

GDDA707 – Advanced Data Engineering

16 August 2024

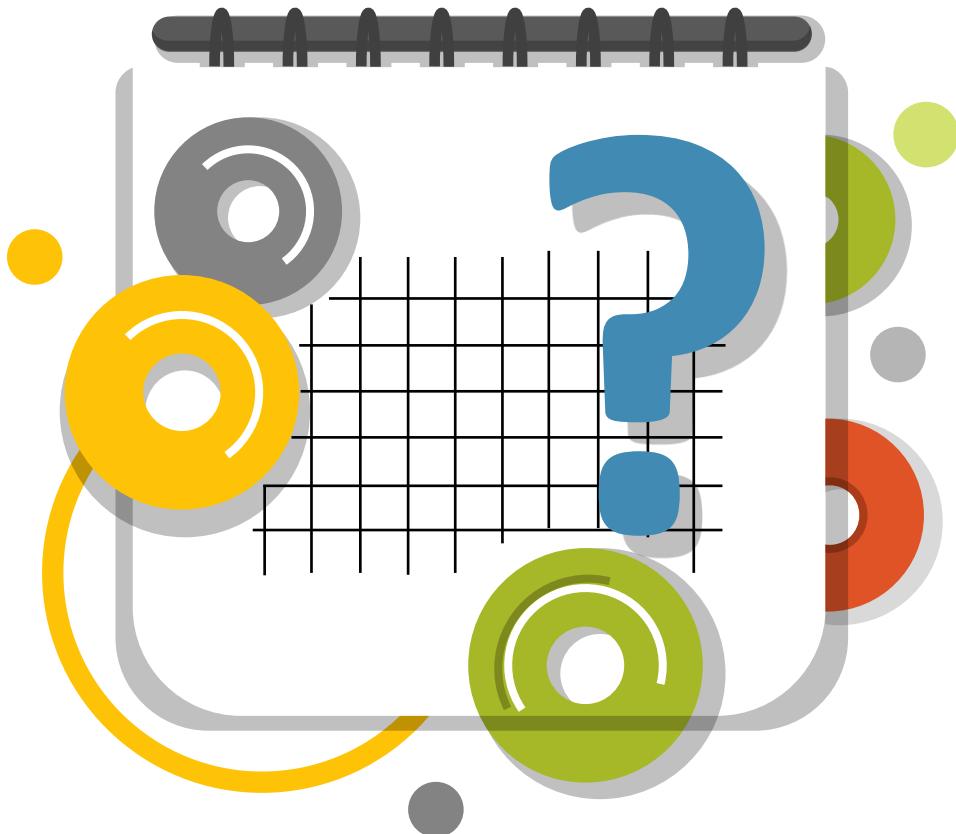


Table of Contents

GDDA707 – Advanced Data Engineering	0
1. Introduction.....	2
Overview	2
Purpose	2
Part A: ETL Operations	3
Task 1: Selection of Dataset and Tools	3
Task 2: Load and Pre-processing.....	3
Task 3: ETL Operations and Integration.....	8
Part B: Big Data Analysis and Application of Engineering Techniques	11
Task 1	11
Task 2	12
Task 3	15
Task 4	18
Project Report Requirements (2000 words exclude reference)	22
Section 1: Summary of the project (500 words) – using APA-V7 style in academic format.....	22
Section2: Task-Specific Description and Technical Details	22
Section 3: Ethical Data Analysis Considerations	22
Section 4: Conclusion	22
Section 5: References.....	22

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.

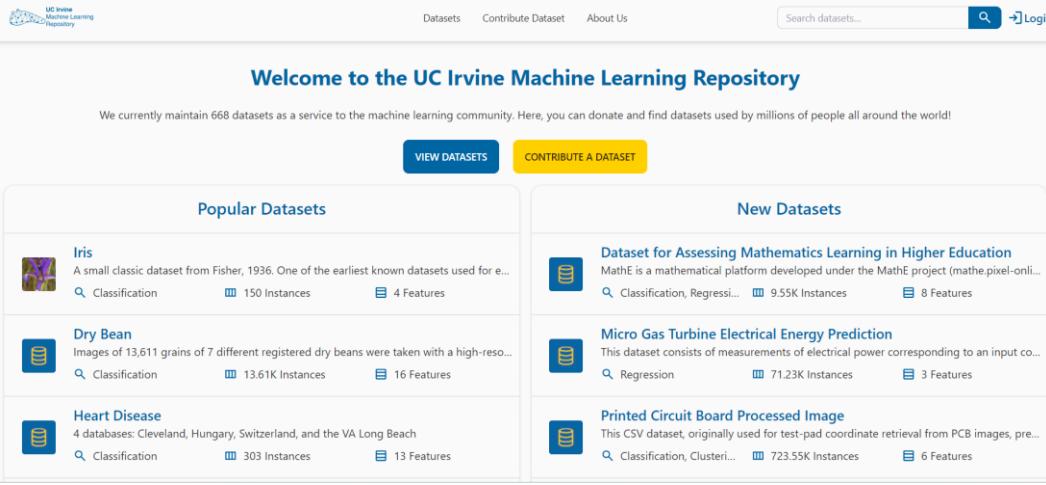
Data Engineering on Big Data Platforms: Implementing data engineering techniques to manage, process, and analyze large volumes of data using cloud-based big data platforms.

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 Tools	
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\\Assessment_2\\customer.csv")
market_df = pd.read_csv("C:\\Users\\patir\\Desktop\\GDDA - Data Analytics\\GDDA707 Advanced Data Engineering\\Assessment_2\\market.csv")
product_df = pd.read_csv("C:\\Users\\patir\\Desktop\\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

4899 rows × 6 columns

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	Pi
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 column 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 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	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.

