```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

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

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/OnlineRetail.csv",encoding="ISO-8859-1")
df
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Count
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	17850.0	Unit Kingd
1	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	17850.0	Unit Kingd
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.75	17850.0	Unit Kingd
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.39	17850.0	Unit Kingd
			RED WOOLLY					

```python
df.shape
```

```
(49445, 8)
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49445 entries, 0 to 49444
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   InvoiceNo    49445 non-null  object 
 1   StockCode    49445 non-null  object 
 2   Description  49303 non-null  object 
 3   Quantity     49445 non-null  int64  
 4   InvoiceDate  49445 non-null  object 
 5   UnitPrice    49445 non-null  float64
 6   CustomerID   31563 non-null  float64
 7   Country      49444 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 3.0+ MB
```

```python
df.describe()
```

	Quantity	UnitPrice	CustomerID
count	49445.000000	49445.000000	31563.000000
mean	8.315239	6.458260	15423.016823
std	55.832343	167.436019	1753.298111
min	-9360.000000	0.000000	12347.000000
25%	1.000000	1.250000	14051.000000
50%	2.000000	2.510000	15464.000000
75%	8.000000	4.250000	17041.000000
max	2880.000000	16888.020000	18283.000000

```python
df.isnull().sum()
```

```
InvoiceNo          0
StockCode          0
Description      142
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     17882
Country            1
dtype: int64
```

```python
df=df.dropna()
df.isnull().sum()
```

```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

```python
df.shape
```

```
(31563, 8)
```

```python
df['CustomerID']=df['CustomerID'].astype(str)
```

```
<ipython-input-17-d4b6fccbd77f>:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['CustomerID']=df['CustomerID'].astype(str)
```

```python
df['Amount']=df['UnitPrice']*df['Quantity']
fm_m=df.groupby('CustomerID')['Amount'].sum()
fm_m=fm_m.reset_index()
fm_m.columns=['CustomerID','Spend_Amount']
fm_m.head()
```

```
<ipython-input-18-523d3fc75744>:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
  df['Amount']=df['UnitPrice']*df['Quantity']
```

	CustomerID	Spend_Amount
0	12347.0	711.79
1	12348.0	892.80
2	12370.0	1868.02
3	12377.0	1001.52
4	12383.0	600.72

```python
fm_f=df.groupby('CustomerID')['InvoiceNo'].count()
fm_f=fm_f.reset_index()
fm_f.columns=['CustomerID','Frequency']
fm_f.head()
```

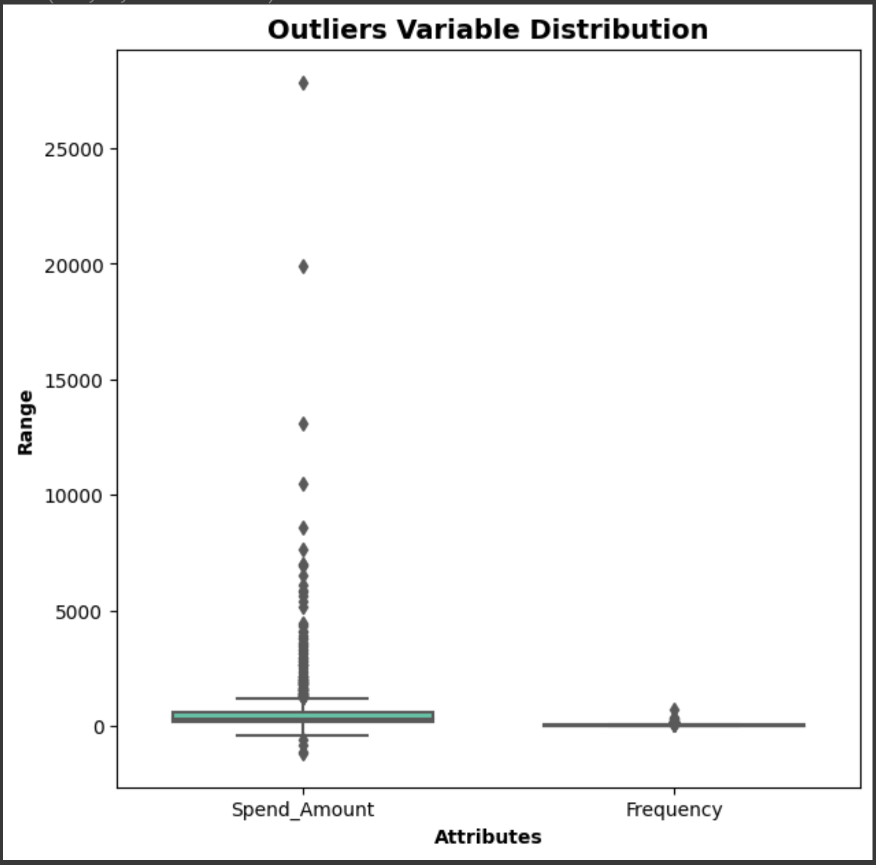
	CustomerID	Frequency
0	12347.0	31
1	12348.0	17
2	12370.0	91
3	12377.0	43
4	12383.0	37

```python
fm=pd.merge(fm_m,fm_f,on='CustomerID',how='inner')
fm.head()
```

