databricks Ecomm_big_data

(https://databricks.com)

1

```
spark
```

SparkSession - hive

SparkContext

Spark UI

Version
    v3.5.2
Master
    local[*, 4]
AppName
    Databricks Shell

FileStore/tables/ecommerce_dataset.csv

3

```
%python
%fs ls /FileStore/tables/ecommerce_dataset.csv
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/ecommerce_dataset.csv	ecommerce_dataset.csv	10405498	1734629826000

1 row

4

```
ecommercedatadf=spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/FileStore/t
```

ecommercedatadf: pyspark.sql.dataframe.DataFrame = [Product ID: string, Product Name: string ... 14 more fields]

5

```
ecommercedatadf.show()
```

```
+----------+---------------+---------------+-------+--------+--------+-----------+-----------+------------------+--
-----------------+---------------+-------------+---------------+-----------+-----------+----------------+
|Product ID|   Product Name|       Category|  Price|Discount|Tax Rate|Stock Level|Supplier ID|Customer Age Group|
Customer Location|Customer Gender|Shipping Cost|Shipping Method|Return Rate|Seasonality|Popularity Index|
+----------+---------------+---------------+-------+--------+--------+-----------+-----------+------------------+--
-----------------+---------------+-------------+---------------+-----------+-----------+----------------+
|     P6879|         Jacket|        Apparel|  53.85|       5|      15|        150|       S535|             35-44|
New York, USA|           Male|        23.32|       Standard|       4.49|        Yes|              56|
|     P5132|         Camera|    Electronics| 761.26|      10|      15|        224|       S583|             25-34|
London, UK|         Female|        20.88|      Overnight|      16.11|         No|              79|
|     P2941|        Sneakers|       Footwear|1756.76|       5|       8|        468|       S118|             25-34|
Tokyo, Japan|     Non-Binary|        16.43|       Standard|       4.93|         No|              40|
|     P8545|      Cookbooks|          Books| 295.24|      10|      15|         25|       S104|             18-24|
Paris, France|         Female|        27.49|       Standard|       1.31|         No|              93|
|     P4594|         Camera|    Electronics|  832.0|      10|      12|        340|       S331|                55|
Tokyo, Japan|           Male|        45.93|      Overnight|       4.37|         No|              56|
|     P1388|    Non-Fiction|          Books| 584.19|      15|       8|        204|       S523|             45-54|
Singapore|         Female|        40.12|        Express|      19.03|         No|              91|
|     P7313|  Running Shoes|       Footwear|1343.95|       0|      10|        493|       S878|             18-24|
Sydney, Australia|         Female|        35.91|      Overnight|      17.73|        Yes|              41|
|     P1060|        Blender|Home Appliances|1873.52|       0|      15|        349|       S396|             18-24|
```

6

```
# Replace invalid characters in column names
cleaned_columns = [col.strip().replace(" ", "_").replace(".", "_") for col in ecommercedatadf.columns]
ecommercedatadf = ecommercedatadf.toDF(*cleaned_columns)

# Show the updated schema
ecommercedatadf.printSchema()

# Save raw data as a Delta table
ecommercedatadf.write.format("delta").mode("overwrite").saveAsTable("ecommerce_bronze")
print("Raw data saved to Bronze Layer!")
```

ecommercedatadf: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 14 more fields]

```
root
 |-- Product_ID: string (nullable = true)
 |-- Product_Name: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Price: double (nullable = true)
 |-- Discount: integer (nullable = true)
 |-- Tax_Rate: integer (nullable = true)
 |-- Stock_Level: integer (nullable = true)
 |-- Supplier_ID: string (nullable = true)
 |-- Customer_Age_Group: string (nullable = true)
 |-- Customer_Location: string (nullable = true)
 |-- Customer_Gender: string (nullable = true)
 |-- Shipping_Cost: double (nullable = true)
 |-- Shipping_Method: string (nullable = true)
 |-- Return_Rate: double (nullable = true)
 |-- Seasonality: string (nullable = true)
 |-- Popularity_Index: integer (nullable = true)

Raw data saved to Bronze Layer!
```

7

```
# Query the Bronze Layer Delta table and display the first 10 rows
bronze_layer_data = spark.sql("SELECT * FROM ecommerce_bronze LIMIT 10")

bronze_layer_data.show()
```

bronze_layer_data: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 14 more fields]

