C. Task 3: Customer Segmentation / Clustering:

Step 1: Load Data

```
uploaded = files.upload()

customers = pd.read_csv('Customers.csv')

transactions = pd.read_csv('Transactions.csv')
```

Step 2: Merge and Prepare Data

Combine customer and transaction data to create a feature set.

```
 # Merge datasets

 merged_data = transactions.merge(customers, on='CustomerID', how='left')

  # Aggregate transaction data for each customer

   customer_features = merged_data.groupby('CustomerID').agg({

  'TotalValue': 'sum',  # Total revenue

   'TransactionID': 'count',  # Number of transactions

    }).reset_index()

   # Merge aggregated data with customer profiles

   customer_features = customer_features.merge(customers, on='CustomerID', how='left')

    # Encode categorical data (e.g., Region)

    customer_features = pd.get_dummies(customer_features, columns=['Region'], drop_first=True)

   # Drop non-relevant columns

   customer_features = customer_features.drop(columns=['CustomerID', 'CustomerName',

   'SignupDate'])
```

Step 3: Normalize the Features

Normalize the data for clustering.

```
 scaler = StandardScaler()

 scaled_features = scaler.fit_transform(customer_features)
```

Step 4: Perform Clustering

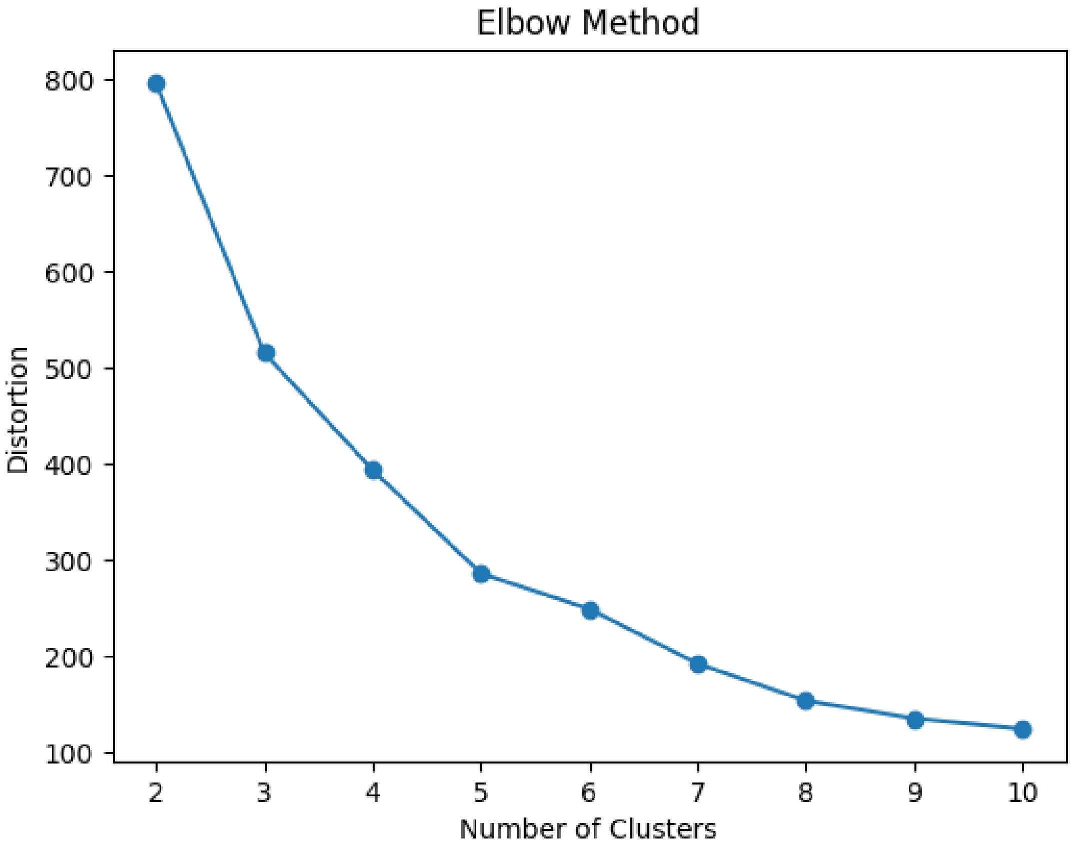
Use K-Means to cluster customers. Choose the number of clusters between 2 and 10.

