In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('sample_dataset.csv')
df
```

Out[2]:

	index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Cu
0	209268	555200	71459	HANGING JAM JAR T-LIGHT HOLDER	24	6/1/2011 12:05	0.85	
1	207108	554974	21128	GOLD FISHING GNOME	4	5/27/2011 17:14	6.95	
2	167085	550972	21086	SET/6 RED SPOTTY PAPER CUPS	4	4/21/2011 17:05	0.65	
3	471836	576652	22812	PACK 3 BOXES CHRISTMAS PANETTONE	3	11/16/2011 10:39	1.95	
4	115865	546157	22180	RETROSPOT LAMP	2	3/10/2011 8:40	9.95	
...	...	...	...	...	...	...	...	...
2705	174534	551821	84970S	HANGING HEART ZINC T-LIGHT HOLDER	12	5/4/2011 12:20	0.85	
2706	386654	570242	23048	SET OF 10 LANTERNS FAIRY LIGHT STAR	200	10/9/2011 15:40	3.75	
2707	494968	578285	20728	LUNCH BAG CARS BLUE	1	11/23/2011 13:57	1.65	
2708	531871	580974	22865	HAND WARMER OWL DESIGN	12	12/6/2011 15:17	2.10	
2709	110490	545688	22773	GREEN DRAWER KNOB ACRYLIC EDWARDIAN	12	3/6/2011 12:42	1.25	

2710 rows × 9 columns

In [3]:
```
df.head()
```

Out[3]:

	index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custo
0	209268	555200	71459	HANGING JAM JAR T-LIGHT HOLDER	24	6/1/2011 12:05	0.85	17
1	207108	554974	21128	GOLD FISHING GNOME	4	5/27/2011 17:14	6.95	14
2	167085	550972	21086	SET/6 RED SPOTTY PAPER CUPS	4	4/21/2011 17:05	0.65	14
3	471836	576652	22812	PACK 3 BOXES CHRISTMAS PANETTONE	3	11/16/2011 10:39	1.95	17
4	115865	546157	22180	RETROSPOT LAMP	2	3/10/2011 8:40	9.95	13

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2710 entries, 0 to 2709
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   index        2710 non-null   int64  
 1   InvoiceNo    2710 non-null   object 
 2   StockCode    2710 non-null   object 
 3   Description  2704 non-null   object 
 4   Quantity     2710 non-null   int64  
 5   InvoiceDate  2710 non-null   object 
 6   UnitPrice    2710 non-null   float64
 7   CustomerID   2003 non-null   float64
 8   Country      2710 non-null   object 
dtypes: float64(2), int64(2), object(5)
memory usage: 190.7+ KB
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['index', 'InvoiceNo', 'StockCode', 'Description', 'Quantity',
       'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [6]:
```
df.dtypes
```

Out[6]:
```
index            int64
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

In [7]:
```
df.shape
```

Out[7]:
```
(2710, 9)
```

In [8]:
```
df = df.drop('CustomerID', axis=1, errors='ignore')
```

In [9]:
```
df.shape
```

Out[9]:
```
(2710, 8)
```

In [10]:
```
df = pd.get_dummies(df, drop_first=True)
df
```

Out[10]:

	index	Quantity	UnitPrice	InvoiceNo_536401	InvoiceNo_536408	InvoiceNo_5364
0	209268	24	0.85	0	0	
1	207108	4	6.95	0	0	
2	167085	4	0.65	0	0	
3	471836	3	1.95	0	0	
4	115865	2	9.95	0	0	
...	...	...	...	...	...	
2705	174534	12	0.85	0	0	
2706	386654	200	3.75	0	0	
2707	494968	1	1.65	0	0	
2708	531871	12	2.10	0	0	
2709	110490	12	1.25	0	0	

2710 rows × 7258 columns

In [11]:
```
df.fillna(df.mean(), inplace=True)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
index                          0
Quantity                       0
UnitPrice                      0
InvoiceNo_536401               0
InvoiceNo_536408               0
                              ..
Country_Switzerland            0
Country_USA                    0
Country_United Arab Emirates   0
Country_United Kingdom         0
Country_Unspecified            0
Length: 7258, dtype: int64
```

