Data Warehousing and OLAP Coursework (7 BUS 010W)

Course work-2

Module leader: Dr. Panagiotis Chountas

Report by: Mayur V chakalasiya(W1872798)

Introduction **3**

1. Data Understanding **3**
1.1 Data Cleaning 3
1.2 Distribution analysis 3
1.4 Statistical exploration 4
1.5 Correlation analysis 5
1.6 Suitable transformations of variables 5
1.7 Elimination of redundant variables 5
1.8 Management of missing values 7

2. Perform RFM Segmentation **7**
2.1 Definition of RFM metrics 7
3.1 Build of K-Means Model in Python 10
3.2 Correct Justification of K value 10
3.3 Testing of K-Means Model in Python 11

4. Review of Results **12**
4.1 Identification of business value customer segments 12
4.2 Correct Justification of their business value 13

5 Data Mart Design **13**
5.1 Identification of Dimensions 13
5.2 Justification of Selected Dimensions 14
5.3 Identification of Measures 14
Abbreviations 15
Reference 15
Appendix 15

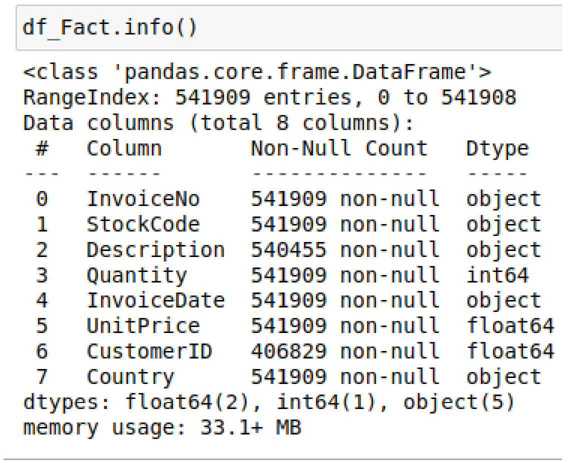
Introduction

As part of the coursework, customer retail data is provided from the UCI Machine Learning Repository to analysis on customer segmentation and provide a report on increased customer loyalty and customer lifetime value. It will provide insightful details to the marketing department. I will use sqlite3 and python to complete the analysis. For analysis and customer segmentation, I will pre-process the given dataset and derive the RFM model, and customer segmentation based on RFM values. After that, I will use the K-mean clustering algorithm to create a customer cluster and provide my analysis. In the end, I will create a data mart based on my analysis.

Note: In the reports, I provide the code snippet reference with the line number. For that, refer to the cw_2_dataware_house.py file with line numbers.

