In [239…
```
#importing all relevant liabraries

import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
pd.options.display.max_columns=None
import seaborn as sns
import plotly.express as px

import datetime as dt
from tqdm import tqdm
import time

from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan


import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
import scipy.stats as stats
```

In [199…
```
df=pd.read_excel('sales_data.xlsx')
```

In [200…
```
df.head(10)
```

Out[200…

	CustomerID	TOTAL_ORDERS	REVENUE	AVERAGE_ORDER_VALUE	CARRIAGE_REVENUE	AVERAGESHIPPING	FIRST_ORDER_DATE	LATEST_ORDE
0	22	124	11986.54	96.67	529.59	4.27	2016-12-30	202
1	29	82	11025.96	134.46	97.92	1.19	2018-03-31	202
2	83	43	7259.69	168.83	171.69	3.99	2017-11-30	202
3	95	44	6992.27	158.92	92.82	2.11	2019-04-09	202
4	124	55	6263.44	113.88	179.04	3.26	2020-10-23	202
5	153	49	5841.24	119.21	96.84	1.98	2015-07-26	202
6	187	43	5470.27	127.22	128.77	2.99	2019-01-14	202
7	219	54	5200.53	96.31	237.53	4.40	2019-11-19	202
8	258	19	4967.06	261.42	51.91	2.73	2021-03-03	202
9	308	21	4726.38	225.07	63.88	3.04	2020-01-06	202

Dataset Insights

In [201…
```
df.shape
```

Out[201… (5000, 42)

Dataset Desciption

The dataset folder contains the following files:
train.csv: *5000, 40*

EDA

Features names-

In [202…
```
# get all column names
df.columns
```

Out[202…
```
Index(['CustomerID', 'TOTAL_ORDERS', 'REVENUE', 'AVERAGE_ORDER_VALUE',
       'CARRIAGE_REVENUE', 'AVERAGESHIPPING', 'FIRST_ORDER_DATE',
       'LATEST_ORDER_DATE', 'AVGDAYSBETWEENORDERS', 'DAYSSINCELASTORDER',
       'MONDAY_ORDERS', 'TUESDAY_ORDERS', 'WEDNESDAY_ORDERS',
       'THURSDAY_ORDERS', 'FRIDAY_ORDERS', 'SATURDAY_ORDERS', 'SUNDAY_ORDERS',
       'MONDAY_REVENUE', 'TUESDAY_REVENUE', 'WEDNESDAY_REVENUE',
       'THURSDAY_REVENUE', 'FRIDAY_REVENUE', 'SATURDAY_REVENUE',
       'SUNDAY_REVENUE', 'WEEK1_DAY01_DAY07_ORDERS',
       'WEEK2_DAY08_DAY15_ORDERS', 'WEEK3_DAY16_DAY23_ORDERS',
       'WEEK4_DAY24_DAY31_ORDERS', 'WEEK1_DAY01_DAY07_REVENUE',
       'WEEK2_DAY08_DAY15_REVENUE', 'WEEK3_DAY16_DAY23_REVENUE',
       'WEEK4_DAY24_DAY31_REVENUE', 'TIME_0000_0600_ORDERS',
       'TIME_0601_1200_ORDERS', 'TIME_1200_1800_ORDERS',
       'TIME_1801_2359_ORDERS', 'TIME_0000_0600_REVENUE',
       'TIME_0601_1200_REVENUE', 'TIME_1200_1800_REVENUE',
       'TIME_1801_2359_REVENUE', 'Unnamed: 40', 'Unnamed: 41'],
      dtype='object')
```

Data type and count of features-

In [203…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 42 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   CustomerID                 5000 non-null   int64
 1   TOTAL_ORDERS               5000 non-null   int64
 2   REVENUE                    5000 non-null   float64
 3   AVERAGE_ORDER_VALUE        5000 non-null   float64
 4   CARRIAGE_REVENUE           5000 non-null   float64
 5   AVERAGESHIPPING            5000 non-null   float64
 6   FIRST_ORDER_DATE           5000 non-null   datetime64[ns]
 7   LATEST_ORDER_DATE          5000 non-null   datetime64[ns]
 8   AVGDAYSBETWEENORDERS       5000 non-null   float64
 9   DAYSSINCELASTORDER         5000 non-null   int64
 10  MONDAY_ORDERS              5000 non-null   int64
 11  TUESDAY_ORDERS             5000 non-null   int64
 12  WEDNESDAY_ORDERS           5000 non-null   int64
 13  THURSDAY_ORDERS            5000 non-null   int64
 14  FRIDAY_ORDERS              5000 non-null   int64
 15  SATURDAY_ORDERS            5000 non-null   int64
 16  SUNDAY_ORDERS              5000 non-null   int64
 17  MONDAY_REVENUE             5000 non-null   float64
 18  TUESDAY_REVENUE            5000 non-null   float64
 19  WEDNESDAY_REVENUE          5000 non-null   float64
 20  THURSDAY_REVENUE           5000 non-null   float64
 21  FRIDAY_REVENUE             5000 non-null   float64
 22  SATURDAY_REVENUE           5000 non-null   float64
 23  SUNDAY_REVENUE             5000 non-null   float64
 24  WEEK1_DAY01_DAY07_ORDERS   5000 non-null   int64
 25  WEEK2_DAY08_DAY15_ORDERS   5000 non-null   int64
 26  WEEK3_DAY16_DAY23_ORDERS   5000 non-null   int64
 27  WEEK4_DAY24_DAY31_ORDERS   5000 non-null   int64
 28  WEEK1_DAY01_DAY07_REVENUE  5000 non-null   float64
 29  WEEK2_DAY08_DAY15_REVENUE  5000 non-null   float64
 30  WEEK3_DAY16_DAY23_REVENUE  5000 non-null   float64
 31  WEEK4_DAY24_DAY31_REVENUE  5000 non-null   float64
 32  TIME_0000_0600_ORDERS      5000 non-null   int64
 33  TIME_0601_1200_ORDERS      5000 non-null   int64
 34  TIME_1200_1800_ORDERS      5000 non-null   int64
 35  TIME_1801_2359_ORDERS      5000 non-null   int64
 36  TIME_0000_0600_REVENUE     5000 non-null   float64
 37  TIME_0601_1200_REVENUE     5000 non-null   float64
 38  TIME_1200_1800_REVENUE     5000 non-null   float64
 39  TIME_1801_2359_REVENUE     5000 non-null   float64
 40  Unnamed: 40                0 non-null      float64
 41  Unnamed: 41                1 non-null      float64
dtypes: datetime64[ns](2), float64(22), int64(18)
memory usage: 1.6 MB
```

1. After checking the Dtypes of all the columns
   A. object - String values
   B. float64 - Numerical values
2. There are more String values than the numerical values in the dataset

The describe() method returns description of feature and their interrealtions-

count - The number of not-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile.
50% - The 50% percentile.
75% - The 75% percentile*.
max - the maximum value.

In [204…
```
df.describe()
```

Out[204…

	CustomerID	TOTAL_ORDERS	REVENUE	AVERAGE_ORDER_VALUE	CARRIAGE_REVENUE	AVERAGESHIPPING	AVGDAYSBETWEENORDE
count	5000.000000	5000.00000	5000.000000	5000.000000	5000.000000	5000.000000	5000.0000
mean	40709.227800	12.87040	1681.523840	136.537378	46.036376	3.592574	163.1596
std	49949.848017	12.67988	1998.618678	91.651569	47.879226	2.021360	259.6994
min	1.000000	1.00000	38.500000	10.680000	0.000000	0.000000	0.0000
25%	1687.500000	3.00000	315.097500	83.025000	9.980000	2.500000	21.6700
50%	13765.000000	8.00000	966.725000	113.160000	24.985000	3.660000	57.6350
75%	71891.500000	20.00000	2493.072500	160.272500	76.862500	4.790000	170.3575
max	277160.000000	156.00000	34847.400000	1578.880000	529.590000	35.990000	1409.5000

No of unique values for each column/feature-

In [205…
```
for x in df.columns:
    print(x,'-',len(df[x].unique()))
```