	<pre>2]: # Show data set after merge merge_df</pre> <table border="1"> <thead> <tr> <th></th><th>Customer_Name</th><th>Province</th><th>Region</th><th>Customer_Segment</th><th>Cust_id</th><th>Ord_id</th><th>Prod_id</th><th>Ship_id</th><th>Sales</th><th>Discount</th><th>Order_Quantity</th><th>Profit</th><th>Shipp</th></tr> </thead> <tbody> <tr><td>0</td><td>MUHAMMED MACINTYRE</td><td>NUNAVUT</td><td>NUNAVUT</td><td>SMALL BUSINESS</td><td>Cust_1</td><td>Ord_1</td><td>Prod_1</td><td>SHP_1</td><td>261.54</td><td>0.04</td><td>6</td><td>-213.25</td><td></td></tr> <tr><td>1</td><td>CARLOS SOLTERO</td><td>NUNAVUT</td><td>NUNAVUT</td><td>CONSUMER</td><td>Cust_4</td><td>Ord_20</td><td>Prod_1</td><td>SHP_27</td><td>370.48</td><td>0.04</td><td>28</td><td>-5.45</td><td></td></tr> <tr><td>2</td><td>CARL JACKSON</td><td>NUNAVUT</td><td>NUNAVUT</td><td>CORPORATE</td><td>Cust_5</td><td>Ord_5</td><td>Prod_1</td><td>SHP_7</td><td>905.08</td><td>0.09</td><td>22</td><td>127.70</td><td></td></tr> <tr><td>3</td><td>MONICA FEDERLE</td><td>NUNAVUT</td><td>NUNAVUT</td><td>CORPORATE</td><td>Cust_6</td><td>Ord_6</td><td>Prod_1</td><td>SHP_8</td><td>2781.82</td><td>0.07</td><td>21</td><td>-695.26</td><td></td></tr> <tr><td>4</td><td>NICOLE HANSEN</td><td>NUNAVUT</td><td>NUNAVUT</td><td>SMALL BUSINESS</td><td>Cust_24</td><td>Ord_55</td><td>Prod_1</td><td>SHP_72</td><td>3338.98</td><td>0.00</td><td>40</td><td>-846.73</td><td></td></tr> <tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td></td></tr> <tr><td>8394</td><td>SHUI TOM</td><td>ALBERTA</td><td>WEST</td><td>HOME OFFICE</td><td>Cust_1816</td><td>Ord_5400</td><td>Prod_17</td><td>SHP_7540</td><td>5793.46</td><td>0.01</td><td>38</td><td>2477.77</td><td></td></tr> <tr><td>8395</td><td>FRANK HAWLEY</td><td>ALBERTA</td><td>WEST</td><td>HOME OFFICE</td><td>Cust_1817</td><td>Ord_5450</td><td>Prod_17</td><td>SHP_7690</td><td>12007.05</td><td>0.09</td><td>16</td><td>2713.95</td><td></td></tr> <tr><td>8396</td><td>AARON BERGMAN</td><td>ALBERTA</td><td>WEST</td><td>CORPORATE</td><td>Cust_1818</td><td>Ord_5485</td><td>Prod_17</td><td>SHP_7664</td><td>4233.15</td><td>0.08</td><td>35</td><td>1219.87</td><td></td></tr> <tr><td>8397</td><td>ADRIAN SHAMI</td><td>ALBERTA</td><td>WEST</td><td>CONSUMER</td><td>Cust_1820</td><td>Ord_5489</td><td>Prod_17</td><td>SHP_7671</td><td>754.92</td><td>0.00</td><td>7</td><td>-129.57</td><td></td></tr> <tr><td>8398</td><td>HARRY GREENE</td><td>ALBERTA</td><td>WEST</td><td>CORPORATE</td><td>Cust_1830</td><td>Ord_5504</td><td>Prod_17</td><td>SHP_7696</td><td>10071.09</td><td>0.10</td><td>41</td><td>1977.69</td><td></td></tr> </tbody> </table> <p>8399 rows x 16 columns</p>		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	
	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																																																																																																																																																													
	<table border="1"> <thead> <tr> <th>Discount</th><th>Order_Quantity</th><th>Profit</th><th>Shipping_Cost</th><th>Product_Base_Margin</th><th>Product_Category</th><th>Product_Sub_Category</th></tr> </thead> <tbody> <tr><td>0.04</td><td>6</td><td>-213.25</td><td>35.00</td><td>0.80</td><td>OFFICE SUPPLIES</td><td>STORAGE & ORGANIZATION</td></tr> <tr><td>0.04</td><td>28</td><td>-5.45</td><td>4.51</td><td>0.59</td><td>OFFICE SUPPLIES</td><td>STORAGE & ORGANIZATION</td></tr> <tr><td>0.09</td><td>22</td><td>127.70</td><td>6.22</td><td>NaN</td><td>OFFICE SUPPLIES</td><td>STORAGE & ORGANIZATION</td></tr> <tr><td>0.07</td><td>21</td><td>-695.26</td><td>35.00</td><td>NaN</td><td>OFFICE SUPPLIES</td><td>STORAGE & ORGANIZATION</td></tr> <tr><td>0.00</td><td>40</td><td>-846.73</td><td>35.00</td><td>0.81</td><td>OFFICE SUPPLIES</td><td>STORAGE & ORGANIZATION</td></tr> <tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr> <tr><td>0.01</td><td>38</td><td>2477.77</td><td>13.99</td><td>0.38</td><td>TECHNOLOGY</td><td>OFFICE MACHINES</td></tr> <tr><td>0.09</td><td>16</td><td>2713.95</td><td>55.30</td><td>0.40</td><td>TECHNOLOGY</td><td>OFFICE MACHINES</td></tr> <tr><td>0.08</td><td>35</td><td>1219.87</td><td>26.30</td><td>0.38</td><td>TECHNOLOGY</td><td>OFFICE MACHINES</td></tr> <tr><td>0.00</td><td>7</td><td>-129.57</td><td>19.99</td><td>0.52</td><td>TECHNOLOGY</td><td>OFFICE MACHINES</td></tr> <tr><td>0.10</td><td>41</td><td>1977.69</td><td>17.86</td><td>0.58</td><td>TECHNOLOGY</td><td>OFFICE MACHINES</td></tr> </tbody> </table>	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																																																																																				
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.																																																																																																																																																																								

<p>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.</p> <pre>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.java:45)</pre>	<p>Problems when I tried to start the spark session in Python.</p> <pre># Read customer.csv customer_df = spark.read.csv("C:\\Users\\PatiR\\Desktop\\\\6004 - Data Analytics\\6004787 Advanced Data Engineering\\Assessment_2\\customer.csv", header=True, inferSchema=True) #+py4jJavaError: line 2 # Cell In[7], line 2 # 1 # Read customer.csv #-> 2 customer_df = spark.read.csv("C:\\Users\\PatiR\\Desktop\\\\6004 - Data Analytics\\6004787 Advanced Data Engineering\\Assessment_2\\customer.csv", header=True, inferSchema=True) # File ~/anaconda3/lib/python/site-packages/pyspark/sql/session.py:1706, in SparkSession.read(self) 1699 property 1700 def read(self) -> DataFrameReader: 1701 """ Returns a <class 'DataFrameReader'> that can be used to read data 1702 in as a <class 'DataFrame'>. 1703 """ 1704 +-----+ 1705 ... 1706 return DataFrameReader(self) # File ~/anaconda3/lib/python/site-packages/pyspark/sql/readwriter.py:70, in DataFrameReader.__init__(self, spark) 69 def __init__(self, spark: "SparkSession"): --> 70 self._reader = spark._jsparkSession.read() 71 self._spark = spark # File ~/anaconda3/lib/python/site-packages/pyspark/jvm_gateway.py:1322, in JavaMember.__call__(self, *args) 1310 command = proto.CALL_COMMAND_NAME +: 1311 self.command_header + 1312 args_command + 1313 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"):</pre>
<p>Loading CSV with spark I got this answer.</p> <p>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.</p>	

