39.1.jar;C:\kafka\libs\jersey-container-servlet-2.39.1.jar;C:\kafka\libs\jersey-contain
er-servlet-core-2.39.1.jar;C:\kafka\libs\jersey-hk2-2.39.1.jar;C:\kafka\libs\jersey-server-2.39.1.jar;C:\kafka\libs\jetty-client-9.4.53.v20231009.jar;C:\kaf
ka\libs\jetty-continuation-9.4.53.v20231009.jar;C:\kafka\libs\jetty-http-9.4.53.v20231009.jar;C:\kafka\libs\jetty-io-9.4.53.v20231009.jar;C:\kafka\libs\jett
y-security-9.4.53.v20231009.jar;C:\kafka\libs\jetty-server-9.4.53.v20231009.jar;C:\kafka\libs\jetty-servlet-9.4.53.v20231009.jar;C:\kafka\libs\jetty-servlet
s-9.4.53.v20231009.jar;C:\kafka\libs\jetty-util-9.4.53.v20231009.jar;C:\kafka\libs\jetty-util-ajax-9.4.53.v20231009.jar;C:\kafka\libs\jline-3.22.0.jar;C:\ka
fka\libs\jopt-simple-5.0.4.jar;C:\kafka\libs\jose4j-0.9.4.jar;C:\kafka\libs\jsr305-3.0.2.jar;C:\kafka\libs\kafka-clients-3.7.0.jar;C:\kafka\libs\kafka-group
-coordinator-3.7.0.jar;C:\kafka\libs\kafka-log4j-appender-3.7.0.jar;C:\kafka\libs\kafka-metadata-3.7.0.jar;C:\kafka\libs\kafka-raft-3.7.0.jar;C:\kafka\libs\
kafka-server-3.7.0.jar;C:\kafka\libs\kafka-server-common-3.7.0.jar;C:\kafka\libs\kafka-shell-3.7.0.jar;C:\kafka\libs\kafka-storage-3.7.0.jar;C:\kafka\libs\k
afka-storage-api-3.7.0.jar;C:\kafka\libs\kafka-streams-3.7.0.jar;C:\kafka\libs\kafka-streams-examples-3.7.0.jar;C:\kafka\libs\kafka-streams-scala_2.13-3.7.0
.jar;C:\kafka\libs\kafka-streams-test-utils-3.7.0.jar;C:\kafka\libs\kafka-tools-3.7.0.jar;C:\kafka\libs\kafka-tools-api-3.7.0.jar;C:\kafka\libs\kafka_2.13-3
.7.0.jar;C:\kafka\libs\lz4-java-1.8.0.jar;C:\kafka\libs\maven-artifact-3.8.8.jar;C:\kafka\libs\metrics-core-2.2.0.jar;C:\kafka\libs\metrics-core-4.1.12.1.ja
r;C:\kafka\libs\netty-buffer-4.1.100.Final.jar;C:\kafka\libs\netty-codec-4.1.100.Final.jar;C:\kafka\libs\netty-common-4.1.100.Final.jar;C:\kafka\libs\netty-
handler-4.1.100.Final.jar;C:\kafka\libs\netty-resolver-4.1.100.Final.jar;C:\kafka\libs\netty-transport-4.1.100.Final.jar;C:\kafka\libs\netty-transport-class
es-epoll-4.1.100.Final.jar;C:\kafka\libs\netty-transport-native-epoll-4.1.100.Final.jar;C:\kafka\libs\netty-transport-native-unix-common-4.1.100.Final.jar;C
:\kafka\libs\opentelemetry-proto-1.0.0-alpha.jar;C:\kafka\libs\osgi-resource-locator-1.0.3.jar;C:\kafka\libs\paranamer-2.8.jar;C:\kafka\libs\pcollections-4.
0.1.jar;C:\kafka\libs\plexus-utils-3.3.1.jar;C:\kafka\libs\protobuf-java-3.23.4.jar;C:\kafka\libs\reflections-0.10.2.jar;C:\kafka\libs\reload4j-1.2.25.jar;C
:\kafka\libs\rocksdbjni-7.9.2.jar;C:\kafka\libs\scala-collection-compat_2.13-2.10.0.jar;C:\kafka\libs\scala-java8-compat_2.13-1.0.2.jar;C:\kafka\libs\scala-
library-2.13.12.jar;C:\kafka\libs\scala-logging_2.13-3.9.4.jar;C:\kafka\libs\scala-reflect-2.13.12.jar;C:\kafka\libs\slf4j-api-1.7.36.jar;C:\kafka\libs\slf4
j-reload4j-1.7.36.jar;C:\kafka\libs\snappy-java-1.1.10.5.jar;C:\kafka\libs\swagger-annotations-2.2.8.jar;C:\kafka\libs\trogdor-3.7.0.jar;C:\kafka\libs\zooke
eper-3.8.3.jar;C:\kafka\libs\zookeeper-jute-3.8.3.jar;C:\kafka\libs\zstd-jni-1.5.5-6.jar (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,041] INFO Server environment:java.library.path=C:\Program Files\Java\jdk-11\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;
C:\Program Files\Common Files\Oracle\Java\javapath;C:\app\patir\product\21c\dbhomeXE\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\
System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;%SPARK_HOME%\bin;%JAVA_HOME%\bin;%CASSANDRA_HOME%\bin;C:\Program Files\Docker\Docker\resources
\bin;%HADOOP_HOME%\bin;C:\Users\patir\AppData\Local\Programs\Python\Launcher\;C:\Users\patir\AppData\Local\Microsoft\WindowsApps;C:\Users\patir\AppData\Loca
l\Programs\Microsoft VS Code\bin;C:\Users\patir\AppData\Local\GitHubDesktop\bin;. (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,041] INFO Server environment:java.io.tmpdir=C:\Users\patir\AppData\Local\Temp\ (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,041] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,041] INFO Server environment:os.name=Windows 11 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,041] INFO Server environment:os.arch=amd64 (org.apache.zookeeper.server.ZooKeeperServer)
```





Kafka is used to stream large volumes of data in an efficient and fault-tolerant manner. It is a good choice for building real-time data pipelines. Architecture kafka is based on a distributed publish/subscribe (pub/sub) system, where producers send messages to topics, and consumers read these messages asynchronously. Data is distributed into partitions within each topic, and these partitions are replicated across different brokers (cluster nodes) to ensure fault tolerance. Zookeeper is used to coordinate and manage the state of the cluster, ensuring the high availability and resilience of Kafka.

Project Report Requirements (2000 words exclude reference)

Divide your project report into the following four sections:

	Section 1: Summary of the project (500 words) – using APA-V7 style in the academic format
	Provide a concise executive summary of the project, highlighting its key objectives, methodologies employed, and major findings. Present a background of the project, outlining the context, significance, and any relevant historical information that sets the stage for your work.
	Please, check in the end of this document I have another pdf only writing about this section.

	Section2: Task-Specific Description and Technical Details
	Provide a specific description, and the technical aspects of tasks assigned in parts A and B. Clearly articulate the goals, methodologies, and any relevant technical details that will help the grader understand the scope and execution of your work.
	Please, check in the end of this document I have another pdf only writing about this section.

	Section 3: Ethical Data Analysis Considerations
	Provide a short description of the source of your dataset and describe how the data was collected, including any tools or methods used. Indicate whether you have the necessary permissions or consent to use the data, especially if it involves personal or sensitive information. Explain the measures taken to ensure the privacy and anonymity of individuals represented in the data. Ensure that your data usage complies with relevant ethical guidelines and regulations.
