```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from datetime import timedelta
from pandas import ExcelWriter
```

```
pip install openpyxl
```

```
Collecting openpyxlNote: you may need to restart the kernel to use
updated packages.

  Downloading openpyxl-3.1.2-py2.py3-none-any.whl (249 kB)
     ---------------------------------------- 0.0/250.0 kB ? eta
-:--:--
     -----------------------------------  245.8/250.0 kB 15.7 MB/s
eta 0:00:01
     ---------------------------------------- 250.0/250.0 kB 3.1 MB/s
eta 0:00:00
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.2
```

```python
data = pd.read_excel('Online Retail.xlsx')
data.head()
```

```
  InvoiceNo StockCode                          Description  Quantity

0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6  \
1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
```

Data Cleaning

Perform descriptive analytics on the given data

- Check for missing data
- Remove duplicate data records
- Perform descriptive analytics on the given data

```
data.describe().T
```

```
                count                             mean
min
Quantity     541909.0                          9.55225             -
80995.0  \
InvoiceDate    541909  2011-07-04 13:34:57.156386048  2010-12-01
08:26:00
UnitPrice    541909.0                         4.611114             -
11062.06
CustomerID   406829.0                      15287.69057
12346.0

                              25%                  50%
75%
Quantity                      1.0                  3.0
10.0  \
InvoiceDate  2011-03-28 11:34:00  2011-07-19 17:17:00  2011-10-19
11:27:00
UnitPrice                    1.25                 2.08
4.13
CustomerID                13953.0              15152.0
16791.0

                              max          std
Quantity                  80995.0   218.081158
InvoiceDate  2011-12-09 12:50:00          NaN
UnitPrice                 38970.0    96.759853
CustomerID                18287.0  1713.600303
```

```
# Dropping rows with negative quantity
data.drop(data[data['Quantity']<=0].index, inplace=True)

# Dropping rows with $0.00 sales
data.drop(data[data['UnitPrice'] == 0].index, inplace=True)

# Checking for duplicate rows in database

duplicate = data[data.duplicated()]
print(f'There are {len(duplicate)} duplacate rows in this data file')
```

```
There are 5226 duplacate rows in this data file
```

```python
# Removing duplicate rows in database and re-checking to be sure the database is clear of duplicates.

data = data.drop_duplicates()
duplicate_check = data[data.duplicated()]
print(f'There are {len(duplicate_check)} duplacate rows in this data file')
```

```
There are 0 duplacate rows in this data file
```

```python
# Checking for missing values

data.isna().any()
```

```
InvoiceNo      False
StockCode      False
Description    False
Quantity       False
InvoiceDate    False
UnitPrice      False
CustomerID      True
Country        False
dtype: bool
```

```python
Customer_id_isna = data[pd.isnull(data['CustomerID'])]
print('There are: ' +
str(len(pd.unique(Customer_id_isna['InvoiceNo']))) + ' Invoices with no Customer ID')
Customer_id_isna
```

```
There are: 1430 Invoices with no Customer ID

       InvoiceNo StockCode                      Description  Quantity

1443      536544     21773  DECORATIVE ROSE BATHROOM BOTTLE         1
\
1444      536544     21774  DECORATIVE CATS BATHROOM BOTTLE         2

1445      536544     21786              POLKADOT RAIN HAT           4

1446      536544     21787            RAIN PONCHO RETROSPOT         2

1447      536544     21790               VINTAGE SNAP CARDS         9

...          ...       ...                              ...       ...

541536    581498    85099B          JUMBO BAG RED RETROSPOT         5

541537    581498    85099C   JUMBO  BAG BAROQUE BLACK WHITE         4
```

```
541538    581498     85150    LADIES & GENTLEMEN METAL SIGN         1

541539    581498     85174                 S/4 CACTI CANDLES         1

541540    581498       DOT                    DOTCOM POSTAGE         1


               InvoiceDate  UnitPrice  CustomerID         Country
1443   2010-12-01 14:32:00       2.51         NaN  United Kingdom
1444   2010-12-01 14:32:00       2.51         NaN  United Kingdom
1445   2010-12-01 14:32:00       0.85         NaN  United Kingdom
1446   2010-12-01 14:32:00       1.66         NaN  United Kingdom
1447   2010-12-01 14:32:00       1.66         NaN  United Kingdom
...                    ...        ...         ...             ...
541536 2011-12-09 10:26:00       4.13         NaN  United Kingdom
541537 2011-12-09 10:26:00       4.13         NaN  United Kingdom
541538 2011-12-09 10:26:00       4.96         NaN  United Kingdom
541539 2011-12-09 10:26:00      10.79         NaN  United Kingdom
541540 2011-12-09 10:26:00    1714.17         NaN  United Kingdom

[132188 rows x 8 columns]
```

```python
# Will drop all rows with no Customer ID

data.dropna(subset=['CustomerID'], inplace=True)

# Checking for missing values again

data.isna().any()
```

```
InvoiceNo      False
StockCode      False
Description    False
Quantity       False
InvoiceDate    False
UnitPrice      False
CustomerID     False
Country        False
dtype: bool
```

Data Transformation

Perform cohort analysis (a cohort is a group of subjects that share a defining characteristic). Observe how a cohort behaves across time and compare it to other cohorts.

- Create month cohorts and analyze active customers for each cohort
- Analyze the retention rate of customers

```python
data['Month'] = data['InvoiceDate'].dt.to_period('M')
data.head()
```

```
  InvoiceNo StockCode                          Description  Quantity

0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6  \
1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


          InvoiceDate  UnitPrice  CustomerID         Country    Month

0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom  2010-12

1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12

2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom  2010-12

3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12

4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12
```

```python
# Convert to InvoiceDate to Year-Month format
data['month_year'] = data['InvoiceDate'].dt.to_period('M')
data['month_year'].nunique()
```

```
13
```

```python
month_cohort = data.groupby('month_year')['CustomerID'].nunique()
month_cohort
```

```
month_year
2010-12     885
2011-01     741
2011-02     758
2011-03     974
2011-04     856
2011-05    1056
2011-06     991
2011-07     949
2011-08     935
2011-09    1266
2011-10    1364
2011-11    1664
2011-12     615
Freq: M, Name: CustomerID, dtype: int64
```

```python
data['cohort'] = data.groupby('CustomerID')['InvoiceDate'] \
                 .transform('min') \
                 .dt.to_period('M')

data.tail(50)
```

```
       InvoiceNo StockCode                          Description
Quantity
541859    581580     37500          TEA TIME TEAPOT IN GIFT BOX
1  \
541860    581581     23562  SET OF 6 RIBBONS PERFECTLY PRETTY
6
541861    581581     23561                SET OF 6 RIBBONS PARTY
6
541862    581581     23681          LUNCH BAG RED VINTAGE DOILY
10
541863    581582     23552         BICYCLE PUNCTURE REPAIR KIT
6
541864    581582     23498                CLASSIC BICYCLE CLIPS
12
541865    581583     20725              LUNCH BAG RED RETROSPOT
40
541866    581583     85038      6 CHOCOLATE LOVE HEART T-LIGHTS
36
541867    581584     20832     RED FLOCK LOVE HEART PHOTO FRAME
72
541868    581584     85038      6 CHOCOLATE LOVE HEART T-LIGHTS
48
541869    581585     22481        BLACK TEA TOWEL CLASSIC DESIGN
12
541870    581585     22915         ASSORTED BOTTLE TOP  MAGNETS
24
541871    581585     22178      VICTORIAN GLASS HANGING T-LIGHT
12
541872    581585     22460        EMBOSSED GLASS TEALIGHT HOLDER
12
541873    581585     84832     ZINC WILLIE WINKIE  CANDLE STICK
24
541874    581585     23084                   RABBIT NIGHT LIGHT
12
541875    581585     84879         ASSORTED COLOUR BIRD ORNAMENT
16
541876    581585     84945   MULTI COLOUR SILVER T-LIGHT HOLDER
24
541877    581585     22113          GREY HEART HOT WATER BOTTLE
4
541878    581585     23356                LOVE HOT WATER BOTTLE
3
541879    581585     22726           ALARM CLOCK BAKELIKE GREEN
8
541880    581585     22727            ALARM CLOCK BAKELIKE RED
```