1. Data Understanding

To load data, I create a retails_details table and load data in sqlite3. After that, I create the connect to sqlite3 using the sqlite3 package and load data into the data frame. Refer to Image-1 in the Appendix [Refer to code in lines 28-40].



```
df_Fact.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Image-2

As shown in Image-2, using the info() method of the data frame, I get detailed information about columns' names, counts, and their data types. Mainly three data types are available in the dataset which are Object, int64, and float64. There are two columns (CustomerID and Country) for customers, three columns related to the products (StockeCode, Description, Unit price), and three columns related to invoices (InviceNo, InvoiceDate, and Quantity). Refer to Image-3 for the top five rows in the appendix.

1.1 Data Cleaning

To clean data, I add a few conditions in the where clause of the select query, while loading data in the data frame. Conditions are as follows:

1) To remove canceled invoice Number: InvoiceNo NOT LIKE '%C%'
2) To remove blank and null customer numbers: CustomerID IS NOT NULL and CustomerID <> " "
3) To remove Transaction with zero unit price: unitprice != 0

Refer to Image-4 for SQL query and top five columns after cleaning data. [Refer to code in lines 46-51].

1.2 Distribution analysis

It is crucial to do distribution analysis, which helps to understand the distribution of important columns and find outliers. Outliers can impact the overall performance model and analysis. If there are any outliers, I will derive outlier cutoff values and remove outliers from the data.

I choose the two columns quality and unit price and draw a distribution plot using the matplotlib library. Refer to image-5 for the distribution plot of both columns. Refer to the code snippet in Images 6 & 7 in the appendix. [Refer to code in lines 62-76].

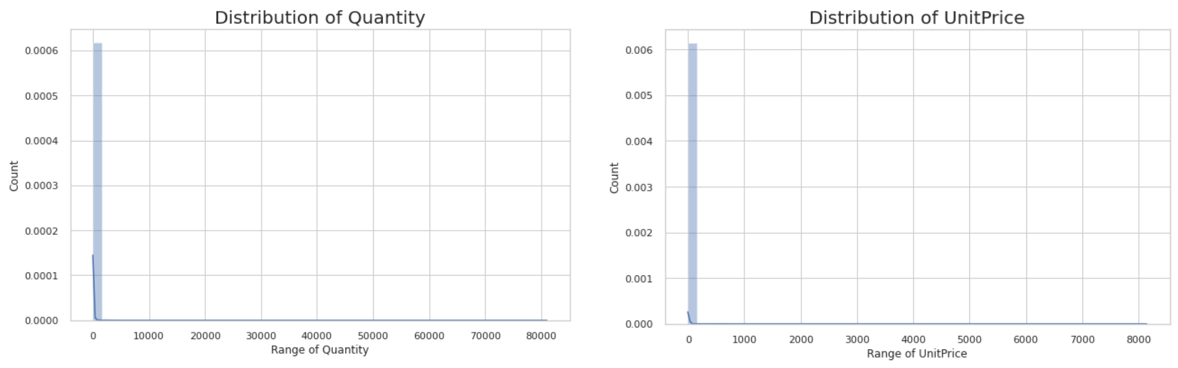


Image-5

There are outliers available for both columns as per the above images. If there are any values above (Mean + 3*SD) and below (Mean - 3*SD), I will remove them from the dataset. [Refer to code in lines 93-97].

- **Highest allowed UnitPrice:** 69.41011764784909
- **Lowest allowed UnitPrice:** -63.17714213740865
- **Highest allowed Quantity:** 550.9835626675814
- **Lowest allowed Quantity:** -525.0070871118918

Using the above values, I remove the outliers and plot distribution for unit price and Quantity. Refer to Image-8 for distribution plots. Refer to the code snippet in Image-9 in the appendix. [Refer to code in lines 103-120].

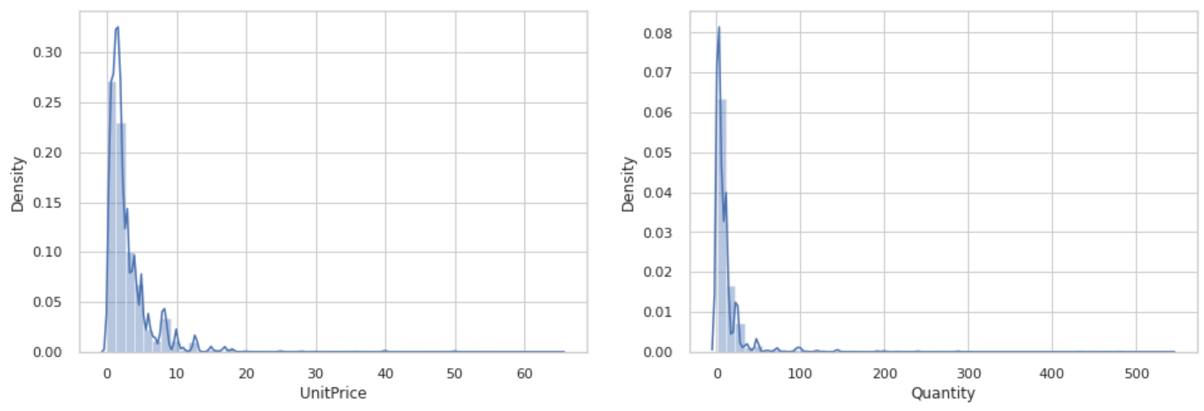


Image-8

1.4 Statistical exploration

I use the panda profiling library for in-depth analysis of the given dataset. The report for analysis is given in the Online_Retail_Data_Statistics_Report.html file. The report contains an analysis of correlation, interaction, sample values, and below information for the dataset and individual columns. Refer to Image-10 for the code snippet in the appendix. [Refer to code in lines 188-189].

The Number of variables: 9
Number of observations: 397590
Missing cells: 0
Missing cells (%): 0.0%
Duplicate rows: 0
Duplicate rows (%): 0.0%
Total size in memory: 27.3 MiB
Average record size in memory: 72.0 B

1.5 Correlation analysis

From the statistics analysis (refer to section 1.4), I find correlation matrics for df_index, quality, and Unit price, and Customer ID. Refer to Image-11 for correlation analysis.

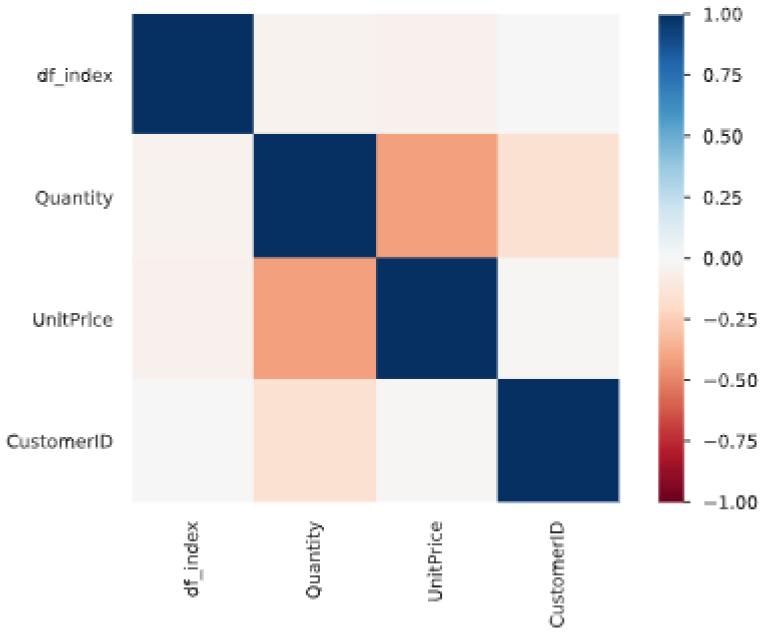


Image-11

Outcome of correlation analysis:

- df_index is highly correlated with InvoiceNo High correlation
- InvoiceNo is highly correlated with df_index High correlation
- CustomerID is highly correlated with Country High correlation
- Country is highly correlated with CustomerID High correlation

1.6 Suitable transformations of variables

As part of data understanding, I have identified a few columns which require transformation from suitable data types. For example, the Invoice Date is the object column that should be transformed to DateTime data type in python or SQL. The data type of Country and Description Is an object which should be a string.

1.7 Elimination of redundant variables