```
CustomerID - 5000
TOTAL_ORDERS - 85
REVENUE - 4635
AVERAGE_ORDER_VALUE - 4113
CARRIAGE_REVENUE - 1175
AVERAGESHIPPING - 567
FIRST_ORDER_DATE - 1928
LATEST_ORDER_DATE - 207
AVGDAYSBETWEENORDERS - 3563
DAYSSINCELASTORDER - 207
MONDAY_ORDERS - 19
TUESDAY_ORDERS - 22
WEDNESDAY_ORDERS - 22
THURSDAY_ORDERS - 23
FRIDAY_ORDERS - 24
SATURDAY_ORDERS - 24
SUNDAY_ORDERS - 23
MONDAY_REVENUE - 2151
TUESDAY_REVENUE - 2201
WEDNESDAY_REVENUE - 2262
THURSDAY_REVENUE - 2582
FRIDAY_REVENUE - 2378
SATURDAY_REVENUE - 2149
SUNDAY_REVENUE - 2573
WEEK1_DAY01_DAY07_ORDERS - 27
WEEK2_DAY08_DAY15_ORDERS - 34
WEEK3_DAY16_DAY23_ORDERS - 33
WEEK4_DAY24_DAY31_ORDERS - 32
WEEK1_DAY01_DAY07_REVENUE - 3026
WEEK2_DAY08_DAY15_REVENUE - 2900
WEEK3_DAY16_DAY23_REVENUE - 2947
WEEK4_DAY24_DAY31_REVENUE - 3386
TIME_0000_0600_ORDERS - 22
TIME_0601_1200_ORDERS - 41
TIME_1200_1800_ORDERS - 41
TIME_1801_2359_ORDERS - 36
TIME_0000_0600_REVENUE - 1413
TIME_0601_1200_REVENUE - 3203
TIME_1200_1800_REVENUE - 3512
TIME_1801_2359_REVENUE - 3128
Unnamed: 40 - 1
Unnamed: 41 - 2
```

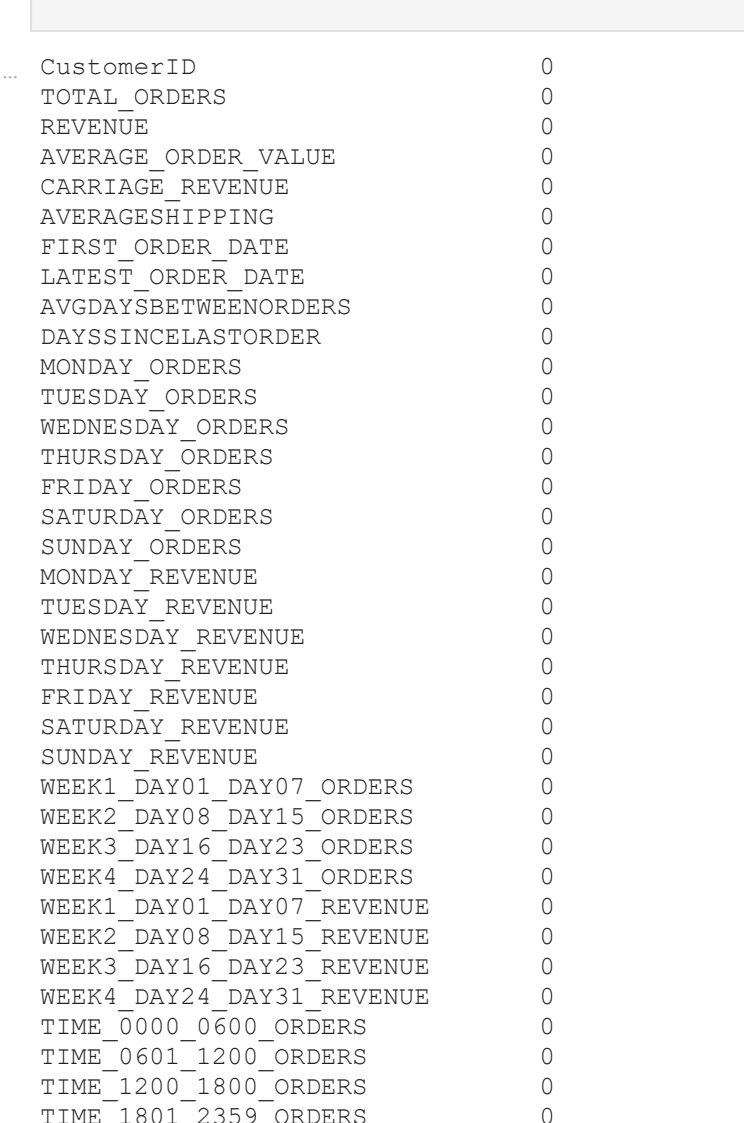
Counting Null (missing values) values-

In [206…
```
# Check for missing values in all the columnns of the train_dataset
df.isnull().sum()
```

Out[206…
```
CustomerID                      0
TOTAL_ORDERS                    0
REVENUE                         0
AVERAGE_ORDER_VALUE             0
CARRIAGE_REVENUE                0
AVERAGESHIPPING                 0
FIRST_ORDER_DATE                0
LATEST_ORDER_DATE               0
AVGDAYSBETWEENORDERS            0
DAYSSINCELASTORDER              0
MONDAY_ORDERS                   0
TUESDAY_ORDERS                  0
WEDNESDAY_ORDERS                0
THURSDAY_ORDERS                 0
FRIDAY_ORDERS                   0
SATURDAY_ORDERS                 0
SUNDAY_ORDERS                   0
MONDAY_REVENUE                  0
TUESDAY_REVENUE                 0
WEDNESDAY_REVENUE               0
THURSDAY_REVENUE                0
FRIDAY_REVENUE                  0
SATURDAY_REVENUE                0
SUNDAY_REVENUE                  0
WEEK1_DAY01_DAY07_ORDERS        0
WEEK2_DAY08_DAY15_ORDERS        0
WEEK3_DAY16_DAY23_ORDERS        0
WEEK4_DAY24_DAY31_ORDERS        0
WEEK1_DAY01_DAY07_REVENUE       0
WEEK2_DAY08_DAY15_REVENUE       0
WEEK3_DAY16_DAY23_REVENUE       0
WEEK4_DAY24_DAY31_REVENUE       0
TIME_0000_0600_ORDERS           0
TIME_0601_1200_ORDERS           0
TIME_1200_1800_ORDERS           0
TIME_1801_2359_ORDERS           0
TIME_0000_0600_REVENUE          0
TIME_0601_1200_REVENUE          0
TIME_1200_1800_REVENUE          0
TIME_1801_2359_REVENUE          0
Unnamed: 40                  5000
Unnamed: 41                  4999
dtype: int64
```

In [207…
```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[207… <AxesSubplot:>

