In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

In [8]:
```
df= pd.read_excel("C:\\Users\\HP\\Desktop\\Copy of online_retail_II.xlsx")
df.head()
```

Out[8]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [9]:
```
## Data preprocessing
#Handling missing values
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)
df.dropna(inplace=True)

#Keeping only positive values
df = df[df['Quantity'] > 0]
df = df[df['Price'] > 0]

# Converting InvoiceDate to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

```
Missing values in each column:
 Invoice             0
StockCode           0
Description      2928
Quantity            0
InvoiceDate         0
Price               0
Customer ID    107927
Country             0
dtype: int64
```

In [13]:
```
## Feature Engineering
# Compute RFM (Recency, Frequency, Monetary) to represent the customer behaviour
current_date = df['InvoiceDate'].max() + pd.DateOffset(days=1)
df['TotalPrice'] = df['Quantity'] * df['Price']
rfm = df.groupby('Customer ID').agg({
    'InvoiceDate': lambda x: (current_date - x.max()).days,
    'Invoice': 'count',
    'TotalPrice': lambda x: np.sum(x)
}).rename(columns={'InvoiceDate': 'Recency', 'Invoice': 'Frequency', 'TotalPrice': 'Monetary'})
print("RFM Dataset:")
rfm.head()
```

```
RFM Dataset:
```

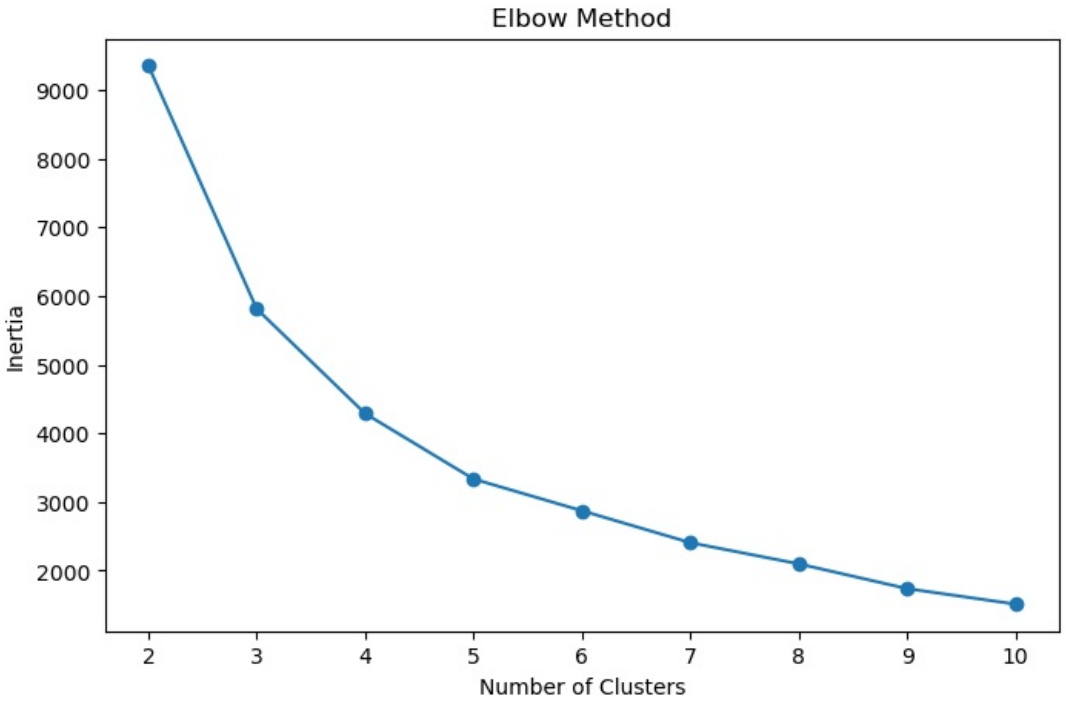
Out[13]:

Customer ID	Recency	Frequency	Monetary
12346.0	165	33	372.86
12347.0	3	71	1323.32
12348.0	74	20	222.16
12349.0	43	102	2671.14
12351.0	11	21	300.93

In [15]:
```
## Scaling the data to normalise the rfm values for effective clustering
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm)
```

In [16]:
```
# Finding optimal clusters using the elbow method
inertia = []
k_range = range(2, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=35)
    kmeans.fit(rfm_scaled)
    inertia.append(kmeans.inertia_)
```

In [17]:
```
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```



In [18]:
```
# Selecting optimal K using silhouette score
best_k = 0
best_score = -1
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(rfm_scaled)
    score = silhouette_score(rfm_scaled, labels)
    if score > best_score:
        best_k, best_score = k, score

print(f'Optimal K: {best_k}, Silhouette Score: {best_score}')
```

```
Optimal K: 4, Silhouette Score: 0.6094127396489585
```

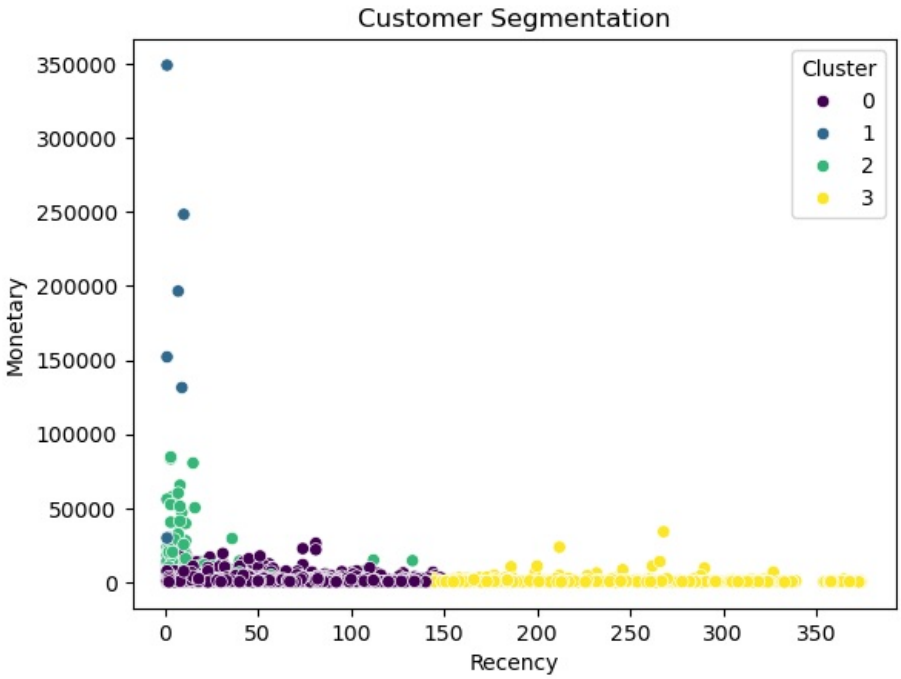
In [19]:
```
# Applying K-Means with optimal K
kmeans = KMeans(n_clusters=best_k, random_state=35)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)
print("RFM Dataset after clustering:")
rfm.head()
```

```
RFM Dataset after clustering:
```

Out[19]:

Customer ID	Recency	Frequency	Monetary	Cluster
12346.0	165	33	372.86	3
12347.0	3	71	1323.32	0
12348.0	74	20	222.16	0
12349.0	43	102	2671.14	0
12351.0	11	21	300.93	0

In [20]:
```
# Visualizing clusters
sns.scatterplot(data=rfm, x='Recency', y='Monetary', hue='Cluster', palette='viridis')
plt.title('Customer Segmentation')
plt.show()
```



In [ ]: