In [1]:

```python
import pandas as pd
import numpy as np
#viz Libraries
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import seaborn as sns
#datetime
import datetime as dt
#StandardSccaler
from sklearn.preprocessing import StandardScaler
#KMeans
from sklearn.cluster import KMeans
```

In [11]:

```python
df = pd.read_csv(r"C:\Users\kavet\Desktop\BE_Sem_7_Assignments-main\BE_Ser
```

```python

df.shape #Dimensions of the data


```

In [12]:

```python
df.head() #Glimpse of the data


```

Out[12]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDA
0	10107	30	95.70	2	2871.00	2/24/20 0
1	10121	34	81.35	5	2765.90	5/7/2003 0
2	10134	41	94.74	2	3884.34	7/1/2003 0
3	10145	45	83.26	6	3746.70	8/25/20 0
4	10159	49	100.00	14	5205.27	10/10/20 0

5 rows × 25 columns

In [13]:

```python
#Removing the variables which dont add significant value fot the analysis.
to_drop = ['PHONE','ADDRESSLINE1','ADDRESSLINE2','STATE','POSTALCODE']
df = df.drop(to_drop, axis=1)

df.isnull().sum()


```

Out[13]:

```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
```

In [13]:

```python
#Removing the variables which dont add significant value fot the analysis.
to_drop = ['PHONE','ADDRESSLINE1','ADDRESSLINE2','STATE','POSTALCODE']
df = df.drop(to_drop, axis=1)

df.isnull().sum()


```

Out[13]:

```
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
SALES                  0
ORDERDATE              0
STATUS                 0
QTR_ID                 0
MONTH_ID               0
YEAR_ID                0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
CITY                   0
COUNTRY                0
TERRITORY           1074
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64
```

In [14]:

```python
df.dtypes

df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

df['ORDERDATE'] = [d.date() for d in df['ORDERDATE']]
df.head()
```

Out[14]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDA
0	10107	30	95.70	2	2871.00	2003-02-
1	10121	34	81.35	5	2765.90	2003-05-
2	10134	41	94.74	2	3884.34	2003-07-
3	10145	45	83.26	6	3746.70	2003-08-
4	10159	49	100.00	14	5205.27	2003-10-

In [15]:

```python
# Calculate Recency, Frequency and Monetary value for each customer
latest_date = df['ORDERDATE'].max() + dt.timedelta(days=1) #latest date i
df_RFM = df.groupby(['CUSTOMERNAME'])

df_RFM = df_RFM.agg({
    'ORDERDATE': lambda x: (latest_date - x.max()).days,
    'ORDERNUMBER': 'count',
    'SALES':'sum'})
```

In [16]:

```python
#Renaming the columns
df_RFM.rename(columns={'ORDERDATE': 'Recency',
                       'ORDERNUMBER': 'Frequency',
                       'SALES': 'MonetaryValue'}, inplace=True)
```

In [17]:

```python
data = df_RFM[['Recency','Frequency','MonetaryValue']]
data.head()
```

Out[17]:

	Recency	Frequency	MonetaryValue
CUSTOMERNAME			
AV Stores, Co.	196	51	157807.81
Alpha Cognac	65	20	70488.44
Amica Models & Co.	265	26	94117.26
Anna's Decorations, Ltd	84	46	153996.13

In [16]:

```
#Renaming the columns
df_RFM.rename(columns={'ORDERDATE': 'Recency',
                       'ORDERNUMBER': 'Frequency',
                       'SALES': 'MonetaryValue'}, inplace=True)
```

In [17]:

```
data = df_RFM[['Recency','Frequency','MonetaryValue']]
data.head()
```

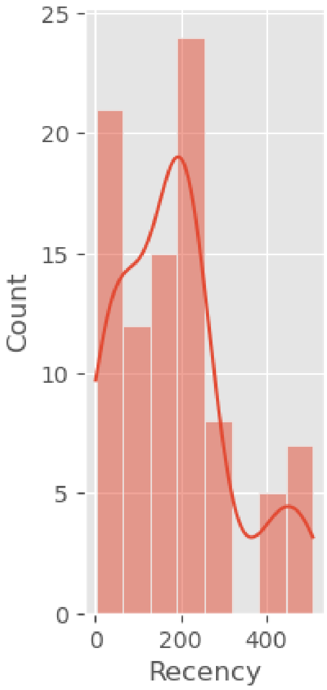
Out[17]:

CUSTOMERNAME	Recency	Frequency	MonetaryValue
AV Stores, Co.	196	51	157807.81
Alpha Cognac	65	20	70488.44
Amica Models & Co.	265	26	94117.26
Anna's Decorations, Ltd	84	46	153996.13
Atelier graphique	188	7	24179.96

In [18]:

```
plt.subplot(1,3,1)
sns.histplot(data['Recency'], kde=True)
```

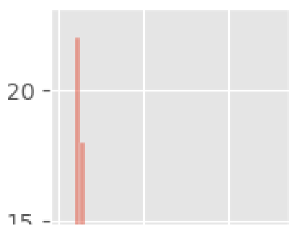
Out[18]: <Axes: xlabel='Recency', ylabel='Count'>



In [19]:

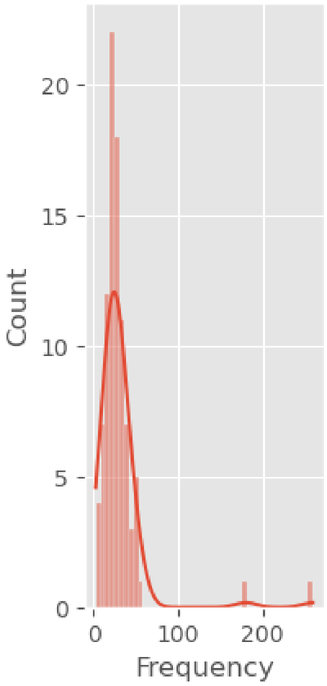
```
plt.subplot(1,3,2)
sns.histplot(data['Frequency'], kde=True)
```

Out[19]: <Axes: xlabel='Frequency', ylabel='Count'>



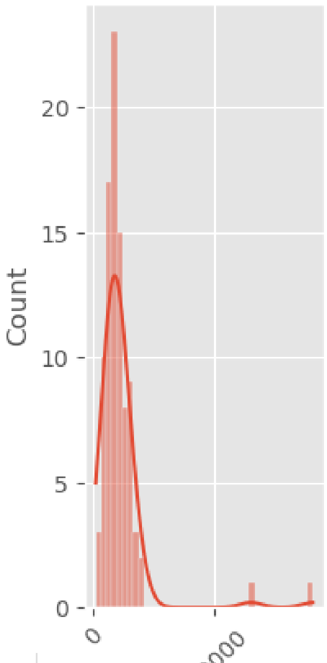
In [19]:
```python
plt.subplot(1,3,2)
sns.histplot(data['Frequency'], kde=True)
```

Out[19]: <Axes: xlabel='Frequency', ylabel='Count'>



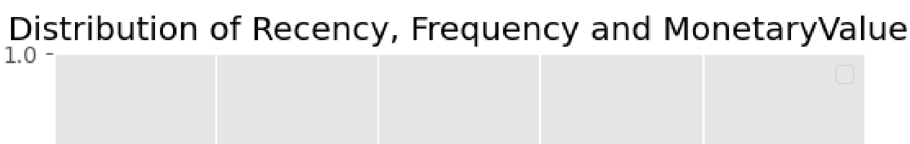
In [20]:
```python
plt.subplot(1,3,3)
plt.xticks(rotation = 45)
sns.histplot(data['MonetaryValue'], kde=True)
```

