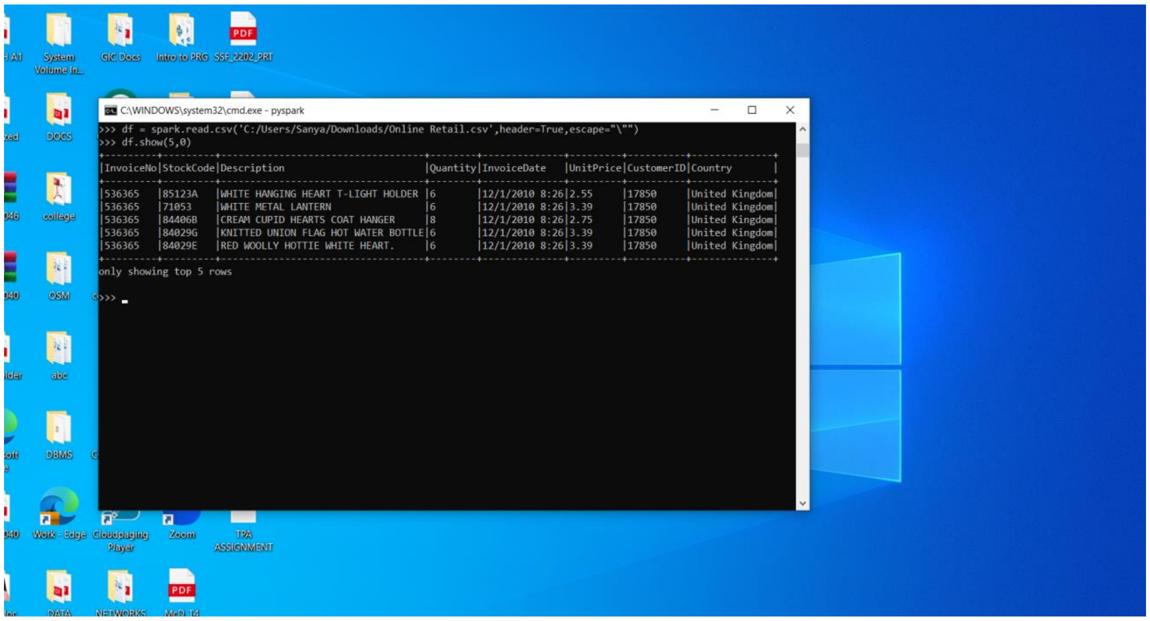
Machine Learning with PySpark: Customer Segmentation

SCREENSHOTS:

TASK 1: Creating a SparkSession

TASK 2: CREATING A DATA FRAME

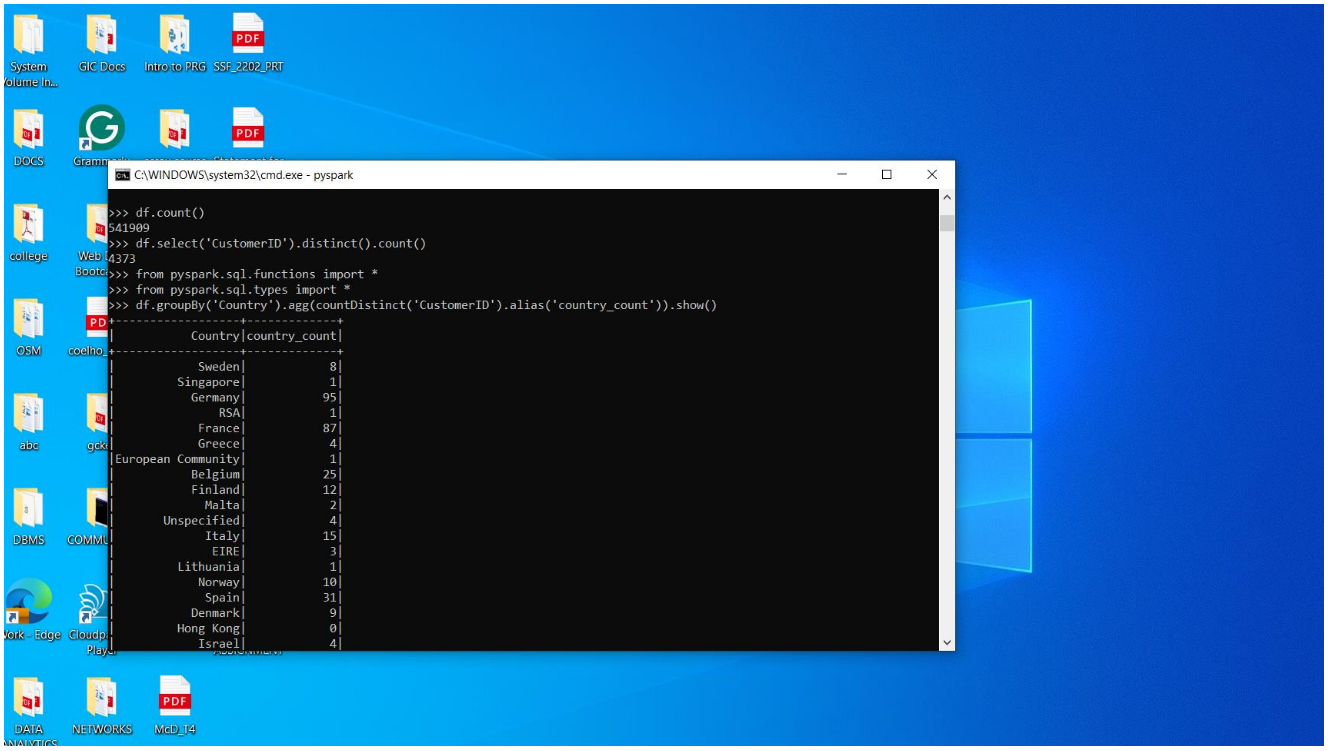


```
>>> df = spark.read.csv('C:/Users/Sanya/Downloads/Online Retail.csv',header=True,escape="\"")
>>> df.show(5,0)
+---------+---------+-----------------------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|Description                        |Quantity|InvoiceDate   |UnitPrice|CustomerID|Country       |
+---------+---------+-----------------------------------+--------+--------------+---------+----------+--------------+
|536365   |85123A   |WHITE HANGING HEART T-LIGHT HOLDER |6       |12/1/2010 8:26|2.55     |17850     |United Kingdom|
|536365   |71053    |WHITE METAL LANTERN                |6       |12/1/2010 8:26|3.39     |17850     |United Kingdom|
|536365   |84406B   |CREAM CUPID HEARTS COAT HANGER     |8       |12/1/2010 8:26|2.75     |17850     |United Kingdom|
|536365   |84029G   |KNITTED UNION FLAG HOT WATER BOTTLE|6       |12/1/2010 8:26|3.39     |17850     |United Kingdom|
|536365   |84029E   |RED WOOLLY HOTTIE WHITE HEART.     |6       |12/1/2010 8:26|3.39     |17850     |United Kingdom|
+---------+---------+-----------------------------------+--------+--------------+---------+----------+--------------+
only showing top 5 rows

>>> 
```