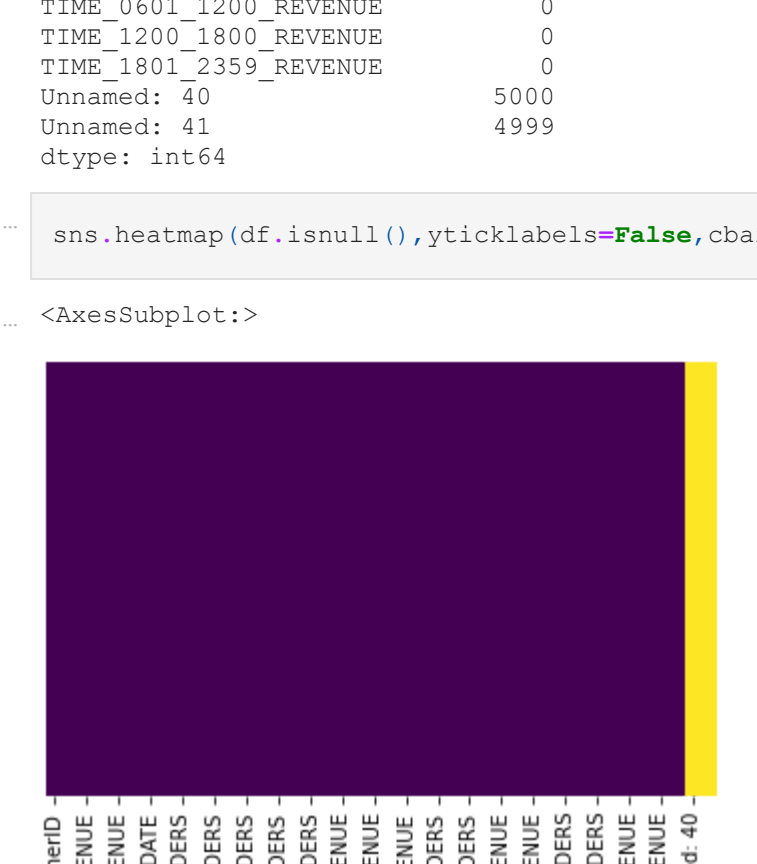


Dropping irrelevant "Unnamed" columns

In [208…
```
df.drop(['Unnamed: 40', 'Unnamed: 41'], axis=1, inplace= True)
```

In [209…
```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[209… <AxesSubplot:>



For finding correlation, creating two seperate list to store categorical column names and numerical column names respectively

In [210…
```
# looping on whole dataset for geting list of categorical and numerical data column name and storing in respect

categorical_list=[]
var_list=[]


for x in df.columns:
    if df[x].dtype=='object':
        categorical_list.append(x)
    elif df[x].dtype=='int64':
        var_list.append(x)
    elif df[x].dtype=='float64':
        var_list.append(x)