Part B: Big Data Analysis and Application of Engineering Techniques

Task 1	
	<p>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.).</p>
	<pre>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:/# </pre> <p>I made this folder transaction in hdfs in docker-Hadoop.</p> <pre>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 container 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> </pre> <p>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.</p> <p>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.</p> <pre>root@5b57735ac4de:/# ls -l /tmp total 616 drwxr-xr-x 3 root root 4096 Aug 14 23:25 Jetty_0_0_0_50070_hdfs____w2cu08 drwxr-xr-x 1 root root 4096 Aug 14 23:47 hsuperdata_root -rwxr-xr-x 1 root root 619850 Jul 9 08:07 market.csv root@5b57735ac4de:/# </pre> <pre>drwxr-xr-x 3 root root 4096 Aug 14 23:25 Jetty_0_0_0_50070_hdfs____w2cu08 drwxr-xr-x 1 root root 4096 Aug 14 23:47 hsuperdata_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:/# </pre>

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.																																																
	<pre>df = pd.read_csv("C:\\\\Users\\\\patir\\\\project_jupyter\\\\online_retail_II.csv")</pre> <pre>df.head()</pre> <table border="1"> <thead> <tr> <th>Invoice</th><th>StockCode</th><th>Description</th><th>Quantity</th><th>InvoiceDate</th><th>Price</th><th>Customer ID</th><th>Country</th></tr> </thead> <tbody> <tr> <td>0</td><td>489434</td><td>85048 15CM CHRISTMAS GLASS BALL 20 LIGHTS</td><td>12</td><td>2009-12-01 07:45:00</td><td>6.95</td><td>13085.0</td><td>United Kingdom</td></tr> <tr> <td>1</td><td>489434</td><td>79323P PINK CHERRY LIGHTS</td><td>12</td><td>2009-12-01 07:45:00</td><td>6.75</td><td>13085.0</td><td>United Kingdom</td></tr> <tr> <td>2</td><td>489434</td><td>79323W WHITE CHERRY LIGHTS</td><td>12</td><td>2009-12-01 07:45:00</td><td>6.75</td><td>13085.0</td><td>United Kingdom</td></tr> <tr> <td>3</td><td>489434</td><td>22041 RECORD FRAME 7" SINGLE SIZE</td><td>48</td><td>2009-12-01 07:45:00</td><td>2.10</td><td>13085.0</td><td>United Kingdom</td></tr> <tr> <td>4</td><td>489434</td><td>21232 STRAWBERRY CERAMIC TRINKET BOX</td><td>24</td><td>2009-12-01 07:45:00</td><td>1.25</td><td>13085.0</td><td>United Kingdom</td></tr> </tbody> </table> <p>Loading data set online retail II and show columns</p> <pre>df.info()</pre> <pre><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</pre> <p>Information about columns, types, and ranges.</p>	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
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																																										

```
#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_MongoDB'), '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,
 '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'}]
```

Print on register ready to save.

localhost:27017 > gdda707_MongoDB > gdda707_MongoDB_collection_v3

Documents 1.1M Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) [Find](#) [Explain](#) [Reset](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

1–20 of 1067371 [<](#) [>](#)

```
_id: ObjectId('66b96282b3de891b5fa79418')
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
Country : "United Kingdom"

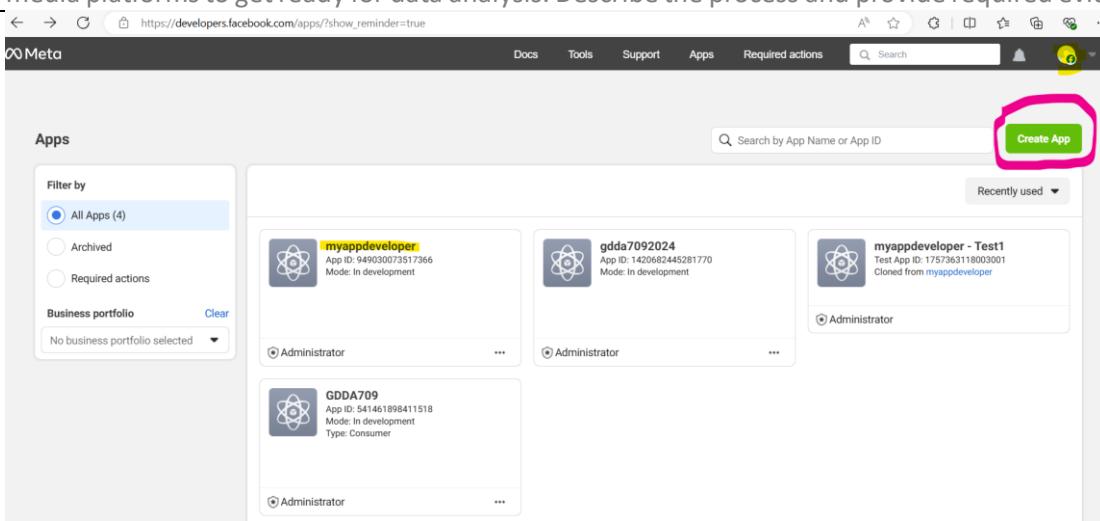
_id: ObjectId('66b96282b3de891b5fa79419')
index : 1
Invoice : "489434"
StockCode : "79323P"
Description : "PINK CHERRY LIGHTS"
Quantity : 12
InvoiceDate : "2009-12-01 07:45:00"
Price : 6.75
Customer ID : 13085
Country : "United Kingdom"

_id: ObjectId('66b96282b3de891b5fa7941a')
index : 2
Invoice : "489434"
StockCode : "79323M"
```

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

The screenshot shows the Facebook Graph API Explorer interface. At the top, there is a navigation bar with links to Docs, Tools, Support, Apps, Required actions, and a search bar. Below the navigation bar, the URL is set to `GET graph.facebook.com/v2.0/me?fields=id,name`. On the right side of the screen, there is a sidebar titled "Access Token" which contains fields for "Meta App" (set to "patir@facebook"), "User or Page" (set to "User Token"), and "Permissions" (set to "public_profile"). A large blue button labeled "Generate Access Token" is prominently displayed. The main panel of the explorer shows a JSON response with the user's ID and name.