As shown in the Online_Retail_Data_Statistics_Report.html report (refer to section 1.4), there are no duplicate rows in the dataset. I already removed the canceled Invoice number and other redundant values during data cleaning. [Refer to code in lines 142-144].

1.8 Data visualization

There are two important two areas that help us understand our analysis, which is the retail market (Country wise transactions) and sales (UnitPrice X Quantity) for all given countries except the UK. I use a bar plot using the matplotlib library for market and sales for given countries except for the UK.

Refer to Images 12 & 13. [Refer to code in lines 154-174]. Top five countries for sales and marketing for UK bases retail stores are as below :

Market: Germany, France, BRE, Spain, Netherlands, Belgium

Sales: Netherlands, BRE, Germany, France, Australia

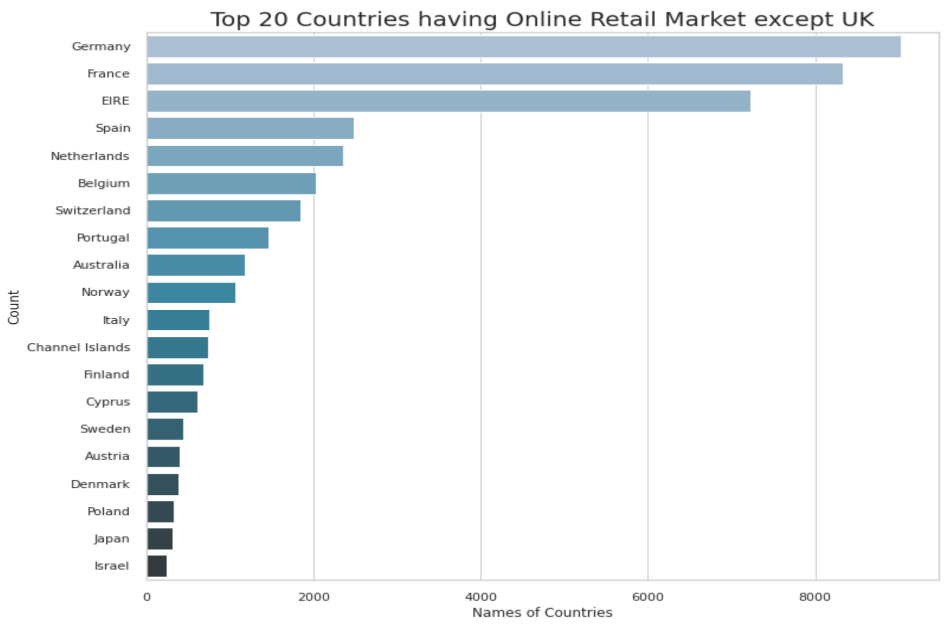


Image-12

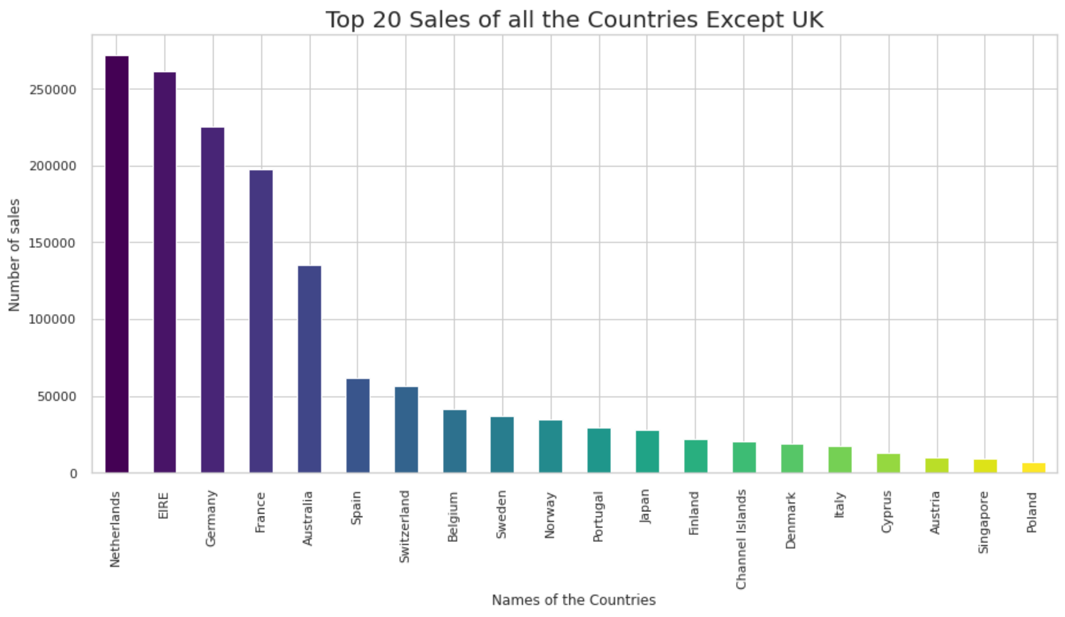
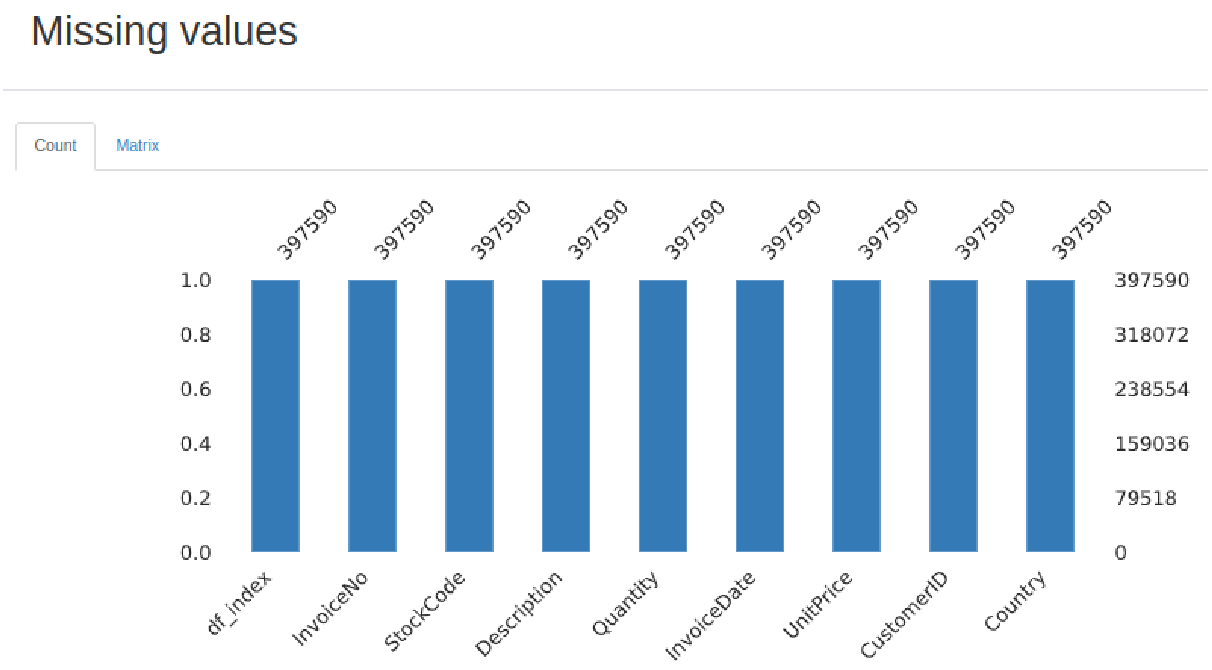


Image-13

1.8 Management of missing values

From the statistic analysis report (refer to section 1.4), I find below bar chat for missing values of all columns. It is clearly displayed that there are no missing values for any columns hence no treatment is required for missing values. For More details refer to detailed statistics reports for missing values.



2. Perform RFM Segmentation

RFM segmentation is a great method to identify the group of customers. It will help the marketing department determine the required customer treatment based on RFM values. It is better than traditional because there is no human intervention so no human errors and analyzes the whole dataset. RFM stands for recency, frequency, and monetary. RFM model help create segmentation for customers, target the customer base for the segmentation, and helps to reposition the marketing team with customer interaction to increase the customer lifetime value.

2.1 Definition of RFM metrics

Let's define the terms recency, frequency, and monetary. ***Recency*** indicates the last when a customer purchases the product or service. A high recency score means a customer appreciates the product/service. ***Frequency*** represents how frequently customers opt for the services/product in a fixed time period. A customer with a high-frequency score is a loyal customer. ***Monetary*** shows how much customers spend on a product or service. A high monetary value represents customers who spend the highest amount of money on the product and service.

To derive the RFM values, I need to assign values to RFM for all customers. I have created the table with four tiers as follows:

Recency	**Frequency**	**Monetary**
R-Tier-1 (most recent)	F-Tier-1 (most frequent)	M-Tier-1 (highest spend)
R-Tier-2	F-Tier-2	M-Tier-2

R-Tier-3	F-Tier-2	M-Tier-3
R-Tier-4 (least recent)	M-Tier-4 (low transaction)	M-Tier-4 (low Spend)

Table-1

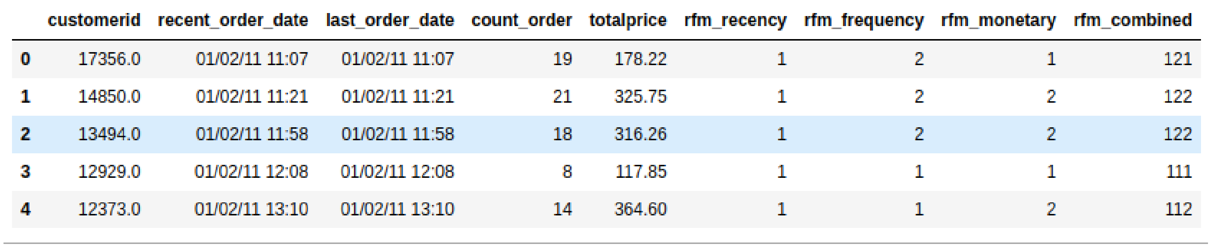
Using above, I will derive customer's tier values for RFM attributes based on given data. Possible Customer segments are 64 (4X4X4). After Customer RFM segmentation, I will create customer groups based on RFM values. It will help me create the visualization. Before implementing the customer group, I need to define the group. As shown in table-2, I have created 11 customer groups based on the below definition.

Customer Segment	**Recency Score Range**	**Frequency & Monetary Combined Score Range**
Champions	4-5	4-5
Loyal Customers	2-5	3-5
Potential Loyalist	3-5	1-3
Recent Customers	4-5	'0-1
Promising	3-4	'0-1
Customers Needing Attention	2-3	2-3
About To Sleep	2-3	0-2
At Risk	0-2	2-5
Can't Lose Them	'0-1	4-5
Hibernating	1-2	1-2
Lost	0-2	0-2

Table-2

2.2 Implementation in Python correct metrics

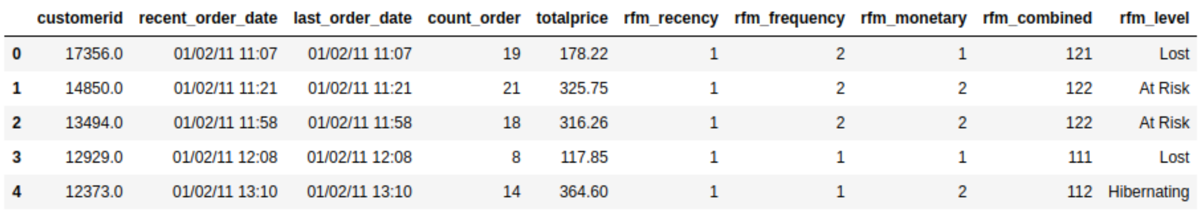
To derive RFM values for each customer, I use python and sqlite3. In the database, There are transaction-related details are available. I group all transaction details with customerID, then get MAX invoice date (recent order date), the sum of unit price multiplied by the quantity (total price), and order count (total order) using group by, aggregation function like max(), count() and other function like sum() in select query. All three values represent recency, monetary, and frequency respectively. After that, I will assign tier values for the RFM attribute to each customer using NTILE() function SQL query. Next, load data from SQL query to data frame and display. Refer to Image-14 for customers with RFM values. For the code snippet, Refer to Image-18 in the appendix. [Refer to code in lines 200-239].



