```
#Load the customer and transaction datasets using Pandas.
import pandas as pd
```

Clean the data by handling missing values and duplicates.

```
# Check for missing values
customers.dropna(inplace=True)
transactions.dropna(inplace=True)
# Remove duplicates
customers.drop_duplicates(inplace=True)
transactions.drop_duplicates(inplace=True)
# Load datasets
customers = pd.read_csv('Customers.csv')
transactions = pd.read_csv('Transactions.csv')
# Example feature engineering
data['Total_Spending'] = data.groupby('CustomerID')['Amount'].transform('sum')
data['Purchase_Frequency'] = data.groupby('CustomerID')['TransactionID'].transform('count')
data['Recency'] = (data['Date'].max() - data['Date']).dt.days
#Merge the customer and transaction data on a common identifier (e.g., Customer ID).
data = pd.merge(customers, transactions, on='CustomerID')
```

Use Matplotlib or Seaborn to visualize the clusters.

```
#Create relevant features for clustering, such as total spending, frequency of purchases, and recency of purchases.
# Example feature engineering
data['Total_Spending'] = data.groupby('CustomerID')['Amount'].transform('sum')
data['Purchase_Frequency'] = data.groupby('CustomerID')['TransactionID'].transform('count')
data['Recency'] = (data['Date'].max() - data['Date']).dt.days
from sklearn.cluster import KMeans
# Select features for clustering
features = data[['Total_Spending', 'Purchase_Frequency', 'Recency']]
# Determine optimal number of clusters using Elbow method
inertia = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(features)
    inertia.append(kmeans.inertia_)
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Total_Spending', y='Purchase_Frequency', hue='Cluster', palette='viridis')
plt.title('Customer Segmentation Clusters')
plt.xlabel('Total Spending')
plt.ylabel('Purchase Frequency')
plt.legend(title='Cluster')
plt.show()
# Fit the K-Means model with the chosen number of clusters.
optimal_k = 4  # Chosen based on Elbow method analysis
kmeans = KMeans(n_clusters=optimal_k)
data['Cluster'] = kmeans.fit_predict(features)
# Calculate the Davies-Bouldin Index to evaluate clustering quality.
from sklearn.metrics import davies_bouldin_score
db_index = davies_bouldin_score(features, data['Cluster'])
```