	Please, check in the end of this document I have another pdf only writing about this section.

	Section 4: Conclusion
	Discuss the insights derived from your analysis. Identify and elaborate on the challenges encountered during the project. Outline your solutions implemented to overcome the identified challenges. Explain the rationale behind your chosen solutions and their effectiveness. Propose suggestions for future work or improvements based on the outcomes of your project.
	Please, check in the end of this document I have another pdf only writing about this section.

	Section 5: References
	Provide a list of references in APA-V7 format that you have referred to external sources. Ensure accurate citation for any data, literature, or methodologies borrowed from other works.
	Please, check in the end of this document I have another pdf only writing about this section.

Project Report Requirement: Online Shopping

Patricia Rangel Santos

Graduate Diploma in Data Analytics, NZSE New Zealand Skills & Education Group – Auckland

Advanced-Data Engineering

Dra Sara Zandi

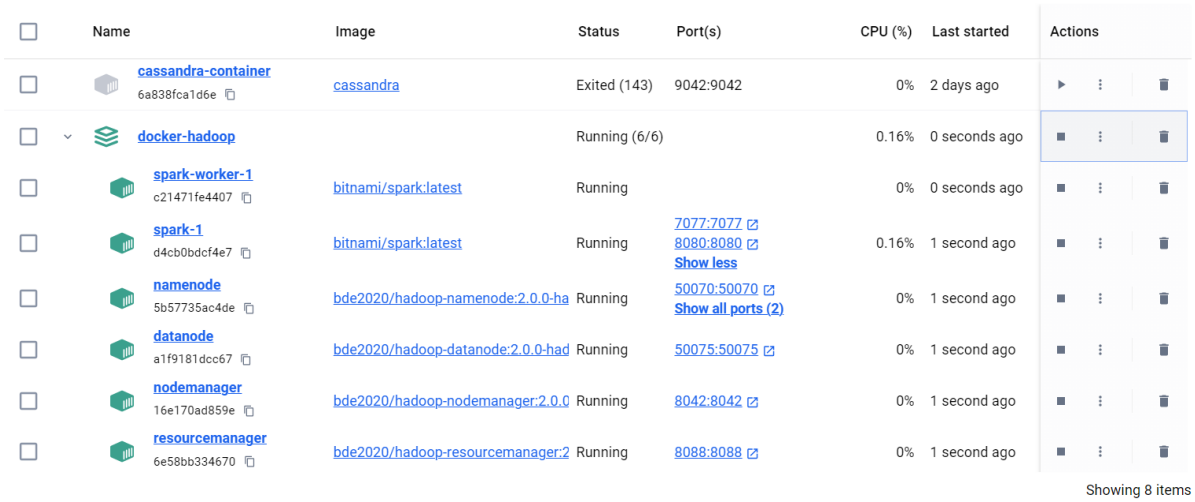
16/08/2024

Summary of the Project Online Shopping

In the Online Shopping project, I am applying data analysis practices to key variables in this scenario to implement changes in my business with more precision. I used the Linear Regression algorithm for the variables sales, discount, order quantity, profit, shipping cost, and product base margin. By studying customer profiles and the product categories that are most sold in the online store, I aim to identify where customers feel most comfortable making purchases and their average consumption of my products. This allows me to offer more targeted and effective promotions and deals. This phase focuses solely on data analysis using Python and its tools.

Now, let's explore a bit more about the tools needed to maintain an information system environment using Big Data technologies, such as the non-relational database MongoDB, Docker-Hadoop, Apache Spark, and Kafka. This entire structure allows us to explore information more efficiently, enabling more accurate decision-making. For further analysis, we will focus on clients who are purchasing very few products and develop a marketing strategy for those regions to understand why this is happening in certain areas.

We configured the environment for Docker-Hadoop as shown in the image below, as we need the server to manage cluster resources and ensure that services are always available.

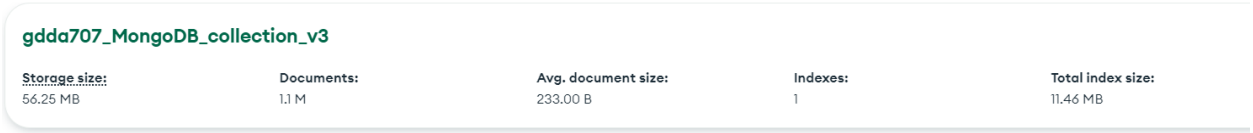


Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
cassandra-container 6a838fca1d6e	cassandra	Exited (143)	9042:9042	0%	2 days ago	
docker-hadoop		Running (6/6)		0.16%	0 seconds ago	
spark-worker-1 c21471fe4407	bitnami/spark:latest	Running		0%	0 seconds ago	
spark-1 d4cb0bdcf4e7	bitnami/spark:latest	Running	7077:7077 8080:8080 Show less	0.16%	1 second ago	
namenode 5b57735ac4de	bde2020/hadoop-namenode:2.0.0-ha	Running	50070:50070 Show all ports (2)	0%	1 second ago	
datanode a1f9181dcc67	bde2020/hadoop-datanode:2.0.0-had	Running	50075:50075	0%	1 second ago	
nodemanager 16e170ad859e	bde2020/hadoop-nodemanager:2.0.0	Running	8042:8042	0%	1 second ago	
resourcemanager 6e58bb334670	bde2020/hadoop-resourcemanager:2	Running	8088:8088	0%	1 second ago	

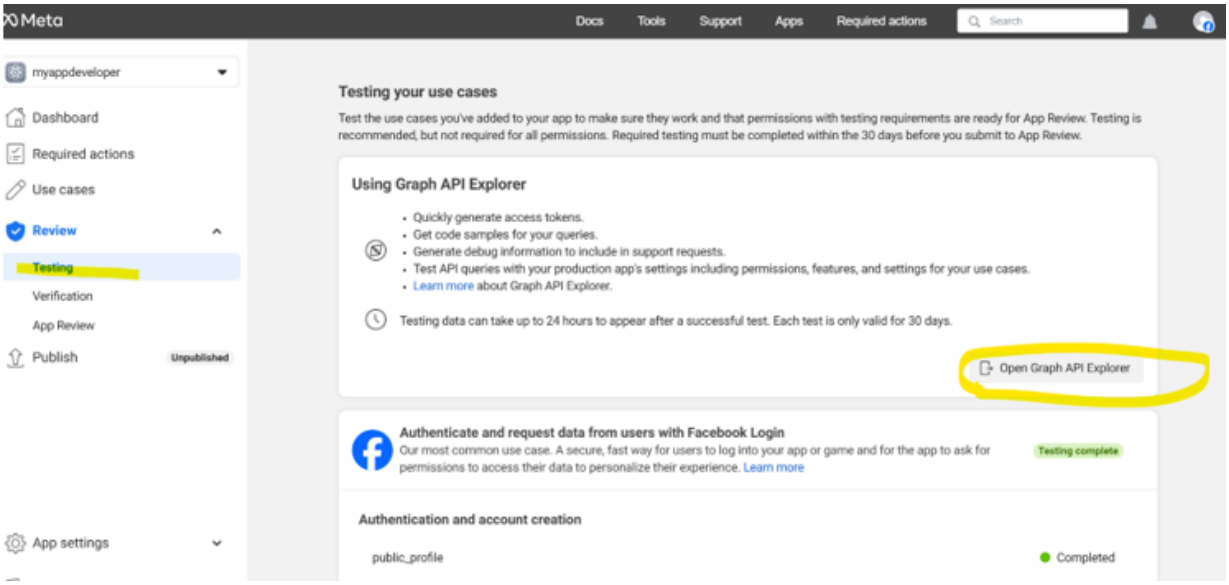
Showing 8 items

We set up the environment for Apache Spark by downloading **spark-3.3.0-bin-hadoop3** and configuring the environment variables. As part of this process, we also installed Java 11, which is required for Apache Spark.

We will use MongoDB in this environment, where we will insert a new collection into the database, as shown in the image below.



We also set up an environment for monitoring social networks using Apache Spark because we already have the system configured for it. We will use Facebook with Meta's API to configure this monitoring.



The final part of the configuration setup involves installing and configuring Kafka, where I will need to create the pipeline for this project. Here, I will explain all the details of this entire installation and configuration, along with the data analysis. This entire structure supports the essential business decisions for the shopping center, allowing us to better target marketing campaigns, offers, and services to our customers while striving for excellence in our partnerships. Consequently, a modern architecture for processing and analysing sales data is employed, utilizing technologies like Apache Kafka, Apache Spark, HDFS, and MongoDB. The need to process large volumes of transactional and customer data in real time is crucial for companies seeking agile insights and strategic decisions. Together, these technologies provide a comprehensive and real-time view of sales patterns, enabling better business management and more effective adaptation to market dynamics**.**

Tasks Part A and Part B in Online Shopping

Part A: I downloaded the dataset from the site https://archive.ics.uci.edu/ and searched for sites related to online shopping because I wanted a large dataset with many records to work

with Big Data. For the analysis, I used pandas with Python, loading the CSVs, organizing, and cleaning the data.

I decided to start the analysis with the market.csv, product.csv, and customer.csv datasets. I loaded the data, and below, I show the outputs of the executions.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8399 entries, 0 to 8398
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Customer_Name         8399 non-null   object
 1   Province              8399 non-null   object
 2   Region                8399 non-null   object
 3   Customer_Segment      8399 non-null   object
 4   Sales                 8399 non-null   float64
 5   Discount              8399 non-null   float64
 6   Order_Quantity        8399 non-null   int64
 7   Profit                8399 non-null   float64
 8   Shipping_Cost         8399 non-null   float64
 9   Product_Base_Margin   8336 non-null   float64
 10  Product_Category      8399 non-null   object
 11  Product_Sub_Category  8399 non-null   object
dtypes: float64(5), int64(1), object(6)
memory usage: 787.5+ KB
```

