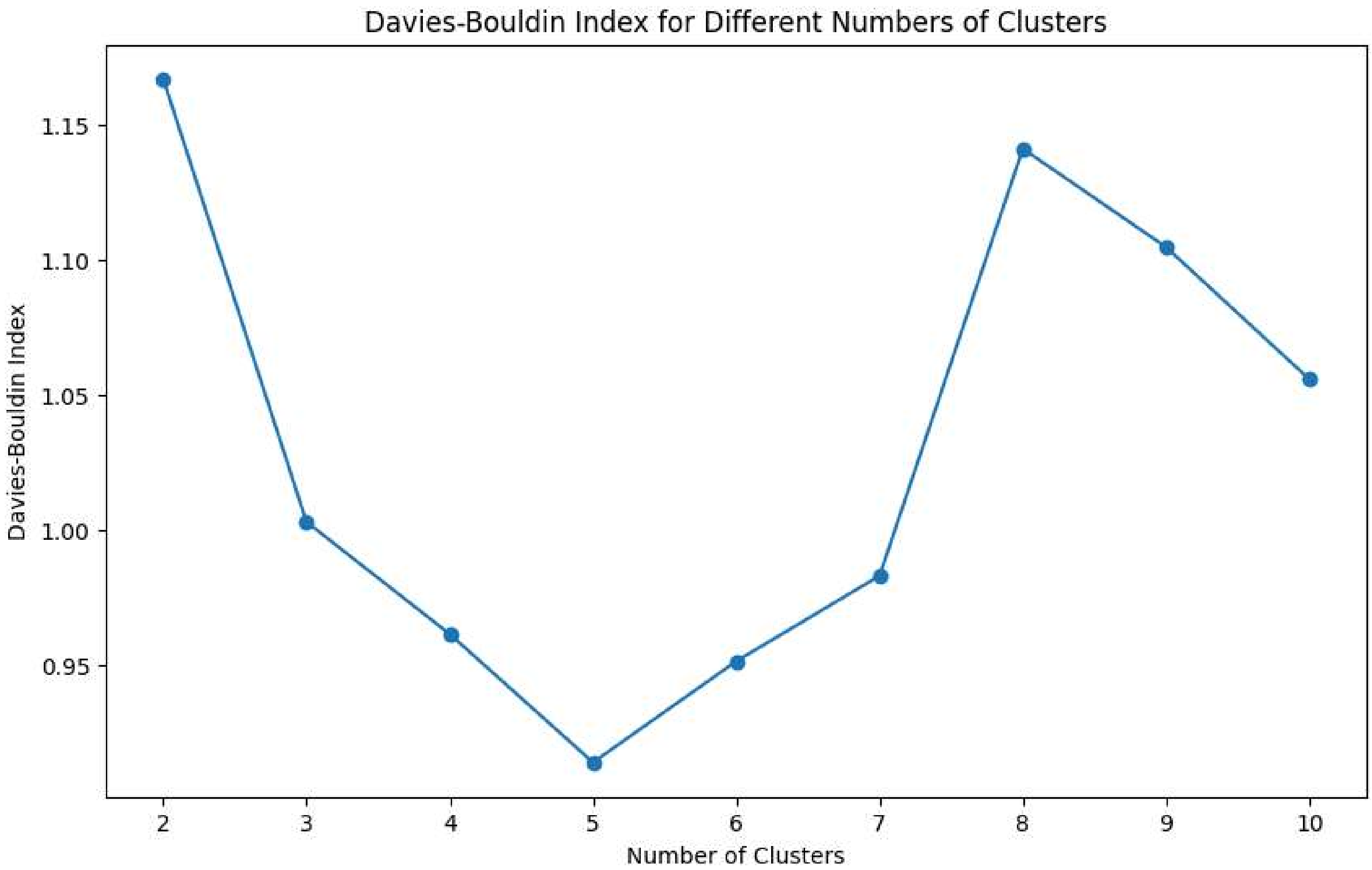
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
import warnings

warnings.filterwarnings('ignore')

# Load the datasets
customers_df = pd.read_csv('Customers.csv')
products_df = pd.read_csv('Products (1).csv')
transactions_df = pd.read_csv('Transactions.csv')

# Display the first few rows of each dataset
print(customers_df.head())
print(products_df.head())
print(transactions_df.head())

# Check for missing values
print(customers_df.isnull().sum())
print(products_df.isnull().sum())
print(transactions_df.isnull().sum())