```
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------+---------------+-------------+---------------+-----------+-----------+----------------+
|Product_ID| Product_Name|       Category|  Price|Discount|Tax_Rate|Stock_Level|Supplier_ID|Customer_Age_Group|   C
ustomer_Location|Customer_Gender|Shipping_Cost|Shipping_Method|Return_Rate|Seasonality|Popularity_Index|
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------+---------------+-------------+---------------+-----------+-----------+----------------+
|     P6879|       Jacket|        Apparel|  53.85|       5|      15|        150|       S535|             35-44|    
New York, USA|           Male|        23.32|       Standard|       4.49|        Yes|              56|
|     P5132|       Camera|    Electronics| 761.26|      10|      15|        224|       S583|             25-34|    
London, UK|         Female|        20.88|      Overnight|      16.11|         No|              79|
|     P2941|     Sneakers|       Footwear|1756.76|       5|       8|        468|       S118|             25-34|    
Tokyo, Japan|     Non-Binary|        16.43|       Standard|       4.93|         No|              40|
|     P8545|    Cookbooks|          Books| 295.24|      10|      15|         25|       S104|             18-24|    
Paris, France|         Female|        27.49|       Standard|       1.31|         No|              93|
|     P4594|       Camera|    Electronics|  832.0|      10|      12|        340|       S331|                55|    
Tokyo, Japan|           Male|        45.93|      Overnight|       4.37|         No|              56|
|     P1388|  Non-Fiction|          Books| 584.19|      15|       8|        204|       S523|             45-54|    
Singapore|         Female|        40.12|        Express|      19.03|         No|              91|
|     P7313|Running Shoes|       Footwear|1343.95|       0|      10|        493|       S878|             18-24|   S
ydney, Australia|         Female|        35.91|      Overnight|      17.73|        Yes|              41|
|     P1060|      Blender|Home Appliances|1873.52|       0|      15|        349|       S396|             18-24|
```

8

```
from pyspark.sql.functions import col

# Clean the data
cleaned_data = ecommercedatadf \
    .filter((col("Price").isNotNull()) & (col("Stock_Level").isNotNull())) \
    .filter((col("Price") > 0) & (col("Stock_Level") > 0)) \
    .withColumn("Price", col("Price").cast("double")) \
    .withColumn("Stock_Level", col("Stock_Level").cast("int"))

# Show the cleaned data
cleaned_data.show()

# Save cleaned data as Delta table
cleaned_data.write.format("delta").mode("overwrite").saveAsTable("ecommerce_silver")
print("Cleaned data saved to Silver Layer!")
```

cleaned_data: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 14 more fields]

```
+----------+---------------+---------------+-------+--------+--------+-----------+-----------+------------------+--
-----------------+---------------+-------------+---------------+-----------+-----------+----------------+
|Product_ID|   Product_Name|       Category|  Price|Discount|Tax_Rate|Stock_Level|Supplier_ID|Customer_Age_Group|
Customer_Location|Customer_Gender|Shipping_Cost|Shipping_Method|Return_Rate|Seasonality|Popularity_Index|
+----------+---------------+---------------+-------+--------+--------+-----------+-----------+------------------+--
-----------------+---------------+-------------+---------------+-----------+-----------+----------------+
|     P6879|         Jacket|        Apparel|  53.85|       5|      15|        150|       S535|             35-44|
New York, USA|           Male|        23.32|       Standard|       4.49|        Yes|              56|
|     P5132|         Camera|    Electronics| 761.26|      10|      15|        224|       S583|             25-34|
London, UK|         Female|        20.88|      Overnight|      16.11|         No|              79|
|     P2941|       Sneakers|       Footwear|1756.76|       5|       8|        468|       S118|             25-34|
Tokyo, Japan|     Non-Binary|        16.43|       Standard|       4.93|         No|              40|
|     P8545|      Cookbooks|          Books| 295.24|      10|      15|         25|       S104|             18-24|
Paris, France|         Female|        27.49|       Standard|       1.31|         No|              93|
|     P4594|         Camera|    Electronics|  832.0|      10|      12|        340|       S331|                55|
Tokyo, Japan|           Male|        45.93|      Overnight|       4.37|         No|              56|
|     P1388|    Non-Fiction|          Books| 584.19|      15|       8|        204|       S523|             45-54|
Singapore|         Female|        40.12|        Express|      19.03|         No|              91|
|     P7313|  Running Shoes|       Footwear|1343.95|       0|      10|        493|       S878|             18-24|
Sydney, Australia|         Female|        35.91|      Overnight|      17.73|        Yes|              41|
|     P1060|        Blender|Home Appliances|1873.52|       0|      15|        349|       S396|             18-24|
```

Gold Layer

Perform aggregations to create meaningful datasets and save them as Delta tables.

Total Sales by Category

10