Data set information with its fields, types and database size.

	Customer_Name	Province	Region	Customer_Segment	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	261.54	0.04	6	-213.25	35.00	0.80
1	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	370.48	0.04	28	-5.45	4.51	0.59
2	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	905.08	0.09	22	127.70	6.22	NaN
3	MONICA FEDERLE	NUNAVUT	NUNAVUT	CORPORATE	2781.82	0.07	21	-695.26	35.00	NaN
4	NICOLE HANSEN	NUNAVUT	NUNAVUT	SMALL BUSINESS	3338.98	0.00	40	-846.73	35.00	0.81
...	...	...	...	...	...	...	...	...	...	...
8394	SHUI TOM	ALBERTA	WEST	HOME OFFICE	5793.46	0.01	38	2477.77	13.99	0.38
8395	FRANK HAWLEY	ALBERTA	WEST	HOME OFFICE	12007.05	0.09	16	2713.95	55.30	0.40
8396	AARON BERGMAN	ALBERTA	WEST	CORPORATE	4233.15	0.08	35	1219.87	26.30	0.38
8397	ADRIAN SHAMI	ALBERTA	WEST	CONSUMER	754.92	0.00	7	-129.57	19.99	0.52
8398	HARRY GREENE	ALBERTA	WEST	CORPORATE	10071.09	0.10	41	1977.69	17.86	0.58

8399 rows × 12 columns

Product_Category	Product_Sub_Category
OFFICE SUPPLIES	STORAGE & ORGANIZATION
OFFICE SUPPLIES	STORAGE & ORGANIZATION
OFFICE SUPPLIES	STORAGE & ORGANIZATION
OFFICE SUPPLIES	STORAGE & ORGANIZATION
OFFICE SUPPLIES	STORAGE & ORGANIZATION
...	...
TECHNOLOGY	OFFICE MACHINES
TECHNOLOGY	OFFICE MACHINES
TECHNOLOGY	OFFICE MACHINES
TECHNOLOGY	OFFICE MACHINES
TECHNOLOGY	OFFICE MACHINES

The fields in the Online Shopping dataset allow us to perform analysis and identify key areas that are driving sales, as well as where improvements can be made in the online sales business. I organized and validated the data before merging the datasets as follows:

```
: # Cheking missing values
  merge_df.isnull().sum()
```

```
: Customer_Name            0
  Province                 0
  Region                   0
  Customer_Segment         0
  Sales                    0
  Discount                 0
  Order_Quantity           0
  Profit                   0
  Shipping_Cost            0
  Product_Base_Margin     63
  Product_Category         0
  Product_Sub_Category     0
  dtype: int64
```

```
: # Fill missing salaries with the median product base margin
  merge_df['Product_Base_Margin'].fillna(merge_df['Product_Base_Margin'].median(), inplace=True)
```

Include the average for the product base margin field because they all had incorrect values and could not be used in our analysis.

I decided to delete fields that were not necessary in my analysis to simplify and not perform calculations without the need for information. As the cust id, ord id, prod id fields were all used to perform the merge and did not need to be used in the calculations.

```
# Dropped some column in my data set.
merge_df.drop(columns= ['Cust_id', 'Ord_id', 'Prod_id', 'Ship_id'], inplace = True)
merge_df
```

After I did the statistical summary of the data where I can have as a response the average of the fields, minimum, maximum, std and other totals. As in the image below and thus it makes it easier to have an idea of the base in question.

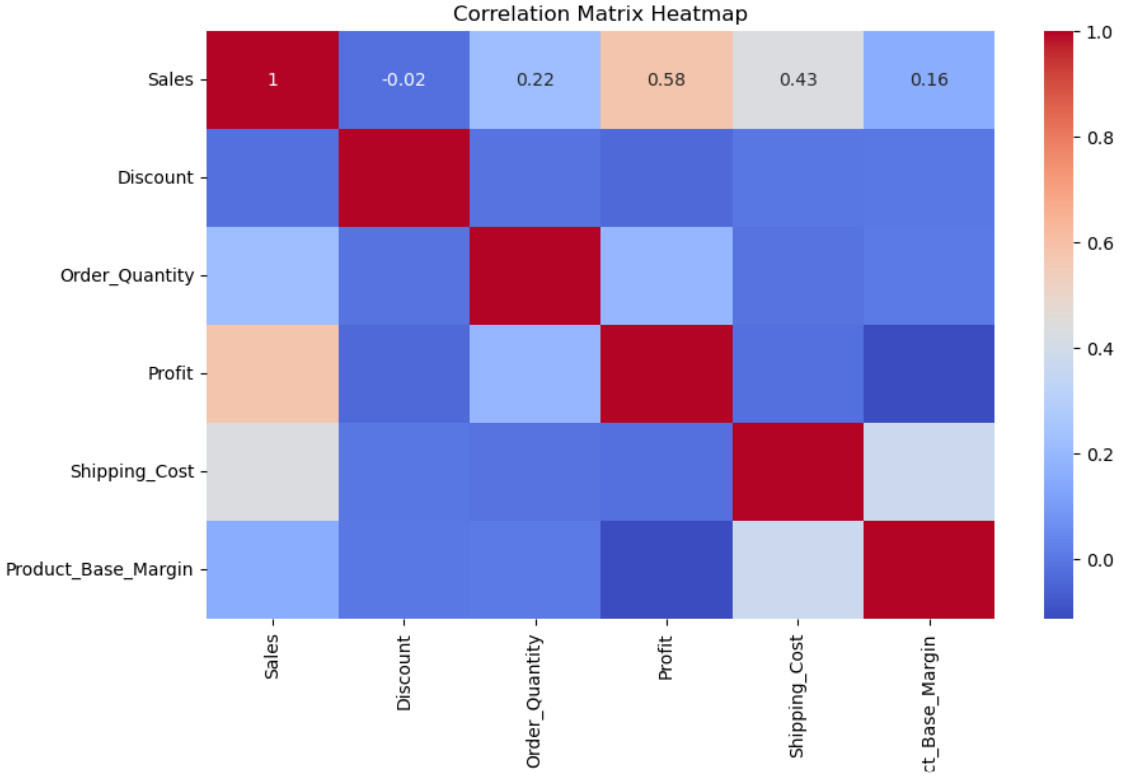
```
# Display summary statistics of the dataset
print("\nSummary Statistics:")
print(merge_df.describe())
```

```
Summary Statistics:
              Sales     Discount  Order_Quantity        Profit  Shipping_Cost  \
count   8399.000000  8399.000000     8399.000000   8399.000000    8399.000000
mean    1775.878179     0.049671       25.571735    181.184424      12.838557
std     3585.050525     0.031823       14.481071   1196.653371      17.264052
min        2.240000     0.000000        1.000000 -14140.700000       0.490000
25%      143.195000     0.020000       13.000000    -83.315000       3.300000
50%      449.420000     0.050000       26.000000     -1.500000       6.070000
75%     1709.320000     0.080000       38.000000    162.750000      13.990000
max    89061.050000     0.250000       50.000000  27220.690000     164.730000

       Product_Base_Margin
count          8336.000000
mean              0.512513
std               0.135589
min               0.350000
25%               0.380000
50%               0.520000
75%               0.590000
max               0.850000
```

We have the correlation output of the variables below along with the values and the graph. And it shows us the relationship between the variables.

```python
# Correlation Analysis
correlation_matrix = merge_df.select_dtypes(include=[np.number]).corr()
print("Correlation matrix:")
print(correlation_matrix)

# Heatmap of correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
Correlation matrix:
                       Sales  Discount  Order_Quantity    Profit  \
Sales                1.000000 -0.019686        0.220582  0.581960
Discount            -0.019686  1.000000       -0.009649 -0.037128
Order_Quantity       0.220582 -0.009649        1.000000  0.194655
Profit               0.581960 -0.037128        0.194655  1.000000
Shipping_Cost        0.434578 -0.001956       -0.011457 -0.021362
Product_Base_Margin  0.156759  0.004079        0.007839 -0.112985

                     Shipping_Cost  Product_Base_Margin
Sales                     0.434578             0.156759
Discount                 -0.001956             0.004079
Order_Quantity           -0.011457             0.007839
Profit                   -0.021362            -0.112985
Shipping_Cost             1.000000             0.373826
Product_Base_Margin       0.373826             1.000000
```



```
# Sales by Product and Category
sales_by_category = merge_df.groupby('Product_Category')['Sales'].sum().sort_values(ascending=False)

print("\nSales by Product Category:")
print(sales_by_category)
```

```
Sales by Product Category:
Product_Category
TECHNOLOGY         5984248.182
FURNITURE          5178590.542
OFFICE SUPPLIES    3752762.100
Name: Sales, dtype: float64
```

In this filter group by Product Category then add Sales and put in ascending order we can see the values sold by product category and we have Technology as the product category that has the highest sales index.

```
# Calculate the mean, median, and mode of the salary column
mean_sales = merge_df['Sales'].mean()
median_sales = merge_df['Sales'].median()
mode_sales = merge_df['Sales'].mode()[0]

print(f"\nMean Sales: {mean_sales}")
print(f"Median Sales: {median_sales}")
print(f"Mode Sales: {mode_sales}")
```

```
Mean Sales: 1775.878178830813
Median Sales: 449.42
Mode Sales: 10.48
```

Calculation of mean, average, and sales mode for everyone.