1. **Elbow Method (Optional)**: Determine the optimal number of clusters:

```
distortions = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    distortions.append(kmeans.inertia_)

plt.plot(range(2, 11), distortions, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Distortion')
plt.show()
```

OUTPUT:



2. **Fit K-Means: Select a cluster count (e.g., 4):**

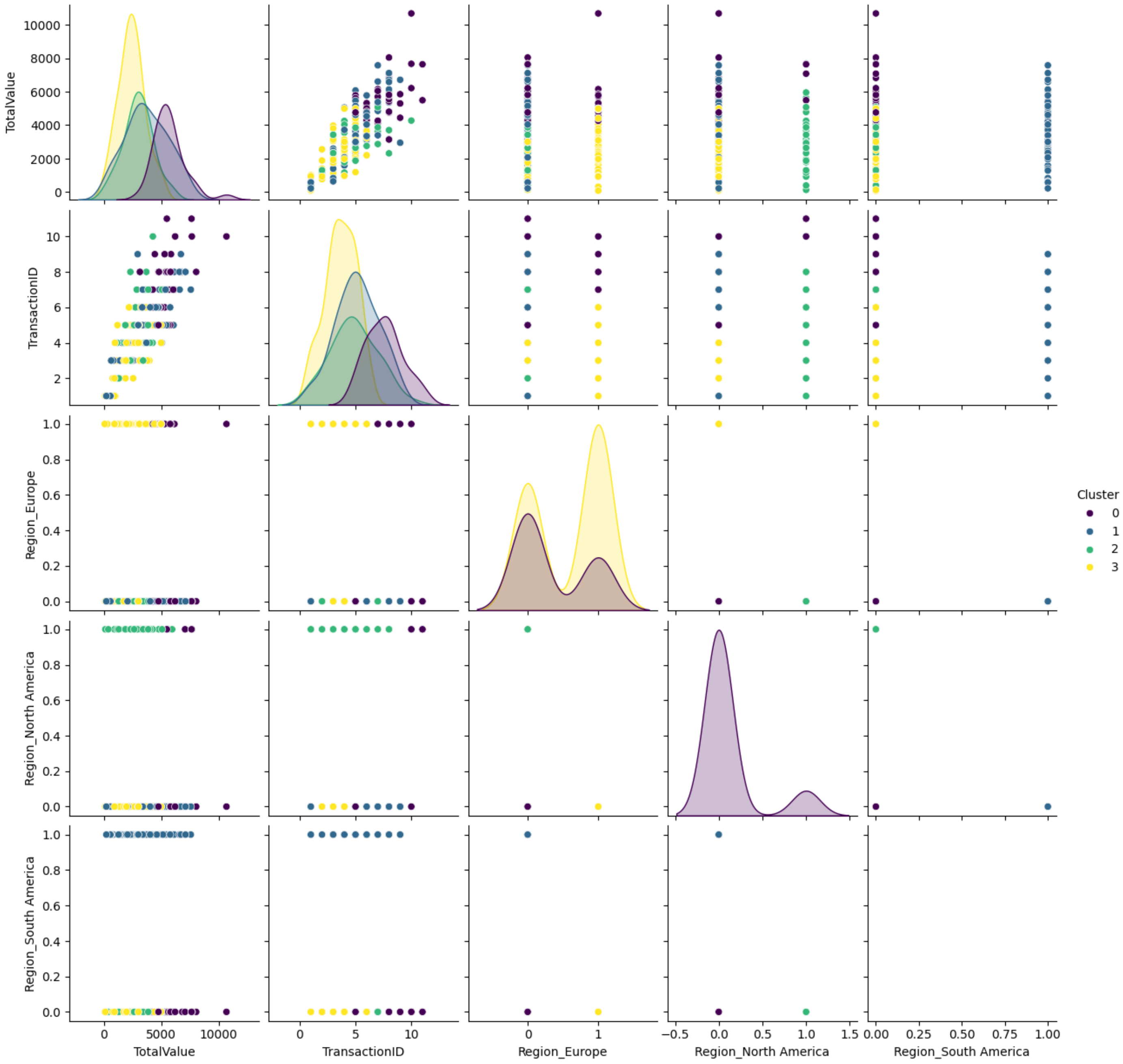
```
kmeans = KMeans(n_clusters=4, random_state=42)
labels = kmeans.fit_predict(scaled_features)
customer_features['Cluster'] = labels
```

Step 5: Evaluate Clustering

```
# Calculate DB Index
db_index = davies_bouldin_score(scaled_features, labels)
print(f"Davies-Bouldin Index: {db_index}")

# Visualize clusters using pair plots
sns.pairplot(customer_features, hue='Cluster', palette='viridis')
plt.show()
```

OUTPUT:


TotalValue
TransactionID
Region_Europe
Region_North America
Region_South America
TotalValue
TransactionID
Region_Europe
Region_North America
Region_South America
Cluster
● 0
● 1
● 2
● 3