Out[20]: <Axes: xlabel='MonetaryValue', ylabel='Count'>



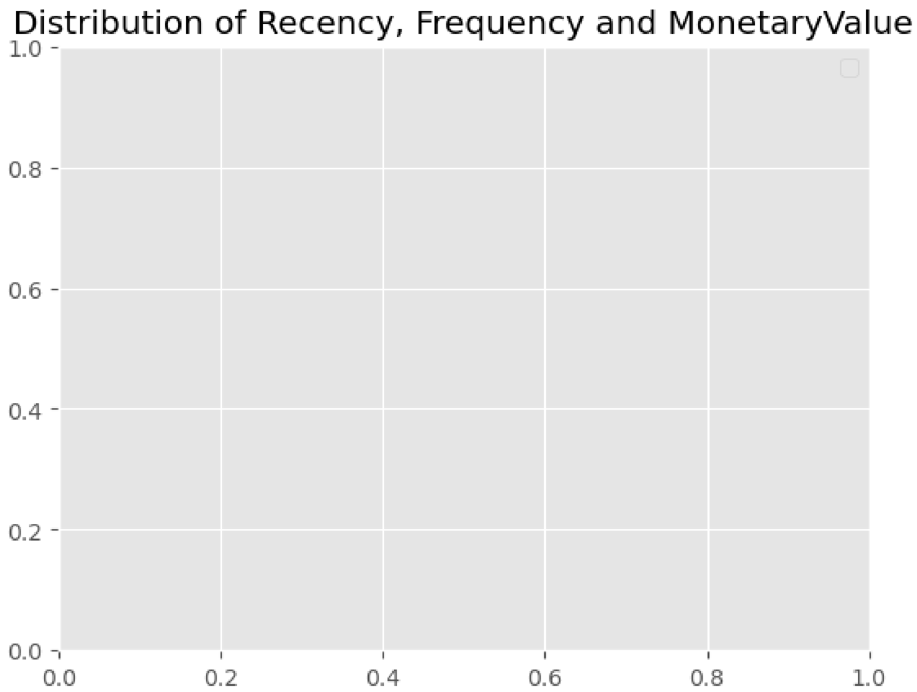
In [21]:
```python
plt.title('Distribution of Recency, Frequency and MonetaryValue')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



In [21]:
```
plt.title('Distribution of Recency, Frequency and MonetaryValue')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



In [22]:
```
data_log = np.log(data)
```

In [23]:
```
data_log.head()
```

Out[23]:

CUSTOMERNAME	Recency	Frequency	MonetaryValue
AV Stores, Co.	5.278115	3.931826	11.969133
Alpha Cognac	4.174387	2.995732	11.163204
Amica Models & Co.	5.579730	3.258097	11.452297
Anna's Decorations, Ltd	4.430817	3.828641	11.944683
Atelier graphique	5.236442	1.945910	10.093279

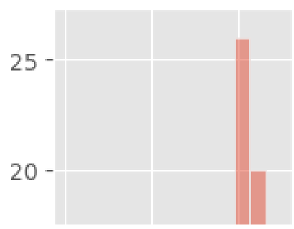
In [24]:
```
plt.figure(figsize=(10,6))
```

Out[24]:
```
<Figure size 1000x600 with 0 Axes>

<Figure size 1000x600 with 0 Axes>
```

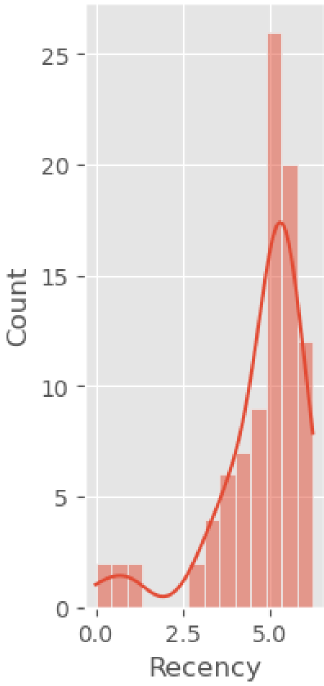
In [25]:
```
plt.subplot(1,3,1)
sns.histplot(data_log['Recency'], kde=True)
```

Out[25]:
```
<Axes: xlabel='Recency', ylabel='Count'>
```



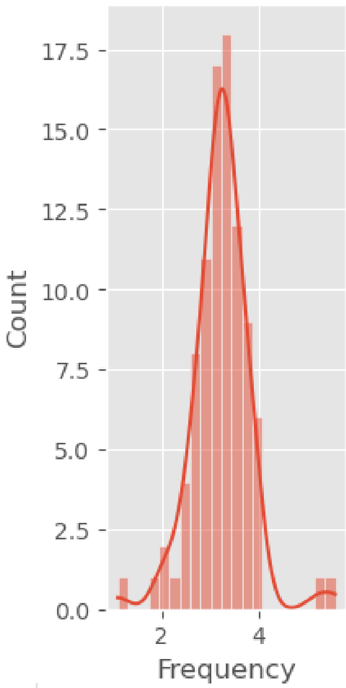
In [25]:
```python
plt.subplot(1,3,1)
sns.histplot(data_log['Recency'], kde=True)
```

Out[25]: <Axes: xlabel='Recency', ylabel='Count'>



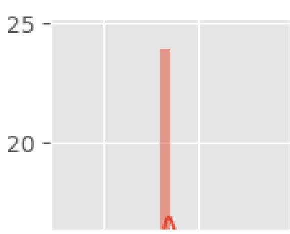
In [26]:
```python
plt.subplot(1,3,2)
sns.histplot(data_log['Frequency'], kde=True)
```

Out[26]: <Axes: xlabel='Frequency', ylabel='Count'>



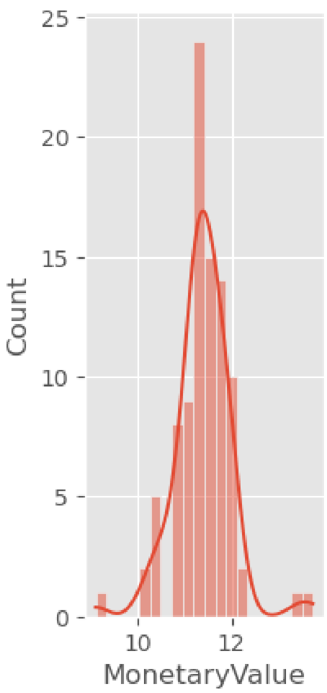
In [27]:
```python
plt.subplot(1,3,3)
sns.histplot(data_log['MonetaryValue'], kde=True)
```

Out[27]: <Axes: xlabel='MonetaryValue', ylabel='Count'>



In [27]:
```python
plt.subplot(1,3,3)
sns.histplot(data_log['MonetaryValue'], kde=True)
```

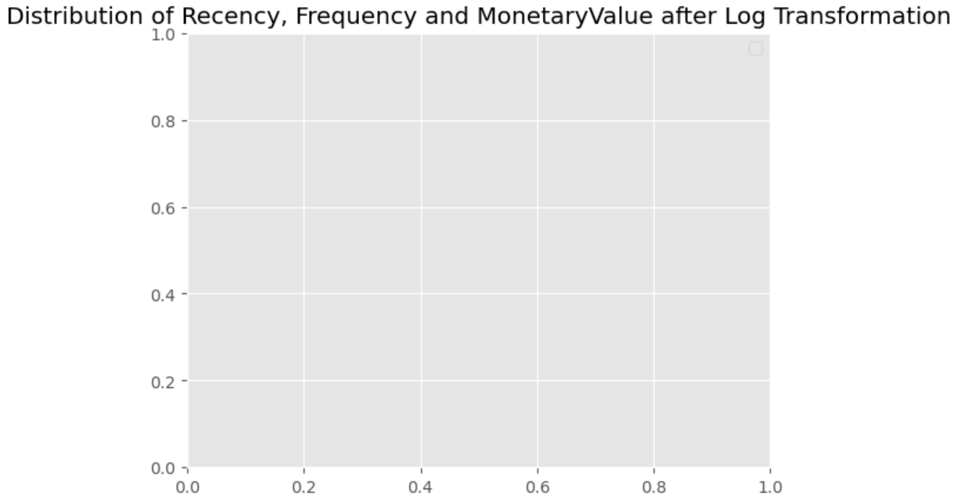
Out[27]: <Axes: xlabel='MonetaryValue', ylabel='Count'>



In [29]:
```python
plt.title('Distribution of Recency, Frequency and MonetaryValue after Log
plt.legend()
```

No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argumen t.

Out[29]: <matplotlib.legend.Legend at 0x1e3e66fce90>



In [30]:
```python
plt.show()
```

In [32]:
```python
# Fit the scaler
scaler.fit(data_log)
```

In [31]:
```python
# Initialize a scaler
scaler = StandardScaler()
```

Out[32]:
```
StandardScaler
StandardScaler()
```

In [33]:
```python
# Scale and center the data
data_normalized = scaler.transform(data_log)
```

In [34]:
```python
# Create a pandas DataFrame
```

In [30]:
```
plt.show()
```

In [32]:
```
# Fit the scaler
scaler.fit(data_log)
```

In [31]:
```
# Initialize a scaler
scaler = StandardScaler()
```

Out[32]:
```
StandardScaler()
```

In [33]:
```
# Scale and center the data
data_normalized = scaler.transform(data_log)
```

In [34]:
```
# Create a pandas DataFrame
data_normalized = pd.DataFrame(data_normalized, index=data_log.index, colu
```

In [35]:
```
# Print summary statistics
data_normalized.describe().round(2)
```

Out[35]:

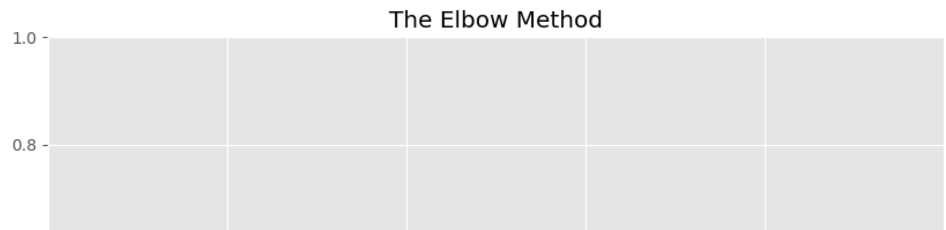
	Recency	Frequency	MonetaryValue
count	92.00	92.00	92.00
mean	0.00	-0.00	0.00
std	1.01	1.01	1.01
min	-3.51	-3.67	-3.82
25%	-0.24	-0.41	-0.39
50%	0.37	0.06	-0.04
75%	0.53	0.45	0.52
max	1.12	4.03	3.92

In [36]:
```
# Fit KMeans and calculate SSE for each k
sse={}
for k in range(1, 21):
    kmeans = KMeans(n_clusters=k, random_state=1)
    kmeans.fit(data_normalized)
    sse[k] = kmeans.inertia_
```

```
ans.py:1412: 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
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1412: 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
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kme
```

In [37]:
```
plt.figure(figsize=(10,6))
plt.title('The Elbow Method')
```

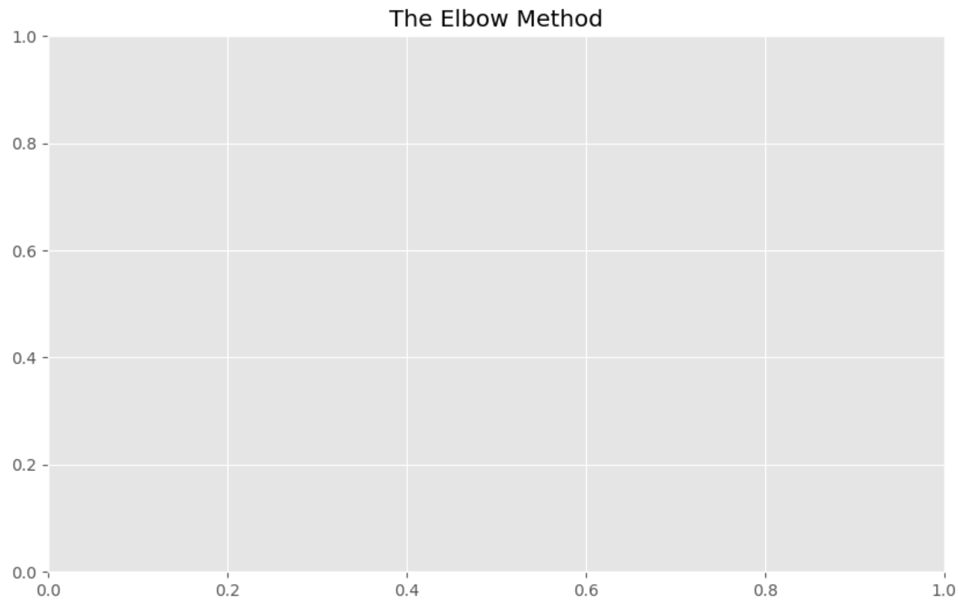
Out[37]:
```
Text(0.5, 1.0, 'The Elbow Method')
```



In [37]:

```python
plt.figure(figsize=(10,6))
plt.title('The Elbow Method')
```

Out[37]: Text(0.5, 1.0, 'The Elbow Method')