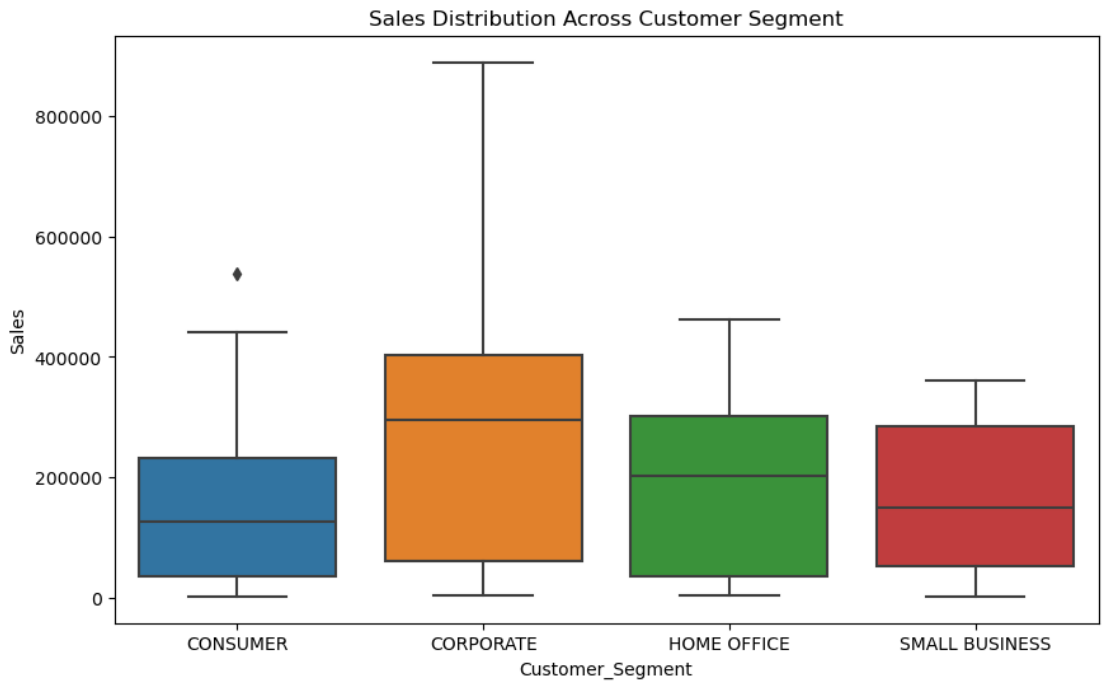
```
# Perform analysis: Product categories sold by segment
sales_by_segment_category = merge_df.groupby(['Customer_Segment', 'Product_Category', 'Product_Sub_Category'])['Sales'].sum().re
print("\nProduct categories sold by Segment:")
print(sales_by_segment_category)
```

Another filter where we can analyse a grouping of customer segments, product categories, and product sub category with the sum of sales. Presenting the output below:

```
Product categories sold by Segment:
   Customer_Segment Product_Category          Product_Sub_Category  \
0          CONSUMER        FURNITURE                     BOOKCASES
1          CONSUMER        FURNITURE           CHAIRS & CHAIRMATS
2          CONSUMER        FURNITURE           OFFICE FURNISHINGS
3          CONSUMER        FURNITURE                       TABLES
4          CONSUMER  OFFICE SUPPLIES                   APPLIANCES
..              ...              ...                          ...
63   SMALL BUSINESS  OFFICE SUPPLIES       STORAGE & ORGANIZATION
64   SMALL BUSINESS       TECHNOLOGY         COMPUTER PERIPHERALS
65   SMALL BUSINESS       TECHNOLOGY              COPIERS AND FAX
66   SMALL BUSINESS       TECHNOLOGY              OFFICE MACHINES
67   SMALL BUSINESS       TECHNOLOGY  TELEPHONES AND COMMUNICATION

          Sales
0   184419.7300
1   374635.0800
2   127840.0300
3   441912.3740
4   112337.4600
..          ...
63  220661.9800
64  149788.9000
65  284825.0800
66  331913.9400
67  360186.4035
```

Another view where we can analyze customer segment and sales and their distribution by customer product is that corporate has the highest performance in online shopping sales.

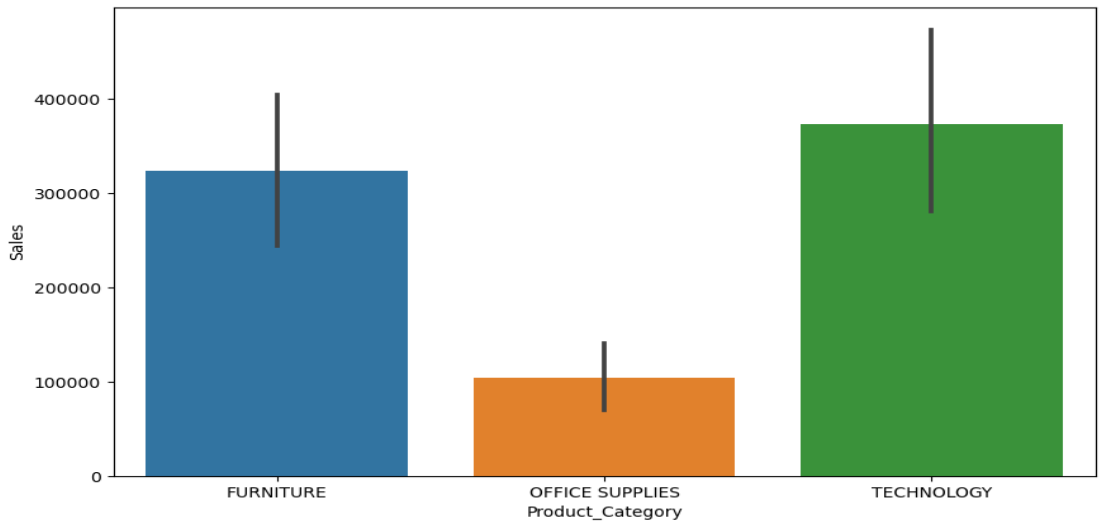
```
# Box plot of salary distribution across departments
plt.figure(figsize=(10, 6))
sns.boxplot(data=sales_by_segment_category, x='Customer_Segment', y='Sales')
plt.title('Sales Distribution Across Customer Segment')
plt.show()
```



In this other view we have the grouping of product categories and the best performance is with Technology.

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'Product_Category', y = 'Sales', data = sales_by_segment_category)
```

```
<Axes: xlabel='Product_Category', ylabel='Sales'>
```



With this calculation, we can have the same answer as the graph above where technology has the best performance in sales.

```
#highest_avg_sales_product_category

highest_avg_product_category = sales_by_segment_category.groupby('Product_Category')['Sales'].mean().idxmax()
print(f"\nProduct Category with the highest average sales: {highest_avg_product_category}")
```

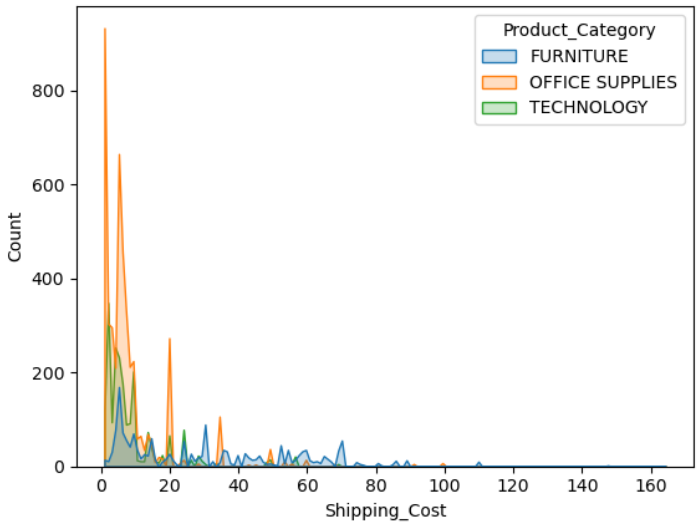
```
Product Category with the highest average sales: TECHNOLOGY
```

In this grouping, we have the fields product category, sales, and shipping cost with the average, as shown in the image below.

