Finding best Customer using Customer Segmentation analysis in Python

1. Summary:

- Found top customers for retailer using RFM segmentation technique which uses past data to divide customers into groups.
- For this analysis we have taken Basket Market Data Retail sales data. We fill find top customers for this retailor.

2. Data:

Invoice No: Unique number assigned to order.

StockCode: Code assigned to each product

Quantity: Number of products ordered.

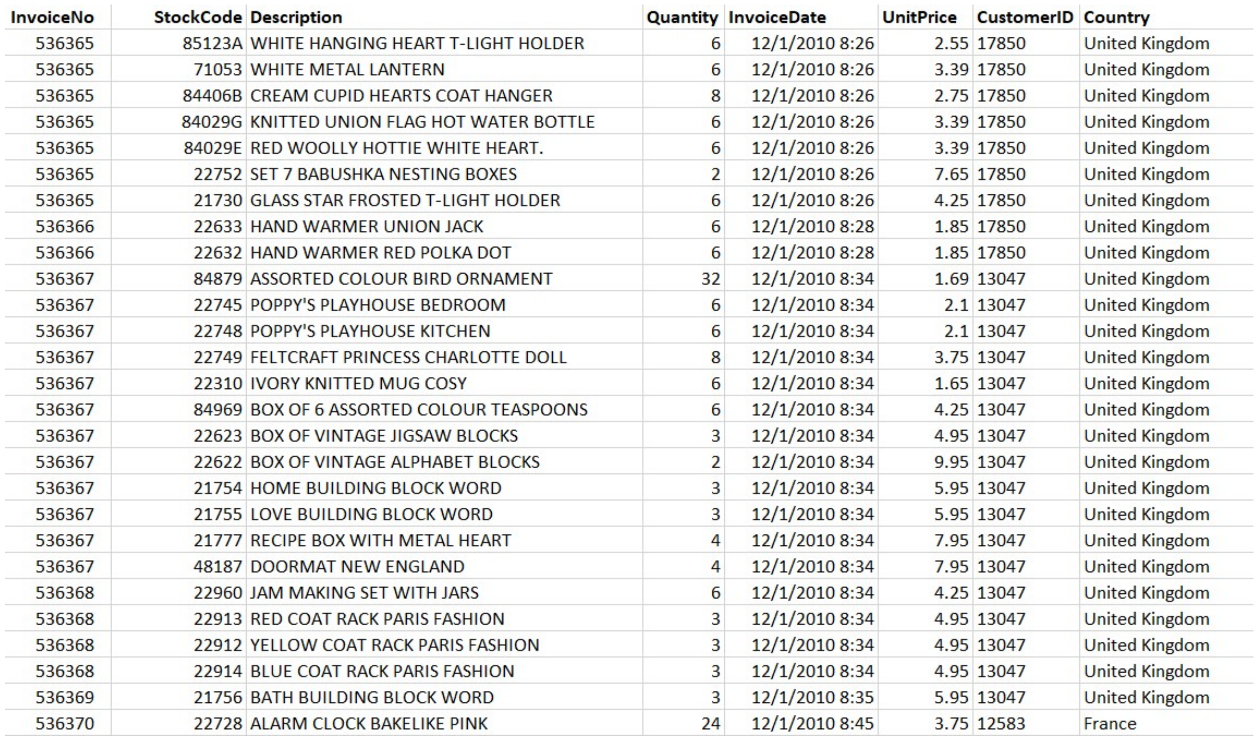
InvocieDate: The purchased order date.

Unit Price: The price of each product.

Customer ID: ID assinged to each customer to differetiate between other customers.

Country: Customer belong to different customers.

Below is the screenshot of the Data used in the analysis



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
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010 8:34	3.75	13047	United Kingdom
536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047	United Kingdom
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010 8:34	4.25	13047	United Kingdom
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047	United Kingdom
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047	United Kingdom
536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010 8:34	7.95	13047	United Kingdom
536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010 8:34	7.95	13047	United Kingdom
536368	22960	JAM MAKING SET WITH JARS	6	12/1/2010 8:34	4.25	13047	United Kingdom
536368	22913	RED COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom
536368	22912	YELLOW COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom
536368	22914	BLUE COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom
536369	21756	BATH BUILDING BLOCK WORD	3	12/1/2010 8:35	5.95	13047	United Kingdom
536370	22728	ALARM CLOCK BAKELIKE PINK	24	12/1/2010 8:45	3.75	12583	France

Data Preprocessing:

1. Missing values in important columns;
2. Customers' distribution in each country;
3. Unit price and Quantity should > 0;

4. Invoice date should < today.

- df1.Country.nunique()