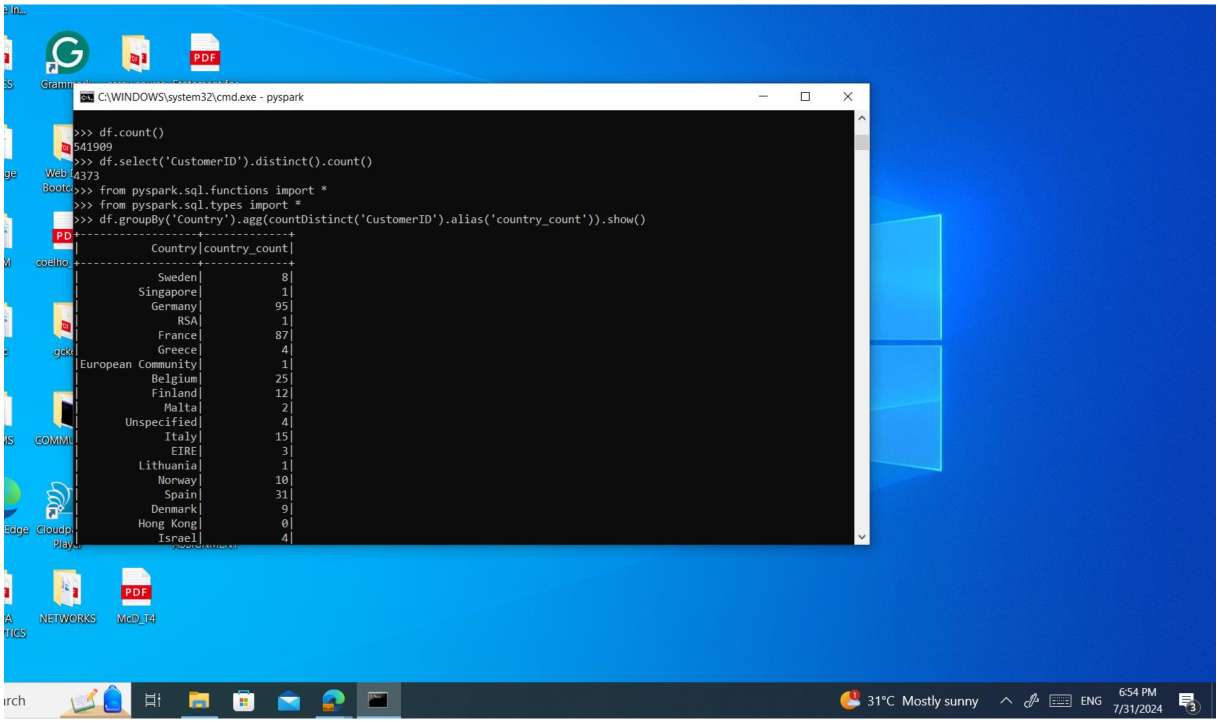
Task 3: Exploratory Data Analysis

1. **Counting the number of rows**
2. **Counting unique customers**

3. Country which purchases the maximum quantity

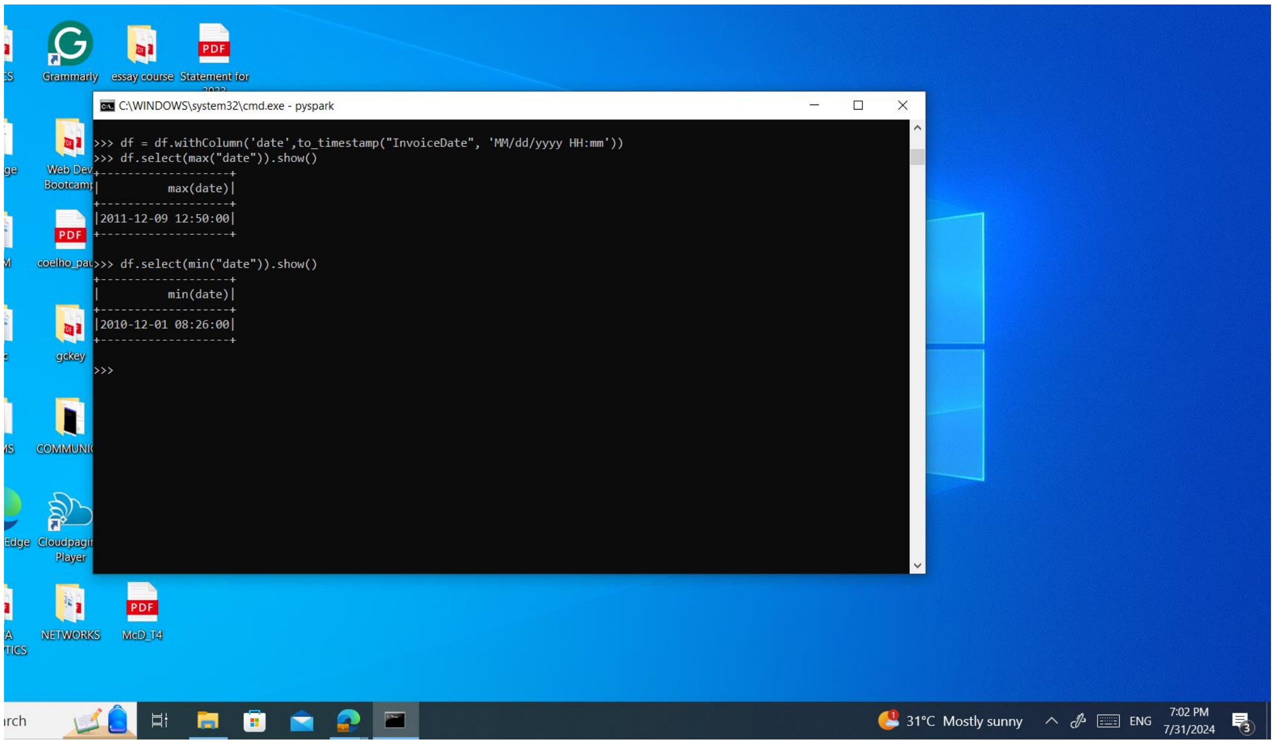


```
>>> df.count()
541909
>>> df.select('CustomerID').distinct().count()
4373
>>> from pyspark.sql.functions import *
>>> from pyspark.sql.types import *
>>> df.groupBy('Country').agg(countDistinct('CustomerID').alias('country_count')).show()
+------------------+-------------+
|           Country|country_count|
+------------------+-------------+
|            Sweden|            8|
|         Singapore|            1|
|           Germany|           95|
|               RSA|            1|
|            France|           87|
|            Greece|            4|
|European Community|            1|
|           Belgium|           25|
|           Finland|           12|
|             Malta|            2|
|       Unspecified|            4|
|             Italy|           15|
|              EIRE|            3|
|         Lithuania|            1|
|            Norway|           10|
|             Spain|           31|
|           Denmark|            9|
|         Hong Kong|            0|
|            Israel|            4|
```



```
>>> df.count()
541909
>>> df.select('CustomerID').distinct().count()
4373
>>> from pyspark.sql.functions import *