In [13]:
```
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [14]:
```
#      Standardization: Transform data to have a mean of 0 and a standa
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
df_scaled
```

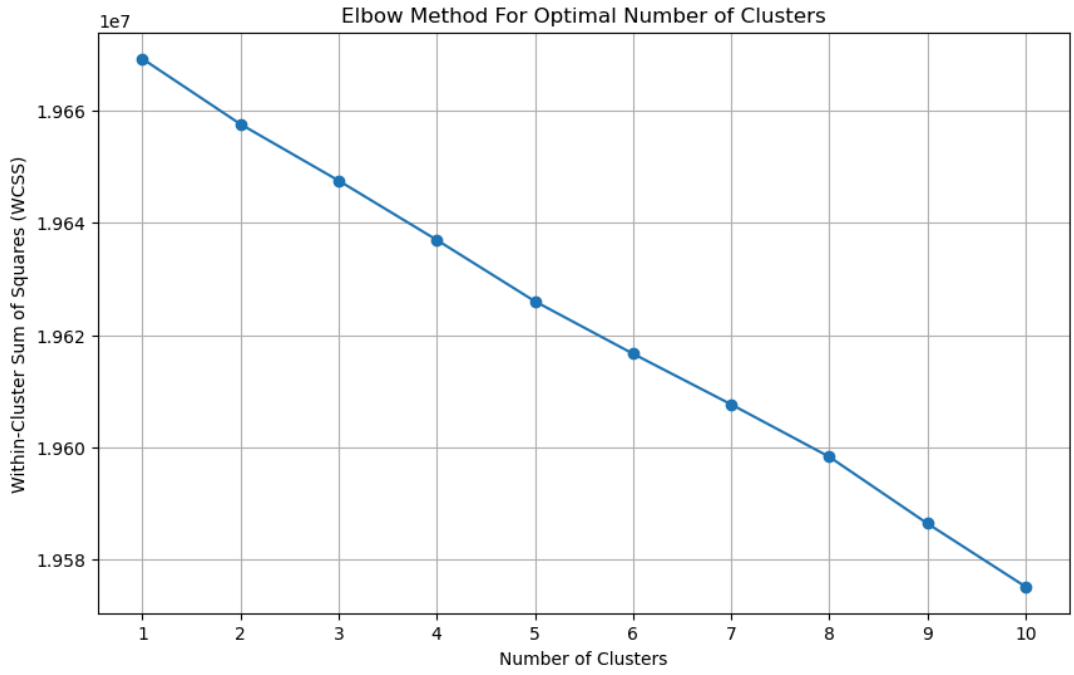
Out[14]:
```
array([[-0.35542256,  0.20527148, -0.35519615, ..., -0.01921301,
         0.32645691, -0.03844732],
       [-0.36907174, -0.08680334,  0.45273428, ..., -0.01921301,
         0.32645691, -0.03844732],
       [-0.62197965, -0.08680334, -0.38168567, ..., -0.01921301,
         0.32645691, -0.03844732],
       ...,
       [ 1.4499341 , -0.13061457, -0.24923806, ..., -0.01921301,
         0.32645691, -0.03844732],
       [ 1.68312653,  0.03002659, -0.18963663, ..., -0.01921301,
         0.32645691, -0.03844732],
       [-0.97960709,  0.03002659, -0.3022171 , ..., -0.01921301,
         0.32645691, -0.03844732]])
```

In [15]:
```
from sklearn.cluster import KMeans
wcss = []
range_n_clusters = range(1, 11)

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++',
                    max_iter=300, n_init=10, random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)

# Plot the elbow graph
plt.figure(figsize=(10, 6))
plt.plot(range_n_clusters, wcss, marker='o')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()
```

```
C:\Users\senap\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:
1036: UserWarning: KMeans is known to have a memory leak on Windows wi
th MKL, when there are less chunks than available threads. You can avo
id it by setting the environment variable OMP_NUM_THREADS=11.
  warnings.warn(
```



In [16]:
```
optimal_clusters = 3
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++',
                max_iter=300, n_init=10, random_state=42)
y_kmeans = kmeans.fit_predict(df_scaled)
y_kmeans
```

Out[16]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [17]:
```
df['Cluster'] = y_kmeans
```

In [18]:
```
from sklearn.decomposition import PCA
```

In [19]:
```
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)

df_pca = pd.DataFrame(df_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = y_kmeans
```

In [20]:
```
plt.figure(figsize=(10, 6))
for cluster in range(optimal_clusters):
    cluster_data = df_pca[df_pca['Cluster'] == cluster]
    plt.scatter(cluster_data['PC1'], cluster_data['PC2'],
                label=f'Cluster {cluster}')
plt.title('Customer Clusters')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```