	CustomerID	Spend_Amount	Frequency
0	12347.0	711.79	31
1	12348.0	892.80	17
2	12370.0	1868.02	91

```python
attributes=['Spend_Amount','Frequency']
plt.rcParams['figure.figsize']=[7,7]
sns.boxplot(data=fm[attributes],orient="X",palette="Set2", whis=1.5,saturation=1,width=0.7)
plt.title("Outliers Variable Distribution",fontsize=14,fontweight='bold')
plt.ylabel("Range",fontweight='bold')
plt.xlabel("Attributes",fontweight='bold')

```

```
Text(0.5, 0, 'Attributes')
```



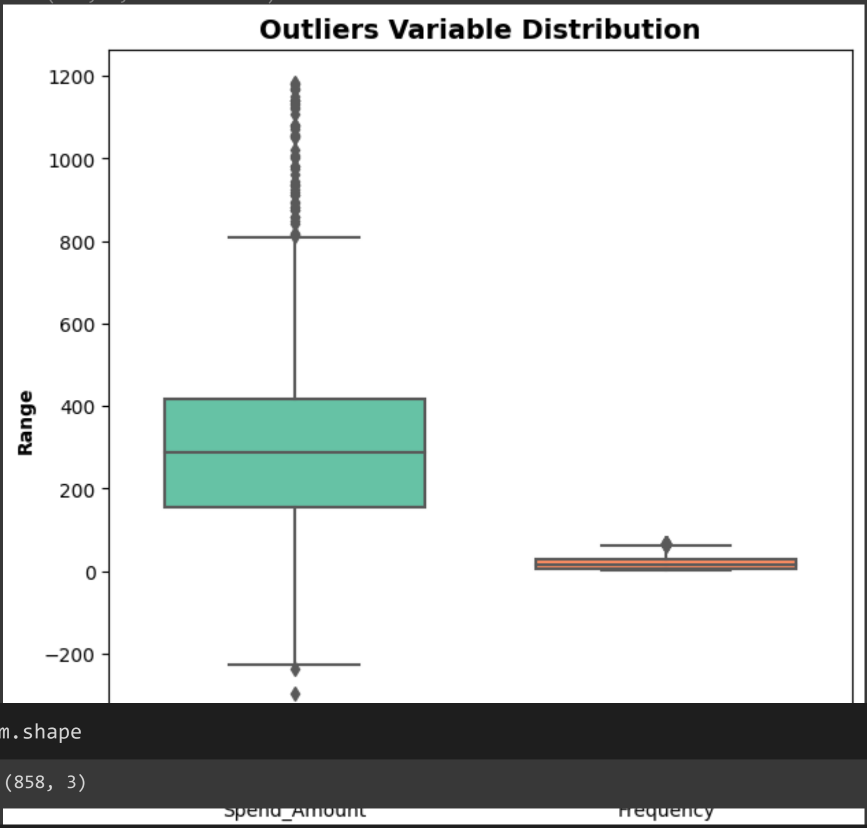
```python
fm.shape
```

```
(1027, 3)
```

```python
Q1=fm.Spend_Amount.quantile(0.25)
Q3=fm.Spend_Amount.quantile(0.75)
IQR=Q3-Q1
fm=fm[(fm.Spend_Amount>=Q1-1.5*IQR)& (fm.Spend_Amount<=Q3+1.5*IQR)]

Q1=fm.Frequency.quantile(0.25)
Q3=fm.Frequency.quantile(0.75)
IQR=Q3-Q1
fm=fm[(fm.Frequency>=Q1-1.5*IQR)& (fm.Frequency<=Q3+1.5*IQR)]