>>> from pyspark.sql.types import *
>>> df.groupBy('Country').agg(countDistinct('CustomerID').alias('country_count')).show()
+------------------+-------------+
|           Country|country_count|
+------------------+-------------+
|            Sweden|            8|
|         Singapore|            1|
|           Germany|           95|
|               RSA|            1|
|            France|           87|
|            Greece|            4|
|European Community|            1|
|           Belgium|           25|
|           Finland|           12|
|             Malta|            2|
|       Unspecified|            4|
|             Italy|           15|
|              EIRE|            3|
|         Lithuania|            1|
|            Norway|           10|
|             Spain|           31|
|           Denmark|            9|
|         Hong Kong|            0|
|            Israel|            4|
```

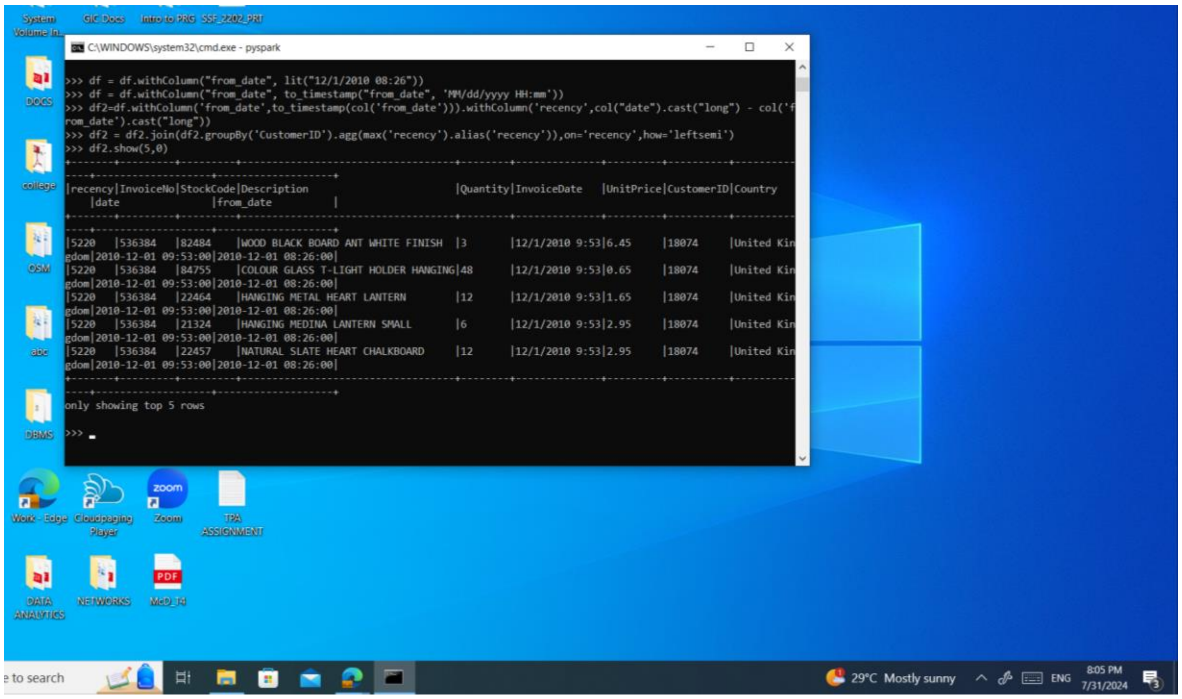
MOST RECENT PURCHASE AND MINIMUM DATE



```
>>> df = df.withColumn('date',to_timestamp("InvoiceDate", 'MM/dd/yyyy HH:mm'))
>>> df.select(max("date")).show()
+-------------------+
|          max(date)|
+-------------------+
|2011-12-09 12:50:00|
+-------------------+

>>> df.select(min("date")).show()
+-------------------+
|          min(date)|
+-------------------+
|2010-12-01 08:26:00|
+-------------------+

>>>
```