# Basic information about the datasets
print(customers_df.info())
print(products_df.info())
print(transactions_df.info())
```

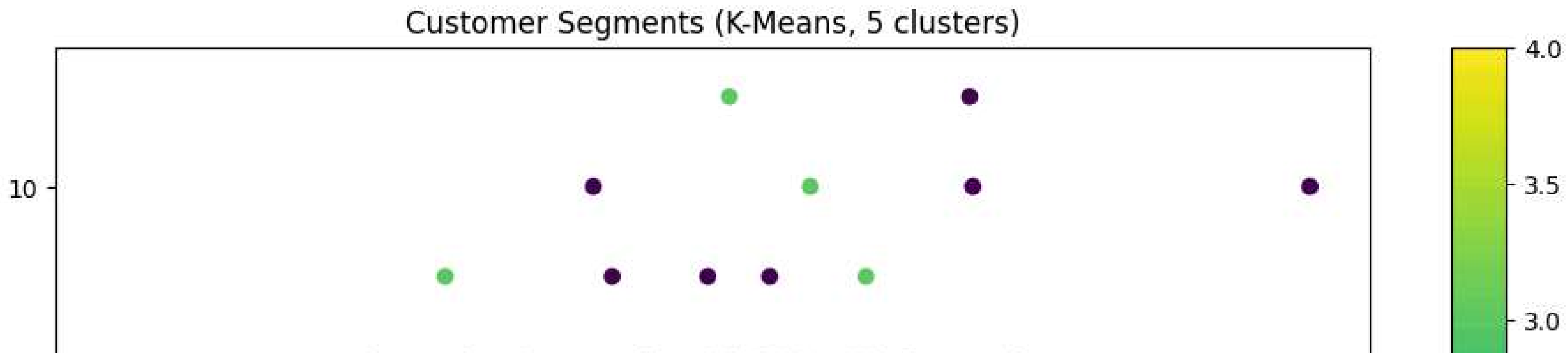
```
Category       0
Price          0
dtype: int64
TransactionID      0
CustomerID         0
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price              0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
```

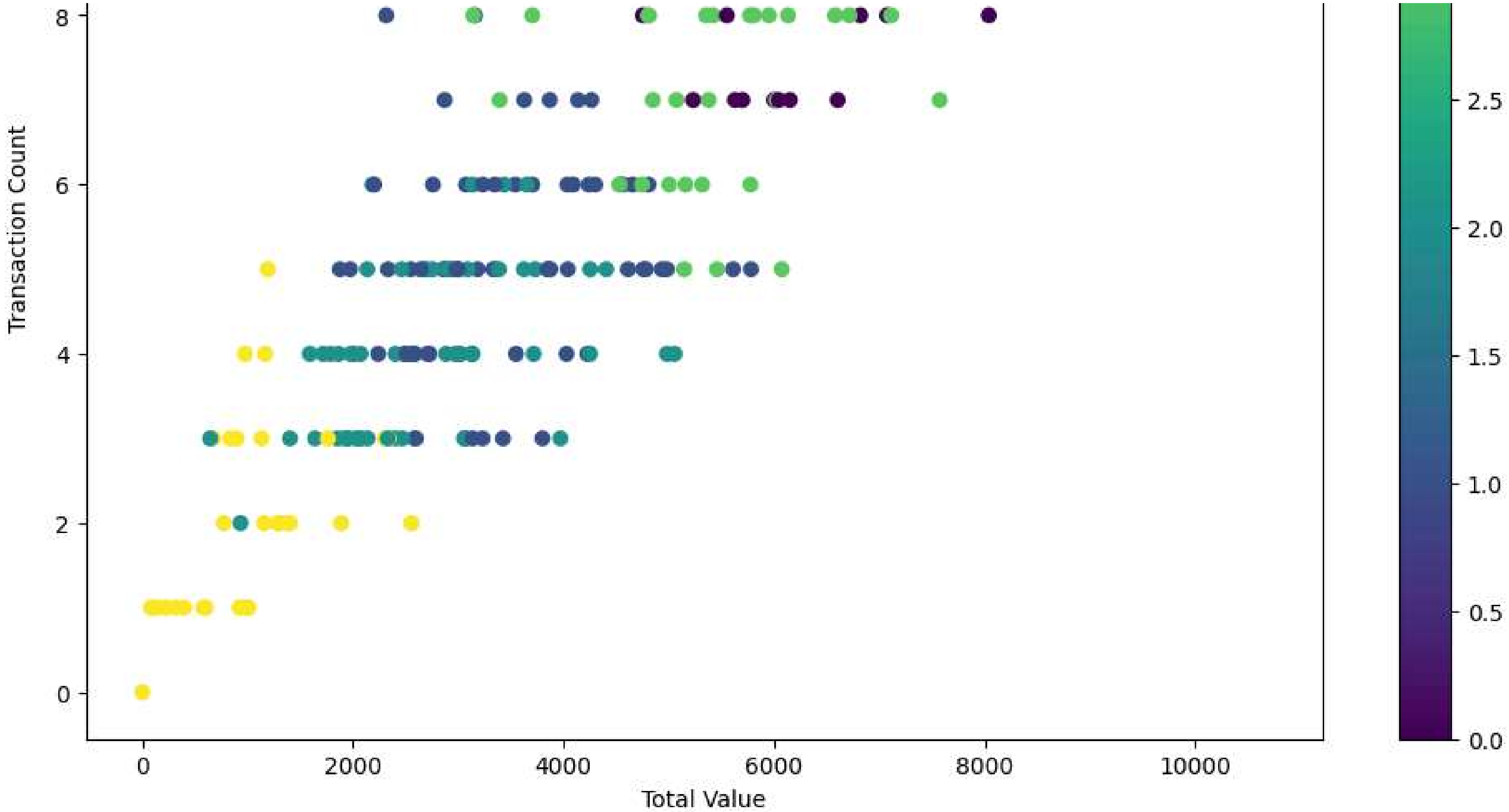
```
 3   TransactionDate  1000 non-null   object
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB
None
```