	customerid	recent_order_date	last_order_date	count_order	totalprice	rfm_recency	rfm_frequency	rfm_monetary	rfm_combined
0	17356.0	01/02/11 11:07	01/02/11 11:07	19	178.22	1	2	1	121
1	14850.0	01/02/11 11:21	01/02/11 11:21	21	325.75	1	2	2	122
2	13494.0	01/02/11 11:58	01/02/11 11:58	18	316.26	1	2	2	122
3	12929.0	01/02/11 12:08	01/02/11 12:08	8	117.85	1	1	1	111
4	12373.0	01/02/11 13:10	01/02/11 13:10	14	364.60	1	1	2	112

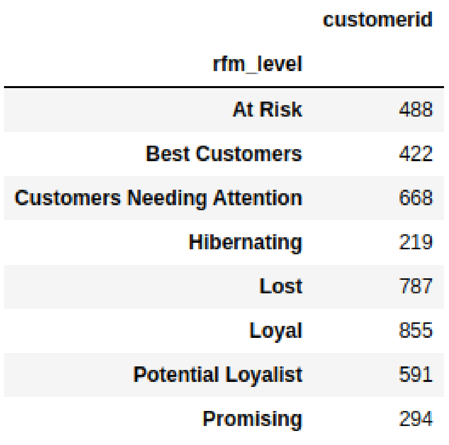
Image-14

Using RFM values, I have created 11 customer groups which are defined in the table-2. To assign a group to each customer, I write the python function rfm_level() and apply it to all the rows of the data frame. For the code snippet, Refer to Image-19 in the appendix. [Refer to code in lines 247-266]. Data frame with customer group look like as below Image-15:



	customerid	recent_order_date	last_order_date	count_order	totalprice	rfm_recency	rfm_frequency	rfm_monetary	rfm_combined	rfm_level
0	17356.0	01/02/11 11:07	01/02/11 11:07	19	178.22	1	2	1	121	Lost
1	14850.0	01/02/11 11:21	01/02/11 11:21	21	325.75	1	2	2	122	At Risk
2	13494.0	01/02/11 11:58	01/02/11 11:58	18	316.26	1	2	2	122	At Risk
3	12929.0	01/02/11 12:08	01/02/11 12:08	8	117.85	1	1	1	111	Lost
4	12373.0	01/02/11 13:10	01/02/11 13:10	14	364.60	1	1	2	112	Hibernating

Image-15



	customerid
rfm_level	
At Risk	488
Best Customers	422
Customers Needing Attention	668
Hibernating	219
Lost	787
Loyal	855
Potential Loyalist	591
Promising	294

Image-16

As shown in Image-16, I calculate the customer count in each group using the group() and agg() functions of the data frame. 855 customers are loyal customers, whereas 787 customers are lost. I pass the result of grouping and aggregation to the plot function to create a visualization of all groups. refer to Image 17. [Refer to code in lines 274-290].

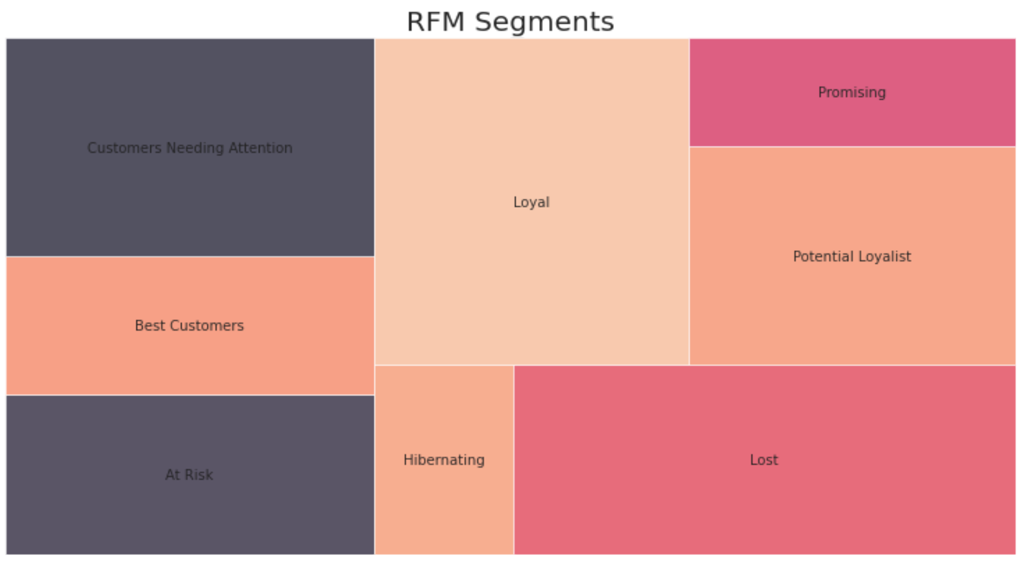


Image-17

3 Customer segmentation with k-means

K-mean is an unsupervised algorithm to create clusters for the given database. The Input features to K-mean clustering are the data derived from the RFM modeling of the customers. I will use PCA component analysis to select features for K-mean modeling. To decide the ideal cluster number, I will use the elbow method. After clustering, I will visualize the customer data with clusters and discuss each cluster.

3.1 Build of K-Means Model in Python

From the RFM segmentation, I take columns for clustering. To select features, I perform the PCA analysis on the selected five columns. [Refer to code in lines 298-312]. Variances for all components are as below:

pca.explained_variance_ratio_: [**0.50254045, 0.20975561, 0.13658656,** 0.10297548, 0.0481419]
pca.n_components_: 5

After finding variance for all given features, I decided to select the first 3 components which cover the 83%+ total variance. After that, I use PCA data to build the K-mean model. In K-mean, K indicates the number of clusters for given data. For this problem, I used k=5 which is derived using the elbow method. Cluster-wise customer counts are as follows

Cluster-0:	1122 Customers
Cluster-1:	1349 Customers
Cluster-2:	12 Customers
Cluster-3:	1012 Customers
Cluster-4:	829 Customers

3.2 Correct Justification of K value

It is crucial to derive K value in K-mean clustering as it will decide the number of clusters. [Refer to code in lines 319-344] Below are steps for the elbow method:

1) Fit pca data into the K-mean model

2) Find kmeans.inertia_ from step-1 and add to list wcss
3) Iterate steps 1 & 2 with K values from 1 to 22.
4) After the 22nd Iteration, plot the wcss list using KElbowVisualizer from Yellowbrick Visualiser.

Refer to Image-17 for the line graph plot from step-4. It displays ideal K=5 values for the dataset after 22 iterations.

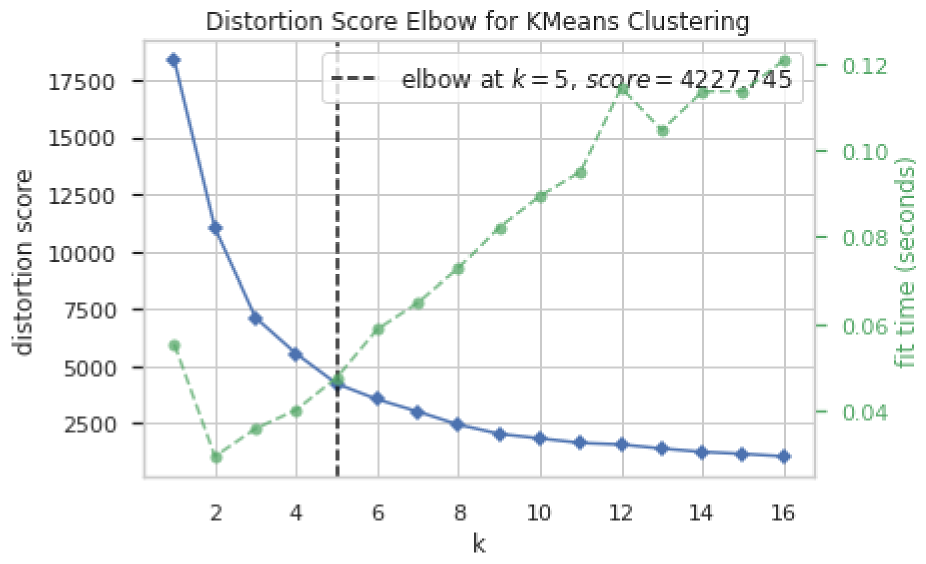
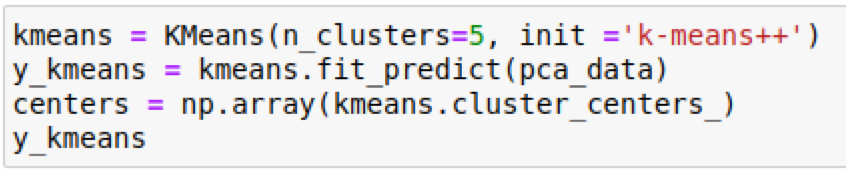


Image-17

3.3 Testing of K-Means Model in Python

As shown in Image-18, PCA data pass to Kmeans with k=5 and fit the data to get a cluster for all the transactions. [Refer to code in lines 350-392]