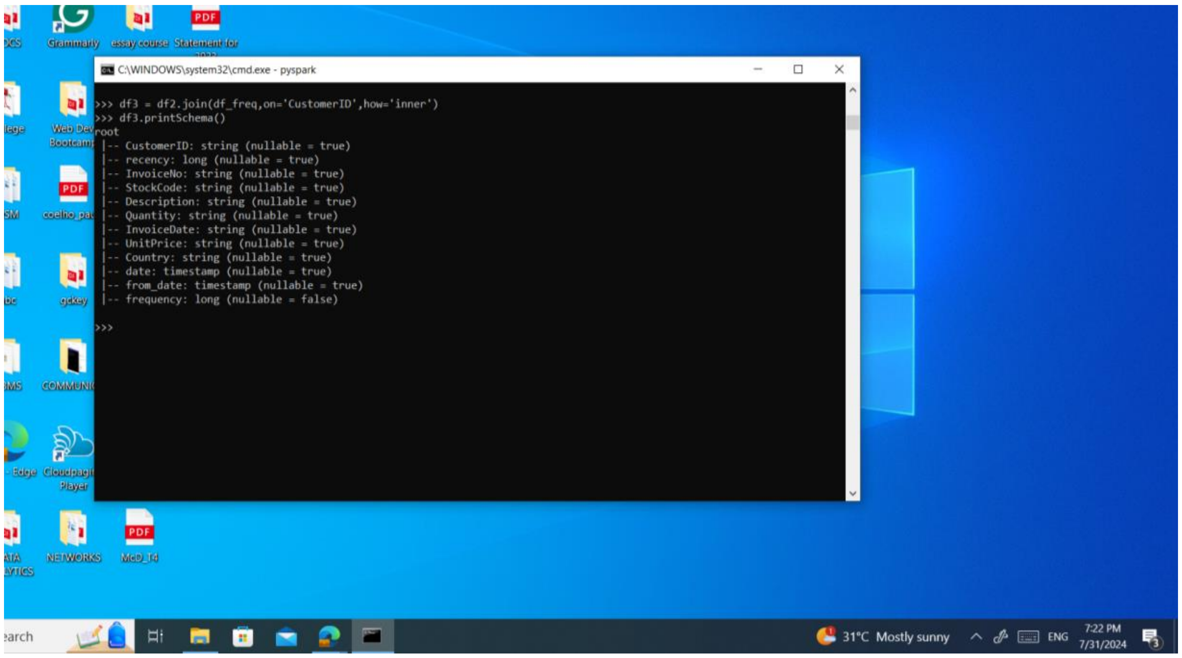
TASK 4: DATA PRE-PROCESSING – RFM

RECENCY



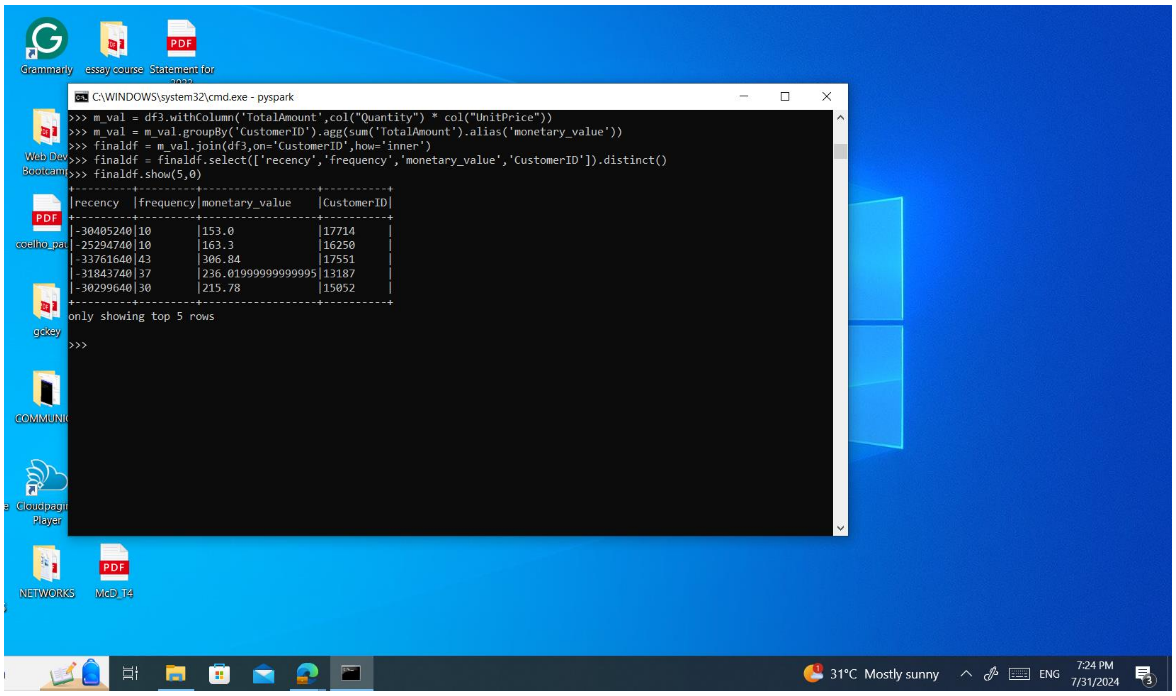
```
>>> df = df.withColumn("from_date", lit("12/1/2010 08:26"))
>>> df = df.withColumn("from_date", to_timestamp("from_date", 'MM/dd/yyyy HH:mm'))
>>> df2=df.withColumn('from_date',to_timestamp(col('from_date'))).withColumn('recency',col("date").cast("long") - col('f
rom_date').cast("long"))
>>> df2 = df2.join(df2.groupBy('CustomerID').agg(max('recency').alias('recency')),on='recency',how='leftsemi')
>>> df2.show(5,0)
+-------+---------+---------+-----------------------------------+--------+---------------+---------+----------+--------
----+-------------------+-------------------+
|recency|InvoiceNo|StockCode|Description                        |Quantity|InvoiceDate    |UnitPrice|CustomerID|Country
    |date               |from_date          |
+-------+---------+---------+-----------------------------------+--------+---------------+---------+----------+--------
----+-------------------+-------------------+
|5220   |536384   |82484    |WOOD BLACK BOARD ANT WHITE FINISH  |3       |12/1/2010 9:53|6.45     |18074     |United Kin
gdom|2010-12-01 09:53:00|2010-12-01 08:26:00|
|5220   |536384   |84755    |COLOUR GLASS T-LIGHT HOLDER HANGING|48      |12/1/2010 9:53|0.65     |18074     |United Kin
gdom|2010-12-01 09:53:00|2010-12-01 08:26:00|
|5220   |536384   |22464    |HANGING METAL HEART LANTERN        |12      |12/1/2010 9:53|1.65     |18074     |United Kin
gdom|2010-12-01 09:53:00|2010-12-01 08:26:00|
|5220   |536384   |21324    |HANGING MEDINA LANTERN SMALL       |6       |12/1/2010 9:53|2.95     |18074     |United Kin
gdom|2010-12-01 09:53:00|2010-12-01 08:26:00|
|5220   |536384   |22457    |NATURAL SLATE HEART CHALKBOARD     |12      |12/1/2010 9:53|2.95     |18074     |United Kin
gdom|2010-12-01 09:53:00|2010-12-01 08:26:00|
+-------+---------+---------+-----------------------------------+--------+---------------+---------+----------+--------
----+-------------------+-------------------+
only showing top 5 rows

>>>
```