```
#  Group by Product Category and Sales - Mean shipping cost
product_category_shipping = merge_df.groupby(['Product_Category', 'Sales'])['Shipping_Cost'].mean().reset_index()
```

Then I present a graph with the information where we can analyze the data where the product category Office Supplies has a high average shipping cost.

```
: sns.histplot(product_category_shipping, x="Shipping_Cost", hue="Product_Category", element="poly")
: <Axes: xlabel='Shipping_Cost', ylabel='Count'>
```



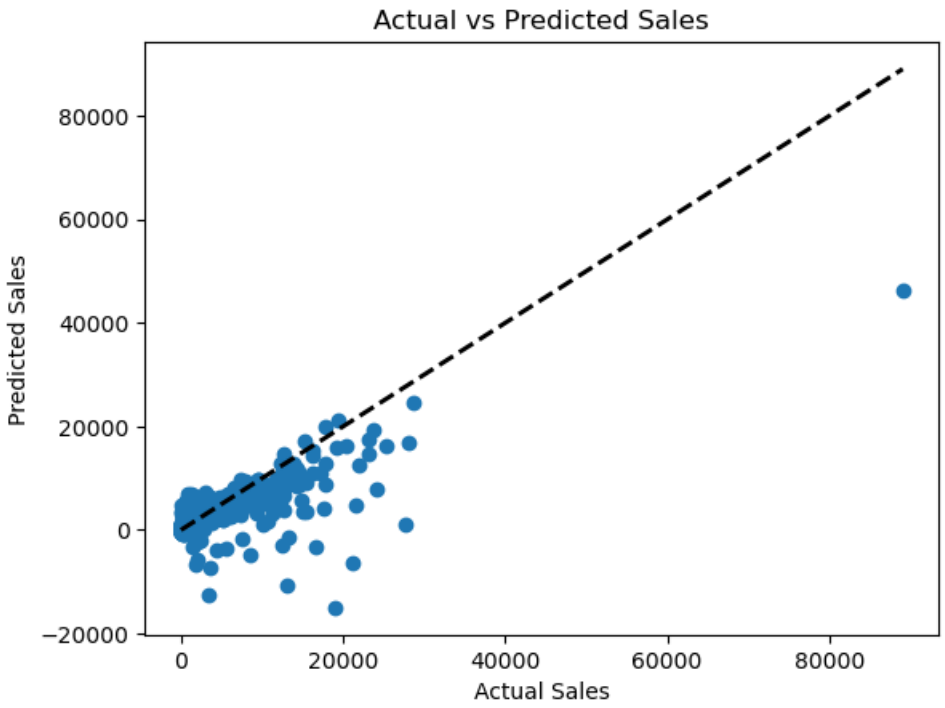
Using the Linear Regression model with prediction for the Online Shopping scenario, I analyze the target variables: discount, order quantity, profit, shipping cost, product base margin, and sales.

```
# Define features (X) and target variable (y)
X = merge_df[['Discount', 'Order_Quantity', 'Profit', 'Shipping_Cost', 'Product_Base_Margin']]
y = merge_df[['Sales']]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error adn
mse = mean_squared_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)


print("Mean Squared Error:", mse)
print("R-squared:", r_squared)

# Plot the best fit line
plt.scatter(y_test, y_pred)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)  # Plot the diagonal line
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.show()
```

```
Mean Squared Error: 6709614.698436826
R-squared: 0.6032294217288994
```

Mean Squared Error: In this case, the value of 6,709,614.70 is relatively high, suggesting that the model may be making significant errors in its predictions. This could indicate that the model is not capturing the relationships between the predictor variables and the dependent variable well, or that there is considerable variability in the data that the model cannot explain.

R-Squared: In this case, the value of 0.6032 means that approximately 60.32% of the variance in the dependent variable is explained by the model. This indicates that the model has moderate predictive power, but there is still 39.68% of the variance unexplained by the model, suggesting that other factors or variables may be influencing the dependent variable.



Graph of Linear Regression showing growth, though other variables still need to be analyzed.

We can start by reporting on the data load created via Python, where I established a connection to the database. In this case, I will use MongoDB for this purpose. I selected a dataset on Online Retail II with a total of 1,067,371 records, as shown in the image below.

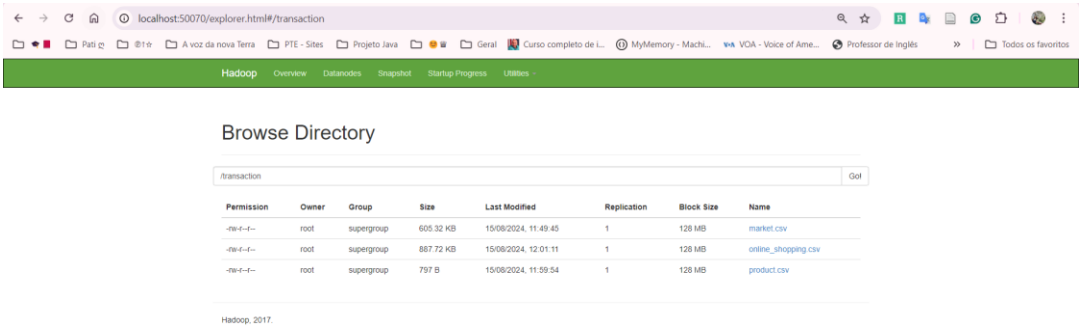
Let's continue with the explanation of HDFS used in Task 1. HDFS (Hadoop Distributed File System) operates on a master-slave architecture where the master manages the metadata and the system's namespace, while the Data Nodes store the data blocks. The data is divided into small blocks, typically 128 MB each, which are replicated across other Data Nodes to ensure fault tolerance and high availability.

In this task, we created an example of the scenario described above, but we do not have a large volume of data to fully assess the performance of the architecture.

```
C:\Users\patir\docker-hadoop>docker exec -it namenode /bin/bash
root@24e12b5bdd4a:/# hdfs dfs -mkdir/transaction
-mkdir/transaction: Unknown command
root@24e12b5bdd4a:/# hdfs dfs -mkdir /transaction
root@24e12b5bdd4a:/#
```

```
PS C:\Users\patir\docker-hadoop> ls


    Directory: C:\Users\patir\docker-hadoop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/08/2024     11:25           2322 docker-compose.yml
-a----        14/08/2024     18:40            407 docker-compose_spark.yml
-a----        09/07/2024     20:07         619850 market.csv
-a----        31/07/2024     18:06         909024 online_shopping.csv
-a----        09/07/2024     20:07            797 product.csv


PS C:\Users\patir\docker-hadoop> # Copie o arquivo para o contêiner
PS C:\Users\patir\docker-hadoop> docker cp C:/Users/patir/docker-hadoop/market.csv namenode:/tmp/market.csv
Successfully copied 622kB to namenode:/tmp/market.csv
PS C:\Users\patir\docker-hadoop>
```

Creating the transaction folder within Hadoop using HDFS, I then loaded three different datasets (market, online_shopping, product) into the server directory. Subsequently, these datasets were displayed on the Hadoop localhost.




```
df.head()
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

Data presented after being loaded into jupyter.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067371 entries, 0 to 1067370
Data columns (total 10 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   _id          1067371 non-null  object 
 1   index        1067371 non-null  int64  
 2   Invoice      1067371 non-null  object 
 3   StockCode    1067371 non-null  object 
 4   Description  1062989 non-null  object 
 5   Quantity     1067371 non-null  int64  
 6   InvoiceDate  1067371 non-null  object 
 7   Price        1067371 non-null  float64
 8   Customer ID  824364 non-null   float64
 9   Country      1067371 non-null  object 
dtypes: float64(2), int64(2), object(6)
memory usage: 81.4+ MB
```

Information about this database such as fields, data types, and the size of the database.