```
kmeans = KMeans(n_clusters=5, init ='k-means++')
y_kmeans = kmeans.fit_predict(pca_data)
centers = np.array(kmeans.cluster_centers_)
y_kmeans
```

Image-18

After deriving the clusters, I merge the actual data frame with pca data and cluster number. Refer to Image-19, visualization of customer clusters.

Cluster No	Customer Count
0	1122
1	1349
2	12
3	1012
4	829

Table-3

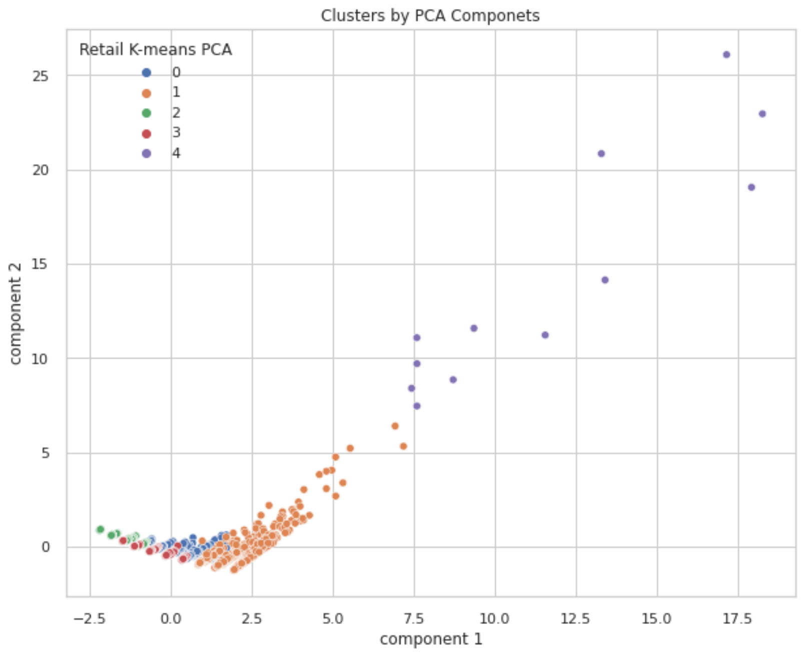


Image-19

Among five groups, cluster-1 has maximum customers, whereas cluster-2 has a minimum number of customers, but all are the best customer in cluster-2. Data of clusters 0,2 & 3 overlap and are concentrated in the graph. Cluster-0 and cluster-3 have customers which need more attention and the marketing team needs to reposition to convert them into best or loyal customers.

4. Review of Results

In task-2, I have calculated RFM values for given customers using 4 tiers. After that, using table-2, I created a segmentation of customers into 11 groups using the python function. In task-3, Use RFM values and other columns as inputs and assigned clusters to each customer using the K-means clustering unsupervised algorithm. In this section, I will try to derive business values from both segmentations and try to justify them to the marketing team.

4.1 Identification of business value customer segments

As an outcoming of RFM segmentation, We have 11 groups of customers which can be further four different categories to decide to improve overall performance.

1) **Platinum Customers:** This category of customers is best for our business because they are recently purchasing products, the quantity of production is higher than in other categories, and spent more money. They are almost 30% of total customers.
    - Best Customer: 422
    - Loyal: 855

2) **Gold Customers:** This category of customers is better for business because they are recently purchased products and we should target them to select our products in the future. They are 20.45% of total customers.
    - Potential Loyalist: 591
    - Promising: 294

3) **Silver Customers:** this category of customers is an area of concern for business their monitory and frequency score is good. They should be targeted to increase the profit of the business. They are 32% of overall customers.

- Customer Needing Attention: 668
- At-Risk: 488
- Hibernating: 219

4) **Bronze Customers:** these customers are low money spenders, and quantity is also low among all the customers. We might lose these customers. We need to study the details of those customers so we can approach our future customers in a better way. 18.20% of customers are lost customers from our dataset.

- Lost: 787

4.2 Correct Justification of their business value

Let's try to justify the customer segmentation with clusters I created using K-mean modeling. Cluster-2 and Cluster-0 container best and loyal customers and cluster-4 have promising and potential Loyalist customers.

As defined categories in section 4.2, Silver customers are spread between cluster 1 and cluster 3. I observed that few clusters have values that should be not part of the clusters. As per my observation, there is scope for improvement in K-mean clustering as some customers do not fall in the right clusters. I took the total price and quantity with the RFM score derived from RFM values to form clusters. I should pass the recent purchase date also to form a cluster or cover more variance in PCA analysis, Which could improve cluster performance.

5 Data Mart Design

Based on the RFM model and K- means clustering, I am going to create a data mart that will help the marketing department to do analysis.

5.1 Identification of Dimensions

To Identify the dimensions of customer segmentation, I build the attribute trees. There are three attribute trees: ***Invoice, RFM values, and time.*** The final attribute tree after combining these attribute trees is shown in Image-20.

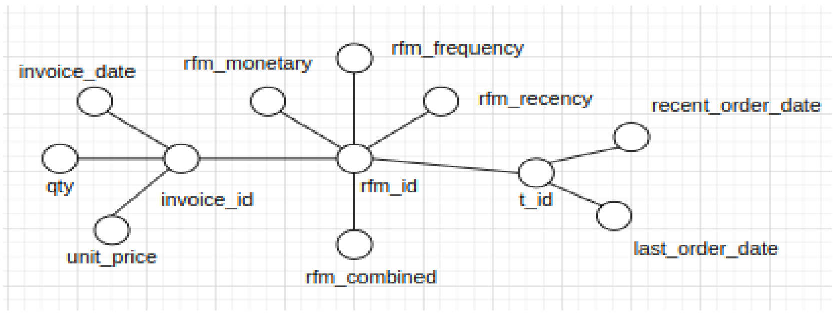


Image-20

Dimensions as below:

1. **Invoice:** invoice_date, qty, unit_price
2. **RFM:** rfm_monetary, rfm_frequency, rfm_recency.
3. **Time:** recent_order_date, last_order_date

5.2 Justification of Selected Dimensions

Dimension trees provide information about how facts can be aggregated for decision-making processes. Below is the justification of the selected dimensions.

1. **Invoice dimensions** provide information about the unit price and quantity of the product with the purchase date. It will provide information about customer monetary spending on products.
2. **RFM values** help to create segmentation of the given customers based on their score of rfm_monetary, rfm_frequency, and rfm_recency.
3. **Time_dimension** gives information about the recent and last purchase dates of the customs. It is crucial information to derive the recency and time duration between the first and latest purchase of customers.

5.3 Identification of Measures

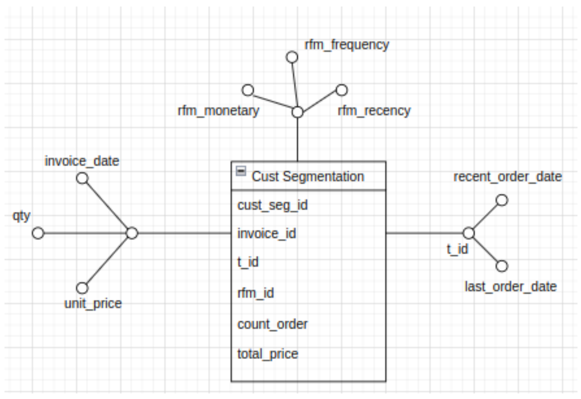


Image-21

Measures are basically numerical values associated with facts of DW. On Measures, I can apply aggregate functions like sum/Average/maximum/Minimum/Count.

Measures are part of fact tables. The fact table for the DFM model is shown in image-21. Measures for the fact table are as follows:

1. Count_order
2. total_price.

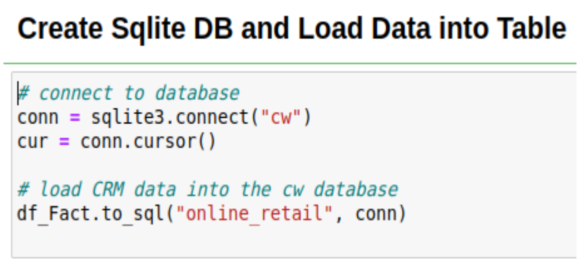
Abbreviations

RFM: Recency, Frequency, and Monetary
PCA: Principal component analysis

Reference

Uci.edu. (2010). *UCI Machine Learning Repository: Online Retail Data Set*. [online] Available at: https://archive.ics.uci.edu/ml/datasets/Online+Retail.

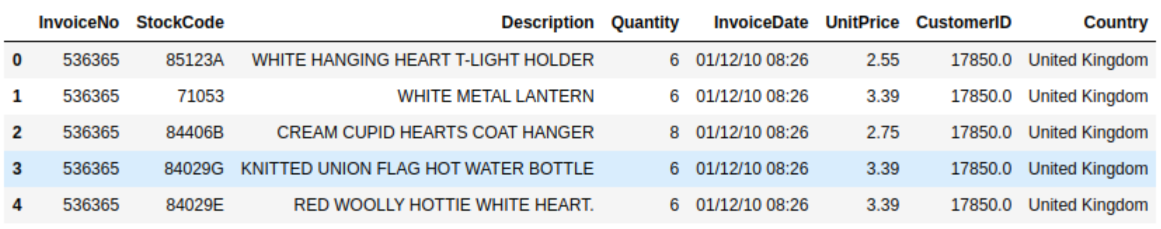
Appendix



Create Sqlite DB and Load Data into Table

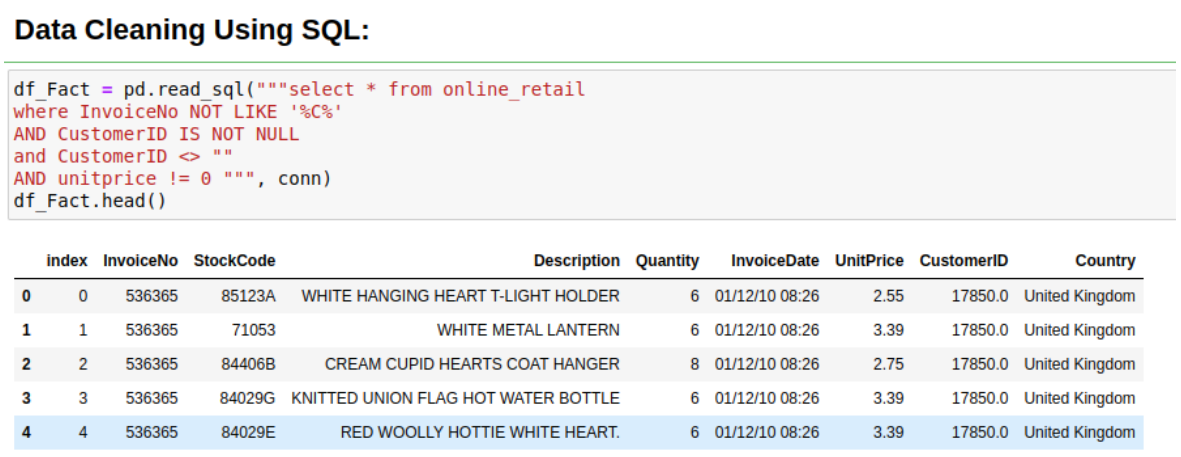
```
# connect to database
conn = sqlite3.connect("cw")
cur = conn.cursor()