```

```python
attributes=['Spend_Amount','Frequency']
plt.rcParams['figure.figsize']=[7,7]
sns.boxplot(data=fm[attributes],orient="X",palette="Set2", whis=1.5,saturation=1,width=0.7)
plt.title("Outliers Variable Distribution",fontsize=14,fontweight='bold')
plt.ylabel("Range",fontweight='bold')
plt.xlabel("Attributes",fontweight='bold')

```

```
Text(0.5, 0, 'Attributes')
```



```
fm.shape
```

```
(858, 3)
```

```
fm.describe()
```

	Spend_Amount	Frequency
count	858.000000	858.000000
mean	323.150501	20.074592
std	258.816060	16.728919
min	-442.500000	1.000000
25%	155.687500	7.000000
50%	288.600000	16.000000
75%	417.540000	29.000000
max	1183.090000	70.000000

```
fm_df=fm[['Spend_Amount','Frequency']]
scaler=StandardScaler()
fm_df_scaled=scaler.fit_transform(fm_df)
fm_df_scaled.shape
```

```
(858, 2)
```

```
fm_df_scaled
```

```
array([[ 1.50248086,  0.65346602],
       [ 2.20226579, -0.18389624],
       [ 2.62257746,  1.3712051 ],
       ...,
       [-0.35741377, -0.54276578],
       [-0.7123127 , -0.72220055],
       [-0.83003245,  2.20856736]])
```

```
fm_df_scaled=pd.DataFrame(fm_df_scaled)
fm_df_scaled.columns=['Amount','Frequency']
fm_df_scaled.head()
```

	Amount	Frequency
0	1.502481	0.653466
1	2.202266	-0.183896
2	2.622577	1.371205
3	1.073084	1.012336
4	0.304446	-0.602577

```
fm_df_scaled.describe()
```

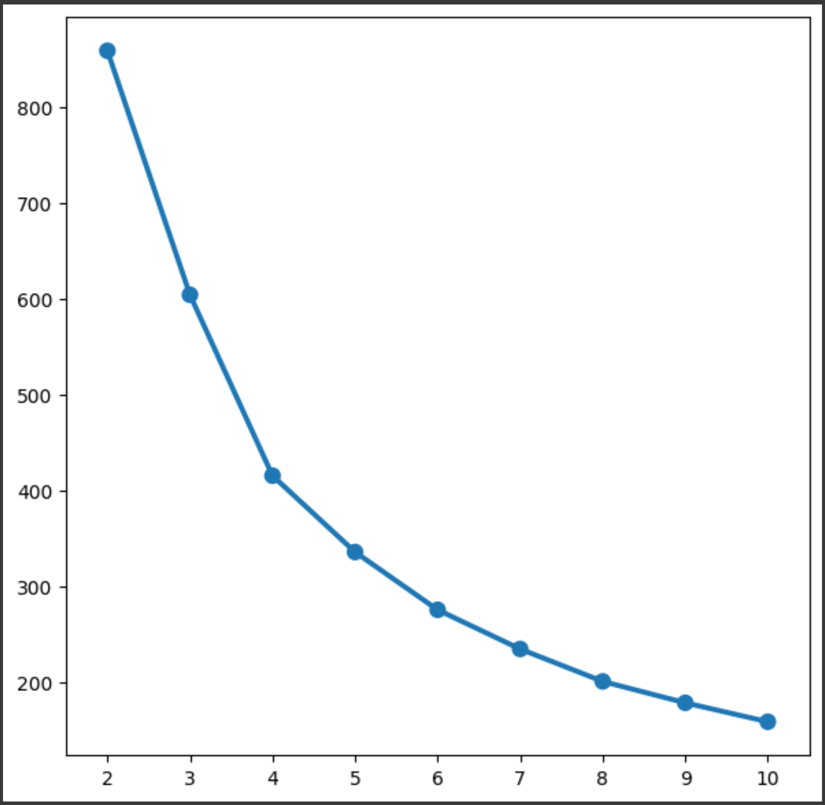
	Amount	Frequency
count	8.580000e+02	8.580000e+02
mean	1.656277e-17	-7.039176e-17
std	1.000583e+00	1.000583e+00
min	-2.960006e+00	-1.140882e+00
25%	-6.474122e-01	-7.820121e-01
50%	-1.335723e-01	-2.437078e-01
75%	3.649099e-01	5.338428e-01
max	3.324527e+00	2.986118e+00

```
track_inertia={}
silhouette={}
for k in range(2,11):
  kmeans=KMeans(n_clusters=k,init='k-means++',random_state=0)
  kmeans.fit(fm_df_scaled)
  track_inertia[k]=kmeans.inertia_
  silhouette[k]=silhouette_score(fm_df_scaled,kmeans.labels_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
```