```
# Total sales by category
total_sales_by_category = cleaned_data.groupBy("Category") \
    .agg({"Price": "sum"}) \
    .withColumnRenamed("sum(Price)", "Total_Sales")

# Save as Delta table
total_sales_by_category.write.format("delta").mode("overwrite").saveAsTable("total_sales_by_category")
print("Total Sales by Category saved to Gold Layer!")
total_sales_by_category.show()
```

total_sales_by_category: pyspark.sql.dataframe.DataFrame = [Category: string, Total_Sales: double]

```
Total Sales by Category saved to Gold Layer!
+---------------+--------------------+
|       Category|         Total_Sales|
+---------------+--------------------+
|        Apparel|2.0063856269999962E7|
|    Electronics| 2.010186938999998E7|
|       Footwear|2.0161514950000018E7|
|          Books|2.0276547950000018E7|
|Home Appliances|1.9960686890000023E7|
+---------------+--------------------+
```

Average Price by Category

12

```
# Average price by category
avg_price_by_category = cleaned_data.groupBy("Category") \
    .agg({"Price": "avg"}) \
    .withColumnRenamed("avg(Price)", "Avg_Price")

# Save as Delta table
avg_price_by_category.write.format("delta").mode("overwrite").saveAsTable("avg_price_by_category")
avg_price_by_category.show()
```

avg_price_by_category: pyspark.sql.dataframe.DataFrame = [Category: string, Avg_Price: double]

```
+---------------+------------------+
|       Category|         Avg_Price|
+---------------+------------------+
|        Apparel|1001.6902780828738|
|    Electronics|1009.3833487321103|
|       Footwear|1004.7099691035041|
|          Books|1009.6374022805367|
|Home Appliances|1012.9757366150735|
+---------------+------------------+
```

Total Quantity Sold by Shipping Method

14

```
# Total quantity sold by shipping method
total_quantity_by_shipping = cleaned_data.groupBy("Shipping_Method") \
    .agg({"Stock_Level": "sum"}) \
    .withColumnRenamed("sum(Stock_Level)", "Total_Quantity")

# Save as Delta table
total_quantity_by_shipping.write.format("delta").mode("overwrite").saveAsTable("total_quantity_by_shipping")
total_quantity_by_shipping.show()
```

total_quantity_by_shipping: pyspark.sql.dataframe.DataFrame = [Shipping_Method: string, Total_Quantity: long]

```
+---------------+--------------+
|Shipping_Method|Total_Quantity|
+---------------+--------------+
|        Express|       8292172|
|      Overnight|       8339453|
|       Standard|       8373928|
+---------------+--------------+
```

Monthly Sales Trends

16

```
from pyspark.sql.functions import month, current_timestamp

# Add a synthetic timestamp column if missing
cleaned_data_with_date = cleaned_data.withColumn("Order_Date", current_timestamp())

# Monthly sales trends
monthly_sales = cleaned_data_with_date.withColumn("Month", month(col("Order_Date"))) \
    .groupBy("Month") \
    .agg({"Price": "sum"}) \
    .withColumnRenamed("sum(Price)", "Total_Sales")

# Save as Delta table
monthly_sales.write.format("delta").mode("overwrite").saveAsTable("monthly_sales")
monthly_sales.show()
```

cleaned_data_with_date: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 15 more fields]

monthly_sales: pyspark.sql.dataframe.DataFrame = [Month: integer, Total_Sales: double]

```
+-----+--------------------+
|Month|         Total_Sales|
+-----+--------------------+
|   12|1.005644754499997E8|
+-----+--------------------+
```

Verifying Layers

Bronze Layer

18

```
bronze_layer_data = spark.sql("SELECT * FROM ecommerce_bronze")
bronze_layer_data.show(10, truncate=False)
```

bronze_layer_data: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 14 more fields]