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

This screenshot shows the same Facebook Graph API Explorer interface. The URL remains the same: `GET graph.facebook.com/v2.0/me?fields=id,name`. The main panel displays the JSON response from the previous step, showing the user's ID and name. The sidebar for "Access Token" is still visible on the right.

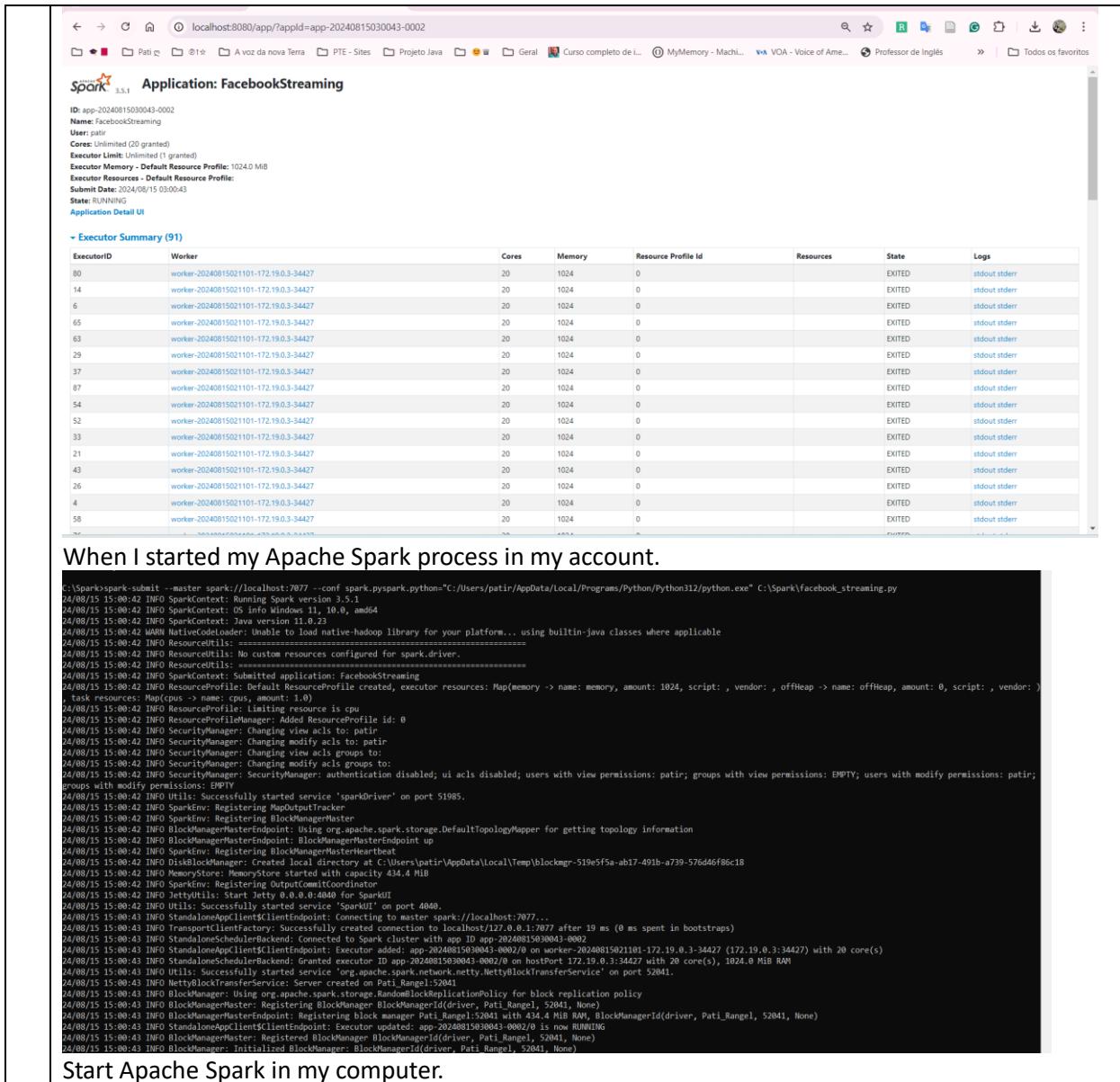
Shows me my ID and name in my account on Facebook.

The screenshot shows the Apache Spark master UI running at `spark://d4cb0bdcf4e7:7077`. The UI provides detailed information about the cluster, including the number of workers, memory usage, and applications. It also lists completed applications with their details. The browser tab title is "Spark Master at spark://d4cb0bdcf4e7:7077".

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240815021101-172.19.0.3-34427	(null) FacebookStreaming	20	1024.0 MiB		2024/08/15 03:00:43	patir	RUNNING	45 s

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240815024256-0001	test_spark	20	1024.0 MiB		2024/08/15 02:42:56	patir	FINISHED	0.4 s
app-20240815023405-0000	test_spark	20	1024.0 MiB		2024/08/15 02:34:05	patir	FINISHED	0.5 s

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



The screenshot shows the Apache Spark Application UI for the "FacebookStreaming" application. The UI includes:

- Application Detail UI:** Shows basic application information like ID, Name, User, Cores, Executor Limit, Executor Memory, Executor Resources, Submit Date, and State.
- Executor Summary (91):** A table listing 91 executors. Each row contains the ExecutorID (e.g., 80), Worker (e.g., worker-20240815021101-172.19.0.3-34427), Cores (20), Memory (1024), Resource Profile Id (0), Resources (EXITED), State (EXITED), and Logs (stdout stderr).
- Logs:** A large text area displaying the command-line output of the application. The logs show the submission of the application, the creation of executors, and various system messages related to security, memory, and network operations.