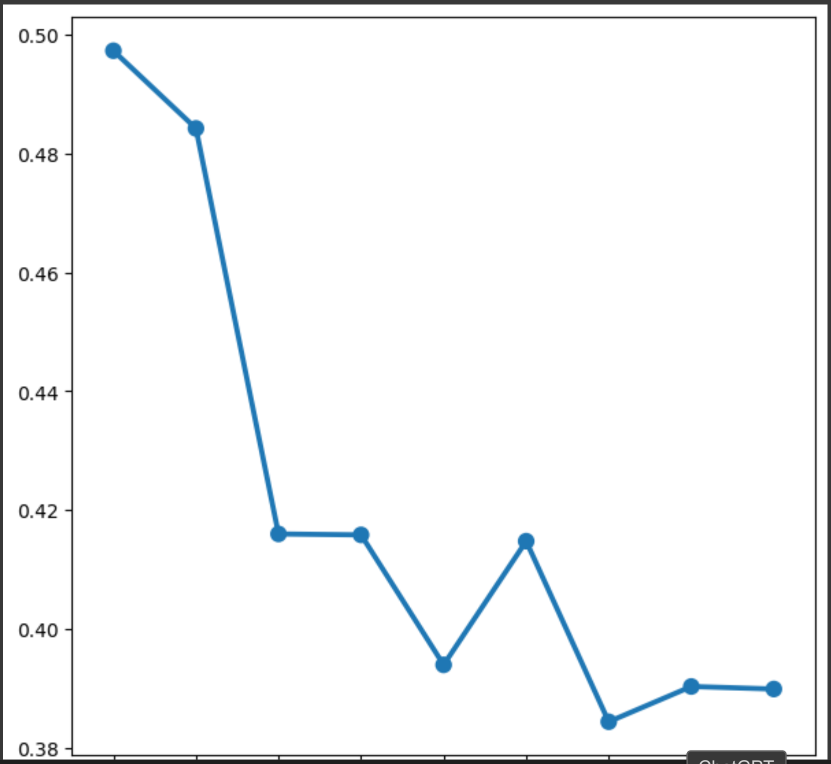
```
sns.pointplot(x=list(track_inertia.keys()),y=list(track_inertia.values()))
```

```
<Axes: >
```



```
sns.pointplot(x=list(silhouette.keys()),y=list(silhouette.values()))
```

<Axes: >



```
kmeans=KMeans(n_clusters=3,init='k-means++',max_iter=300,n_init=10,random_state=0)
kmeans.fit(fm_df_scaled)
print("WCSS for k: %d and silhouette score: %f" %(kmeans.inertia_,silhouette_score(fm_df_scaled,kmeans.labels_)))