4
541881 581585 16016 LARGE CHINESE STYLE SCISSOR
10
541882 581585 21916 SET 12 RETRO WHITE CHALK STICKS
24
541883 581585 84692 BOX OF 24 COCKTAIL PARASOLS
25
541884 581585 84946 ANTIQUE SILVER T-LIGHT GLASS
12
541885 581585 21684 SMALL MEDINA STAMPED METAL BOWL
12
541886 581585 22398 MAGNETS PACK OF 4 SWALLOWS
12
541887 581585 23328 SET 6 SCHOOL MILK BOTTLES IN CRATE
4
541888 581585 23145 ZINC T-LIGHT HOLDER STAR LARGE
12
541889 581585 22466 FAIRY TALE COTTAGE NIGHT LIGHT
12
541890 581586 22061 LARGE CAKE STAND HANGING STRAWBERY
8
541891 581586 23275 SET OF 3 HANGING OWLS OLLIE BEAK
24
541892 581586 21217 RED RETROSPOT ROUND CAKE TINS
24
541893 581586 20685 DOORMAT RED RETROSPOT
10
541894 581587 22631 CIRCUS PARADE LUNCH BOX
12
541895 581587 22556 PLASTERS IN TIN CIRCUS PARADE
12
541896 581587 22555 PLASTERS IN TIN STRONGMAN
12
541897 581587 22728 ALARM CLOCK BAKELIKE PINK
4
541898 581587 22727 ALARM CLOCK BAKELIKE RED
4
541899 581587 22726 ALARM CLOCK BAKELIKE GREEN
4
541900 581587 22730 ALARM CLOCK BAKELIKE IVORY
4
541901 581587 22367 CHILDRENS APRON SPACEBOY DESIGN
8
541902 581587 22629 SPACEBOY LUNCH BOX
12
541903 581587 23256 CHILDRENS CUTLERY SPACEBOY
4
541904 581587 22613 PACK OF 20 SPACEBOY NAPKINS
12
541905 581587 22899 CHILDREN'S APRON DOLLY GIRL

```
6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL
4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT
3

               InvoiceDate  UnitPrice  CustomerID         Country
Month
541859 2011-12-09 12:20:00       4.95     12748.0  United Kingdom
2011-12  \
541860 2011-12-09 12:20:00       2.89     17581.0  United Kingdom
2011-12
541861 2011-12-09 12:20:00       2.89     17581.0  United Kingdom
2011-12
541862 2011-12-09 12:20:00       1.65     17581.0  United Kingdom
2011-12
541863 2011-12-09 12:21:00       2.08     17581.0  United Kingdom
2011-12
541864 2011-12-09 12:21:00       1.45     17581.0  United Kingdom
2011-12
541865 2011-12-09 12:23:00       1.45     13777.0  United Kingdom
2011-12
541866 2011-12-09 12:23:00       1.85     13777.0  United Kingdom
2011-12
541867 2011-12-09 12:25:00       0.72     13777.0  United Kingdom
2011-12
541868 2011-12-09 12:25:00       1.85     13777.0  United Kingdom
2011-12
541869 2011-12-09 12:31:00       0.39     15804.0  United Kingdom
2011-12
541870 2011-12-09 12:31:00       0.19     15804.0  United Kingdom
2011-12
541871 2011-12-09 12:31:00       1.95     15804.0  United Kingdom
2011-12
541872 2011-12-09 12:31:00       1.25     15804.0  United Kingdom
2011-12
541873 2011-12-09 12:31:00       0.85     15804.0  United Kingdom
2011-12
541874 2011-12-09 12:31:00       2.08     15804.0  United Kingdom
2011-12
541875 2011-12-09 12:31:00       1.69     15804.0  United Kingdom
2011-12
541876 2011-12-09 12:31:00       0.85     15804.0  United Kingdom
2011-12
541877 2011-12-09 12:31:00       4.25     15804.0  United Kingdom
2011-12
541878 2011-12-09 12:31:00       5.95     15804.0  United Kingdom
2011-12
```

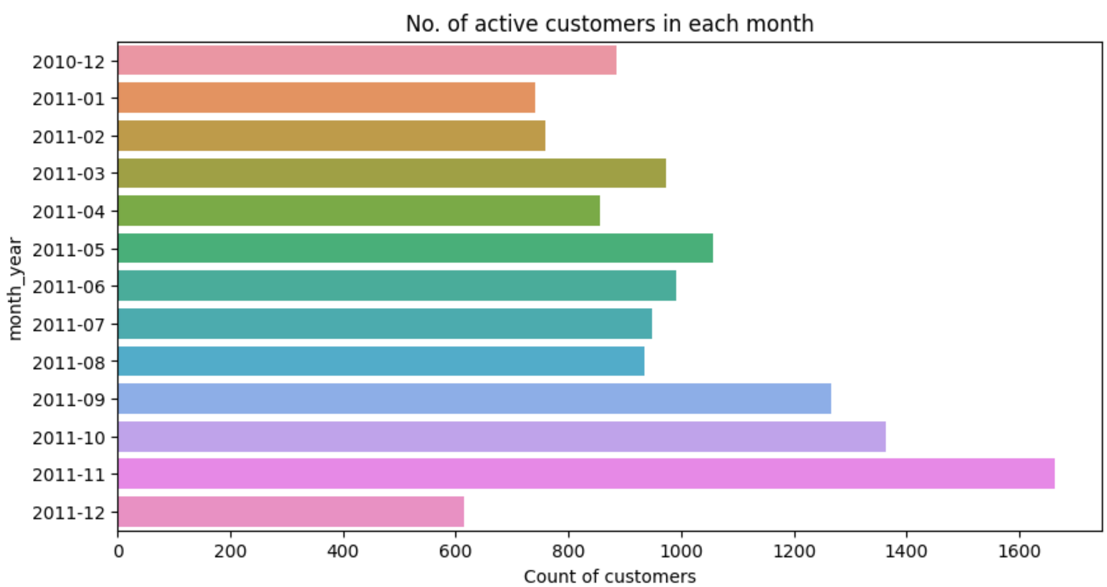
541879 2011-12-09 12:31:00 3.75 15804.0 United Kingdom
2011-12
541880 2011-12-09 12:31:00 3.75 15804.0 United Kingdom
2011-12
541881 2011-12-09 12:31:00 0.85 15804.0 United Kingdom
2011-12
541882 2011-12-09 12:31:00 0.42 15804.0 United Kingdom
2011-12
541883 2011-12-09 12:31:00 0.42 15804.0 United Kingdom
2011-12
541884 2011-12-09 12:31:00 1.25 15804.0 United Kingdom
2011-12
541885 2011-12-09 12:31:00 0.85 15804.0 United Kingdom
2011-12
541886 2011-12-09 12:31:00 0.39 15804.0 United Kingdom
2011-12
541887 2011-12-09 12:31:00 3.75 15804.0 United Kingdom
2011-12
541888 2011-12-09 12:31:00 0.95 15804.0 United Kingdom
2011-12
541889 2011-12-09 12:31:00 1.95 15804.0 United Kingdom
2011-12
541890 2011-12-09 12:49:00 2.95 13113.0 United Kingdom
2011-12
541891 2011-12-09 12:49:00 1.25 13113.0 United Kingdom
2011-12
541892 2011-12-09 12:49:00 8.95 13113.0 United Kingdom
2011-12
541893 2011-12-09 12:49:00 7.08 13113.0 United Kingdom
2011-12
541894 2011-12-09 12:50:00 1.95 12680.0 France
2011-12
541895 2011-12-09 12:50:00 1.65 12680.0 France
2011-12
541896 2011-12-09 12:50:00 1.65 12680.0 France
2011-12
541897 2011-12-09 12:50:00 3.75 12680.0 France
2011-12
541898 2011-12-09 12:50:00 3.75 12680.0 France
2011-12
541899 2011-12-09 12:50:00 3.75 12680.0 France
2011-12
541900 2011-12-09 12:50:00 3.75 12680.0 France
2011-12
541901 2011-12-09 12:50:00 1.95 12680.0 France
2011-12
541902 2011-12-09 12:50:00 1.95 12680.0 France
2011-12
541903 2011-12-09 12:50:00 4.15 12680.0 France
2011-12

```
541904 2011-12-09 12:50:00       0.85     12680.0          France
2011-12
541905 2011-12-09 12:50:00       2.10     12680.0          France
2011-12
541906 2011-12-09 12:50:00       4.15     12680.0          France
2011-12
541907 2011-12-09 12:50:00       4.15     12680.0          France
2011-12
541908 2011-12-09 12:50:00       4.95     12680.0          France
2011-12

       month_year   cohort
541859    2011-12  2010-12
541860    2011-12  2010-12
541861    2011-12  2010-12
541862    2011-12  2010-12
541863    2011-12  2010-12
541864    2011-12  2010-12
541865    2011-12  2010-12
541866    2011-12  2010-12
541867    2011-12  2010-12
541868    2011-12  2010-12
541869    2011-12  2011-05
541870    2011-12  2011-05
541871    2011-12  2011-05
541872    2011-12  2011-05
541873    2011-12  2011-05
541874    2011-12  2011-05
541875    2011-12  2011-05
541876    2011-12  2011-05
541877    2011-12  2011-05
541878    2011-12  2011-05
541879    2011-12  2011-05
541880    2011-12  2011-05
541881    2011-12  2011-05
541882    2011-12  2011-05
541883    2011-12  2011-05
541884    2011-12  2011-05
541885    2011-12  2011-05
541886    2011-12  2011-05
541887    2011-12  2011-05
541888    2011-12  2011-05
541889    2011-12  2011-05
541890    2011-12  2010-12
541891    2011-12  2010-12
541892    2011-12  2010-12
541893    2011-12  2010-12
541894    2011-12  2011-08
541895    2011-12  2011-08
541896    2011-12  2011-08
```

```
541897    2011-12  2011-08
541898    2011-12  2011-08
541899    2011-12  2011-08
541900    2011-12  2011-08
541901    2011-12  2011-08
541902    2011-12  2011-08
541903    2011-12  2011-08
541904    2011-12  2011-08
541905    2011-12  2011-08
541906    2011-12  2011-08
541907    2011-12  2011-08
541908    2011-12  2011-08
```

```
plt.figure(figsize=(10,5))
sns.barplot(y = month_cohort.index, x = month_cohort.values);
plt.xlabel("Count of customers")
plt.title("No. of active customers in each month")
```

```
Text(0.5, 1.0, 'No. of active customers in each month')
```