PRINTING SCHEMA



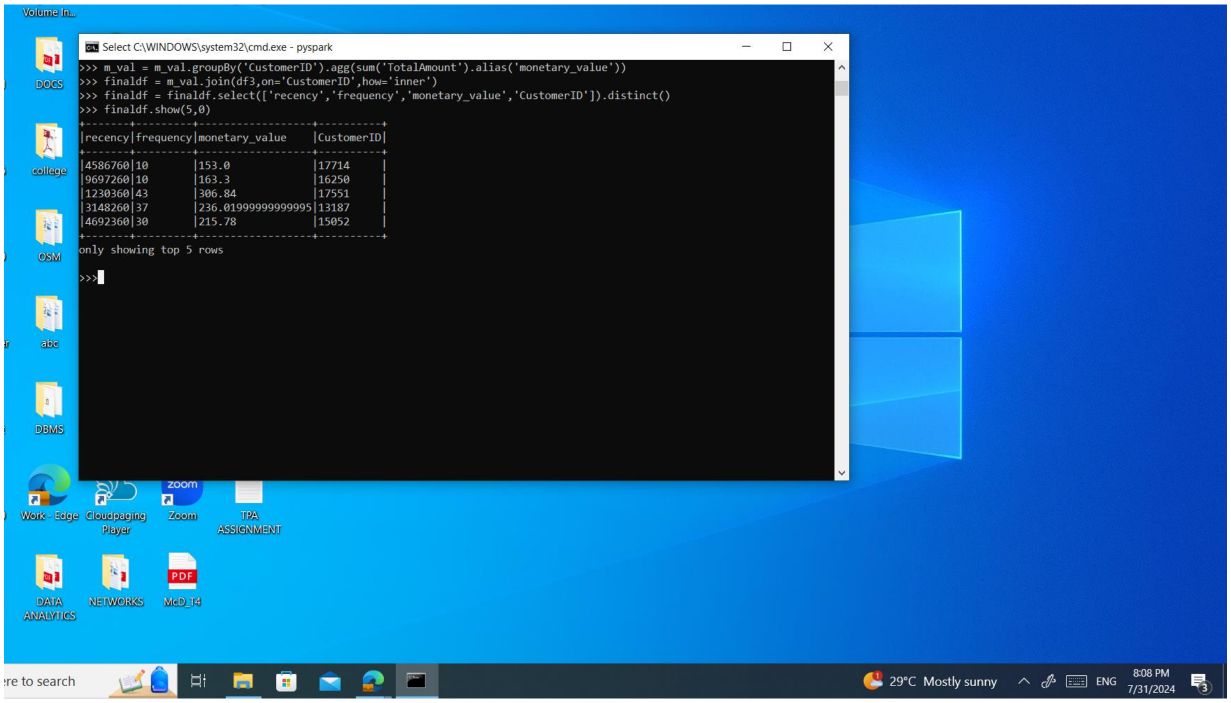
```
>>> df3 = df2.join(df_freq,on='CustomerID',how='inner')
>>> df3.printSchema()
root
 |-- CustomerID: string (nullable = true)
 |-- recency: long (nullable = true)
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: string (nullable = true)
 |-- InvoiceDate: string (nullable = true)
 |-- UnitPrice: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- date: timestamp (nullable = true)
 |-- from_date: timestamp (nullable = true)
 |-- frequency: long (nullable = false)

>>>
```

FREQUENCY



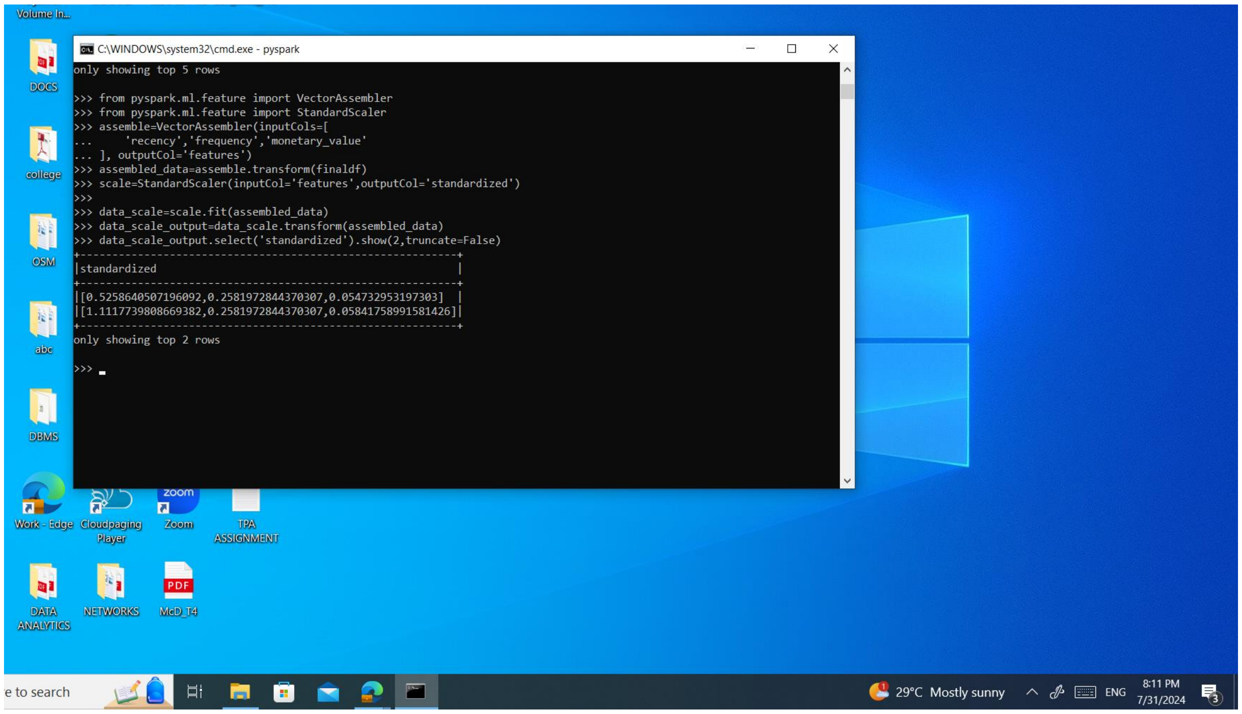
```
C:\WINDOWS\system32\cmd.exe - pyspark
>>> m_val = df3.withColumn('TotalAmount',col("Quantity") * col("UnitPrice"))
>>> m_val = m_val.groupBy('CustomerID').agg(sum('TotalAmount').alias('monetary_value'))
>>> finaldf = m_val.join(df3,on='CustomerID',how='inner')
>>> finaldf = finaldf.select(['recency','frequency','monetary_value','CustomerID']).distinct()
>>> finaldf.show(5,0)
+---------+---------+-----------------+----------+
|recency  |frequency|monetary_value   |CustomerID|
+---------+---------+-----------------+----------+
|-30405240|10       |153.0            |17714     |
|-25294740|10       |163.3            |16250     |
|-33761640|43       |306.84           |17551     |
|-31843740|37       |236.01999999999995|13187     |
|-30299640|30       |215.78           |15052     |
+---------+---------+-----------------+----------+
only showing top 5 rows

>>>
```