```

```
WCSS for k: 603 and silhouette score: 0.484208
```

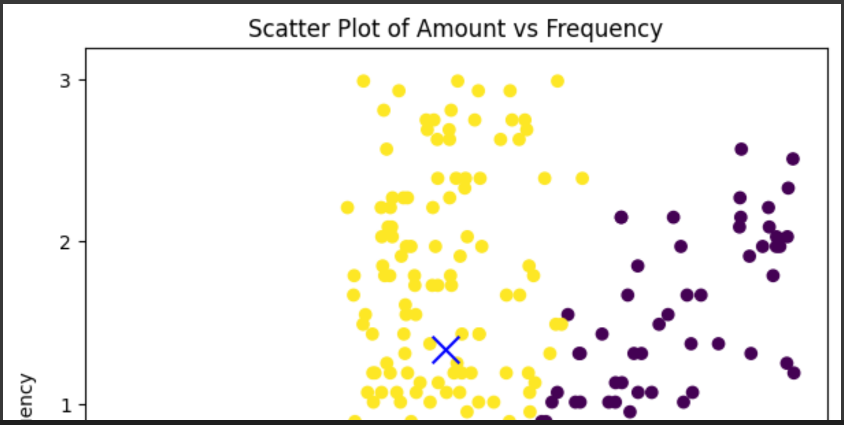
```
kmeans.cluster_centers_
```

```
array([[ 1.94709735,  0.71176975],
       [-0.43576192, -0.56482396],
       [ 0.08037223,  1.33634349]])
```

```
fm['Cluster_Id']=kmeans.labels_
fm.head()
```

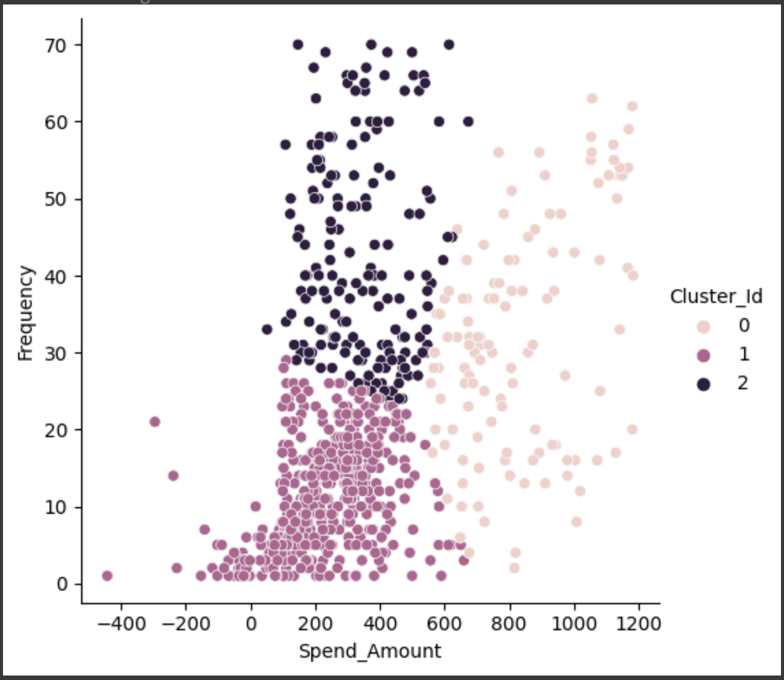
	CustomerID	Spend_Amount	Frequency	Cluster_Id
0	12347.0	711.79	31	0
1	12348.0	892.80	17	0
3	12377.0	1001.52	43	0
4	12383.0	600.72	37	0
5	12386.0	401.90	10	1

```
plt.figure(figsize=(7,7))
plt.scatter(fm_df_scaled["Amount"],fm_df_scaled["Frequency"],c=kmeans.labels_,cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],c='blue',s=200,marker='x')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.title('Scatter Plot of Amount vs Frequency')
plt.show()
```



```
sns.relplot(x='Spend_Amount',y='Frequency', data=fm,hue=fm['Cluster_Id'],height=5)
```

```
<seaborn.axisgrid.FacetGrid at 0x783df54f3a60>
```



```
new_data_point=[390.5,23]
new_data_point_scaled=scaler.transform([new_data_point])
predicted_cluster=kmeans.predict(new_data_point_scaled)
print(f"The new data point belongs to cluster {predicted_cluster[0]}")
```

```
The new data point belongs to cluster 1
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted w
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitted with feat
  warnings.warn(
```