```
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------------+---------------+-------------+---------------+-----------+-----------+----------------+
|Product_ID|Product_Name |Category       |Price  |Discount|Tax_Rate|Stock_Level|Supplier_ID|Customer_Age_Group|Cust
omer_Location      |Customer_Gender|Shipping_Cost|Shipping_Method|Return_Rate|Seasonality|Popularity_Index|
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------------+---------------+-------------+---------------+-----------+-----------+----------------+
|P6879     |Jacket       |Apparel        |53.85  |5       |15      |150        |S535       |35-44             |New 
York, USA          |Male           |23.32        |Standard       |4.49       |Yes        |56              |
|P5132     |Camera       |Electronics    |761.26 |10      |15      |224        |S583       |25-34             |Lond
on, UK             |Female         |20.88        |Overnight      |16.11      |No         |79              |
|P2941     |Sneakers     |Footwear       |1756.76|5       |8       |468        |S118       |25-34             |Toky
o, Japan           |Non-Binary     |16.43        |Standard       |4.93       |No         |40              |
|P8545     |Cookbooks    |Books          |295.24 |10      |15      |25         |S104       |18-24             |Pari
s, France          |Female         |27.49        |Standard       |1.31       |No         |93              |
|P4594     |Camera       |Electronics    |832.0  |10      |12      |340        |S331       |55                |Toky
o, Japan           |Male           |45.93        |Overnight      |4.37       |No         |56              |
|P1388     |Non-Fiction  |Books          |584.19 |15      |8       |204        |S523       |45-54             |Sing
apore              |Female         |40.12        |Express        |19.03      |No         |91              |
|P7313     |Running Shoes|Footwear       |1343.95|0       |10      |493        |S878       |18-24             |Sydn
ey, Australia      |Female         |35.91        |Overnight      |17.73      |Yes        |41              |
|P1060     |Blender      |Home Appliances|1873.52|0       |15      |349        |S396       |18-24             |Phoe
```

Verifying Layers

Silver Layer

20

```
silver_layer_data = spark.sql("SELECT * FROM ecommerce_silver")
silver_layer_data.show(10, truncate=False)
```

silver_layer_data: pyspark.sql.dataframe.DataFrame = [Product_ID: string, Product_Name: string ... 14 more fields]

```
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------------+---------------+-------------+---------------+-----------+-----------+----------------+
```

```
|Product_ID|Product_Name |Category       |Price  |Discount|Tax_Rate|Stock_Level|Supplier_ID|Customer_Age_Group|Cust
omer_Location      |Customer_Gender|Shipping_Cost|Shipping_Method|Return_Rate|Seasonality|Popularity_Index|
+----------+-------------+---------------+-------+--------+--------+-----------+-----------+------------------+----
-------------------+---------------+-------------+---------------+-----------+-----------+----------------+
|P6879     |Jacket       |Apparel        |53.85  |5       |15      |150        |S535       |35-44             |New 
York, USA          |Male           |23.32        |Standard       |4.49       |Yes        |56              |
|P5132     |Camera       |Electronics    |761.26 |10      |15      |224        |S583       |25-34             |Lond
on, UK             |Female         |20.88        |Overnight      |16.11      |No         |79              |
|P2941     |Sneakers     |Footwear       |1756.76|5       |8       |468        |S118       |25-34             |Toky
o, Japan           |Non-Binary     |16.43        |Standard       |4.93       |No         |40              |
|P8545     |Cookbooks    |Books          |295.24 |10      |15      |25         |S104       |18-24             |Pari
s, France          |Female         |27.49        |Standard       |1.31       |No         |93              |
|P4594     |Camera       |Electronics    |832.0  |10      |12      |340        |S331       |55                |Toky
o, Japan           |Male           |45.93        |Overnight      |4.37       |No         |56              |
|P1388     |Non-Fiction  |Books          |584.19 |15      |8       |204        |S523       |45-54             |Sing
apore              |Female         |40.12        |Express        |19.03      |No         |91              |
|P7313     |Running Shoes|Footwear       |1343.95|0       |10      |493        |S878       |18-24             |Sydn
ey, Australia      |Female         |35.91        |Overnight      |17.73      |Yes        |41              |
|P1060     |Blender      |Home Appliances|1873.52|0       |15      |349        |S396       |18-24             |Phoe
```

Verifying Layers

Gold Layers

22

