

```

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import seaborn as sns

# Merge datasets for comprehensive analysis
merged_data = pd.merge(transactions, customers, on='CustomerID')

# Feature Engineering
# Aggregate transaction data for each customer
customer_transactions = merged_data.groupby('CustomerID').agg({
    'TotalValue': 'sum', # Total spending
    'Quantity': 'sum', # Total items purchased
    'TransactionID': 'count' # Number of transactions
}).reset_index()

# Merge with customer profile data
customer_data = pd.merge(customer_transactions, customers, on='CustomerID')

# Encode categorical variables (Region)
encoder = OneHotEncoder()
encoded_region = encoder.fit_transform(customer_data[['Region']]).toarray()
encoded_region_df = pd.DataFrame(encoded_region, columns=encoder.get_feature_names_out(['Region']))

# Combine encoded features with numerical features
customer_data_processed = pd.concat([customer_data[['TotalValue', 'Quantity', 'TransactionID']], encoded_region_df], axis=1)

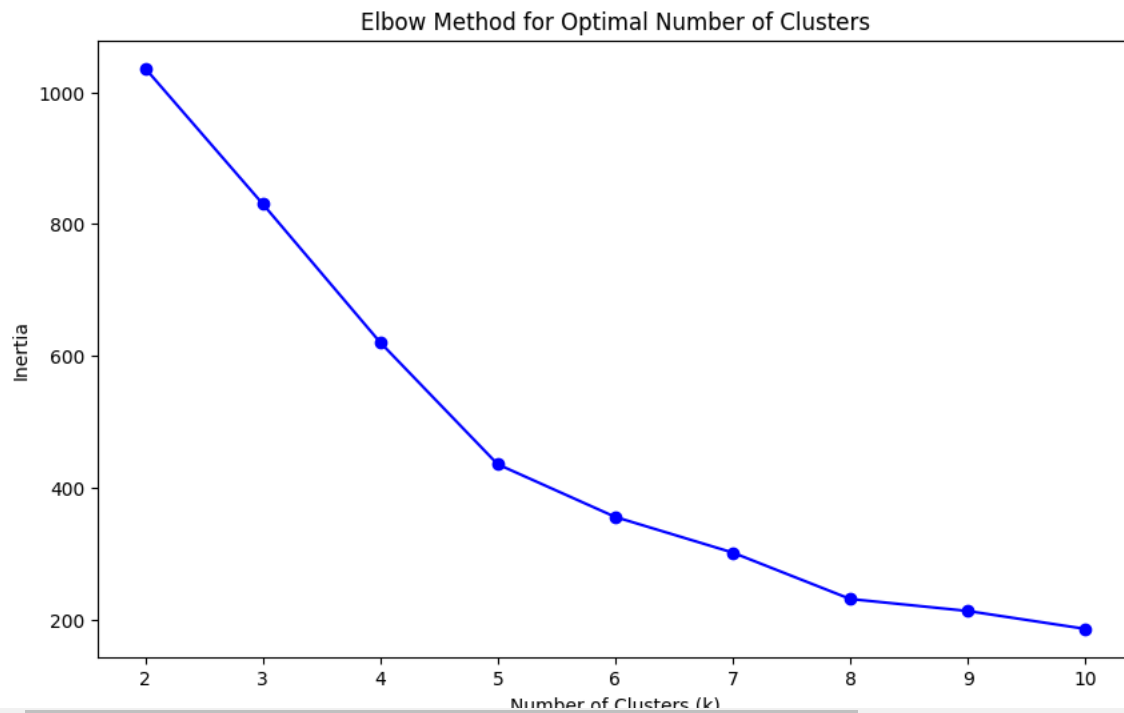
# Standardize the data
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data_processed)

# Determine the optimal number of clusters using the Elbow Method
inertia = []
db_scores = []
k_values = range(2, 11)

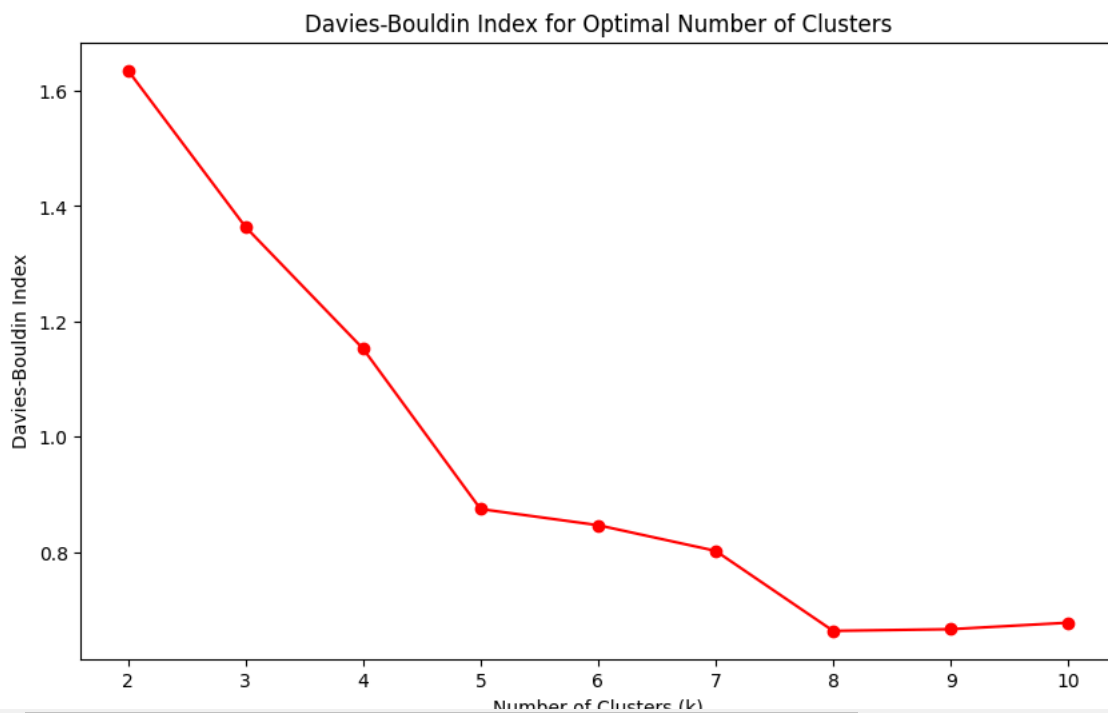
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(customer_data_scaled)
    inertia.append(kmeans.inertia_)
    db_scores.append(davies_bouldin_score(customer_data_scaled, kmeans.labels_))

# Plot the Elbow Method graph
plt.figure(figsize=(10, 6))
plt.plot(k_values, inertia, marker='o', color='blue')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()