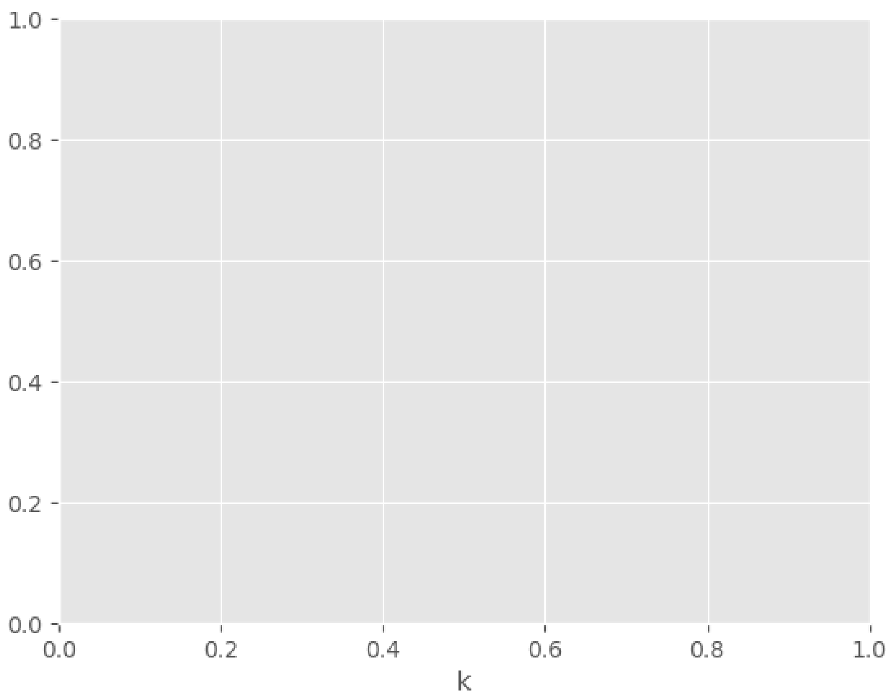


In [38]:

```python
# Add X-axis label "k"
plt.xlabel('k')

```

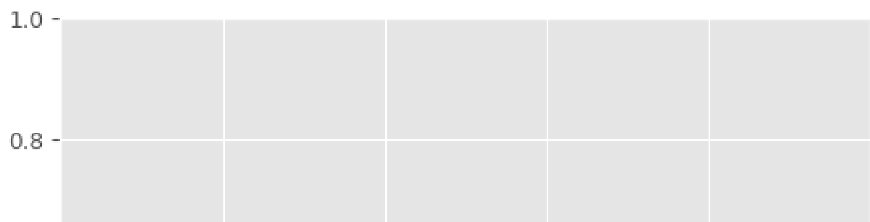
Out[38]: Text(0.5, 0, 'k')



In [39]:

```python
# Add Y-axis label "SSE"
plt.ylabel('SSE')
```

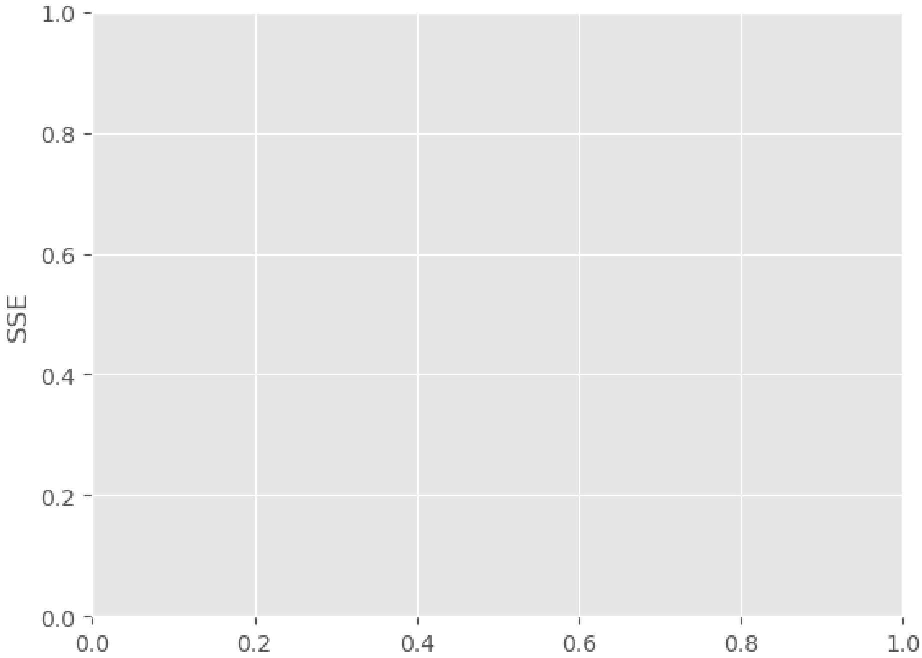
Out[39]: Text(0, 0.5, 'SSE')



In [39]:

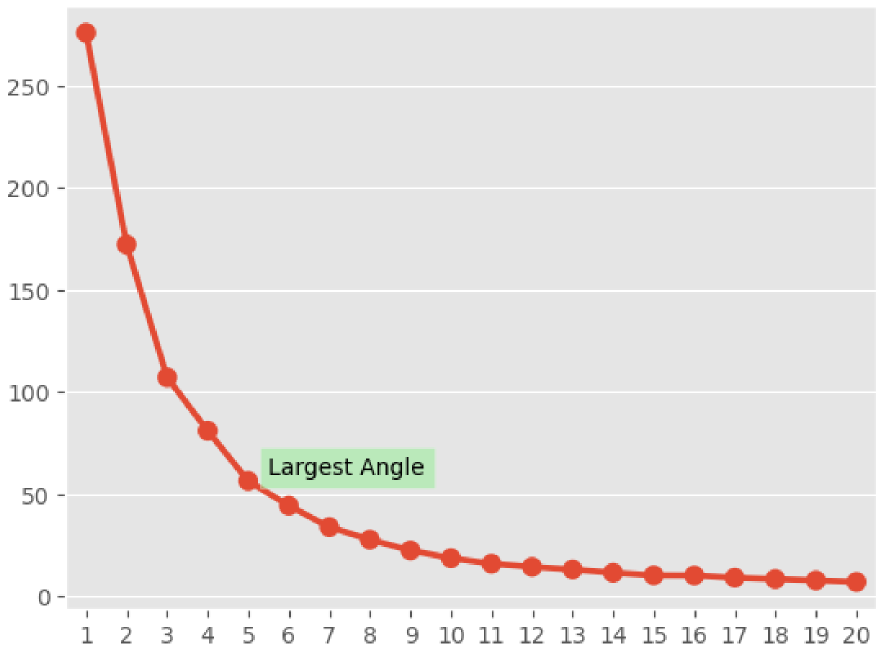
```python
# Add Y-axis label "SSE"
plt.ylabel('SSE')
```

Out[39]: Text(0, 0.5, 'SSE')



In [40]:

```python
# Plot SSE values for each key in the dictionary
sns.pointplot(x=list(sse.keys()), y=list(sse.values()))
plt.text(4.5,60,"Largest Angle",bbox=dict(facecolor='lightgreen', alpha=0
plt.show()
```



In [41]:

```python
# # Running KMeans with 5 clusters
# Initialize KMeans
kmeans = KMeans(n_clusters=5, random_state=1)
```

In [42]:

```python
# Fit K-means clustering on the normalized data set
kmeans.fit(data_normalized)
```

```
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kmean
s.py:1412: FutureWarning: The default value of `n_init` will change from 10 t
o 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with
MKL, when there are less chunks than available threads. You can avoid it by s
etting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [41]:

```
# Running KMeans with 5 clusters
# Initialize KMeans
kmeans = KMeans(n_clusters=5, random_state=1)
```

In [42]:

```
# Fit K-means clustering on the normalized data set
kmeans.fit(data_normalized)
```

```
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kmean
s.py:1412: FutureWarning: The default value of `n_init` will change from 10 t
o 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\kavet\anaconda3\New folder\Lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with
MKL, when there are less chunks than available threads. You can avoid it by s
etting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[42]:

```
KMeans
KMeans(n_clusters=5, random_state=1)
```

In [43]:

```
# Extract cluster labels
cluster_labels = kmeans.labels_
```

In [44]:

```
# Assigning Cluster Labels to Raw Data
# Create a DataFrame by adding a new cluster label column
data_rfm = data.assign(Cluster=cluster_labels)
data_rfm.head()
```

Out[44]:

	Recency	Frequency	MonetaryValue	Cluster
CUSTOMERNAME				
AV Stores, Co.	196	51	157807.81	1
Alpha Cognac	65	20	70488.44	2
Amica Models & Co.	265	26	94117.26	2
Anna's Decorations, Ltd	84	46	153996.13	1
Atelier graphique	188	7	24179.96	0

In [45]:

```
# Group the data by cluster
grouped = data_rfm.groupby(['Cluster'])
```

In [46]:

```
# Calculate average RFM values and segment sizes per cluster value
grouped.agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'MonetaryValue': ['mean', 'count']
  }).round(1)
```

Out[46]:

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
Cluster				
0	324.2	10.7	35628.7	12
1	126.5	37.1	133158.0	31
2	209.2	22.1	78633.2	43
3	2.0	219.5	783576.1	2
4	2.0	38.8	132201.6	4