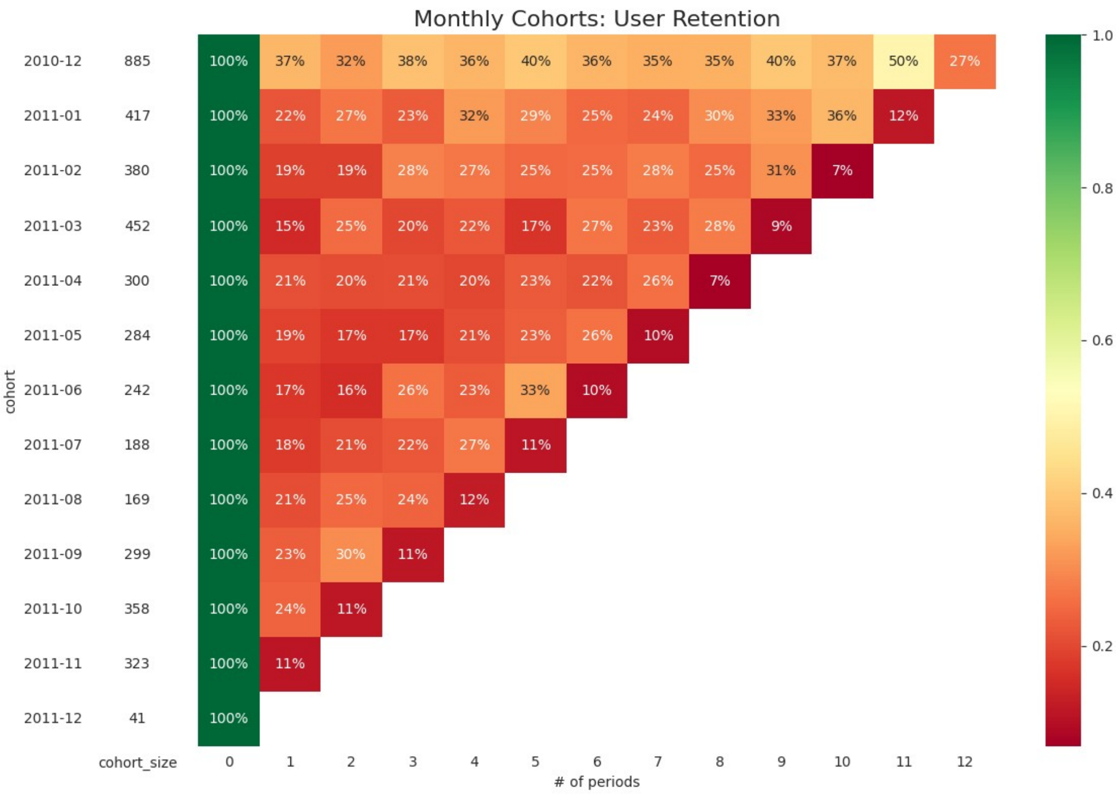

```
from operator import attrgetter
data_cohort = data.groupby(['cohort', 'Month']) \
              .agg(n_customers=('CustomerID', 'nunique')) \
              .reset_index(drop=False)
data_cohort['period_number'] = (data_cohort.Month -
data_cohort.cohort).apply(attrgetter('n'))
```

```
data_cohort.head()
```

	cohort	Month	n_customers	period_number
0	2010-12	2010-12	885	0
1	2010-12	2011-01	324	1
2	2010-12	2011-02	286	2

```
3  2010-12  2011-03          340              3
4  2010-12  2011-04          321              4
```

```
cohort_pivot = data_cohort.pivot_table(index = 'cohort',
                                     columns = 'period_number',
                                     values = 'n_customers')

cohort_size = cohort_pivot.iloc[:,0]
retention_matrix = cohort_pivot.divide(cohort_size, axis = 0)

import seaborn as sns
import matplotlib.colors as mcolors

with sns.axes_style("white"):
    fig, ax = plt.subplots(1, 2, figsize=(12, 8), sharey=True,
gridspec_kw={'width_ratios': [1, 11]})

    # retention matrix
    sns.heatmap(retention_matrix,
                mask=retention_matrix.isnull(),
                annot=True,
                fmt='.0%',
                cmap='RdYlGn',
                ax=ax[1])
    ax[1].set_title('Monthly Cohorts: User Retention', fontsize=16)
    ax[1].set(xlabel='# of periods',
              ylabel='')

    # cohort size
    cohort_size_df = pd.DataFrame(cohort_size).rename(columns={0:
'cohort_size'})
    white_cmap = mcolors.ListedColormap(['white'])
    sns.heatmap(cohort_size_df,
                annot=True,
                cbar=False,
                fmt='g',
                cmap=white_cmap,
                ax=ax[0])

    fig.tight_layout()
```



- There is a significant drop off in retention after the first month.
- Each subsequent month is about 20% to 30% retention
- The first cohort, 2010-12, seems to be the most consistent with the highest retention percents
- December is the least consistent month with much lower retention percent than any other month.

Data Modeling

Build a RFM (Recency Frequency Monetary) model. Recency means the number of days since a customer made the last purchase. Frequency is the number of purchases in a given period. It could be 3 months, 6 months, or 1 year. Monetary is the total amount of money a customer spent in that givn period. Therefore, big spenders will be differentiated among other customers such as MVP (Minimum Viable Product) or VIP.

Calculate RFM Metrics

Build RFM Segments. Give Recency, Frequency, and Monetary scores indivudually by dividing them into quartiles.

- Combine three ratings to gt a RFM segment (as strings)
- Get the RFM Score by adding up the three ratings.
- Analyze the RFM segments by summarizing thm and comment on the findings

Note:

- Rate "Recency" for customer who has been active more recently hgher than less recent customer, because each comapany wants its customers to be recent.
- Rate "Frequency" and "Monetary" higher, because the company wants the customer to visit more often and spend more money.

```
from datetime import datetime
recency_now_date = data['InvoiceDate'].max()
recency = data.groupby('CustomerID', as_index=False)
['InvoiceDate'].max()
recency.columns = ['CustomerID','max_date']
recency['Recency'] = recency['max_date'].apply(lambda row:
(recency_now_date - row).days)
recency.drop(['max_date'], axis=1, inplace=True)
recency.head()
```

```
   CustomerID  Recency
0     12346.0      325
1     12347.0        1
2     12348.0       74
3     12349.0       18
4     12350.0      309
```

```
frequency = data.groupby('CustomerID', as_index=False)
['InvoiceNo'].nunique()
frequency.columns = ['CustomerID', 'Frequency']
frequency.head()
```

```
   CustomerID  Frequency
0     12346.0          1
1     12347.0          7
2     12348.0          4
3     12349.0          1
4     12350.0          1
```

```
data['OrderTotal'] = data['Quantity'] * data['UnitPrice']
monetary = data.groupby('CustomerID', as_index=False)
['OrderTotal'].sum()
monetary.columns = ['CustomerID', 'Monetary']
monetary.head()
```

```
   CustomerID  Monetary
0     12346.0  77183.60
1     12347.0   4310.00
2     12348.0   1797.24
3     12349.0   1757.55
4     12350.0    334.40
```

```
rf_data = pd.merge(recency, frequency, how='right')
rfm_data = pd.merge(rf_data, monetary, how='right')
rfm_data.head()
```

```
   CustomerID  Recency  Frequency  Monetary
0     12346.0      325          1  77183.60
1     12347.0        1          7   4310.00
2     12348.0       74          4   1797.24
3     12349.0       18          1   1757.55
4     12350.0      309          1    334.40
```

```
rfm_data["RecencyScore"] = pd.cut(rfm_data["Recency"],
                                       bins=[-1,

np.percentile(rfm_data["Recency"], 25),

np.percentile(rfm_data["Recency"], 50),

np.percentile(rfm_data["Recency"], 75),

rfm_data["Recency"].max()],
                                       labels=[4, 3, 2,
1]).astype("int")

rfm_data["FrequencyScore"] = pd.cut(rfm_data["Frequency"],
                                       bins=[-1,

np.percentile(rfm_data["Frequency"], 25),

np.percentile(rfm_data["Frequency"], 50),

np.percentile(rfm_data["Frequency"], 75),

rfm_data["Frequency"].max()],
                                       labels=[1, 2, 3,
4]).astype("int")

rfm_data["MonetaryScore"] = pd.cut(rfm_data["Monetary"],
                                       bins=[-1,

np.percentile(rfm_data["Monetary"], 25),

np.percentile(rfm_data["Monetary"], 50),

np.percentile(rfm_data["Monetary"], 75),

rfm_data["Monetary"].max()],
                                       labels=[1, 2, 3,
4]).astype("int")

rfm_data['RFM_Score'] = rfm_data['RecencyScore'] +
rfm_data['FrequencyScore'] + rfm_data['MonetaryScore']
```

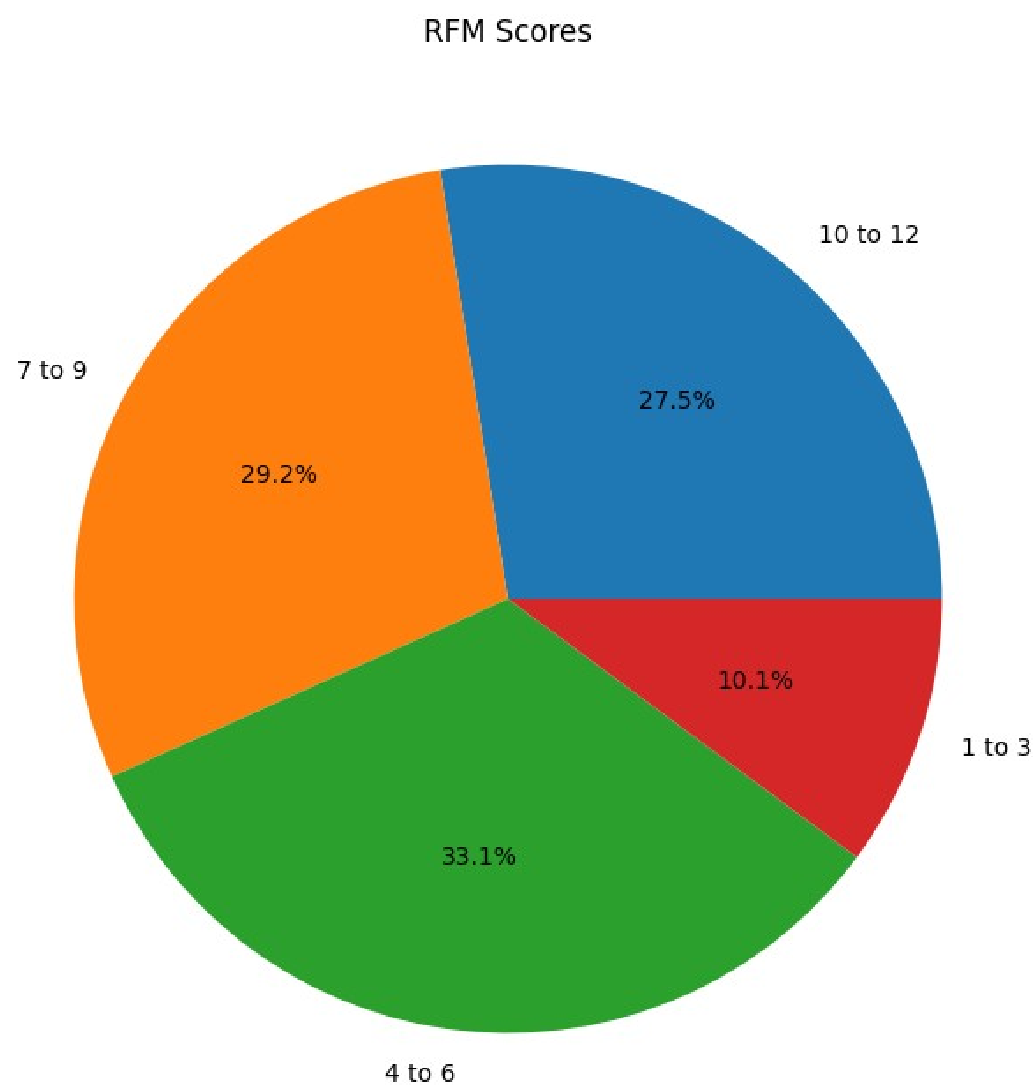
```
# Looking at the RFM data to see how it was segmented.
rfm_segmentation = pd.DataFrame()
rfm_segmentation['RecencyMinValue'] = rfm_data.groupby('RecencyScore')
['Recency'].min()
rfm_segmentation['RecencyMaxValue'] = rfm_data.groupby('RecencyScore')
['Recency'].max()
rfm_segmentation['FrequencyMinValue'] =
rfm_data.groupby('FrequencyScore')['Frequency'].min()
rfm_segmentation['FrequencyMaxValue'] =
rfm_data.groupby('FrequencyScore')['Frequency'].max()
rfm_segmentation['MonetaryMinValue'] =
rfm_data.groupby('MonetaryScore')['Monetary'].min()
rfm_segmentation['MonetaryMaxValue'] =
rfm_data.groupby('MonetaryScore')['Monetary'].max()
rfm_segmentation
```

```
              RecencyMinValue  RecencyMaxValue  FrequencyMinValue
RecencyScore
1                         142              373                  1  \
2                          51              141                  2
3                          18               50                  3
4                           0               17                  6

              FrequencyMaxValue  MonetaryMinValue  MonetaryMaxValue
RecencyScore
1                             1              3.75            306.46
2                             2            306.55            668.56
3                             5            668.58           1659.75
4                           209           1660.88         280206.02
```

Looking at the RFM Segments we can make the following observations:

- 25% of customers have been active in the last 17 days
- More than 25% of all customers have not been active in the last 4 and a half months
- More than 75% of customers have placed no more than 5 orders total
- Less than 25% of customers are responsible for the vast majority of overall sales at over $280k

```
top = len(rfm_data[rfm_data['RFM_Score'] >= 10])
mid_top = len(rfm_data[rfm_data['RFM_Score'].isin(range(7,10))])
mid_bottom = len(rfm_data[rfm_data['RFM_Score'].isin(range(4,7))])
bottom = len(rfm_data[rfm_data['RFM_Score'] <= 3])

pie_data = ([top, mid_top, mid_bottom, bottom])
labels = ['10 to 12', '7 to 9', '4 to 6', '1 to 3']
plt.figure(figsize=(8,8))
plt.title('RFM Scores')
plt.pie(pie_data, labels=labels, autopct='%1.1f%%');
```



Most of the RFM Scores are somewhat equally distributed among all customers with the exception of customers with RFM Scores of 1 to 3 which is only 10% of total customers.

Data Modeling

Create clusters using k-means clustering algorithm.

- Prepare the data for the algorithm. If the data is asymmetrically distributed, manage the skewness with appropriate transformation. Standardiz the data.
- Decide the optimum number of clusters to be formed.
- Analyze these clusters and comment on the results.