```



```
# Plot the Davies-Bouldin Index graph
plt.figure(figsize=(10, 6))
plt.plot(k_values, db_scores, marker='o', color='red')
plt.title('Davies-Bouldin Index for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Davies-Bouldin Index')
plt.show()
```



```
# Choose the optimal number of clusters (e.g., k=4 based on the Elbow Method and DB Index)
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)
```

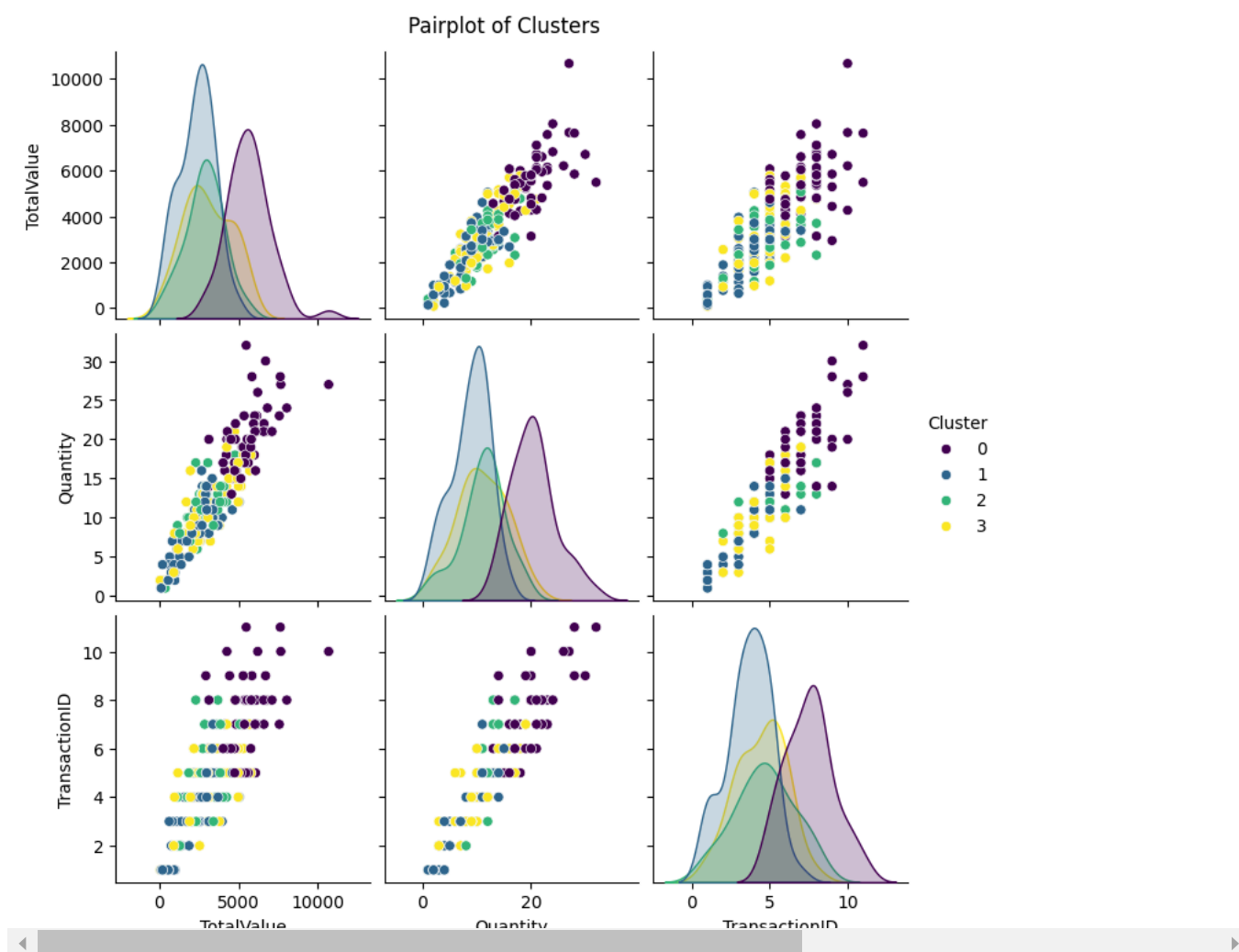
```
# Calculate clustering metrics
db_index = davies_bouldin_score(customer_data_scaled, kmeans.labels_)
print(f"Davies-Bouldin Index for k={optimal_k}: {db_index}")
```




Davies-Bouldin Index for k=4: 1.1529548592400125

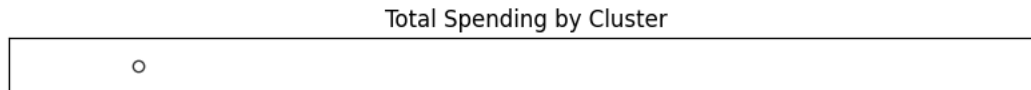
```
# Visualize the clusters
```

```
# Pairplot of TotalValue, Quantity, and TransactionID
sns.pairplot(customer_data, hue='Cluster', vars=['TotalValue', 'Quantity', 'TransactionID'], palette='viridis')
plt.suptitle('Pairplot of Clusters', y=1.02)
plt.show()
```



```
# Boxplot of TotalValue by Cluster
plt.figure(figsize=(10, 6))
sns.boxplot(x='Cluster', y='TotalValue', data=customer_data, palette='viridis')
plt.title('Total Spending by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Total Spending')
plt.show()
```

 <ipython-input-44-4e0bf5f2351d>:3: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend` to `True` to get a legend.
`sns.boxplot(x='Cluster', y='TotalValue', data=customer_data, palette='viridis')`



```
# Boxplot of Quantity by Cluster
plt.figure(figsize=(10, 6))
sns.boxplot(x='Cluster', y='Quantity', data=customer_data, palette='viridis')
plt.title('Total Items Purchased by Cluster')
plt.xlabel('Cluster')
```