```
gold_layer_data = spark.sql("SELECT * FROM total_sales_by_category")
gold_layer_data.show(10, truncate=False)
```

gold_layer_data: pyspark.sql.dataframe.DataFrame = [Category: string, Total_Sales: double]

```
+---------------+--------------------+
|Category       |Total_Sales         |
+---------------+--------------------+
|Apparel        |2.0063856269999962E7|
|Electronics    |2.010186938999998E7 |
|Footwear       |2.0161514950000018E7|
|Books          |2.0276547950000018E7|
|Home Appliances|1.9960686890000023E7|
+---------------+--------------------+
```

23

```
gold_layer_data = spark.sql("SELECT * FROM monthly_sales")
gold_layer_data.show(10, truncate=False)
```

gold_layer_data: pyspark.sql.dataframe.DataFrame = [Month: integer, Total_Sales: double]

```
+-----+-------------------+
|Month|Total_Sales        |
+-----+-------------------+
|12   |1.005644754499997E8|
+-----+-------------------+
```

24

```
gold_layer_data = spark.sql("SELECT * FROM total_quantity_by_shipping")
gold_layer_data.show(10, truncate=False)
```

gold_layer_data: pyspark.sql.dataframe.DataFrame = [Shipping_Method: string, Total_Quantity: long]

```
+---------------+--------------+
|Shipping_Method|Total_Quantity|
+---------------+--------------+
|Express        |8292172       |
|Overnight      |8339453       |
|Standard       |8373928       |
+---------------+--------------+
```

Machine learning Model to Predict Stock Levels (Regression model)

Objective: To Predict the Stock_Level for a product based on features like Price, Discount, and Tax_Rate.

26

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator

# Prepare data for regression
assembler = VectorAssembler(inputCols=["Price", "Discount", "Tax_Rate"], outputCol="features")
regression_data = assembler.transform(cleaned_data).select("features", "Stock_Level")

# Split data into training and testing sets
train_data, test_data = regression_data.randomSplit([0.8, 0.2], seed=42)

# Train Linear Regression model
lr = LinearRegression(featuresCol="features", labelCol="Stock_Level")
lr_model = lr.fit(train_data)

# Evaluate model
predictions = lr_model.transform(test_data)
evaluator = RegressionEvaluator(labelCol="Stock_Level", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)

print(f"Root Mean Squared Error (RMSE): {rmse}")
predictions.show(10, truncate=False)
```

- predictions: pyspark.sql.dataframe.DataFrame
- regression_data: pyspark.sql.dataframe.DataFrame
- test_data: pyspark.sql.dataframe.DataFrame
- train_data: pyspark.sql.dataframe.DataFrame

Loading widget. This should take less than 30 seconds.

Loading widget. This should take less than 30 seconds.

```
Root Mean Squared Error (RMSE): 144.1967092226598
+------------------+-----------+------------------+
|features          |Stock_Level|prediction        |
+------------------+-----------+------------------+
|[10.23,25.0,8.0]  |207        |249.11485704564956|
|[10.46,25.0,8.0]  |114        |249.11522916082723|
|[10.58,5.0,8.0]   |480        |248.56769425997706|
|[10.78,25.0,10.0] |210        |249.72145366904982|
|[11.23,0.0,12.0]  |5          |249.64322719336815|
|[11.56,10.0,12.0] |312        |249.91762562170322|
|[11.77,15.0,8.0]  |293        |248.84348407549822|
|[12.14,15.0,12.0] |128        |250.05549626108262|
|[12.64,15.0,5.0]  |225        |247.93633146746808|
|[12.7,15.0,12.0]  |165        |250.05640228064567|
+------------------+-----------+------------------+
only showing top 10 rows
```

Machine Learning Model : TO Classify Products by Popularity (Classification Model)

Objective:Classify products as "Popular" or "Not Popular" based on their Popularity_Index.