# load CRM data into the cw database
df_Fact.to_sql("online_retail", conn)
```

Image-1



	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01/12/10 08:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	01/12/10 08:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01/12/10 08:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01/12/10 08:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01/12/10 08:26	3.39	17850.0	United Kingdom

Image-3

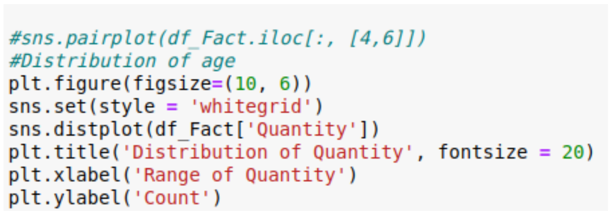


Data Cleaning Using SQL:

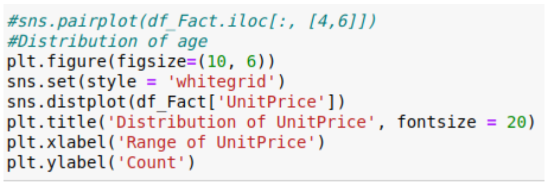
```
df_Fact = pd.read_sql("""select * from online_retail
where InvoiceNo NOT LIKE '%C%'
AND CustomerID IS NOT NULL
and CustomerID <> ""
AND unitprice != 0 """, conn)
df_Fact.head()
```

	index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01/12/10 08:26	2.55	17850.0	United Kingdom
1	1	536365	71053	WHITE METAL LANTERN	6	01/12/10 08:26	3.39	17850.0	United Kingdom
2	2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01/12/10 08:26	2.75	17850.0	United Kingdom
3	3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01/12/10 08:26	3.39	17850.0	United Kingdom
4	4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01/12/10 08:26	3.39	17850.0	United Kingdom

Image-4



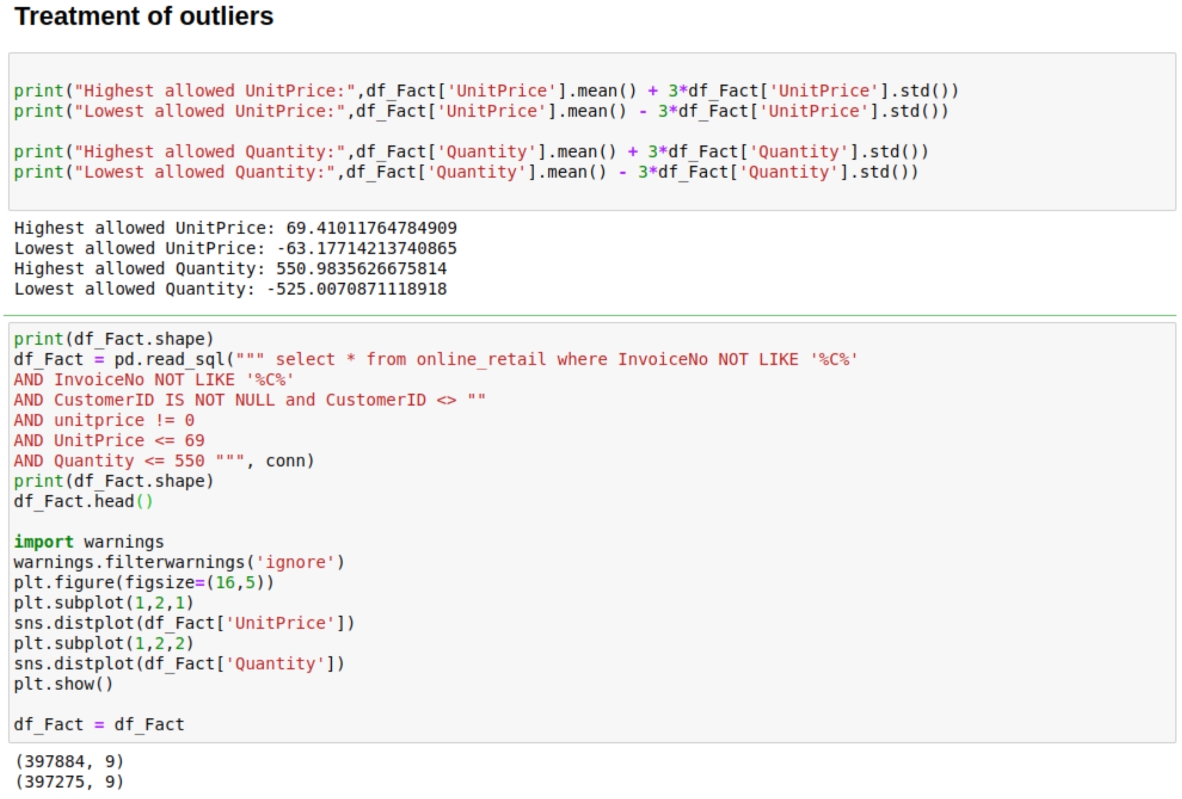
```
#sns.pairplot(df_Fact.iloc[:, [4,6]])
#Distribution of age
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(df_Fact['Quantity'])
plt.title('Distribution of Quantity', fontsize = 20)
plt.xlabel('Range of Quantity')
plt.ylabel('Count')
```