  38

- There were 38 unique countries as follows:

  df1.Country.unique()

```
array([‘United Kingdom’, ‘France’, ‘Australia’, ‘Netherlands’, ‘Germany’,
 ‘Norway’, ‘EIRE’, ‘Switzerland’, ‘Spain’, ‘Poland’, ‘Portugal’,
 ‘Italy’, ‘Belgium’, ‘Lithuania’, ‘Japan’, ‘Iceland’,
 ‘Channel Islands’, ‘Denmark’, ‘Cyprus’, ‘Sweden’, ‘Austria’,
 ‘Israel’, ‘Finland’, ‘Bahrain’, ‘Greece’, ‘Hong Kong’, ‘Singapore’,
 ‘Lebanon’, ‘United Arab Emirates’, ‘Saudi Arabia’, ‘Czech Republic’,
 ‘Canada’, ‘Unspecified’, ‘Brazil’, ‘USA’, ‘European Community’,
 ‘Malta’, ‘RSA’], dtype=object
```

```
customer_country=df1[['Country','CustomerID']].drop_duplicates()
customer_country.groupby(['Country'])['CustomerID'].aggregate('count').reset_index().sort_values('CustomerI
D', ascending=False)
```

- There are 133,600 missing values in the CustomerID column, and since our analysis is based on customers, we will remove these missing values.
  df1 = df1[pd.notnull(df1['CustomerID'])]

- Check the minimum values in UnitPrice and Quantity column.
  df1 = df1[pd.notnull(df1['CustomerID'])]
  0.0
  df1.Quantity.min()
  -80995
- Remove the negative values in Quantity column.
  df1 = df1[(df1['Quantity']>0)]
  df1.shape
  df1.info()

- After cleaning up the data, we are now dealing with 354,345 rows and 8 columns.

  Check unique value for each column.

```
def unique_counts(df1):

 for i in df1.columns:

 count = df1[i].nunique()

  print(i, ": ", count)
```

```
unique_counts(df1)

InvoiceNo : 16649

StockCode : 3645

Description : 3844

Quantity : 294

InvoiceDate : 15615

UnitPrice : 403

CustomerID : 3921

Country : 1
```

- Add a column for total price.

```
df1['TotalPrice'] = df1['Quantity'] * df1['UnitPrice']
```

- Find out the first and last order dates in the data.

```
df1['InvoiceDate'].min()
```

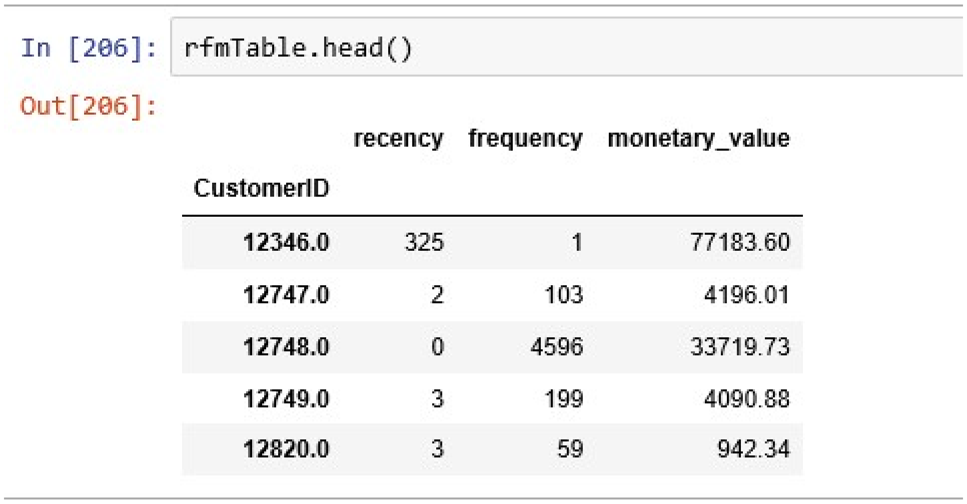
3. Segmentation Method(RFM) to find top customers:

RFM segmentation starts from here.

Create a RFM table

```
rfmTable = df1.groupby('CustomerID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days, 'InvoiceNo': lambda
x: len(x), 'TotalPrice': lambda x: x.sum()})
rfmTable['InvoiceDate'] = rfmTable['InvoiceDate'].astype(int)
rfmTable.rename(columns={'InvoiceDate': 'recency',
                    'InvoiceNo': 'frequency',
                    'TotalPrice': 'monetary_value'}, inplace=True)
```