print('categorical' ,categorical_list)
print('numerical',var_list)
```

```
categorical []
numerical ['CustomerID', 'TOTAL_ORDERS', 'REVENUE', 'AVERAGE_ORDER_VALUE', 'CARRIAGE_REVENUE', 'AVERAGESHIPPIN
G', 'AVGDAYSBETWEENORDERS', 'DAYSSINCELASTORDER', 'MONDAY_ORDERS', 'TUESDAY_ORDERS', 'WEDNESDAY_ORDERS', 'THURS
DAY_ORDERS', 'FRIDAY_ORDERS', 'SATURDAY_ORDERS', 'SUNDAY_ORDERS', 'MONDAY_REVENUE', 'TUESDAY_REVENUE', 'WEDNESD
AY_REVENUE', 'THURSDAY_REVENUE', 'FRIDAY_REVENUE', 'SATURDAY_REVENUE', 'SUNDAY_REVENUE', 'WEEK1_DAY01_DAY07_ORD
ERS', 'WEEK2_DAY08_DAY15_ORDERS', 'WEEK3_DAY16_DAY23_ORDERS', 'WEEK4_DAY24_DAY31_ORDERS', 'WEEK1_DAY01_DAY07_RE
VENUE', 'WEEK2_DAY08_DAY15_REVENUE', 'WEEK3_DAY16_DAY23_REVENUE', 'WEEK4_DAY24_DAY31_REVENUE', 'TIME_0000_0600_
ORDERS', 'TIME_0601_1200_ORDERS', 'TIME_1200_1800_ORDERS', 'TIME_1801_2359_ORDERS', 'TIME_0000_0600_REVENUE',
'TIME_0601_1200_REVENUE', 'TIME_1200_1800_REVENUE', 'TIME_1801_2359_REVENUE']
```

Correlation Matrix

A correlation matrix is a table shows correlation coefficients between variables.

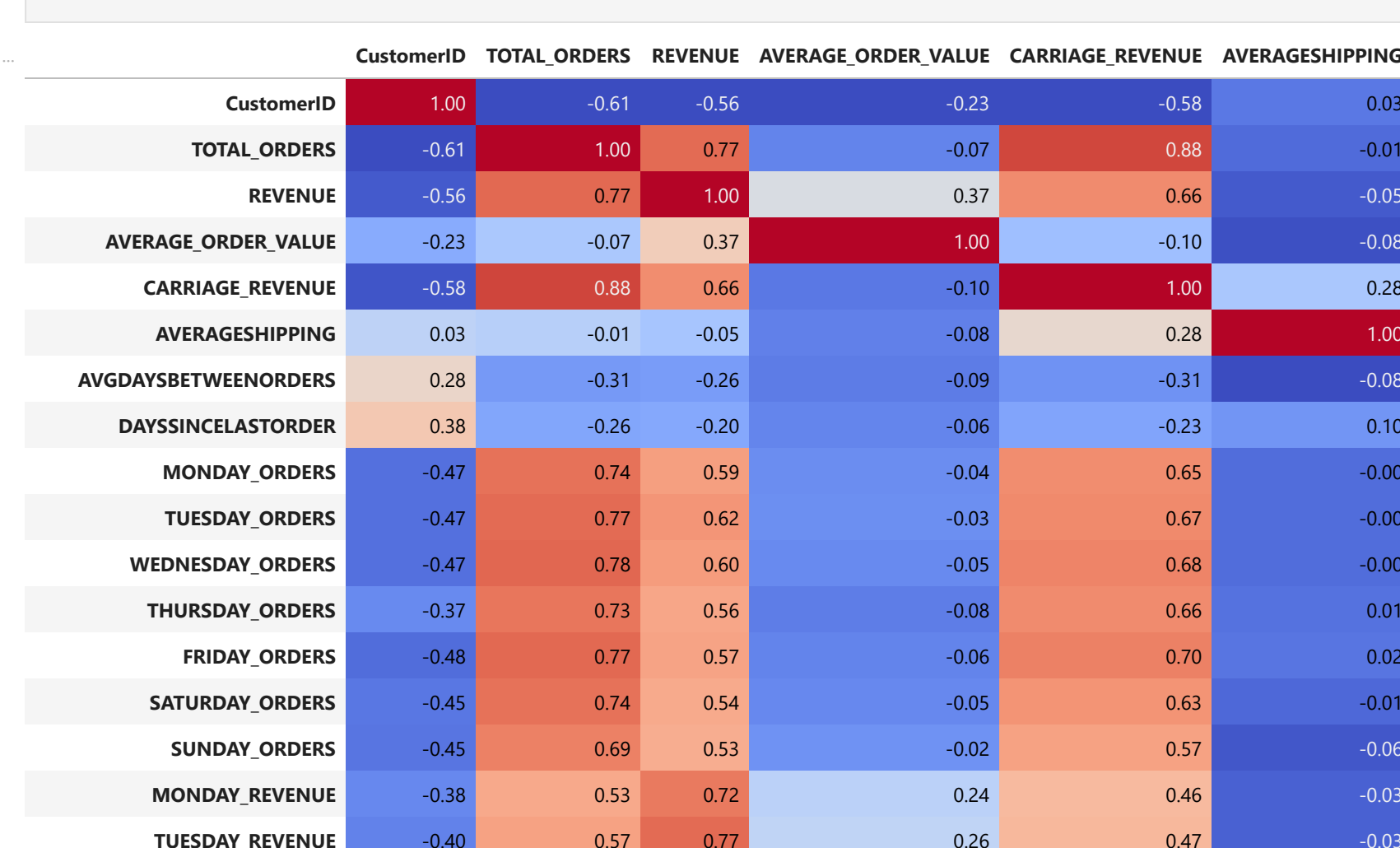
In [211…
```
# Correlation metrix using pandas
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[211…