```
# Prepare data for clustering
cluster_data = customer_data[features].copy()

# Normalize the features
scaler = StandardScaler()
normalized_cluster_data = scaler.fit_transform(cluster_data)

# Function to perform K-Means clustering and calculate DB Index
def perform_kmeans(n_clusters):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(normalized_cluster_data)
    db_index = davies_bouldin_score(normalized_cluster_data, cluster_labels)
    return kmeans, cluster_labels, db_index

# Try different numbers of clusters
cluster_range = range(2, 11)
db_scores = []

for n_clusters in cluster_range:
    _, _, db_index = perform_kmeans(n_clusters)
    db_scores.append(db_index)

# Plot DB Index scores
plt.figure(figsize=(10, 6))
plt.plot(cluster_range, db_scores, marker='o')
plt.title('Davies-Bouldin Index for Different Numbers of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Davies-Bouldin Index')
plt.show()

# Choose the number of clusters with the lowest DB Index
optimal_clusters = cluster_range[db_scores.index(min(db_scores))]
```

```
# Perform final clustering with optimal number of clusters
final_kmeans, final_labels, final_db_index = perform_kmeans(optimal_clusters)

# Add cluster labels to the customer data
customer_data['Cluster'] = final_labels

# Visualize the clusters
plt.figure(figsize=(12, 8))
scatter = plt.scatter(customer_data['TotalValue'], customer_data['TransactionCount'],
                      c=customer_data['Cluster'], cmap='viridis')
plt.colorbar(scatter)
plt.title(f'Customer Segments (K-Means, {optimal_clusters} clusters)')
plt.xlabel('Total Value')
plt.ylabel('Transaction Count')
plt.show()

print(f"Number of clusters: {optimal_clusters}")
print(f"Davies-Bouldin Index: {final_db_index}")

# Calculate other relevant metrics
print("\nCluster sizes:")
print(customer_data['Cluster'].value_counts())

print("\nCluster centroids:")
centroids = scaler.inverse_transform(final_kmeans.cluster_centers_)
centroid_df = pd.DataFrame(centroids, columns=features)
print(centroid_df)
```


Davies-Bouldin Index for Different Numbers of Clusters
Davies-Bouldin Index
1.15
1.10
1.05
1.00
0.95
2
3
4
5
6
7
8
9
10
Number of Clusters


Customer Segments (K-Means, 5 clusters)
10
4.0
3.5
3.0




```
Number of clusters: 5
Davies-Bouldin Index: 0.9140504085547727

Cluster sizes:
Cluster
1    63
2    54
3    32
4    32
0    19
Name: count, dtype: int64

Cluster centroids:
   DaysSinceSignup   TotalValue  TransactionCount
0       324.631579  6285.505263          8.263158
```