Calculate RFM metrics for each customer



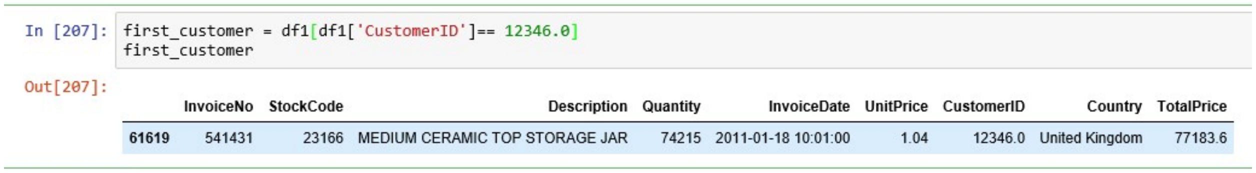
```
In [206]: rfmTable.head()
Out[206]:
```

CustomerID	recency	frequency	monetary_value
12346.0	325	1	77183.60
12747.0	2	103	4196.01
12748.0	0	4596	33719.73
12749.0	3	199	4090.88
12820.0	3	59	942.34

Interpretation:

- CustomerID 12346 has frequency: 1, monetary value: $77,183.60 and recency: 325 days.
- CustomerID 12747 has frequency: 103, monetary value: $4,196.01 and recency: 2 days

Let’s check the details of the first customer.



```
In [207]: first_customer = df1[df1['CustomerID']== 12346.0]
          first_customer
Out[207]:
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346.0	United Kingdom	77183.6

The first customer has shopped only once, bought one product at a huge quantity(74,215). The unit price is very low; perhaps a clearance sale.

- **<u>Split the metrics</u>**

The easiest way to split metrics into segments is by using quartiles.

1. This gives us a starting point for the detailed analysis.
2. 4 segments are easy to understand and explain.

quantiles = rfmTable.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()

- **<u>Create a segmented RFM table</u>**

segmented_rfm = rfmTable

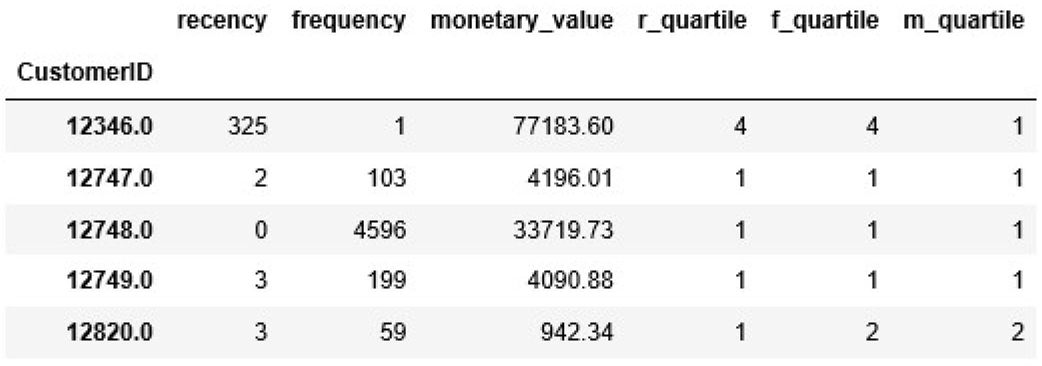
The lowest recency, highest frequency and monetary amounts are our best customers.

```
def RScore(x,p,d):
   if x <= d[p][0.25]:
      return 1
   elif x <= d[p][0.50]:
      return 2
   elif x <= d[p][0.75]:
      return 3
   else:
      return 4

def FMScore(x,p,d):
   if x <= d[p][0.25]:
      return 4
   elif x <= d[p][0.50]:
      return 3
   elif x <= d[p][0.75]:
      return 2
   else:
      return 1
```

- **<u>Add segment numbers to the newly created segmented RFM table</u>**