28

```
from pyspark.ml.classification import LogisticRegression
from pyspark.sql.functions import when
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Add binary column for popularity classification
classification_data = cleaned_data.withColumn("is_popular", when(col("Popularity_Index") >= 50, 1).otherwise(0))

# Prepare features for classification
assembler = VectorAssembler(inputCols=["Price", "Discount", "Tax_Rate", "Stock_Level"], outputCol="features")
classification_data = assembler.transform(classification_data).select("features", "is_popular")

# Split data into training and testing sets
train_data, test_data = classification_data.randomSplit([0.8, 0.2], seed=42)

# Train Logistic Regression model
lr = LogisticRegression(featuresCol="features", labelCol="is_popular", maxIter=10)
lr_model = lr.fit(train_data)

# Evaluate model
predictions = lr_model.transform(test_data)

# Use MulticlassClassificationEvaluator for accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="is_popular", predictionCol="prediction", metricName="accur
accuracy = evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy}")
predictions.show(10, truncate=False)
```

- classification_data: pyspark.sql.dataframe.DataFrame
- predictions: pyspark.sql.dataframe.DataFrame
- test_data: pyspark.sql.dataframe.DataFrame
- train_data: pyspark.sql.dataframe.DataFrame

Loading widget. This should take less than 30 seconds.

Loading widget. This should take less than 30 seconds.

```
Accuracy: 0.5021143777688281
+-----------------------+----------+---------------------------------------------+---------------------------------
--------+----------+
|features               |is_popular|rawPrediction                                |probability
|prediction|
+-----------------------+----------+---------------------------------------------+---------------------------------
--------+----------+
|[10.23,25.0,8.0,207.0] |1         |[0.009160758245145367,-0.009160758245145367] |[0.5022901735454588,0.4977098264
545412] |0.0       |
|[10.46,25.0,8.0,114.0] |0         |[0.009531934371093056,-0.009531934371093056] |[0.5023829655502221,0.4976170344
497779] |0.0       |
|[10.58,5.0,8.0,480.0]  |1         |[-0.010561687068939042,0.010561687068939042] |[0.49735960277724345,0.502640397
2227566]|1.0       |
|[10.78,25.0,10.0,210.0]|1         |[0.00739821539632827,-0.00739821539632827]   |[0.5018495454130679,0.4981504545
869321] |0.0       |
|[11.23,0.0,12.0,5.0]   |1         |[-0.016798815313672933,0.016798815313672933] |[0.4958003949318983,0.5041996050
681017] |1.0       |
|[11.56,10.0,12.0,312.0]|1         |[-0.008729998488339009,0.008729998488339009] |[0.49781751423902365,0.502182485
7609763]|1.0       |
|[11.77,15.0,8.0,293.0] |1         |[-5.162029514218184E-4,5.162029514218184E-4] |[0.49987094926501013,0.500129050
7349898]|1.0       |
```

Machine Learining Model : Customer Segmentation Using K-Means Clustering

Objective: Group customers into segments based on features like total revenue, order count, and average order value.

30

- clustered_customers: pyspark.sql.dataframe.DataFrame
- customer_data: pyspark.sql.dataframe.DataFrame = [Customer_Location: string, Total_Revenue: double ... 2 more fields]
- customer_features: pyspark.sql.dataframe.DataFrame

Loading widget. This should take less than 30 seconds.

Loading widget. This should take less than 30 seconds.

```
+-----------------+--------------------+-----------+------------------+-------+
|Customer_Location|Total_Revenue       |Order_Count|Avg_Order_Value   |Cluster|
+-----------------+--------------------+-----------+------------------+-------+
|Singapore        |1.7024310824700003E9|6695       |1010.3822001493661|0      |
|Toronto, Canada  |1.672599277999999E9 |6585       |1013.0271556567948|1      |
|Mumbai, India    |1.669635879249999E9 |6767       |1005.132818087779 |1      |
|Chicago, USA     |1.693613506369999E9 |6598       |1007.4677826614122|0      |
|Sydney, Australia|1.654938673480001E9 |6527       |1002.6644921096992|1      |
|Dubai, UAE       |1.692078486000001E9 |6759       |1005.3609675987568|0      |
|Phoenix, USA     |1.6739677950999987E9|6643       |1010.5593632394999|1      |
|London, UK       |1.653133941290001E9 |6514       |1006.0021998771882|1      |
|Berlin, Germany  |1.667030874180001E9 |6638       |1002.7849789093117|1      |
|Los Angeles, USA |1.6816780004499977E9|6635       |1013.1032313489078|0      |
+-----------------+--------------------+-----------+------------------+-------+
only showing top 10 rows

Silhouette Score: 0.7126998999023788
```