When I started my Apache Spark process in my account.

```
C:\Spark\spark-submit --master spark://localhost:7077 --conf spark.pyspark.python="C:/Users/patir/AppData/Local/Programs/Python/Python312/python.exe" C:/Spark/facebook_streaming.py
24/08/15 15:00:42 INFO SparkContext: Running Spark version 3.5.1
24/08/15 15:00:42 INFO SparkContext: OS info Windows 11, 10.0, amd64
24/08/15 15:00:42 INFO SparkContext: Java version 11.0.23
24/08/15 15:00:42 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/08/15 15:00:42 INFO ResourceFiles: =====
24/08/15 15:00:42 INFO ResourceFiles: No custom resources configured for spark.driver.
24/08/15 15:00:42 INFO ResourceFiles: =====
24/08/15 15:00:42 INFO SparkContext: Submitted application: FacebookStreaming
24/08/15 15:00:42 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpu -> name: cpus, amount: 1.0)
24/08/15 15:00:42 INFO ResourceProfile: limiting resource is cpu
24/08/15 15:00:42 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/08/15 15:00:42 INFO SecurityManager: Changing view acls to: patir
24/08/15 15:00:42 INFO SecurityManager: Changing modify acls to: patir
24/08/15 15:00:42 INFO SecurityManager: Changing view permissions to: patir
24/08/15 15:00:42 INFO SecurityManager: Changing modify permissions to: patir
24/08/15 15:00:42 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: patir; users with modify permissions: patir; groups with modify permissions: EMPTY
24/08/15 15:00:42 INFO Utils: Successfully started service 'sparkDriver' on port 51985.
24/08/15 15:00:42 INFO SparkEnv: Registering MapOutputTracker
24/08/15 15:00:42 INFO SparkEnv: Registering BlockManagerMasterEndpoint
24/08/15 15:00:42 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/08/15 15:00:42 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/08/15 15:00:42 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/08/15 15:00:42 INFO DiskBlockManager: Created local direct at C:/Users/patir/AppData/Local/Temp/blockmgr-519e5fa-ab17-491b-a739-576d46f86c18
24/08/15 15:00:42 INFO MemoryStore: MemoryStore started with capacity 434.4 MB
24/08/15 15:00:42 INFO SparkEnv: Registering OutputCommitCoordinator
24/08/15 15:00:42 INFO TransportClientFactory: Created transport client for SparkUI
24/08/15 15:00:42 INFO Utils: Successfully started service 'SparkUI' on port 4040.
24/08/15 15:00:43 INFO StandaloneAppClient$ClientEndpoint: Connecting to master spark://localhost:7077...
24/08/15 15:00:43 INFO TransportClientFactory: Successfully created connection to localhost/127.0.0.1:7077 after 19 ms (0 ms spent in bootstraps)
24/08/15 15:00:43 INFO StandaloneSchedulerBackend: Connected to Spark cluster with app ID app-20240815030043-0002
24/08/15 15:00:43 INFO StandaloneAppClient$ClientEndpoint: Executor added: app-20240815030043-0002/0 on worker-20240815021101-172.19.0.3-34427 ((172.19.0.3:34427) with 20 core(s))
24/08/15 15:00:43 INFO StandaloneSchedulerBackend: Granted executor ID app-20240815030043-0002/0 on hostPort 172.19.0.3:34427 with 20 core(s), 1024.0 MiB RAM
24/08/15 15:00:43 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 52041.
24/08/15 15:00:43 INFO NettyBlockTransferService: Server created on Pati_Rangel52041
24/08/15 15:00:43 INFO BlockManagerMasterEndpoint: Registering block storage location policy for block replication policy
24/08/15 15:00:43 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, Pati_Rangel, 52041, None)
24/08/15 15:00:43 INFO BlockManagerMasterEndpoint: Registering block manager Pati_Rangel:52041 with 434.4 MiB RAM, BlockManagerId(driver, Pati_Rangel, 52041, None)
24/08/15 15:00:43 INFO StandaloneAppClient$ClientEndpoint: Executor updated: app-20240815030043-0002/0 is now RUNNING
24/08/15 15:00:43 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, Pati_Rangel, 52041, None)
24/08/15 15:00:43 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, Pati_Rangel, 52041, None)
```

Start Apache Spark in my computer.

	<pre> 24/08/15 15:03:49 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/177 is now EXITED (Command exited with code 1) 24/08/15 15:03:49 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/177 removed: Command exited with code 1 24/08/15 15:03:49 INFO StandaloneSchedulerBackend: Removal of executor 177 requested 24/08/15 15:03:49 INFO BlockManagerMasterEndpoint: Removal of executor 177 requested 24/08/15 15:03:49 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/178 on worker-20240815021101-172.19.0.3:34427 with 20 core(s) 24/08/15 15:03:49 INFO StandaloneSchedulerBackend\$StandaloneDriverEndpoint: Asked to remove non-existent executor 178 24/08/15 15:03:49 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/178 removed: Command exited with code 1 24/08/15 15:03:49 INFO BlockManagerMasterEndpoint: Removal of executor 178 requested 24/08/15 15:03:50 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/179 on worker-20240815021101-172.19.0.3:34427 (172.19.0.3:34427) with 20 core(s) 24/08/15 15:03:50 INFO StandaloneSchedulerBackend\$StandaloneDriverEndpoint: Asked to remove non-existent executor 179 24/08/15 15:03:50 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/179 removed: Command exited with code 1 24/08/15 15:03:50 INFO BlockManagerMasterEndpoint: Removal of executor 179 requested 24/08/15 15:03:50 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/180 on worker-20240815021101-172.19.0.3:34427 (172.19.0.3:34427) with 20 core(s) 24/08/15 15:03:50 INFO BlockManagerMasterEndpoint: Removal of executor 180 requested 24/08/15 15:03:51 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/180 removed: Command exited with code 1 24/08/15 15:03:51 INFO BlockManagerMasterEndpoint: Removal of executor 180 requested 24/08/15 15:03:52 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/181 on worker-20240815021101-172.19.0.3:34427 (172.19.0.3:34427) with 20 core(s) 24/08/15 15:03:52 INFO BlockManagerMasterEndpoint: Removal of executor 181 requested 24/08/15 15:03:53 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/181 removed: Command exited with code 1 24/08/15 15:03:53 INFO BlockManagerMasterEndpoint: Removal of executor 181 requested 24/08/15 15:03:54 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/182 on worker-20240815021101-172.19.0.3:34427 (172.19.0.3:34427) with 20 core(s) 24/08/15 15:03:54 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/182 removed: Command exited with code 1 24/08/15 15:03:54 INFO BlockManagerMasterEndpoint: Removal of executor 182 requested 24/08/15 15:03:54 INFO StandaloneAppClient\$ClientEndpoint: Executor added: app-20240815030043-0002/183 on worker-20240815021101-172.19.0.3:34427 (172.19.0.3:34427) with 20 core(s) 24/08/15 15:03:54 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240815030043-0002/183 is now RUNNING </pre>
Running Apache Spark.	