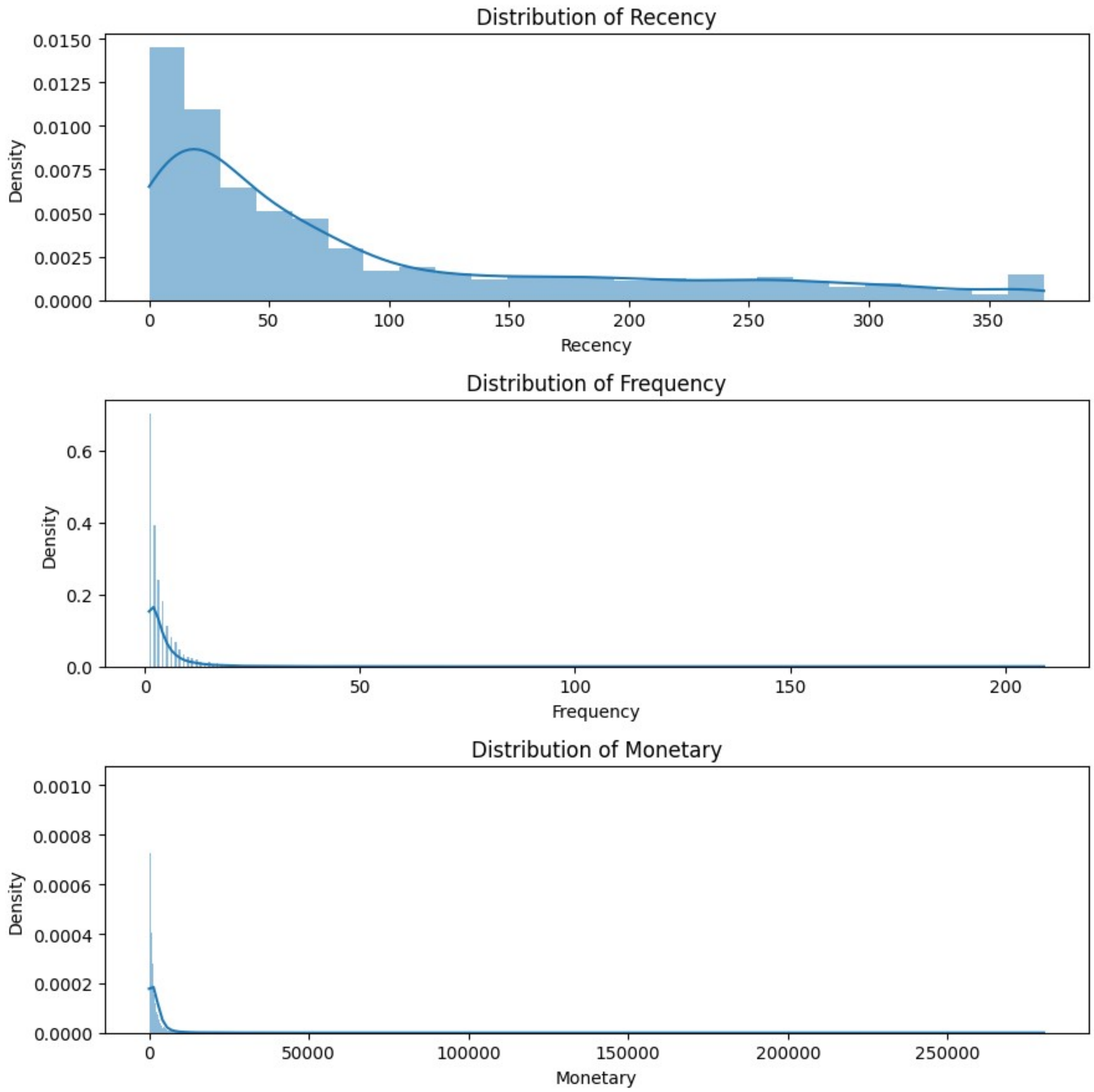
```
from scipy import stats

# Function to check for skewness
def check_skew(df_skew, column):
```

```
    skew = stats.skew(df_skew[column])
    skewtest = stats.skewtest(df_skew[column])
    plt.title('Distribution of ' + column)
    sns.histplot(df_skew[column], kde=True, stat='density',
linewidth=0)
    print("{}'s: Skew {}, : {}".format(column, skew, skewtest))
    return

rfm_segments = rfm_data[['CustomerID', 'Recency', 'Frequency',
'Monetary']]
rfm_segments.head()
```

```
   CustomerID  Recency  Frequency  Monetary
0     12346.0      325          1  77183.60
1     12347.0        1          7   4310.00
2     12348.0       74          4   1797.24
3     12349.0       18          1   1757.55
4     12350.0      309          1    334.40
```

```
plt.figure(figsize=(9,9))

plt.subplot(3,1,1)
check_skew(rfm_segments, 'Recency')

plt.subplot(3,1,2)
check_skew(rfm_segments, 'Frequency')

plt.subplot(3,1,3)
check_skew(rfm_segments, 'Monetary')

plt.tight_layout()
```

```
Recency's: Skew 1.2456166142880103, :
SkewtestResult(statistic=26.606793376917242,
pvalue=5.664292789640091e-156)
Frequency's: Skew 12.062857869870964, :
SkewtestResult(statistic=74.62743613377035, pvalue=0.0)
Monetary's: Skew 19.332680144099353, :
SkewtestResult(statistic=85.01187149828888, pvalue=0.0)
```




```
rfm_data_log = rfm_segments.copy()
rfm_data_log.head()
```

```
   CustomerID  Recency  Frequency  Monetary
0     12346.0      325          1  77183.60
1     12347.0        1          7   4310.00
2     12348.0       74          4   1797.24
3     12349.0       18          1   1757.55
4     12350.0      309          1    334.40
```

```
pip install feature-engine
```

```
Collecting feature-engine
  Downloading feature_engine-1.6.0-py2.py3-none-any.whl (319 kB)
     ---------------------------------------- 0.0/319.4 kB ? eta
-:--:--
     ------------------------------------  317.4/319.4 kB 6.5 MB/s
```