	CustomerID	TOTAL_ORDERS	REVENUE	AVERAGE_ORDER_VALUE	CARRIAGE_REVENUE	AVERAGESHIPPING
CustomerID	1.00	-0.61	-0.56	-0.23	-0.58	0.03
TOTAL_ORDERS	-0.61	1.00	0.77	-0.07	0.88	-0.01
REVENUE	-0.56	0.77	1.00	0.37	0.66	-0.05
AVERAGE_ORDER_VALUE	-0.23	-0.07	0.37	1.00	-0.10	-0.08
CARRIAGE_REVENUE	-0.58	0.88	0.66	-0.10	1.00	0.28
AVERAGESHIPPING	0.03	-0.01	-0.05	-0.08	0.28	1.00
AVGDAYSBETWEENORDERS	0.28	-0.31	-0.26	-0.09	-0.31	-0.08
DAYSSINCELASTORDER	0.38	-0.26	-0.20	-0.06	-0.23	0.10
MONDAY_ORDERS	-0.47	0.74	0.59	-0.04	0.65	-0.00
TUESDAY_ORDERS	-0.47	0.77	0.62	-0.03	0.67	-0.00
WEDNESDAY_ORDERS	-0.47	0.78	0.60	-0.05	0.68	-0.00
THURSDAY_ORDERS	-0.37	0.73	0.56	-0.08	0.66	0.01
FRIDAY_ORDERS	-0.48	0.77	0.57	-0.06	0.70	0.02
SATURDAY_ORDERS	-0.45	0.74	0.54	-0.05	0.63	-0.01
SUNDAY_ORDERS	-0.45	0.69	0.53	-0.02	0.57	-0.06
MONDAY_REVENUE	-0.38	0.53	0.72	0.24	0.46	-0.03
TUESDAY_REVENUE	-0.40	0.57	0.77	0.26	0.47	-0.03
WEDNESDAY_REVENUE	-0.42	0.60	0.77	0.26	0.51	-0.04
THURSDAY_REVENUE	-0.37	0.56	0.72	0.25	0.49	-0.01
FRIDAY_REVENUE	-0.44	0.60	0.72	0.22	0.53	-0.00
SATURDAY_REVENUE	-0.41	0.58	0.67	0.24	0.46	-0.06
SUNDAY_REVENUE	-0.38	0.47	0.66	0.36	0.38	-0.08
WEEK1_DAY01_DAY07_ORDERS	-0.45	0.81	0.60	-0.09	0.72	0.01
WEEK2_DAY08_DAY15_ORDERS	-0.53	0.86	0.69	-0.03	0.75	-0.01
WEEK3_DAY16_DAY23_ORDERS	-0.54	0.87	0.68	-0.04	0.76	-0.00
WEEK4_DAY24_DAY31_ORDERS	-0.53	0.85	0.64	-0.06	0.74	-0.02
WEEK1_DAY01_DAY07_REVENUE	-0.43	0.62	0.80	0.28	0.54	-0.02
WEEK2_DAY08_DAY15_REVENUE	-0.47	0.64	0.83	0.30	0.54	-0.04
WEEK3_DAY16_DAY23_REVENUE	-0.46	0.64	0.86	0.31	0.55	-0.04
WEEK4_DAY24_DAY31_REVENUE	-0.49	0.66	0.84	0.34	0.55	-0.07
TIME_0000_0600_ORDERS	-0.29	0.50	0.39	-0.04	0.46	0.03
TIME_0601_1200_ORDERS	-0.48	0.80	0.61	-0.06	0.71	0.00
TIME_1200_1800_ORDERS	-0.53	0.86	0.66	-0.05	0.77	-0.00
TIME_1801_2359_ORDERS	-0.47	0.76	0.60	-0.04	0.63	-0.04
TIME_0000_0600_REVENUE	-0.27	0.38	0.47	0.16	0.35	0.01
TIME_0601_1200_REVENUE	-0.41	0.59	0.77	0.26	0.50	-0.03
TIME_1200_1800_REVENUE	-0.48	0.66	0.85	0.32	0.57	-0.03
TIME_1801_2359_REVENUE	-0.43	0.57	0.74	0.29	0.46	-0.07

In [212…
```
# Histogram using pandas
df.hist(figsize=(15,8))
```

Out[212…
```
array([[<AxesSubplot:title={'center':'CustomerID'}>,
        <AxesSubplot:title={'center':'TOTAL_ORDERS'}>,
        <AxesSubplot:title={'center':'REVENUE'}>,
        <AxesSubplot:title={'center':'AVERAGE_ORDER_VALUE'}>,
        <AxesSubplot:title={'center':'CARRIAGE_REVENUE'}>,
        <AxesSubplot:title={'center':'AVERAGESHIPPING'}>],
       [<AxesSubplot:title={'center':'FIRST_ORDER_DATE'}>,
        <AxesSubplot:title={'center':'LATEST_ORDER_DATE'}>,
        <AxesSubplot:title={'center':'AVGDAYSBETWEENORDERS'}>,
        <AxesSubplot:title={'center':'DAYSSINCELASTORDER'}>,
        <AxesSubplot:title={'center':'MONDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'TUESDAY_ORDERS'}>],
       [<AxesSubplot:title={'center':'WEDNESDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'THURSDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'FRIDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'SATURDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'SUNDAY_ORDERS'}>,
        <AxesSubplot:title={'center':'MONDAY_REVENUE'}>],
       [<AxesSubplot:title={'center':'TUESDAY_REVENUE'}>,
        <AxesSubplot:title={'center':'WEDNESDAY_REVENUE'}>,
        <AxesSubplot:title={'center':'THURSDAY_REVENUE'}>,
        <AxesSubplot:title={'center':'FRIDAY_REVENUE'}>,
        <AxesSubplot:title={'center':'SATURDAY_REVENUE'}>,
        <AxesSubplot:title={'center':'SUNDAY_REVENUE'}>],
       [<AxesSubplot:title={'center':'WEEK1_DAY01_DAY07_ORDERS'}>,
        <AxesSubplot:title={'center':'WEEK2_DAY08_DAY15_ORDERS'}>,
        <AxesSubplot:title={'center':'WEEK3_DAY16_DAY23_ORDERS'}>,
        <AxesSubplot:title={'center':'WEEK4_DAY24_DAY31_ORDERS'}>,
        <AxesSubplot:title={'center':'WEEK1_DAY01_DAY07_REVENUE'}>,
        <AxesSubplot:title={'center':'WEEK2_DAY08_DAY15_REVENUE'}>],
       [<AxesSubplot:title={'center':'WEEK3_DAY16_DAY23_REVENUE'}>,
        <AxesSubplot:title={'center':'WEEK4_DAY24_DAY31_REVENUE'}>,
        <AxesSubplot:title={'center':'TIME_0000_0600_ORDERS'}>,
        <AxesSubplot:title={'center':'TIME_0601_1200_ORDERS'}>,
        <AxesSubplot:title={'center':'TIME_1200_1800_ORDERS'}>,
        <AxesSubplot:title={'center':'TIME_1801_2359_ORDERS'}>],
       [<AxesSubplot:title={'center':'TIME_0000_0600_REVENUE'}>,
        <AxesSubplot:title={'center':'TIME_0601_1200_REVENUE'}>,
        <AxesSubplot:title={'center':'TIME_1200_1800_REVENUE'}>,
        <AxesSubplot:title={'center':'TIME_1801_2359_REVENUE'}>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



In [213…
```
# group data by total order and plot count plot
df.groupby('TOTAL_ORDERS').count().plot(kind='bar',figsize=(18,8))
```

Out[213… <AxesSubplot:xlabel='TOTAL_ORDERS'>

In [214…
```
# point plot for REVENUE and TOTAL_ORDERS columns
plt.figure(figsize=(20,6))
sns.pointplot(x='TOTAL_ORDERS',y='REVENUE',data=df,palette='rainbow')
plt.show()
```

for highest, lowest and average amount of order placed-

In [215…
```
df['REVENUE'].max()
```

Out[215… 34847.4

In [216…
```
df['REVENUE'].min()
```

Out[216… 38.5

In [217…
```
df['REVENUE'].mean()
```

Out[217… 1681.5238400000312

customer who generate revenue > avg revenue (above average category)

In [218…
```
df1 =df[df['REVENUE']>1681.5238400000312]
```

In [219…
```
df1.shape
```

Out[219… (1845, 40)

customers who generate revenue < avg revenue (below average category)

In [220…
```
df2 =df[df['REVENUE']<1681.5238400000312]
```

In [221…
```
df2.shape
```

Out[221… (3155, 40)

Maximum , minimum and avg amount of orders placed

In [222…
```
df['TOTAL_ORDERS'].max()
```

Out[222… 156

In [223…
```
df['TOTAL_ORDERS'].min()
```

Out[223… 1

In [224…
```
df['TOTAL_ORDERS'].mean()
```

Out[224… 12.8704

RFM:

The RFM model is based on three quantitative factors:

Recency: How recently a customer has made a purchase (DAYSSINCELASTORDER)

Frequency: How often a customer makes a purchase (AVGDAYSBETWEENORDERS)

Monetary Value: How much money a customer spends on purchases (REVENUE)

-Champions Customers: Bought recently, buy often and spend the most => recency-latest | frequency-high | momentary-high

-Potential Customers: Recent customers with average frequency=> recency-latest | frequency-avg

-Need attention: Below average recency and frequency, Some time since they've purchased=> recency-low | frequency-low

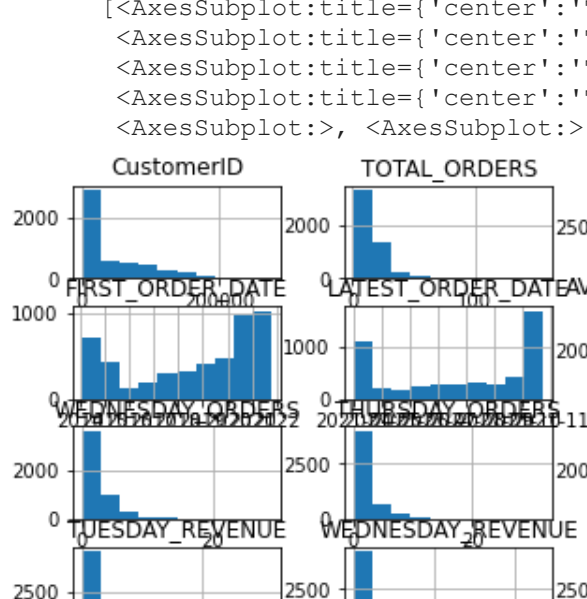
In [225…
```
df['recency']= df['DAYSSINCELASTORDER']
df['frequency']= df['AVGDAYSBETWEENORDERS']
df['revenue']= df['REVENUE']