```
#sns.pairplot(df_Fact.iloc[:, [4,6]])
#Distribution of age
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(df_Fact['UnitPrice'])
plt.title('Distribution of UnitPrice', fontsize = 20)
plt.xlabel('Range of UnitPrice')
plt.ylabel('Count')
```

Image-7

Image-6



Treatment of outliers

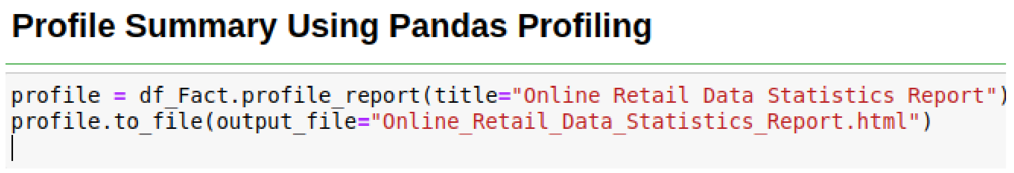
```
print("Highest allowed UnitPrice:",df_Fact['UnitPrice'].mean() + 3*df_Fact['UnitPrice'].std())
print("Lowest allowed UnitPrice:",df_Fact['UnitPrice'].mean() - 3*df_Fact['UnitPrice'].std())

print("Highest allowed Quantity:",df_Fact['Quantity'].mean() + 3*df_Fact['Quantity'].std())
print("Lowest allowed Quantity:",df_Fact['Quantity'].mean() - 3*df_Fact['Quantity'].std())
```

```
Highest allowed UnitPrice: 69.41011764784909
Lowest allowed UnitPrice: -63.17714213740865
Highest allowed Quantity: 550.9835626675814
Lowest allowed Quantity: -525.0070871118918
```

```
print(df_Fact.shape)
df_Fact = pd.read_sql(""" select * from online_retail where InvoiceNo NOT LIKE '%C%'
AND InvoiceNo NOT LIKE '%C%'
AND CustomerID IS NOT NULL and CustomerID <> ""
AND unitprice != 0
AND UnitPrice <= 69
AND Quantity <= 550 """, conn)
print(df_Fact.shape)
df_Fact.head()

import warnings
warnings.filterwarnings('ignore')
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(df_Fact['UnitPrice'])
plt.subplot(1,2,2)
sns.distplot(df_Fact['Quantity'])
plt.show()

df_Fact = df_Fact
```

```
(397884, 9)
(397275, 9)
```

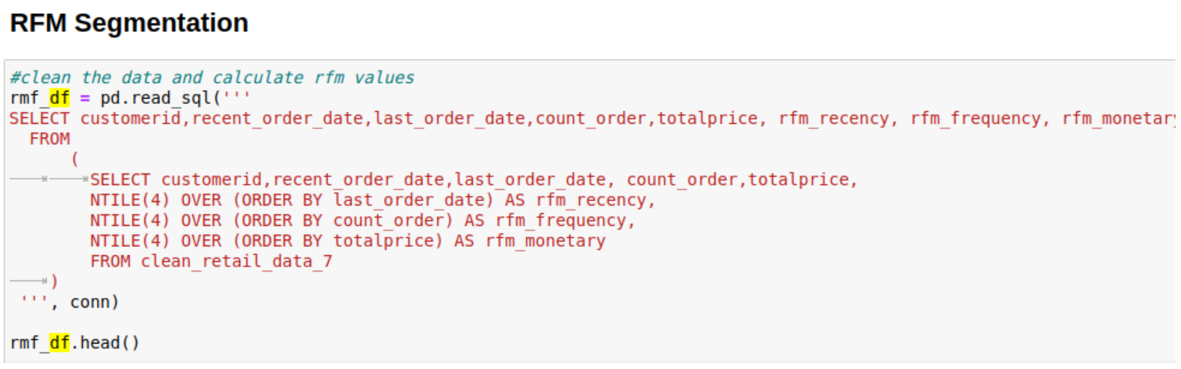
Image-9



Profile Summary Using Pandas Profiling