MONETARY VALUE



```
Select C:\WINDOWS\system32\cmd.exe - pyspark
>>> m_val = m_val.groupBy('CustomerID').agg(sum('TotalAmount').alias('monetary_value'))
>>> finaldf = m_val.join(df3,on='CustomerID',how='inner')
>>> finaldf = finaldf.select(['recency','frequency','monetary_value','CustomerID']).distinct()
>>> finaldf.show(5,0)
+-------+---------+-----------------+----------+
|recency|frequency|monetary_value   |CustomerID|
+-------+---------+-----------------+----------+
|4586760|10       |153.0            |17714     |
|9697260|10       |163.3            |16250     |
|1230360|43       |306.84           |17551     |
|3148260|37       |236.01999999999995|13187     |
|4692360|30       |215.78           |15052     |
+-------+---------+-----------------+----------+
only showing top 5 rows

>>>
```

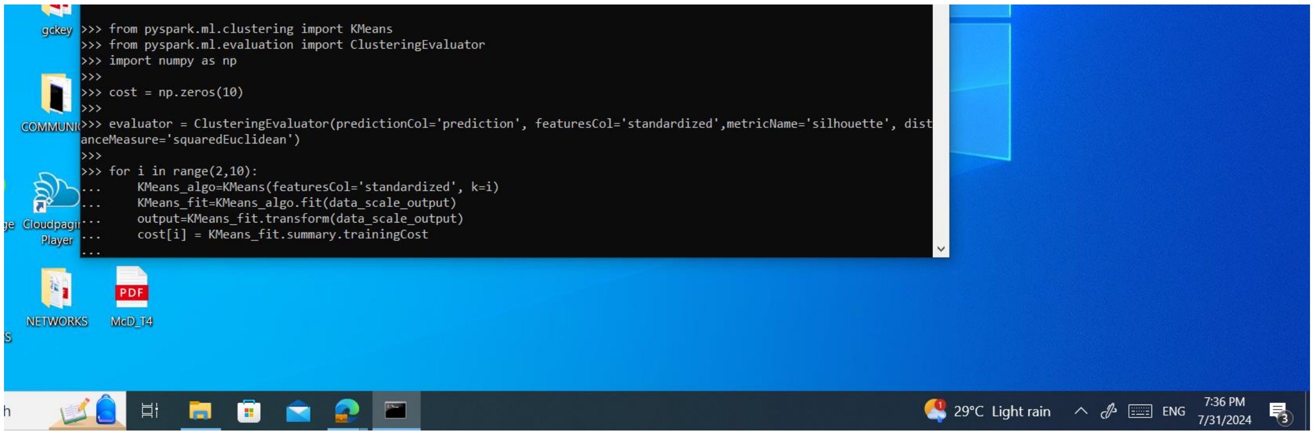
STANDARDIZATION: TO ENSURE VARIABLES ARE AROUND SAME SCALE



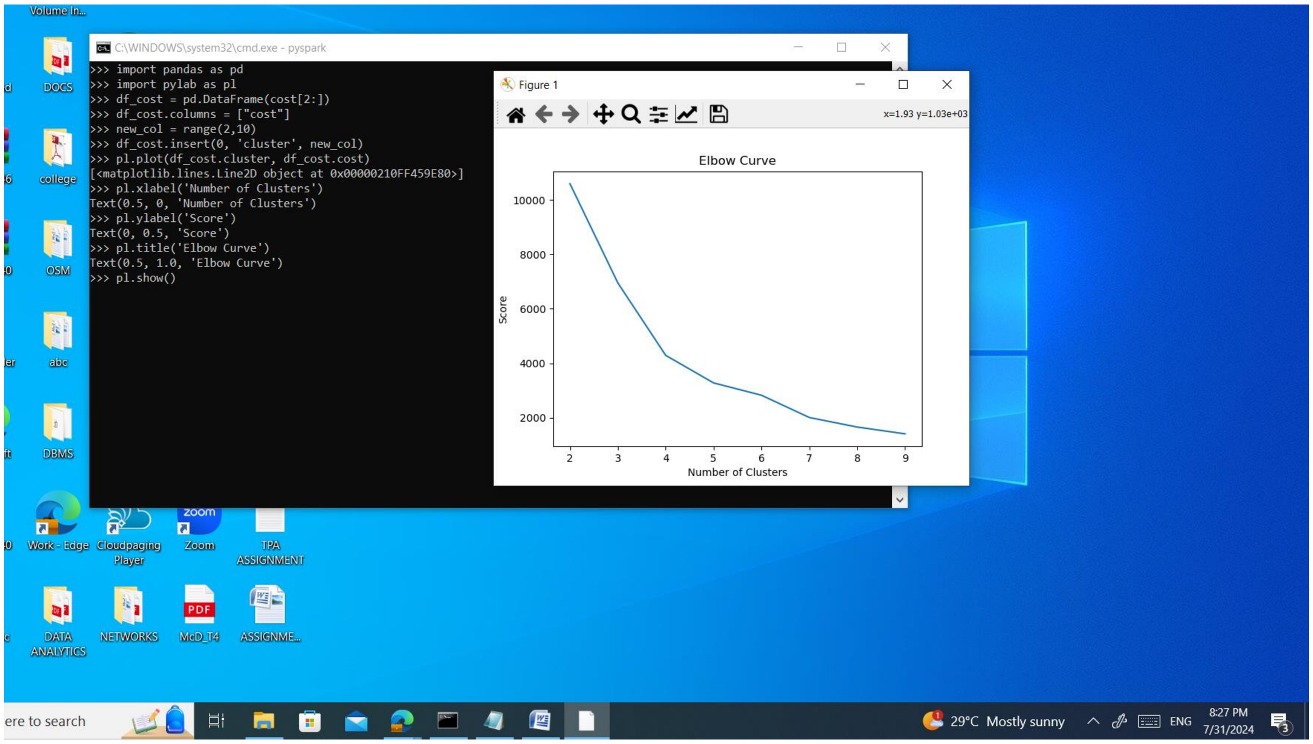
```
only showing top 5 rows

>>> from pyspark.ml.feature import VectorAssembler
>>> from pyspark.ml.feature import StandardScaler
>>> assemble=VectorAssembler(inputCols=[
...     'recency','frequency','monetary_value'
... ], outputCol='features')
>>> assembled_data=assemble.transform(finaldf)
>>> scale=StandardScaler(inputCol='features',outputCol='standardized')