rfm= df[['CustomerID','revenue', 'frequency', 'recency']]
rfm.head(10)
```

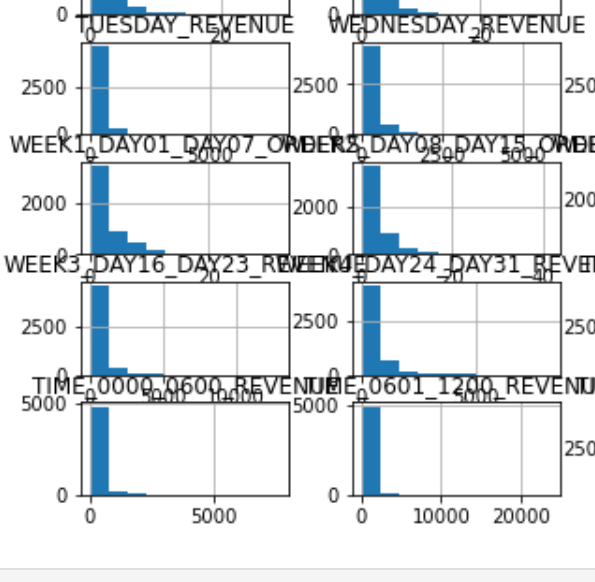
Out[225…

	CustomerID	revenue	frequency	recency
0	22	11986.54	14.19	1
1	29	11025.96	15.89	1
2	83	7259.69	33.12	1
3	95	6992.27	21.11	1
4	124	6263.44	6.65	1
5	153	5841.24	46.57	1
6	187	5470.27	23.58	1
7	219	5200.53	13.06	1
8	258	4967.06	12.37	1
9	308	4726.38	31.29	1

Outliers-

In [226…
```
plt.boxplot(rfm['revenue'])
plt.show()
plt.boxplot(rfm['frequency'])
plt.show()
plt.boxplot(rfm['recency'])
plt.show()
```





Outlier treatment (using interquantiloe range)-

In [227…
```
q1_rev = rfm['revenue'].quantile(0.05)
q3_rev= rfm['revenue'].quantile(0.95)
IQR = q3_rev - q1_rev
rfm = rfm[(rfm['revenue'] >= q1_rev - 1.5*IQR) & (rfm['revenue'] <= q3_rev + 1.5*IQR)]

q1_freq = rfm['frequency'].quantile(0.05)
q3_freq = rfm['frequency'].quantile(0.95)
IQR = q3_freq - q1_freq
rfm = rfm[(rfm['frequency'] >= q1_freq - 1.5*IQR) & (rfm['frequency'] <= q3_freq + 1.5*IQR)]