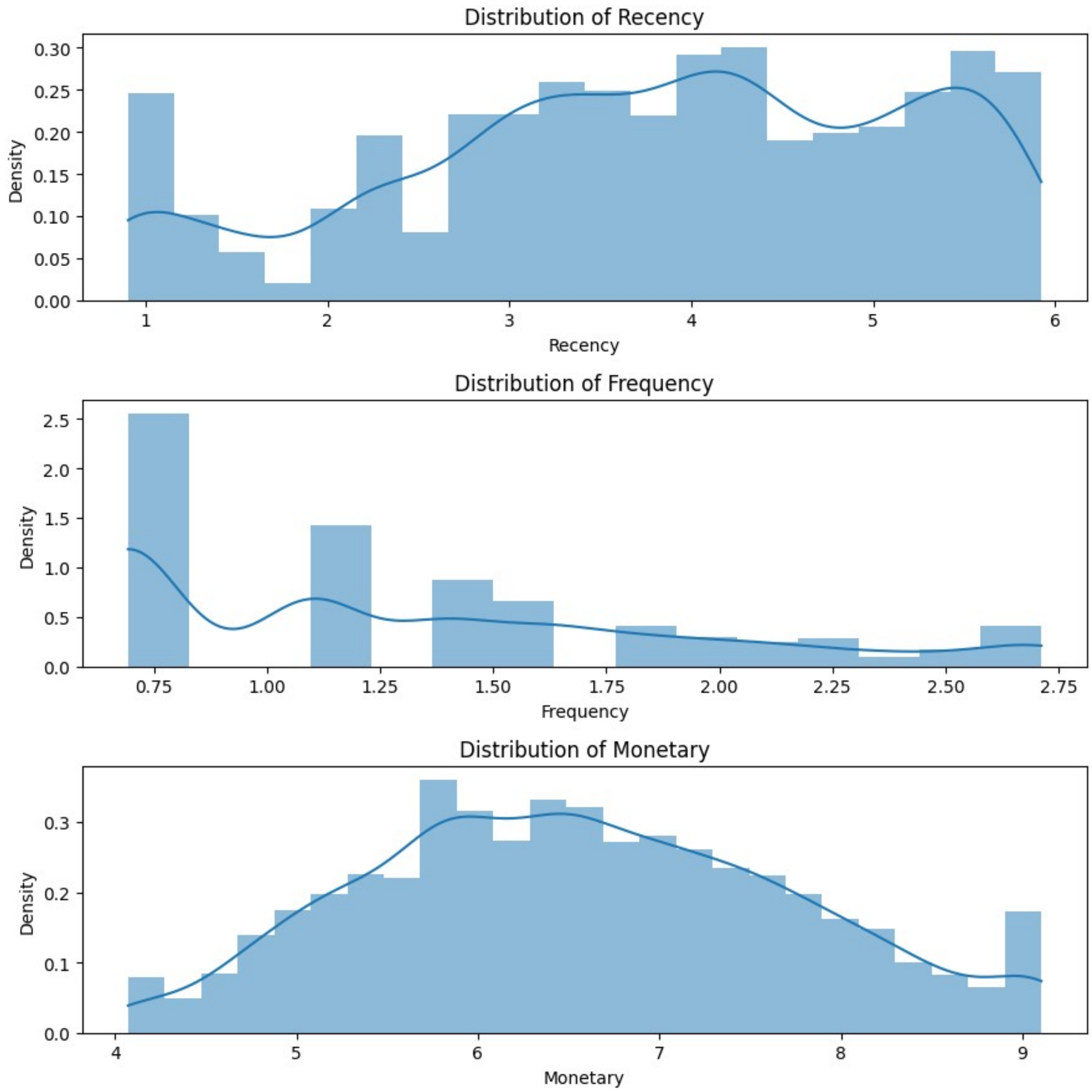
```
eta 0:00:01
     ---------------------------------------- 319.4/319.4 kB 1.7 MB/s
eta 0:00:00
Requirement already satisfied: pandas>=1.0.3 in c:\users\91805\
anaconda1\lib\site-packages (from feature-engine) (2.0.0)
Requirement already satisfied: scipy>=1.4.1 in c:\users\91805\
anaconda1\lib\site-packages (from feature-engine) (1.10.1)
Collecting statsmodels>=0.11.1
  Downloading statsmodels-0.14.0-cp39-cp39-win_amd64.whl (9.4 MB)
     ---------------------------------------- 0.0/9.4 MB ? eta -:--:--
     - -------------------------------------- 0.4/9.4 MB 8.7 MB/s eta
0:00:02
     --- ------------------------------------ 0.9/9.4 MB 9.4 MB/s eta
0:00:01
     ---- ----------------------------------- 1.1/9.4 MB 7.5 MB/s eta
0:00:02
     ------- -------------------------------- 1.8/9.4 MB 9.6 MB/s eta
0:00:01
     --------- ------------------------------ 2.2/9.4 MB 10.2 MB/s eta
0:00:01
     --------- ------------------------------ 2.2/9.4 MB 10.2 MB/s eta
0:00:01
     ---------- ----------------------------- 2.5/9.4 MB 8.0 MB/s eta
0:00:01
     --------------- ------------------------ 3.5/9.4 MB 9.4 MB/s eta
0:00:01
     --------------- ------------------------ 3.5/9.4 MB 9.4 MB/s eta
0:00:01
     --------------- ------------------------ 3.5/9.4 MB 8.1 MB/s eta
0:00:01
     ------------------ --------------------- 4.3/9.4 MB 8.3 MB/s eta
0:00:01
     ------------------- -------------------- 4.6/9.4 MB 8.1 MB/s eta
0:00:01
     ----------------------- ---------------- 5.6/9.4 MB 9.1 MB/s eta
0:00:01
     ------------------------ --------------- 5.8/9.4 MB 8.6 MB/s eta
0:00:01
     ------------------------- -------------- 6.1/9.4 MB 8.8 MB/s eta
0:00:01
     ------------------------- -------------- 6.1/9.4 MB 8.8 MB/s eta
0:00:01
     --------------------------- ------------ 6.5/9.4 MB 8.0 MB/s eta
0:00:01
     ------------------------------- -------- 7.5/9.4 MB 8.2 MB/s eta
0:00:01
     --------------------------------- ------ 7.8/9.4 MB 8.2 MB/s eta
0:00:01
     ---------------------------------- ----- 8.1/9.4 MB 8.1 MB/s eta
0:00:01
```

```
     ----------------------------------- ---- 8.4/9.4 MB 8.0 MB/s eta
0:00:01
     ------------------------------------- -- 8.7/9.4 MB 8.0 MB/s eta
0:00:01
     -------------------------------------- - 9.1/9.4 MB 7.9 MB/s eta
0:00:01
     ---------------------------------------  9.4/9.4 MB 7.9 MB/s eta
0:00:01
     ---------------------------------------  9.4/9.4 MB 7.9 MB/s eta
0:00:01
     ---------------------------------------  9.4/9.4 MB 7.9 MB/s eta
0:00:01
     ---------------------------------------  9.4/9.4 MB 7.9 MB/s eta
0:00:01
     ---------------------------------------- 9.4/9.4 MB 6.8 MB/s eta
0:00:00
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\91805\
anaconda1\lib\site-packages (from feature-engine) (1.2.2)
Requirement already satisfied: numpy>=1.18.2 in c:\users\91805\
anaconda1\lib\site-packages (from feature-engine) (1.23.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
91805\anaconda1\lib\site-packages (from pandas>=1.0.3->feature-engine)
(2.8.2)
Requirement already satisfied: tzdata>=2022.1 in c:\users\91805\
anaconda1\lib\site-packages (from pandas>=1.0.3->feature-engine)
(2023.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\91805\
anaconda1\lib\site-packages (from pandas>=1.0.3->feature-engine)
(2022.7)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\91805\
anaconda1\lib\site-packages (from scikit-learn>=1.0.0->feature-engine)
(2.2.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\91805\
anaconda1\lib\site-packages (from scikit-learn>=1.0.0->feature-engine)
(1.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\91805\
anaconda1\lib\site-packages (from statsmodels>=0.11.1->feature-engine)
(23.0)
Collecting patsy>=0.5.2
  Downloading patsy-0.5.3-py2.py3-none-any.whl (233 kB)
     ---------------------------------------- 0.0/233.8 kB ? eta
-:--:--
     ------------------------------------- - 225.3/233.8 kB 6.7 MB/s
eta 0:00:01
     -------------------------------------- 233.8/233.8 kB 3.5 MB/s
eta 0:00:00
Requirement already satisfied: six in c:\users\91805\anaconda1\lib\
site-packages (from patsy>=0.5.2->statsmodels>=0.11.1->feature-engine)
(1.16.0)
Installing collected packages: patsy, statsmodels, feature-engine
```

```
Successfully installed feature-engine-1.6.0 patsy-0.5.3 statsmodels-
0.14.0
Note: you may need to restart the kernel to use updated packages.
```

```
import feature_engine
from feature_engine.outliers import Winsorizer

rfm_data_log = np.log(rfm_data_log+1)

winsorizer = Winsorizer(tail='both', fold=2, variables=['Recency',
'Frequency', 'Monetary'])
winsorizer.fit(rfm_data_log)

rfm_data_log = winsorizer.transform(rfm_data_log)

plt.figure(figsize=(9,9))

plt.subplot(3,1,1)
check_skew(rfm_data_log, 'Recency')

plt.subplot(3,1,2)
check_skew(rfm_data_log, 'Frequency')

plt.subplot(3,1,3)
check_skew(rfm_data_log, 'Monetary')

plt.tight_layout()
```

```
Recency's: Skew -0.3863807061514661, : SkewtestResult(statistic=-
10.055002925140908, pvalue=8.731884165686116e-24)
Frequency's: Skew 0.7220853981502767, :
SkewtestResult(statistic=17.54001378255881, pvalue=7.091019464639143e-
69)
Monetary's: Skew 0.16491244333780397, :
SkewtestResult(statistic=4.412400335006802,
pvalue=1.0223085847018888e-05)
```




```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(rfm_data_log)
rfm_data_scaled = scaler.transform(rfm_data_log)

from sklearn.cluster import KMeans

from scipy.spatial.distance import cdist
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1, 10)

for k in K:
    # Building and fitting the model
```