	<h3>Task 4</h3> <p>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.</p>
	<pre> @echo off cd C:\kafka call .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties </pre> <p>Zookeeper.properties</p> <pre> @echo off cd C:\kafka call .\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic kafkaLab --from-beginning </pre> <p>Start_consumer.bat</p> <pre> @echo off cd C:\kafka call .\bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --topic kafkaLab </pre> <p>Start_kafka.bat</p>

Start kafka.

```
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, 784] 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 19:42:56.002] INFO zookeeper.maxWriteQueuePollTime = 0 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56.002] INFO zookeeper.maxBatchSize=1000 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56.002] INFO zookeeper.inByteBufferStartingSizeBytes = 1024 (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56.003] INFO Weighted connection throttling is disabled (org.apache.zookeeper.server.BlueThrottle)
2024-08-14 19:42:56.004] INFO minSessionTimeout set to 60000 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56.004] INFO maxSessionTimeout set to 60000 ms (org.apache.zookeeper.server.ZooKeeperServer)
2024-08-14 19:42:56.004] INFO getdata response cache size is initialized with value 400. (org.apache.zookeeper.server.ResponseCache)
2024-08-14 19:42:56.004] 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 60000 ms maxSessionTimeout 60000 ms clientPortListenBacklog -1 datadir ..\kafka\zookeeper-data\version-2 snapdir C:\kafka\zookeeper-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.ServerCnxnFactory)
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\zookeeper-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.DataFree)
2024-08-14 19:42:56.100] INFO Snapshot loaded in 17 ms, highest xid is 0x0, digest is 1371985504 (org.apache.zookeeper.server.ZKDatabase)
2024-08-14 19:42:56.102] INFO Snapshottting: 0x0 to C:\kafka\zookeeper-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 PreRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PreRequestProcessor)
2024-08-14 19:42:56.122] INFO Using checkIntervalMs=50000 maxPerMinute=10000 maxNeverUsedIntervalNs=0 (org.apache.zookeeper.server.ContainerManager)
2024-08-14 19:42:56.123] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
```


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 6a838fc1d6e ⓘ	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.

gdda707_MongoDB_collection_v3

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
56.25 MB	1.1 M	233.00 B	1	11.46 MB

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 screenshot shows the Facebook developer console under the 'myappdeveloper' app. The left sidebar has tabs for Dashboard, Required actions, Use cases, Review (selected), Testing (highlighted with a yellow box), Verification, App Review, Publish, and Unpublished. The main content area is titled 'Testing your use cases' and includes sections for 'Using Graph API Explorer' (with a list of benefits) and 'Authenticate and request data from users with Facebook Login' (status: 'Testing complete'). Below these are sections for 'Authentication and account creation' (status: 'Completed') and 'public_profile'. A yellow circle highlights the 'Open Graph API Explorer' button.

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
...	...
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

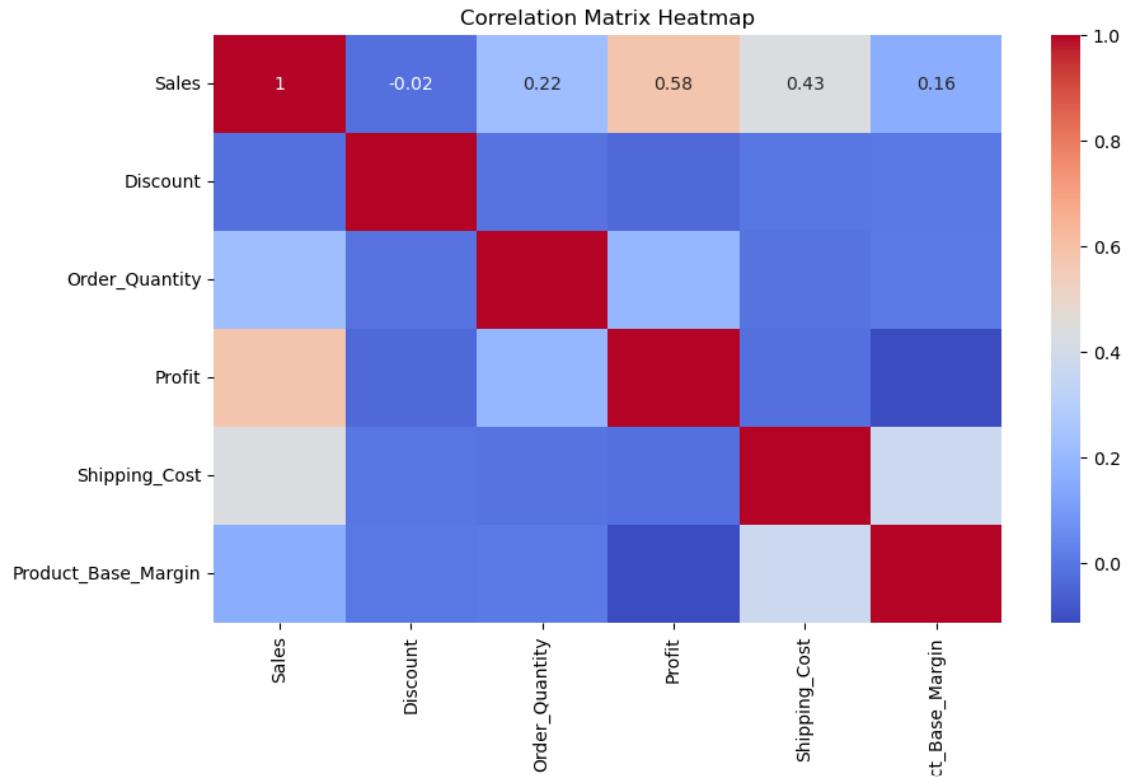
We have the correlation output of the variables below along with the values and the graph. And it shows us the relationship between the variables.

```
# 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)
```