q1_rec = rfm['recency'].quantile(0.05)
q3_rec = rfm['recency'].quantile(0.95)
IQR = q3_rec - q1_rec
rfm = rfm[(rfm['recency'] >= q1_rec - 1.5*IQR) & (rfm['recency'] <= q3_rec + 1.5*IQR)]
```

Scaling-

In [228…
```
#Scaling the features
scale = StandardScaler()
cols = ['revenue', 'frequency', 'recency']
rfm_scaler = scale.fit_transform(rfm[cols])
```

In [229…
```
rfm_scaled = pd.DataFrame(rfm_scaler)
rfm_scaled.columns = ['revenue', 'frequency', 'recency']
rfm_scaled.head()
```

Out[229…

	revenue	frequency	recency
0	6.032833	-0.574806	-1.078028
1	5.473669	-0.568268	-1.078028
2	3.281285	-0.502009	-1.078028
3	3.125618	-0.548194	-1.078028
4	2.701358	-0.603801	-1.078028

Hopkins Statistic- checks if points are well spaced for clustering

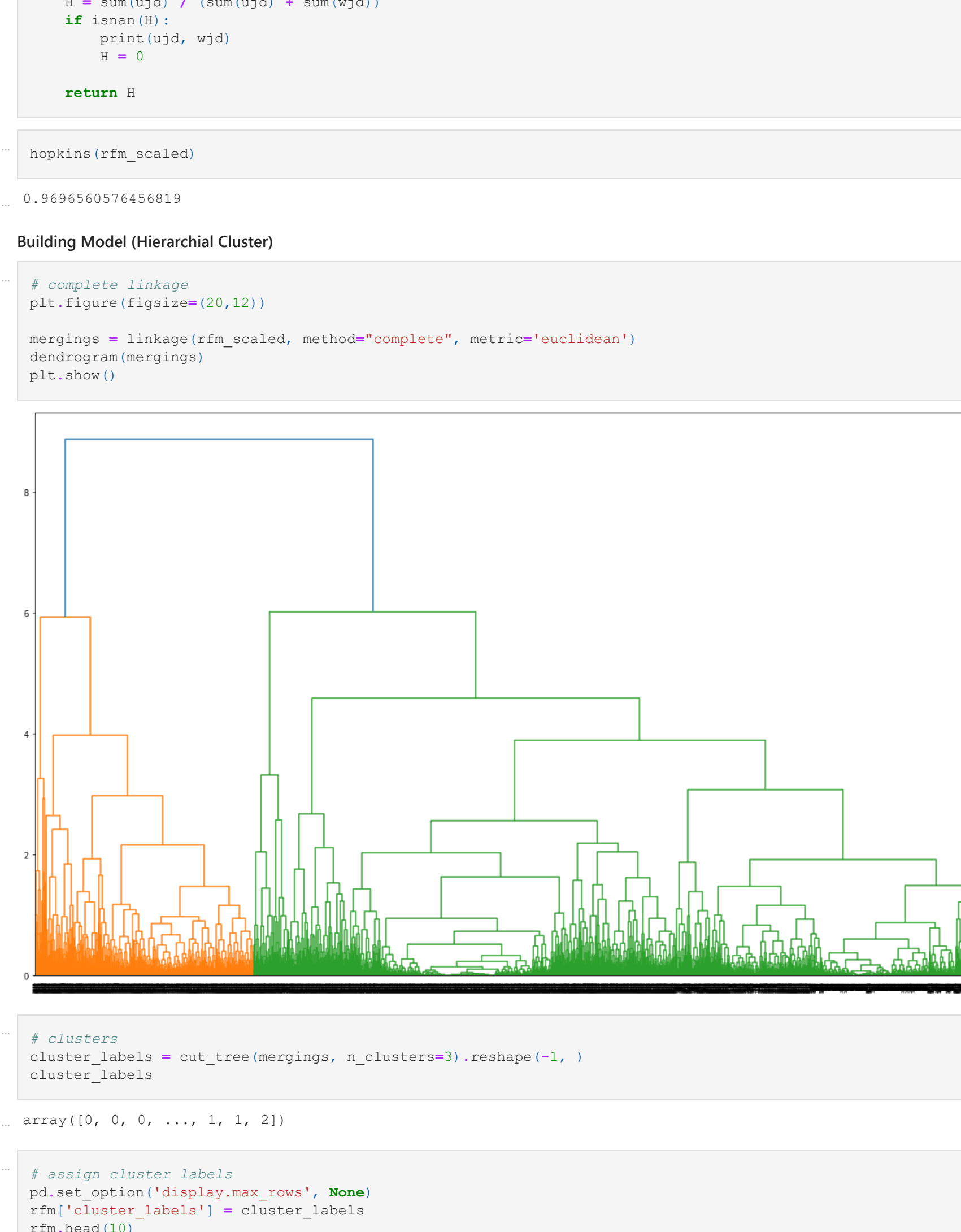
In [230…
```
def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2, return_di
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

In [231…
```
hopkins(rfm_scaled)
```

Out[231… 0.9696560576456819

Building Model (Hierarchial Cluster)

In [232…
```
# complete linkage
plt.figure(figsize=(20,12))

mergings = linkage(rfm_scaled, method="complete", metric='euclidean')
dendrogram(mergings)
plt.show()
```



In [233…
```
# clusters
cluster_labels = cut_tree(mergings, n_clusters=3).reshape(-1, )
cluster_labels
```

Out[233… array([0, 0, 0, ..., 1, 1, 2])

In [234…
```
# assign cluster labels
pd.set_option('display.max_rows', None)
rfm['cluster_labels'] = cluster_labels
rfm.head(10)
```

	CustomerID	revenue	frequency	recency	cluster_labels
0	22	11986.54	14.19	1	0
1	29	11025.96	15.89	1	0
2	83	7259.69	33.12	1	0
3	95	6992.27	21.11	1	0
4	124	6263.44	6.65	1	0
5	153	5841.24	46.57	1	0
6	187	5470.27	23.58	1	0
7	219	5200.53	13.06	1	0
8	258	4967.06	12.37	1	0
9	308	4726.38	31.29	1	0

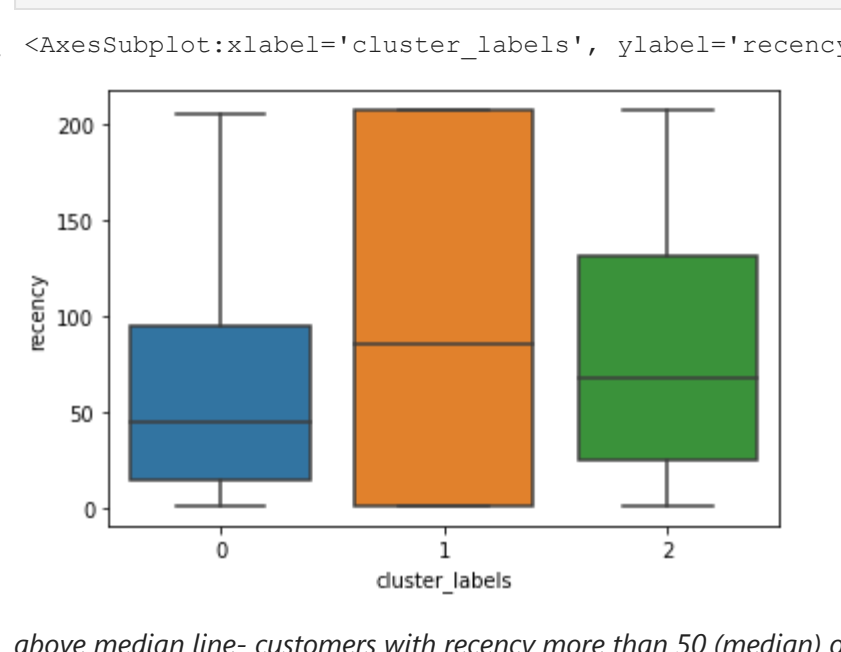
Analysing using box plots of 3 cluster:

-The box in the box plot indicates the range in which the middle 50% of all the data lies
-the upper horizontal edge of box is q3 and similarly lower one is q1 25% data lies above and below q3 and q1 respectively (q3 to q1 is IQR)
-The solid line in box is median
-outside of wiskers are outliers

In [235…

```
# plots
sns.boxplot(x='cluster_labels', y='recency', data=rfm)
```

Out[235… <AxesSubplot:xlabel='cluster_labels', ylabel='recency'>



above median line- customers with recency more than 50 (median) or less recent
below median line- customers with recency less than 50 (median) or more recent

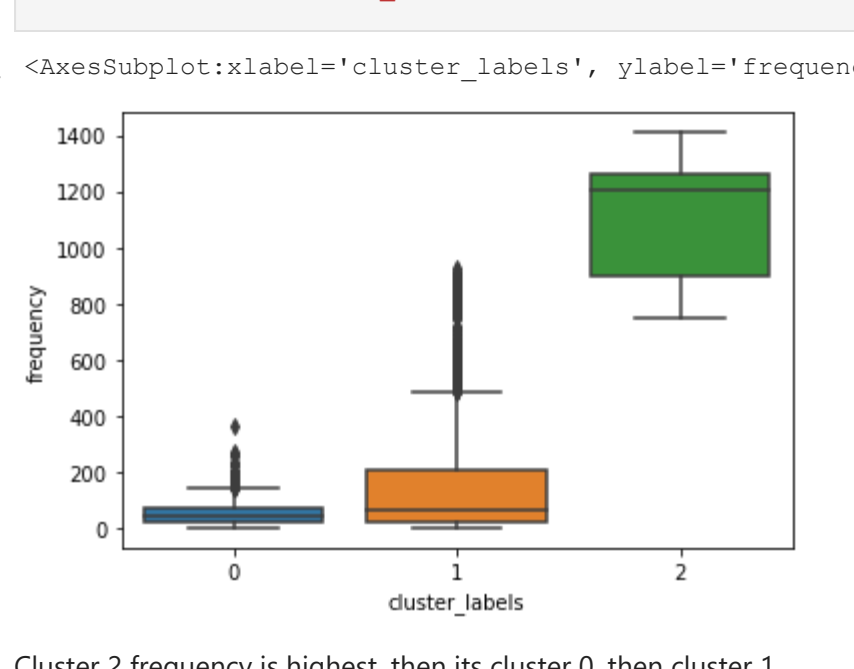
cluster 0 : 20-40 (20) q1 40-90 (50) q3 2:5 less recent coustomers than cluster 1
cluster 1 : 0-80 (80) q1 80-200 (120) q3 2:3
cluster 2 : 30-60 (30) q1 60-130 (70) q3 3:7 least recent coutomers of all or bad recency (less recent orders)

cluster 1> cluster 0> cluster 2 (recency or R)

In [236…

```
# plots
sns.boxplot(x='cluster_labels', y='frequency', data=rfm)
```

Out[236… <AxesSubplot:xlabel='cluster_labels', ylabel='frequency'>



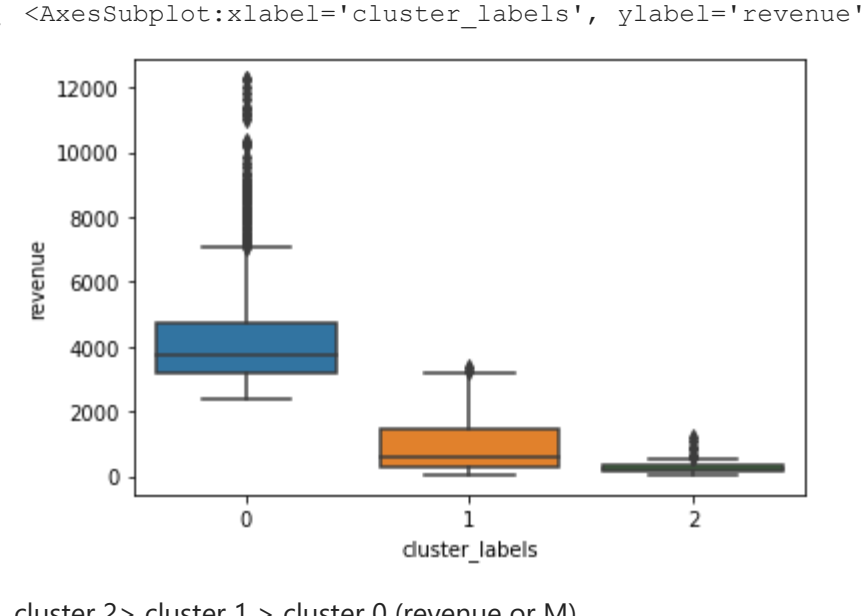
Cluster 2 frequency is highest, then its cluster 0, then cluster 1

cluster 2 > cluster 0 > cluster 1 (Frequency or F)

In [237…

```
# plots
sns.boxplot(x='cluster_labels', y='revenue', data=rfm)
```

Out[237… <AxesSubplot:xlabel='cluster_labels', ylabel='revenue'>



cluster 2> cluster 1 > cluster 0 (revenue or M)

Concluding

cluster 1> cluster 0> cluster 2 (recency or R)

cluster 2 > cluster 0 > cluster 1 (Frequency or F)

cluster 2> cluster 1 > cluster 0 (revenue or M)

-Champions Customers:=> recency-high | frequency-high | momentary-high; So, cluster 2

-Potential Customers:=> recency-high | frequency-avg; So, cluster 1

-Need attention:=> recency-low | frequency-low; So, cluster 0