```
#Connect to MongoDB

#url = "mongodb+srv://patirangelsantos:aUdT7Jxk%40yUx2hu@cluster0.pnlxrbn.mongodb.net/"
url = "mongodb://localhost:27017/"

#client = MongoClient("mongodb+srv://patirangelsantos:aUdT7Jxk%40yUx2hu@cluster0.pnlxrbn.mongodb.net/")
client = MongoClient("mongodb://localhost:27017/")

try:
    # Create MongoClien objective
    db = client['gdda707_MongoDB']
    # Trying to connect to the server
    client.admin.command('ping')
    # Access collection
    collection = db['gdda707_MongoDB_collection_v3']
    print("successfu lconnection")
except ConnectionFailure:
    print("connection fail")
```

```
successfu lconnection
```

figure 1

Below is the information about the connection created with the database. After execution, the connection success message is displayed. We have the creation of the database with the name ‘gdda707_MongoDB’ and the creation of the collection with the name ‘gdda707_MongoDB_collection_v3’, which we will use in this execution.

```
collection
```

```
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'gdda707_MongoD
B'), 'gdda707_MongoDB_collection_v3')
```

figure 2

After executing the code in ‘figure 1’ in the variable defined as ‘collection’, the information in ‘figure 2’ displays all the details that were generated in the initial execution.

```
df.reset_index(inplace=True)
df_dict = df.to_dict('records')
df_dict
```

```
[{'index': 0,
  'Invoice': '489434',
  'StockCode': '85048',
  'Description': '15CM CHRISTMAS GLASS BALL 20 LIGHTS',
  'Quantity': 12,
  'InvoiceDate': '2009-12-01 07:45:00',
  'Price': 6.95,
  'Customer ID': 13085.0,
  'Country': 'United Kingdom'},
 {'index': 1,
  'Invoice': '489434',
  'StockCode': '79323P',
  'Description': 'PINK CHERRY LIGHTS',
  'Quantity': 12,
  'InvoiceDate': '2009-12-01 07:45:00',
  'Price': 6.75,
  'Customer ID': 13085.0,
  'Country': 'United Kingdom'},
 {'index': 2,
```

Figure 3

In the image ‘Figure 3’ we transformed the pandas information that was in a data frame to JSON in the **df.todict**(‘records) section, which can be saved in a collection as is done in a non-relational database.

```
collection.insert_many(df_dict)
```

```
InsertManyResult([ObjectId('66b96282b3de891b5fa79418'), ObjectId('66b96282b3de891b5fa79419'), ObjectId('66b96282b3de891b5fa79
41a'), ObjectId('66b96282b3de891b5fa7941b'), ObjectId('66b96282b3de891b5fa7941c'), ObjectId('66b96282b3de891b5fa7941d'), Obje
ctId('66b96282b3de891b5fa7941e'), ObjectId('66b96282b3de891b5fa7941f'), ObjectId('66b96282b3de891b5fa79420'), ObjectId('66b96
282b3de891b5fa79421'), ObjectId('66b96282b3de891b5fa79422'), ObjectId('66b96282b3de891b5fa79423'), ObjectId('66b96282b3de891b
5fa79424'), ObjectId('66b96282b3de891b5fa79425'), ObjectId('66b96282b3de891b5fa79426'), ObjectId('66b96282b3de891b5fa79427'),
ObjectId('66b96282b3de891b5fa79428'), ObjectId('66b96282b3de891b5fa79429'), ObjectId('66b96282b3de891b5fa7942a'), ObjectId('6
6b96282b3de891b5fa7942b'), ObjectId('66b96282b3de891b5fa7942c'), ObjectId('66b96282b3de891b5fa7942d'), ObjectId('66b96282b3de
891b5fa7942e'), ObjectId('66b96282b3de891b5fa7942f'), ObjectId('66b96282b3de891b5fa79430'), ObjectId('66b96282b3de891b5fa7943
1'), ObjectId('66b96282b3de891b5fa79432'), ObjectId('66b96282b3de891b5fa79433'), ObjectId('66b96282b3de891b5fa79434'), Object
Id('66b96282b3de891b5fa79435'), ObjectId('66b96282b3de891b5fa79436'), ObjectId('66b96282b3de891b5fa79437'), ObjectId('66b9628
2b3de891b5fa79438'), ObjectId('66b96282b3de891b5fa79439'), ObjectId('66b96282b3de891b5fa7943a'), ObjectId('66b96282b3de891b5f
a7943b'), ObjectId('66b96282b3de891b5fa7943c'), ObjectId('66b96282b3de891b5fa7943d'), ObjectId('66b96282b3de891b5fa7943e'), O
bjectId('66b96282b3de891b5fa7943f'), ObjectId('66b96282b3de891b5fa79440'), ObjectId('66b96282b3de891b5fa79441'), ObjectId('66
b96282b3de891b5fa79442'), ObjectId('66b96282b3de891b5fa79443'), ObjectId('66b96282b3de891b5fa79444'), ObjectId('66b96282b3de8
91b5fa79445'), ObjectId('66b96282b3de891b5fa79446'), ObjectId('66b96282b3de891b5fa79447'), ObjectId('66b96282b3de891b5fa7944
8'), ObjectId('66b96282b3de891b5fa79449'), ObjectId('66b96282b3de891b5fa7944a'), ObjectId('66b96282b3de891b5fa7944b'), Object
Id('66b96282b3de891b5fa7944c'), ObjectId('66b96282b3de891b5fa7944d'), ObjectId('66b96282b3de891b5fa7944e'), ObjectId('66b9628
2b3de891b5fa7944f'), ObjectId('66b96282b3de891b5fa79450'), ObjectId('66b96282b3de891b5fa79451'), ObjectId('66b96282b3de891b5f
a79452'), ObjectId('66b96282b3de891b5fa79453'), ObjectId('66b96282b3de891b5fa79454'), ObjectId('66b96282b3de891b5fa79455'), O
```

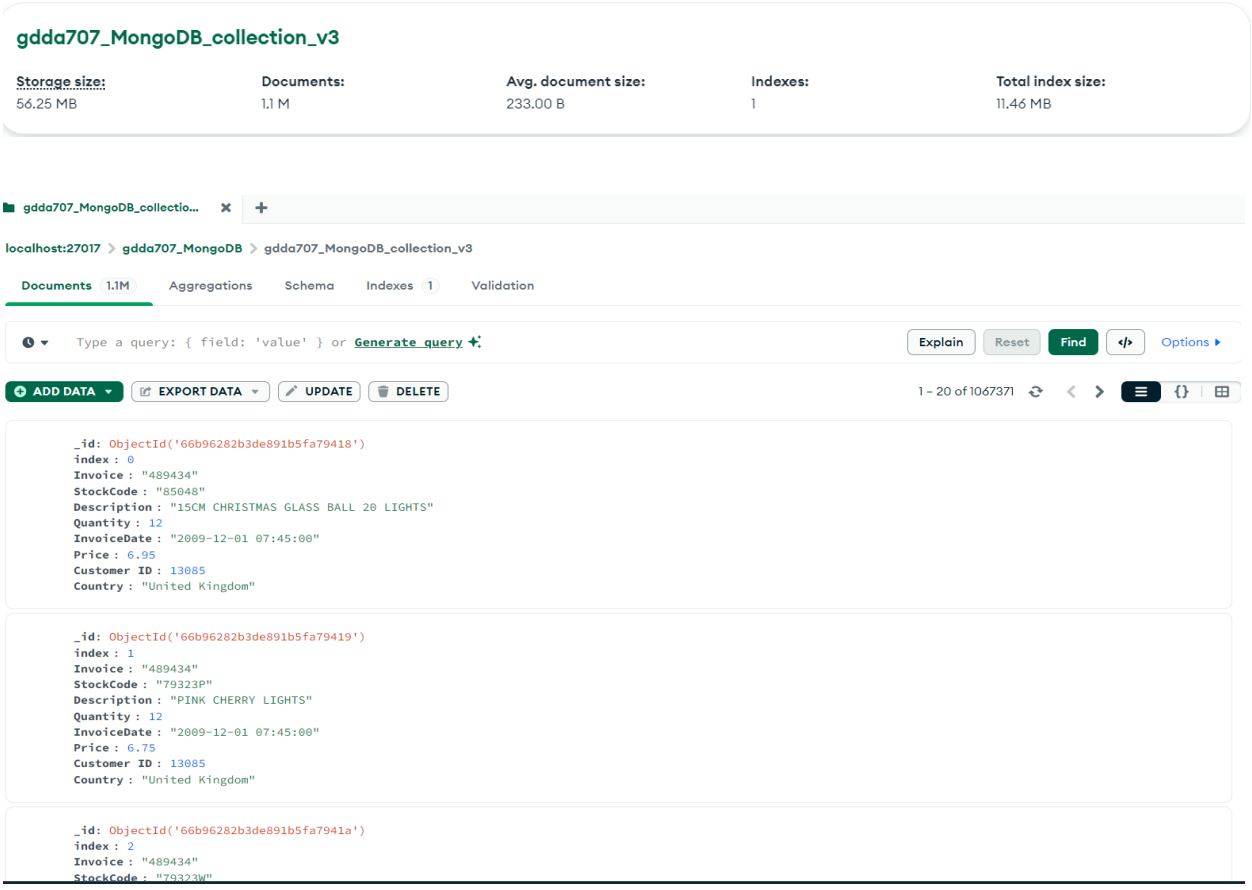
After executing the code in the image above we can see the output generating the Object IDs of the records saved in the collection.