In [47]:

```
#--------------EXTRA---------------#
# # Calculating relative importance of each attribute
# Calculate average RFM values for each cluster
cluster_avg = data_rfm.groupby(['Cluster']).mean()
print(cluster_avg)
```

```
            Recency   Frequency  MonetaryValue
Cluster
0        324.250000   10.666667   35628.653333
1        126.548387   37.129032  133158.014516
2        209.162791   22.093023   78633.205814
3          2.000000  219.500000  783576.085000
4          2.000000   38.750000  132201.635000
```

In [47]:

```python
#---------------EXTRA---------------#
# # Calculating relative importance of each attribute
# Calculate average RFM values for each cluster
cluster_avg = data_rfm.groupby(['Cluster']).mean()
print(cluster_avg)
```

```
            Recency   Frequency  MonetaryValue
Cluster
0        324.250000   10.666667   35628.653333
1        126.548387   37.129032  133158.014516
2        209.162791   22.093023   78633.205814
3          2.000000  219.500000  783576.085000
4          2.000000   38.750000  132201.635000
```

In [48]:

```python
# Calculate average RFM values for the total customer population
population_avg = data.mean()
print(population_avg)
```

```
Recency             182.826087
Frequency            30.684783
MonetaryValue    109050.313587
dtype: float64
```

In [49]:

```python
# Calculate relative importance of cluster's attribute value compared to p
relative_imp = cluster_avg / population_avg - 1
```

In [50]:

```python
# Print relative importance score rounded to 2 decimals
print(relative_imp.round(2))
```

```
         Recency  Frequency  MonetaryValue
Cluster
0           0.77      -0.65          -0.67
1          -0.31       0.21           0.22
2           0.14      -0.28          -0.28
3          -0.99       6.15           6.19
4          -0.99       0.26           0.21
```

In [51]:

```python
#Plot Relative Importance
# Initialize a plot with a figure size of 8 by 2 inches
plt.figure(figsize=(8, 2))
```

Out[51]: <Figure size 800x200 with 0 Axes>

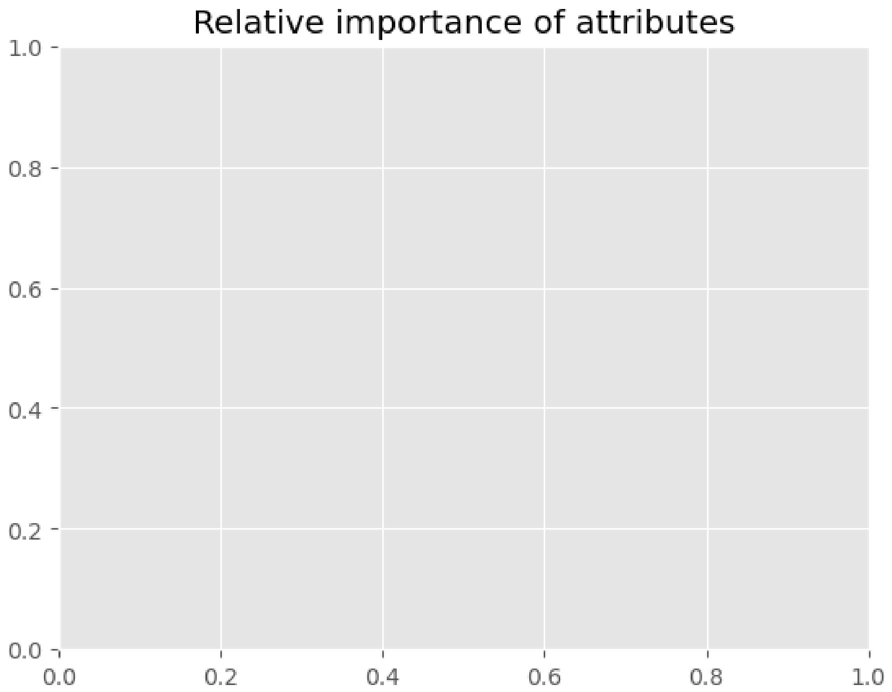
<Figure size 800x200 with 0 Axes>

In [52]:

```python
# Add the plot title
plt.title('Relative importance of attributes')
```

Out[52]: Text(0.5, 1.0, 'Relative importance of attributes')

In [52]:
```
# Add the plot title
plt.title('Relative importance of attributes')
```
Out[52]: Text(0.5, 1.0, 'Relative importance of attributes')



In [53]:
```
# Plot the heatmap
sns.heatmap(data=relative_imp, annot=True, fmt='.2f', cmap='RdYlGn')
plt.show()
```