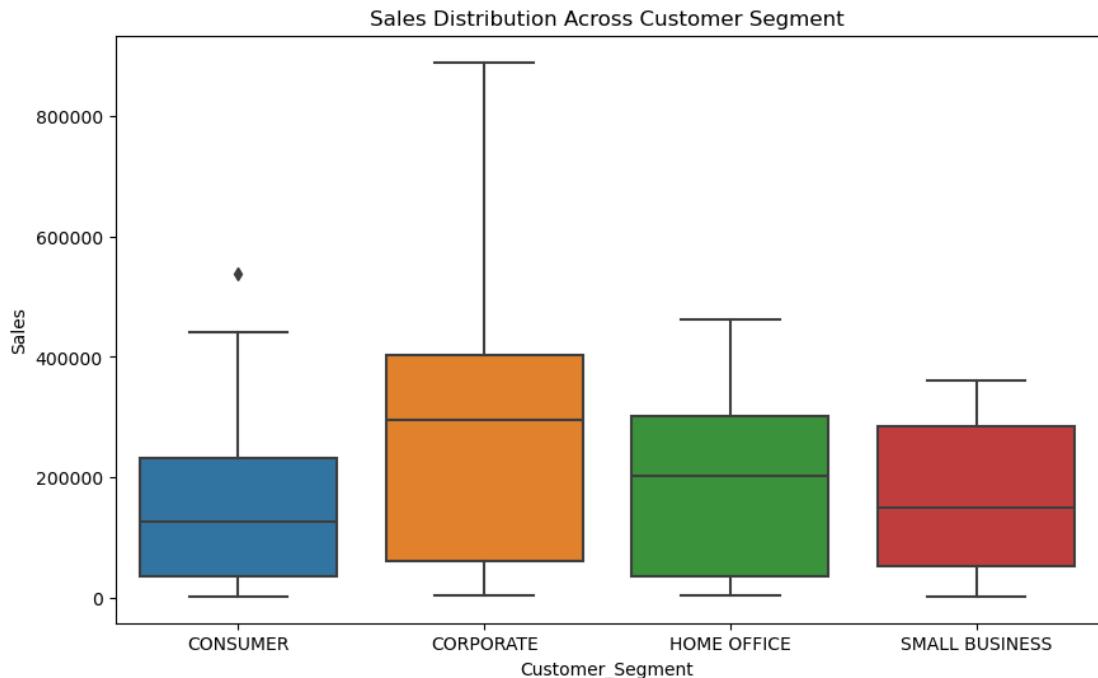
Another filter where we can analyse a grouping of customer segments, product categories, and product sub category with the sum of sales. Presenting the output below:

```
Product categories sold by Segment:
   Customer_Segment Product_Category      Product_Sub_Category \
0           CONSUMER      FURNITURE             BOOKCASES
1           CONSUMER      FURNITURE        CHAIRS & CHAIMATS
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
```

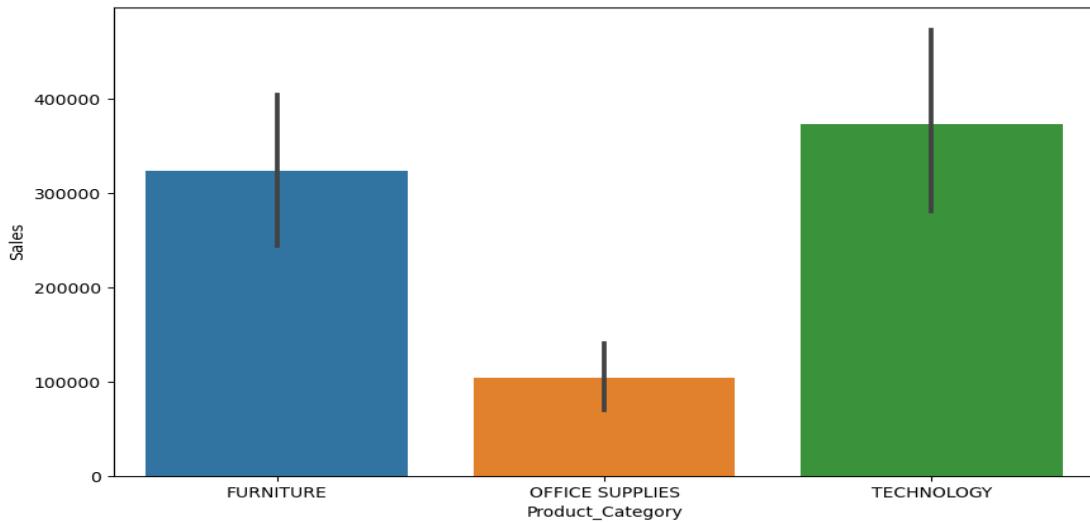
Another view where we can analyze customer segment and sales and their distribution by customer product is that corporate has the highest performance in online shopping sales.

```
# 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}")
```

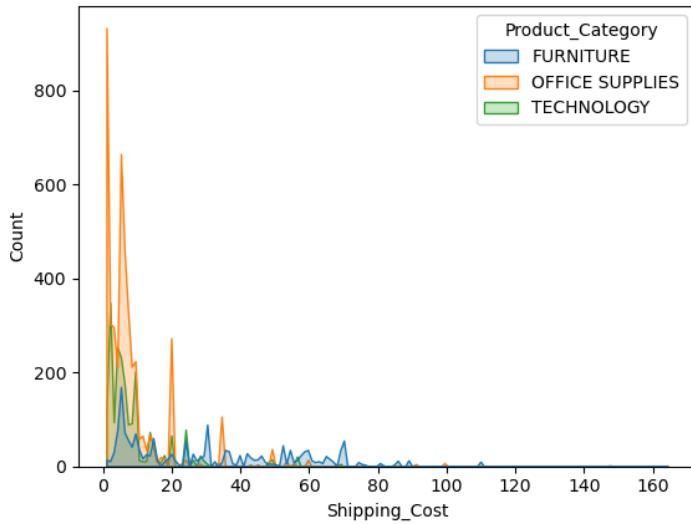
Product Category with the highest average sales: TECHNOLOGY

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 and R-squared
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)

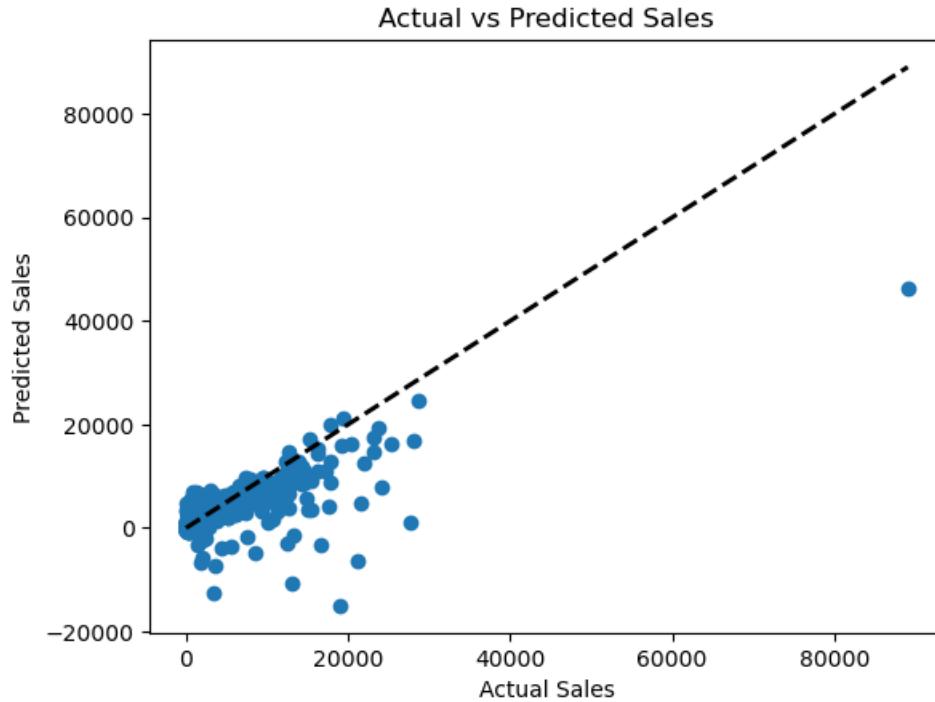
# Plot the best fit line
plt.scatter(y_test, y_pred)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2) # Plot the diagonal line
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.show()

```

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    18:48           487 docker-compose_spark.yml
-a----       09/07/2024    20:07        619856 market.csv
-a----       31/07/2024    18:06      969924 online_shopping.csv
-a----       09/07/2024    20:07           797 product.csv

PS C:\Users\patir\docker-hadoop> # Copie o arquivo para o container
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.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-r--r--r--	root	supergroup	605.32 KB	15/08/2024, 11:49:45	1	128 MB	market.csv
-r--r--r--	root	supergroup	887.72 KB	15/08/2024, 12:01:11	1	128 MB	online_shopping.csv
-r--r--r--	root	supergroup	797 B	15/08/2024, 11:59:54	1	128 MB	product.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

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_MongoDB'), '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,
 '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'}]
```