```
df =  pd.DataFrame(list(collection.find()))
df.head(n =5)
```

	_id	index	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	66b96282b3de891b5fa79418	0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	66b96282b3de891b5fa79419	1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	66b96282b3de891b5fa7941a	2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	66b96282b3de891b5fa7941b	3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	66b96282b3de891b5fa7941c	4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

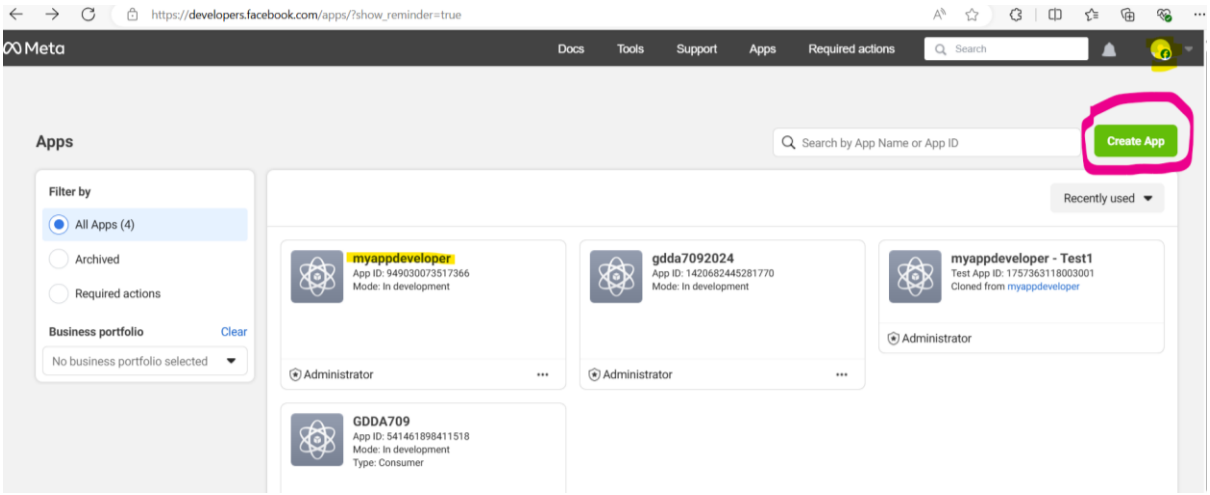
```
client.close()
```

I have the search execution for the collection where it presents the data again in a data frame and we close the Mongo DB session.

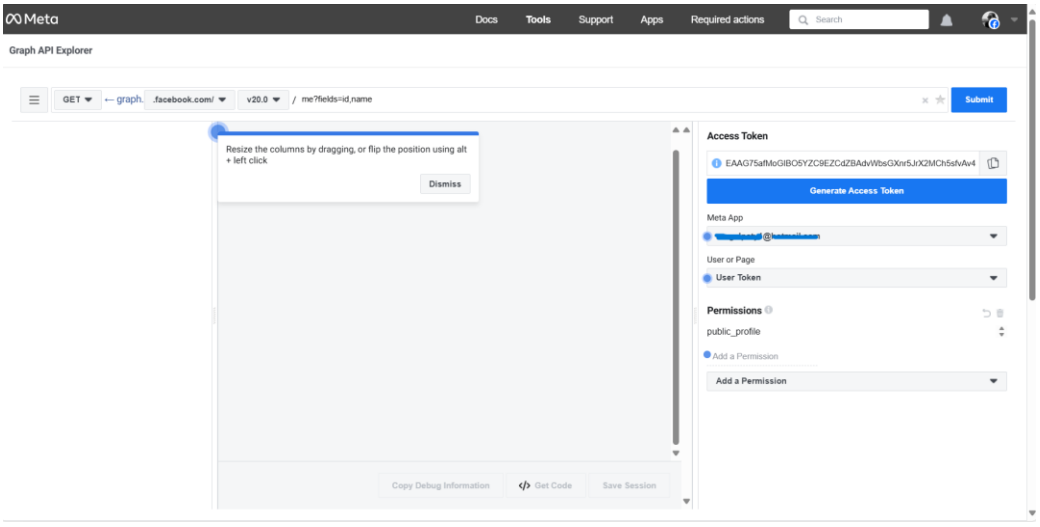


The database created in Mongo DB in the image above and we have the creation of the collection generated in the same database.

We created a real-time scenario with Apache Spark using the Facebook APP, as shown in the image below.



A tela do APPs com o meu login do Facebook



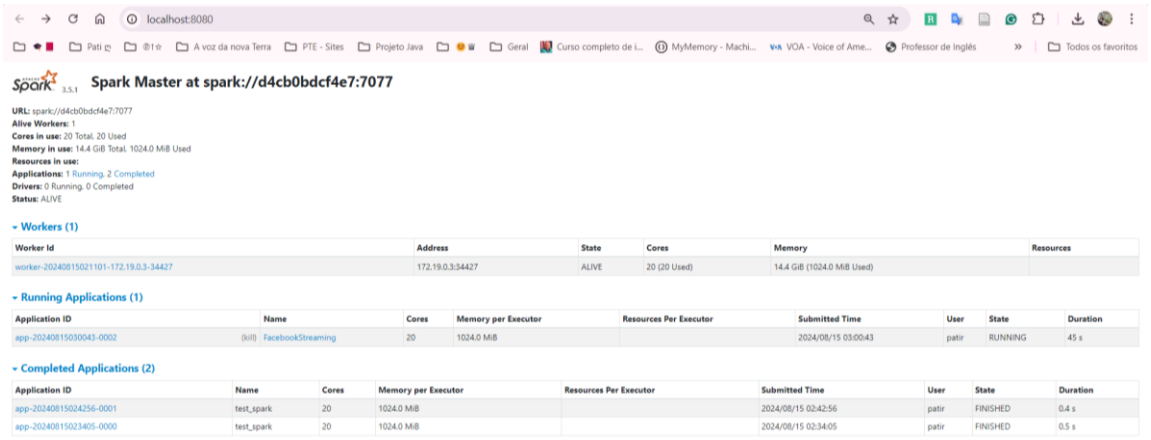
This generates the access token and configures it in Python to release the Spark connection to the application. Below is an excerpt from the code used to perform this simulation, as shown in the image below. Apache Spark has a master-slave architecture, where the Driver Program acts as

the master, controlling the application flow and managing the tasks. Talking to the workers (slaves) that execute the distributed tasks. Spark uses a distributed data model in memory called RDD (Resilient Distributed Datasets), which allows fast and fault-tolerant processing for large volumes of data.

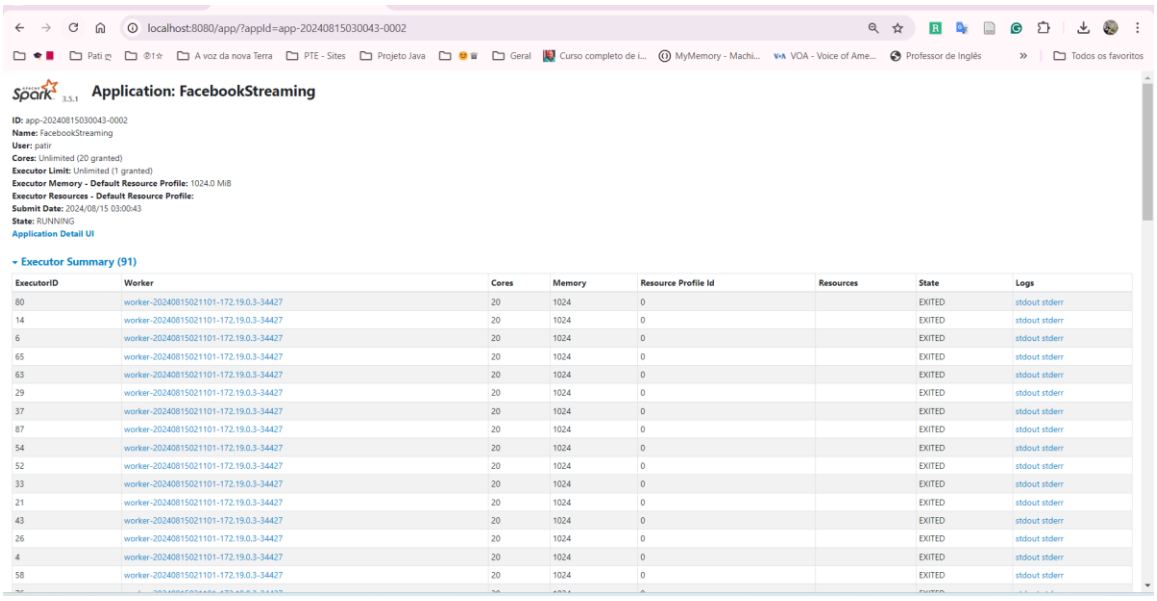
Where I made the **facebook_streaming.py** file available within the Apache Spark directory to be executed in the architecture.