>>>
>>> data_scale=scale.fit(assembled_data)
>>> data_scale_output=data_scale.transform(assembled_data)
>>> data_scale_output.select('standardized').show(2,truncate=False)
+-------------------------------------------------------+
|standardized                                           |
+-------------------------------------------------------+
|[0.5258640507196092,0.2581972844370307,0.054732953197303]  |
|[1.1117739808669382,0.2581972844370307,0.05841758991581426]|
+-------------------------------------------------------+
only showing top 2 rows

>>> _
```

TASK 5: BUILIDNG THE MACHINE LEARNING MODEL USING K-MEANS CLUSTERING

1. FINDING THE NUMBER OF CLUSTERS TO USE



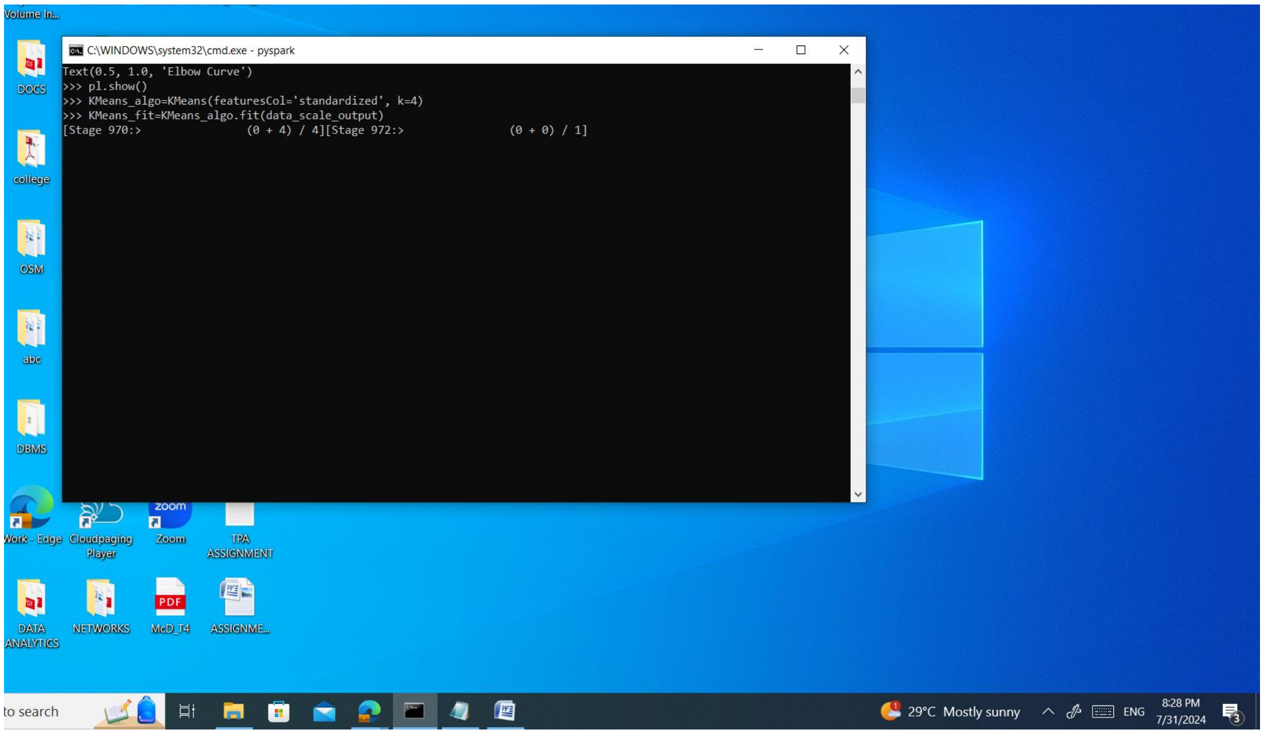
```
>>> from pyspark.ml.clustering import KMeans
>>> from pyspark.ml.evaluation import ClusteringEvaluator
>>> import numpy as np
>>>
>>> cost = np.zeros(10)
>>>
>>> evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='standardized',metricName='silhouette', dist
anceMeasure='squaredEuclidean')
>>>
>>> for i in range(2,10):
...     KMeans_algo=KMeans(featuresCol='standardized', k=i)
...     KMeans_fit=KMeans_algo.fit(data_scale_output)
...     output=KMeans_fit.transform(data_scale_output)
...     cost[i] = KMeans_fit.summary.trainingCost
...
```