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.

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.

gdda707_MongoDB_collection_v3

Storage size: 56.25 MB	Documents: 1.1M	Avg. document size: 233.00 B	Indexes: 1	Total index size: 11.46 MB
---------------------------	--------------------	---------------------------------	---------------	-------------------------------

localhost:27017 > gdda707_MongoDB > gdda707_MongoDB_collection_v3

Documents (1.1M) Aggregations Schema Indexes (1) Validation

Type a query: { field: 'value' } or [Generate query](#)

[Explain](#) [Reset](#) [Find](#) [Options](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

1 – 20 of 1067371

```
_id: ObjectId('66b96282b3de891b5fa79418')
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
Country : "United Kingdom"
```

```
_id: ObjectId('66b96282b3de891b5fa79419')
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
Country : "United Kingdom"
```

```
_id: ObjectId('66b96282b3de891b5fa7941a')
index : 2
Invoice : "489434"
StockCode : "79323W"
```

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.

The screenshot shows the Facebook Developers App dashboard. On the left, there's a sidebar with 'Filter by' options: 'All Apps (4)' (selected), 'Archived', and 'Required actions'. Below that is a 'Business portfolio' section with a dropdown set to 'No business portfolio selected'. The main area displays four app cards:

- myappdeveloper**: App ID: 949030073517366, Mode: In development, Administrator.
- gdda7092024**: App ID: 1420682445281770, Mode: In development, Administrator.
- myappdeveloper - Test1**: Test App ID: 1757363118003001, Cloned from myappdeveloper, Administrator.
- GDDA709**: App ID: 541461898411518, Mode: In development, Type: Consumer, Administrator.

A green rectangular button labeled 'Create App' is located in the top right corner of the dashboard area, and it is circled with a pink oval.

A tela do APPs com o meu login do Facebook

The screenshot shows the Graph API Explorer interface. At the top, there's a search bar and a 'Create Access Token' button. Below the search bar, there are dropdown menus for 'GET', 'graph.facebook.com', 'v2.0.0', and '/me?fields=id,name'. A modal window is open in the center, prompting the user to 'Resize the columns by dragging, or flip the position using alt + left click'. To the right of the modal, there's an 'Access Token' configuration panel. It includes fields for 'Meta App' (set to 'myappdeveloper'), 'User or Page' (set to 'User Token'), and 'Permissions' (set to 'public_profile'). There are also buttons for 'Generate Access Token', 'Copy Debug Information', 'Get Code', and 'Save Session'.

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 = '██████████' # 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 = '████████████████████████████████████████████████████████████████████████████████████████'

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

localhost 7077 spark.

The screenshot shows the Spark Master UI at spark://d4cb0bdcf4e7:7077. It displays the following information:

- URL:** spark://d4cb0bdcf4e7:7077
- Alive Workers:** 1
- Cores in use:** 20 Total, 20 Used
- Memory in use:** 14.4 GiB Total, 1024.0 MiB Used
- Resources in use:**
- Applications:** 1 Running, 2 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20240815021101-172.19.0.3-34427	172.19.0.3:34427	ALIVE	20 (20 Used)	14.4 GiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240815030043-0002	(kill) FacebookStreaming	20	1024.0 MiB		2024/08/15 03:00:43	patr	RUNNING	45 s

Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240815024256-0001	test_spark	20	1024.0 MiB		2024/08/15 02:42:56	patr	FINISHED	0.4 s
app-20240815023405-0000	test_spark	20	1024.0 MiB		2024/08/15 02:34:05	patr	FINISHED	0.5 s

After running the application.

The screenshot shows the Application Detail UI for FacebookStreaming at localhost:8080/app?appId=app-20240815030043-0002. It displays the following information:

- ID:** app-20240815030043-0002
- Name:** FacebookStreaming
- User:** patr
- Cores:** Unlimited (20 granted)
- Executor Limit:** Unlimited (1 granted)
- Executor Memory - Default Resource Profile:** 1024.0 MiB
- Executor Resources - Default Resource Profile:**
- Submit Date:** 2024/08/15 03:00:43
- State:** RUNNING
- Application Detail UI**

Executor Summary (91)

ExecutorID	Worker	Cores	Memory	Resource Profile Id	Resources	State	Logs
80	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
14	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
6	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
65	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
63	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
29	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
37	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
87	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
54	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
52	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
33	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
21	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
43	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
26	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
4	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr
58	worker-20240815021101-172.19.0.3-34427	20	1024	0		EXITED	stdout stderr

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 kafka --partitions 1 --replication-factor 1
```

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.Q
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.QuorumPeerM
ode)
[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.Q
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.zooke
per.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 19:42:56,038] 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=3.8.3-6ad6d364c7c0bcf0de45d54ebefab3058098ab56, built on 2023-10-05 10:34 UTC (org.apa
e.zookeeper.server.ZooKeeperServer)
[2024-08-14 19:42:56,040] INFO Server environment:host.name=Pati_Rangel_znsce.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.jar;C:\kafka\libs\apolliance-repackaged-2.6.1.jar;C:\kaf
```

```

[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 20:19:42,784] 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.tools.TopicCommand)
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

```

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, HDFS 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/>

GitHub

[GDDA707 A2 Online Shopping 850001348](#)