```
profile = df_Fact.profile_report(title="Online Retail Data Statistics Report")
profile.to_file(output_file="Online_Retail_Data_Statistics_Report.html")
```

Image-10

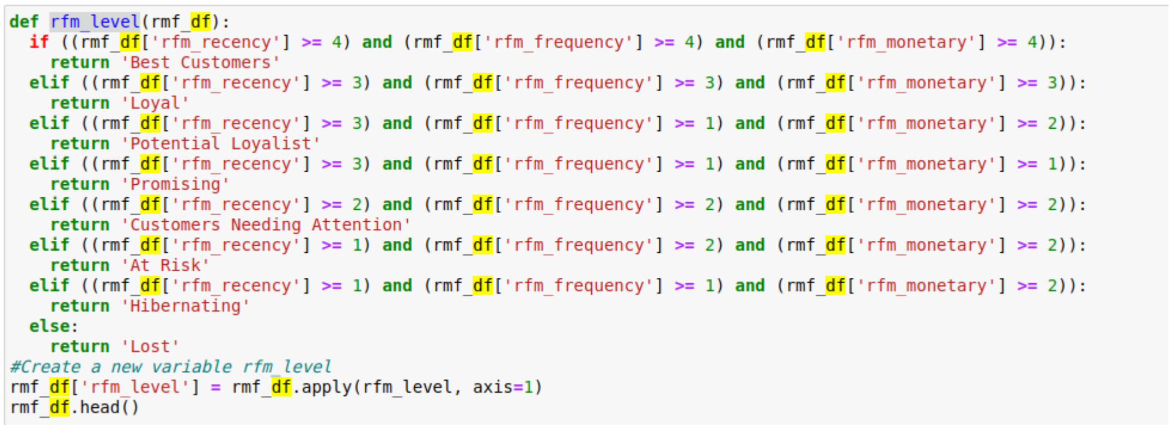


RFM Segmentation

```
#clean the data and calculate rfm values
rmf_df = pd.read_sql('''
SELECT customerid,recent_order_date,last_order_date,count_order,totalprice, rfm_recency, rfm_frequency, rfm_monetar
  FROM
      (
        SELECT customerid,recent_order_date,last_order_date, count_order,totalprice,
        NTILE(4) OVER (ORDER BY last_order_date) AS rfm_recency,
        NTILE(4) OVER (ORDER BY count_order) AS rfm_frequency,
        NTILE(4) OVER (ORDER BY totalprice) AS rfm_monetary
        FROM clean_retail_data_7
      )
 ''', conn)

rmf_df.head()
```

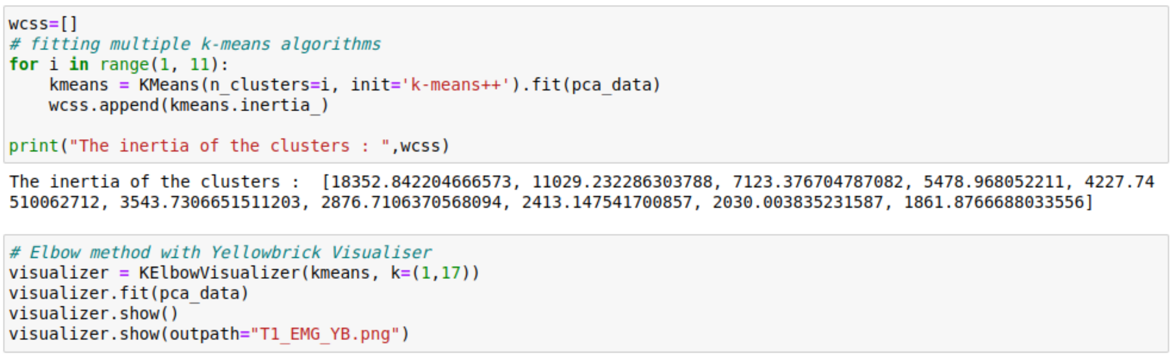
Image-22

Customer Segment based on RFM ScoresCustomer Segment



```python
def rfm_level(rmf_df):
    if ((rmf_df['rfm_recency'] >= 4) and (rmf_df['rfm_frequency'] >= 4) and (rmf_df['rfm_monetary'] >= 4)):
        return 'Best Customers'
    elif ((rmf_df['rfm_recency'] >= 3) and (rmf_df['rfm_frequency'] >= 3) and (rmf_df['rfm_monetary'] >= 3)):
        return 'Loyal'
    elif ((rmf_df['rfm_recency'] >= 3) and (rmf_df['rfm_frequency'] >= 1) and (rmf_df['rfm_monetary'] >= 2)):
        return 'Potential Loyalist'
    elif ((rmf_df['rfm_recency'] >= 3) and (rmf_df['rfm_frequency'] >= 1) and (rmf_df['rfm_monetary'] >= 1)):
        return 'Promising'
    elif ((rmf_df['rfm_recency'] >= 2) and (rmf_df['rfm_frequency'] >= 2) and (rmf_df['rfm_monetary'] >= 2)):
        return 'Customers Needing Attention'
    elif ((rmf_df['rfm_recency'] >= 1) and (rmf_df['rfm_frequency'] >= 2) and (rmf_df['rfm_monetary'] >= 2)):
        return 'At Risk'
    elif ((rmf_df['rfm_recency'] >= 1) and (rmf_df['rfm_frequency'] >= 1) and (rmf_df['rfm_monetary'] >= 2)):
        return 'Hibernating'
    else:
        return 'Lost'
#Create a new variable rfm_level
rmf_df['rfm_level'] = rmf_df.apply(rfm_level, axis=1)
rmf_df.head()
```

Image-23



```python
wcss=[]
# fitting multiple k-means algorithms
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++').fit(pca_data)
    wcss.append(kmeans.inertia_)

print("The inertia of the clusters : ",wcss)
```

```
The inertia of the clusters :  [18352.842204666573, 11029.232286303788, 7123.376704787082, 5478.968052211, 4227.74
510062712, 3543.7306651511203, 2876.7106370568094, 2413.147541700857, 2030.003835231587, 1861.8766688033556]
```

```python
# Elbow method with Yellowbrick Visualiser
visualizer = KElbowVisualizer(kmeans, k=(1,17))
visualizer.fit(pca_data)
visualizer.show()
visualizer.show(outpath="T1_EMG_YB.png")
```

Image-24