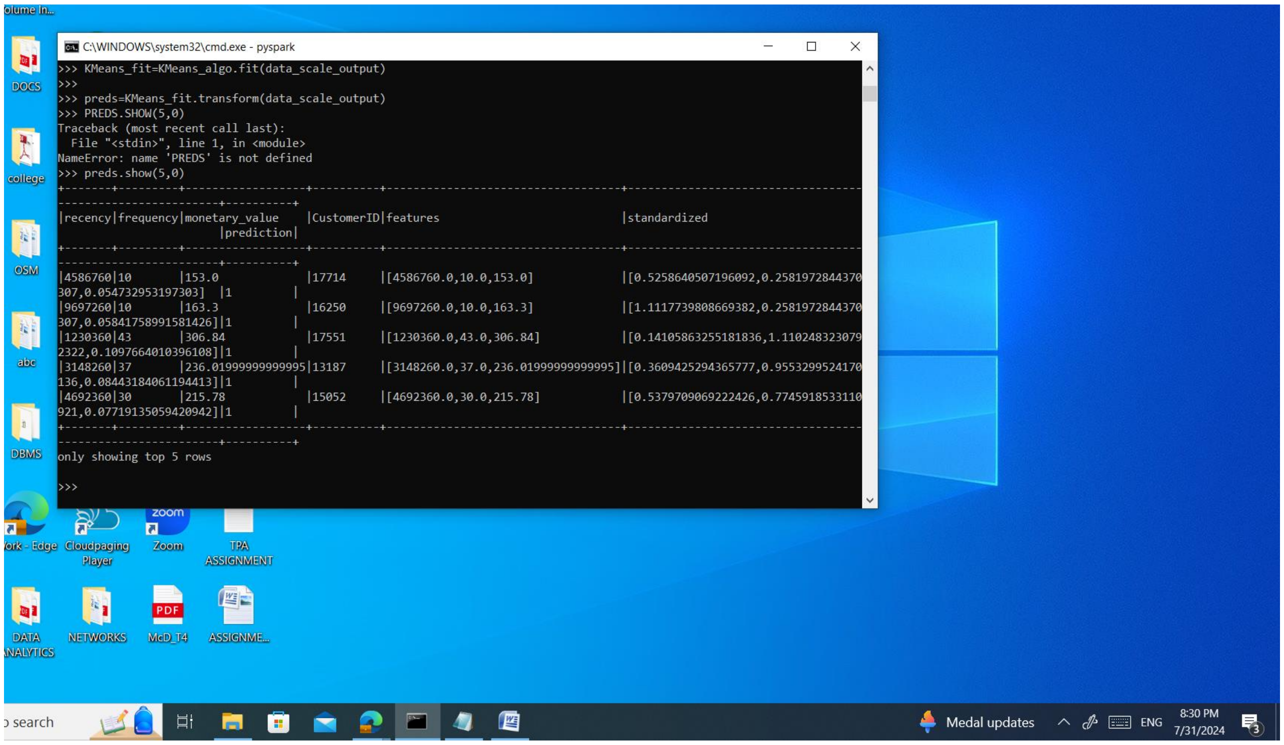
```
>>> import pandas as pd
>>> import pylab as pl
>>> df_cost = pd.DataFrame(cost[2:])
>>> df_cost.columns = ["cost"]
>>> new_col = range(2,10)
>>> df_cost.insert(0, 'cluster', new_col)
>>> pl.plot(df_cost.cluster, df_cost.cost)
[<matplotlib.lines.Line2D object at 0x00000210FF459E80>]
>>> pl.xlabel('Number of Clusters')
Text(0.5, 0, 'Number of Clusters')
>>> pl.ylabel('Score')
Text(0, 0.5, 'Score')
>>> pl.title('Elbow Curve')
Text(0.5, 1.0, 'Elbow Curve')
>>> pl.show()
```

FROM THE ABOVE GRAPH, WE INFER THE NUMBER OF CLUSTERS TO BE 4.

BUILDING THE K-MEANS CLUSTERING MODEL AND MAKING PREDICTIONS



```
Text(0.5, 1.0, 'Elbow Curve')
>>> pl.show()
>>> KMeans_algo=KMeans(featuresCol='standardized', k=4)
>>> KMeans_fit=KMeans_algo.fit(data_scale_output)
[Stage 970:>                  (0 + 4) / 4][Stage 972:>                (0 + 0) / 1]
```



```
>>> KMeans_fit=KMeans_algo.fit(data_scale_output)
>>>
>>> preds=KMeans_fit.transform(data_scale_output)
>>> PREDS.SHOW(5,0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'PREDS' is not defined
>>> preds.show(5,0)
+-------+---------+-----------------+----------+------------------------------------+------------------------------------
------------------------+----------+
|recency|frequency|monetary_value   |CustomerID|features                            |standardized
                        |prediction|
+-------+---------+-----------------+----------+------------------------------------+------------------------------------
------------------------+----------+
|4586760|10       |153.0            |17714     |[4586760.0,10.0,153.0]              |[0.5258640507196092,0.2581972844370
307,0.054732953197303]  |1         |
|9697260|10       |163.3            |16250     |[9697260.0,10.0,163.3]              |[1.1117739808669382,0.2581972844370
307,0.05841758991581426]|1         |
|1230360|43       |306.84           |17551     |[1230360.0,43.0,306.84]             |[0.14105863255181836,1.110248323079
2322,0.1097664010396108]|1         |
|3148260|37       |236.01999999999995|13187    |[3148260.0,37.0,236.01999999999995]|[0.3609425294365777,0.9553299524170
136,0.08443184061194413]|1         |
|4692360|30       |215.78           |15052     |[4692360.0,30.0,215.78]             |[0.5379709069222426,0.7745918533110
921,0.07719135059420942]|1         |
+-------+---------+-----------------+----------+------------------------------------+------------------------------------
------------------------+----------+
only showing top 5 rows

>>>
```