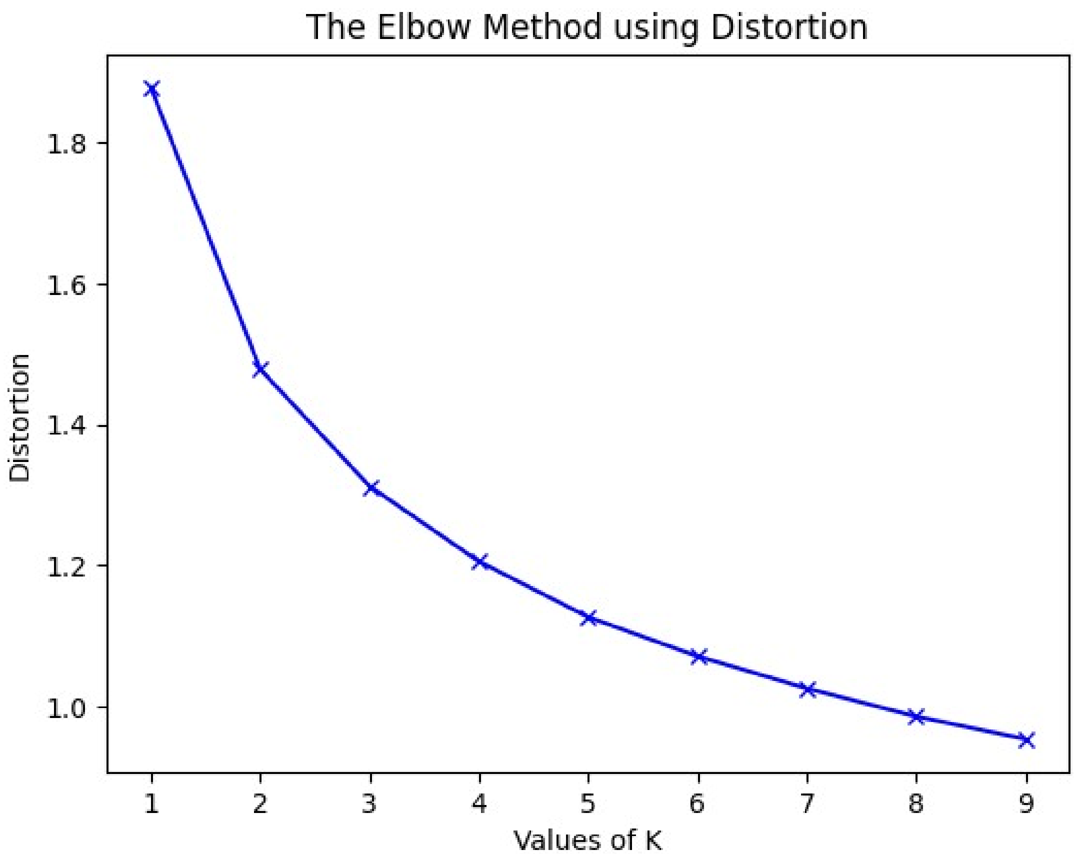
```
    kmeanModel = KMeans(n_clusters=k).fit(rfm_data_scaled)
    kmeanModel.fit(rfm_data_scaled)

    distortions.append(sum(np.min(cdist(rfm_data_scaled,
kmeanModel.cluster_centers_,
                                        'euclidean'), axis=1)) /
rfm_data_scaled.shape[0])
    inertias.append(kmeanModel.inertia_)

    mapping1[k] = sum(np.min(cdist(rfm_data_scaled,
kmeanModel.cluster_centers_,
                                   'euclidean'), axis=1)) /
rfm_data_scaled.shape[0]
    mapping2[k] = kmeanModel.inertia_
```

```
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```

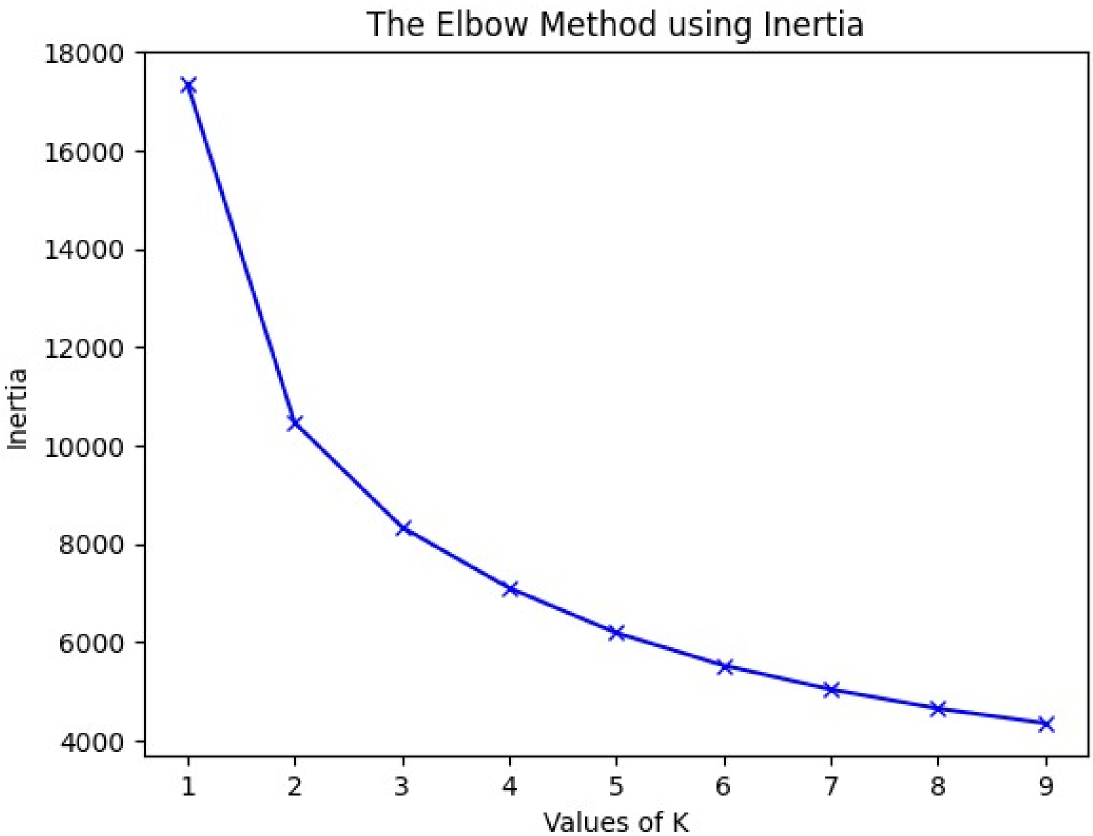
```
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```

```
C:\Users\91805\anaconda1\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```

```python
plt.plot(K, distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```

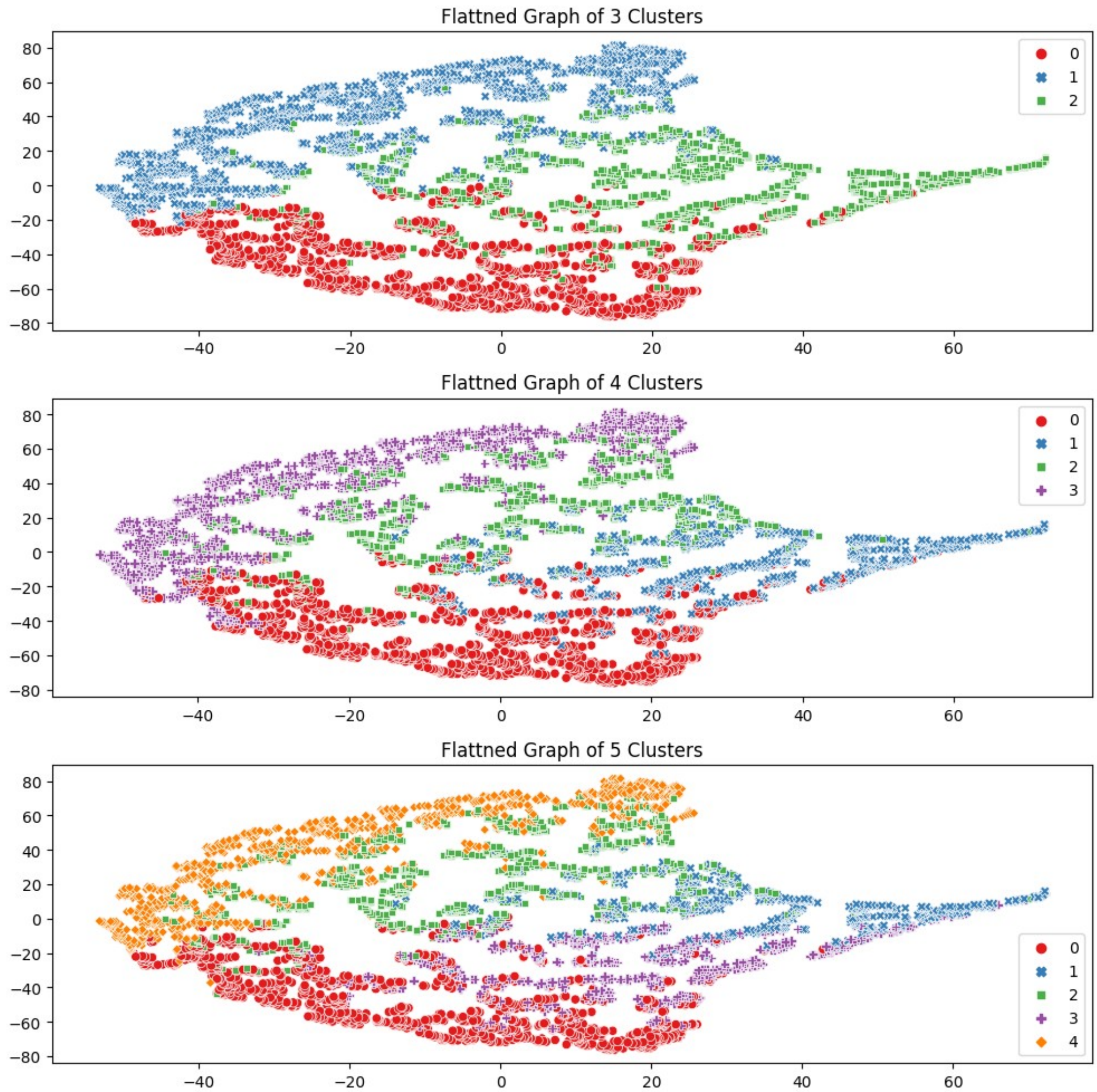



```python
plt.plot(K, inertias, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Inertia')
plt.title('The Elbow Method using Inertia')
plt.show()
```