```
# Function to simulate streaming data by continuously fetching from Facebook API
def fetch_and_stream_data(spark, access_token):
    user_id = '[redacted]'  # Update with your user ID
    while True:
        logger.info("Fetching posts for user ID: %s", user_id)  # Log the user ID being queried
        posts = fetch_facebook_data(access_token, f"{user_id}/posts")

        data = [json.dumps(post) for post in posts.get('data', [])]

        if data:
            logger.info("Received %d posts", len(data))  # Log the number of posts received
            df = spark.read.json(spark.sparkContext.parallelize(data))
            df.write.format("console").option("truncate", "false").save()
        else:
            logger.info("No new posts received.")  # Log if no new posts are available

        time.sleep(30)  # Adjust the sleep interval as needed

if __name__ == "__main__":
    spark = SparkSession.builder \
        .appName("FacebookStreaming") \
        .getOrCreate()

    access_token = '[redacted]'

    # Start the data fetching and streaming process
    logger.info("Starting the Facebook streaming process...")
    fetch_and_stream_data(spark, access_token)
```

Localhost 7077 spark.



After running the application.



We set up the environment for Kafka to generate a pipeline based on Zookeeper, which is highly reliable and used to ensure consistency, elect leaders, manage configurations and synchronize distributed processes, facilitating the coordination of large clusters of services such as Kafka, HDFS and other distributed systems.

We have some configurations to make it easier to understand after downloading Kafka in Confluent and we need to configure properties files such as:

```
@echo off
cd C:\kafka
call .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

Zookeeper-server-start.bat

```
@echo off
cd C:\kafka
call .\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic kafkaLab --from-beginning
```

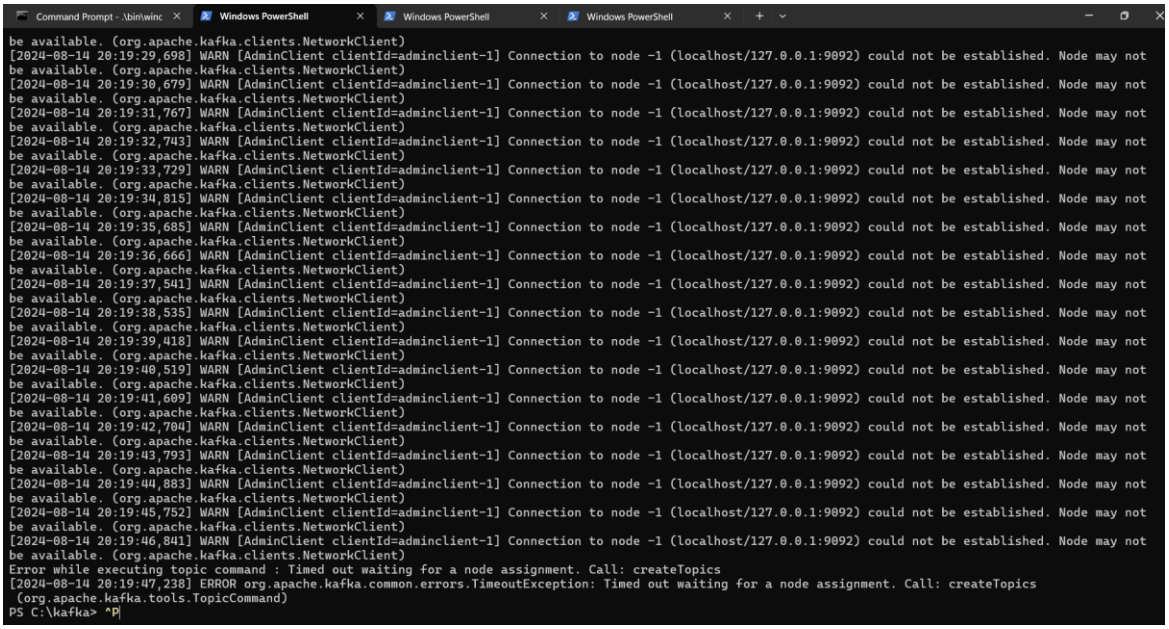
Kafka-console-consumer.bat

```
@echo off
cd C:\kafka
call .\bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --topic kafkaLab
```

Kafka-topics.bat

And we need to perform the executions of all these bats. Where it will present some outputs below the execution of Kafka.

```
[2024-08-14 19:42:55,996] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,010] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2024-08-14 19:42:56,013] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2024-08-14 19:42:56,013] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMai
n)
[2024-08-14 19:42:56,014] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-08-14 19:42:56,015] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,015] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-14 19:42:56,016] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2024-08-14 19:42:56,016] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-08-14 19:42:56,025] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@2e222612 (org.apache.zookeep
er.server.ServerMetrics)
[2024-08-14 19:42:56,027] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-08-14 19:42:56,027] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-08-14 19:42:56,029] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-08-14 19:42:56,038] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO   ______                     _                                                (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  |___  /                    | |                                               (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO     / /    ___     ___     | |  __   ___    ___   _ __     ___   _ __      (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO    / /    / _ \   / _ \    | | /_/  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO   / /__  | (_) | | (_) |   |  <    |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  /_____|  \___/   \___/   |_|\_\   \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO                                                 | |                          (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO                                                 |_|                          (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,038] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apach
e.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:host.name=Pati_Rangel.nzse.com (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.version=11.0.23 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.home=C:\Program Files\Java\jdk-11 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:java.class.path=C:\kafka\libs\activation-1.1.1.jar;C:\kafka\libs\aopalliance-repackaged-2.6.1.jar;C:\kafka
```



Ethical Data Analysis Considerations

My data source to perform this task was the website https://archive.ics.uci.edu/ where I could navigate and check the possible scenarios for my analysis. I looked for databases that had more records.

I used Python with pandas for analysis and calculations, I checked in my databases that I had information such as name and address that I normally try not to use in my analyses.

We need to pay close attention at this point so as not to violate ethics and take care of the information we are going to analyse. Always inform yourself at the company, client, or place of operation about the best practices related to the topic. When performing the analysis, try to use more general data or, if necessary, transform this information so that it is understood what is really necessary for the business and for people without harming or exposing the environment unnecessarily, taking care of everyone's well-being.

Normally, for analysis, we need to send the information worked on to other areas and at this point, we can unnecessarily expose sensitive information to the scenario, whenever possible, check with the person responsible for the topic and the best solution, or use the most open data.

Always stay up to date with the best practices in the places where we operate.

Conclusion

The conclusion of this project was that companies were able to observe in which product category they are investing the most in technology, and the sector that invests the most is the corporate sector. The category with the least sales would be the consumer sector. We can explore marketing campaigns or check why this category has this performance and make internal improvements. By checking the filter in region and province, we can see that the highest sales performance is in the West, Atlantic, and Prairie regions, which together have a percentage of 61.53% of sales, and the other regions have a percentage of 38.45% of total sales, where we can perform checks and analyses to improve the performance of these areas. All this analysis can be improved with the use of Apache Spark, Kafka, Hadoop, and Mongo DB technologies. With Kafka, with real-time processing, we can extract information from social networks and explore sales data. Apache Spark allows us to perform quick analyses, HDSF allows us to store a large volume of data, and Mongo DB allows us flexibility in data queries, such as customer profiles and transactions. With this entire structure, we can perform robust sales analysis, enabling new business possibilities in real-time.

References

What is a Kafka Topic? (confluent.io):

https://developer.confluent.io/courses/apachekafka/topics/.

Apache Hadoop: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Apache Spark: https://spark.apache.org/