```
segmented_rfm['r_quartile'] = segmented_rfm['recency'].apply(RScore, args=('recency',quantiles,))
segmented_rfm['f_quartile'] = segmented_rfm['frequency'].apply(FMScore, args=('frequency',quantiles,))
segmented_rfm['m_quartile'] = segmented_rfm['monetary_value'].apply(FMScore,
args=('monetary_value',quantiles,))
segmented_rfm.head()
```

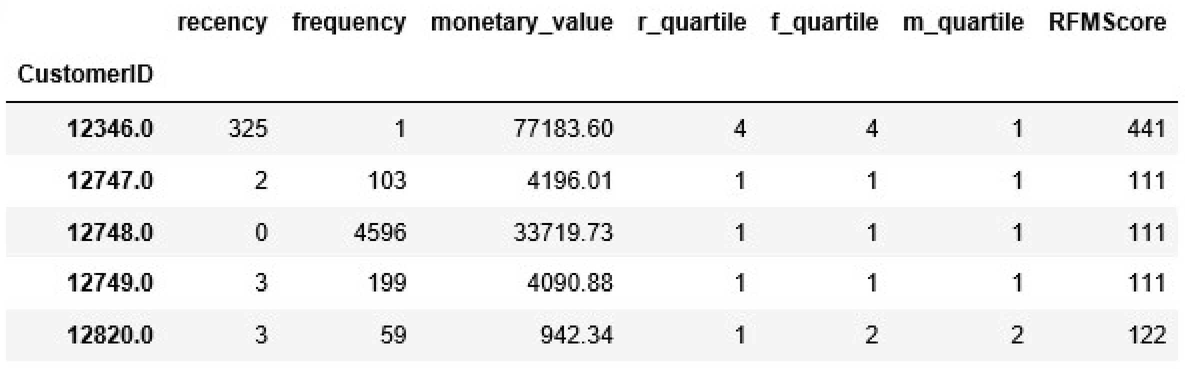


	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile
CustomerID						
12346.0	325	1	77183.60	4	4	1
12747.0	2	103	4196.01	1	1	1
12748.0	0	4596	33719.73	1	1	1
12749.0	3	199	4090.88	1	1	1
12820.0	3	59	942.34	1	2	2

RFM segments split the customer base into an imaginary 3D cube which is hard to visualize. However, we can sort it out.

Add a new column to combine RFM score: 111 is the highest score as we determined earlier.

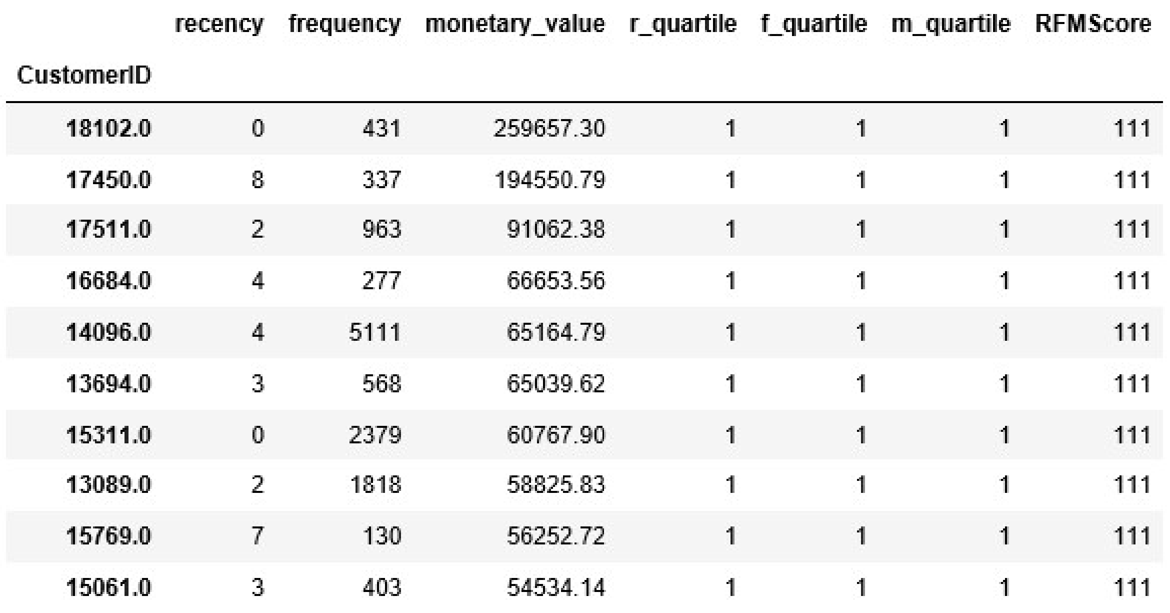
```
segmented_rfm['RFMScore'] = segmented_rfm.r_quartile.map(str)
                    + segmented_rfm.f_quartile.map(str)
                    + segmented_rfm.m_quartile.map(str)
segmented_rfm.head()
```



CustomerID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore
12346.0	325	1	77183.60	4	4	1	441
12747.0	2	103	4196.01	1	1	1	111
12748.0	0	4596	33719.73	1	1	1	111
12749.0	3	199	4090.88	1	1	1	111
12820.0	3	59	942.34	1	2	2	122

It is obvious that the first customer is not our best customer at all. Who are the top 10 of our best customers!

```
segmented_rfm[segmented_rfm['RFMScore']=='111'].sort_values('monetary_value',
ascending=False).head(10)
```



CustomerID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore
18102.0	0	431	259657.30	1	1	1	111
17450.0	8	337	194550.79	1	1	1	111
17511.0	2	963	91062.38	1	1	1	111
16684.0	4	277	66653.56	1	1	1	111
14096.0	4	5111	65164.79	1	1	1	111
13694.0	3	568	65039.62	1	1	1	111
15311.0	0	2379	60767.90	1	1	1	111
13089.0	2	1818	58825.83	1	1	1	111
15769.0	7	130	56252.72	1	1	1	111
15061.0	3	403	54534.14	1	1	1	111