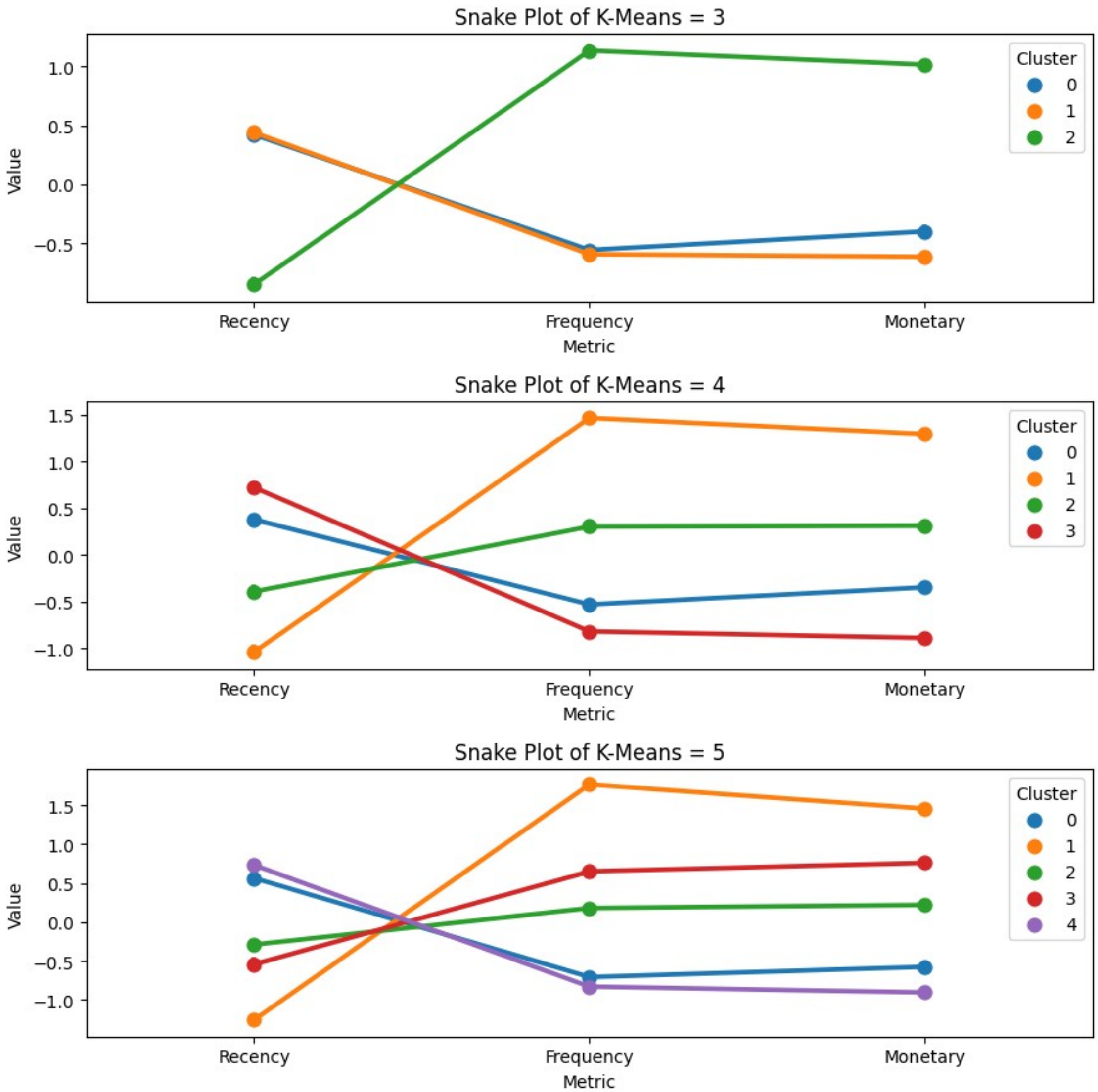

```
from sklearn.manifold import TSNE

def kmeans(normalised_df_rfm, clusters_number, original_df_rfm):
    kmeans = KMeans(n_clusters = clusters_number, random_state=1)
    kmeans.fit(normalised_df_rfm)

    cluster_labels = kmeans.labels_

    df_new = original_df_rfm.assign(Cluster = cluster_labels)

    model = TSNE(random_state=1)
    transformed = model.fit_transform(df_new)

    plt.title('Flattned Graph of {} Clusters'.format(clusters_number))
    sns.scatterplot(x=transformed[:,0], y=transformed[:, 1],
hue=cluster_labels, style=cluster_labels, palette='Set1')

    return df_new

import warnings
warnings.filterwarnings('ignore')

plt.figure(figsize=(10,10))
```

```
plt.subplot(3,1,1)
df_rfm_k3 = kmeans(rfm_data_scaled, 3, rfm_segments)

plt.subplot(3,1,2)
df_rfm_k4 = kmeans(rfm_data_scaled, 4, rfm_segments)

plt.subplot(3,1,3)
df_rfm_k5 = kmeans(rfm_data_scaled, 5, rfm_segments)

plt.tight_layout()
```




```
def snake_plot(normalised_df_rfm, df_rfm_kmeans, df_rfm_original):
    normalised_df_rfm = pd.DataFrame(normalised_df_rfm,
                                     index=rfm_segments.index,
                                     columns=rfm_segments.columns)
    normalised_df_rfm['Cluster'] = df_rfm_kmeans['Cluster']
```

```
    df_melt = pd.melt(normalised_df_rfm.reset_index(),
                      id_vars=['CustomerID', 'Cluster'],
                      value_vars=['Recency', 'Frequency', 'Monetary'],
                      var_name='Metric',
                      value_name='Value')
    plt.xlabel('Metric')
    plt.ylabel('Value')
    sns.pointplot(data=df_melt, x='Metric', y='Value', hue='Cluster')

    return

plt.figure(figsize=(9,9))

plt.subplot(3,1,1)
plt.title('Snake Plot of K-Means = 3')
snake_plot(rfm_data_scaled, df_rfm_k3, rfm_segments)

plt.subplot(3,1,2)
plt.title('Snake Plot of K-Means = 4')
snake_plot(rfm_data_scaled, df_rfm_k4, rfm_segments)

plt.subplot(3,1,3)
plt.title('Snake Plot of K-Means = 5')
snake_plot(rfm_data_scaled, df_rfm_k5, rfm_segments)

plt.tight_layout()
```




```
df_rfm_k4.groupby('Cluster').agg({
    'Recency':'mean',
    'Frequency':'mean',
    'Monetary':['mean', 'count']
}).round(0)
```

```
        Recency Frequency Monetary
           mean      mean     mean count
Cluster
0         117.0       2.0    747.0  1234
1          18.0      12.0   6967.0   888
2          42.0       4.0   1362.0  1033
3         164.0       1.0    315.0  1183
```

Based on the resulting clustered data and going with the 4 cluster selection as that seems to be most appropriate separation of data, we can make the following statements regarding the clusters.

- Cluster 0. Less Recently Active - Less Frequent - Medium Monetary Value.
- Cluster 1. Most Recently Active - Most Frequent - Highest Monetary Value
- Cluster 2. Recently Active - Frequent - Good Monetary Value
- Cluster 3. Least Recently Active - One Time Frequency - Least Monetary Value
- Cluster 1 and 3 are the current valued customers worth trying to retain, with Cluster 1 being the Most Valued Customers.
- Clusters 0 and 3 are most likely passerby customers that are possibly already lost or on the verge of being lost