```
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score, silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
```

```
# Step 1: Load the data
customers = pd.read_csv("Customers.csv")
transactions = pd.read_csv("Transactions.csv")
```

```
customers.head(5)
```

	CustomerID	CustomerName	Region	SignupDate
0	C0001	Lawrence Carroll	South America	2022-07-10
1	C0002	Elizabeth Lutz	Asia	2022-02-13
2	C0003	Michael Rivera	South America	2024-03-07
3	C0004	Kathleen Rodriguez	South America	2022-10-09
4	C0005	Laura Weber	Asia	2022-08-15

Next steps: Generate code with customers | View recommended plots | New interactive sheet

```
transactions.head(5)
```

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	Price
0	T00001	C0199	P067	2024-08-25 12:38:23	1	300.68	300.68
1	T00112	C0146	P067	2024-05-27 22:23:54	1	300.68	300.68
2	T00166	C0127	P067	2024-04-25 07:38:55	1	300.68	300.68
3	T00272	C0087	P067	2024-03-26 22:55:37	2	601.36	300.68
4	T00363	C0070	P067	2024-03-21 15:10:10	3	902.04	300.68

Next steps: Generate code with transactions | View recommended plots | New interactive sheet

```
# Step 2: Data Preprocessing
# Aggregating transaction data
customer_transaction_summary = transactions.groupby('CustomerID').agg(
    TotalValue=('TotalValue', 'sum'),
    AverageTransactionValue=('TotalValue', 'mean'),
    TotalQuantity=('Quantity', 'sum'),
    LastTransactionDate=('TransactionDate', 'max')
).reset_index()
```

```
# Merge with customer data
customers['SignupDate'] = pd.to_datetime(customers['SignupDate'])
customer_transaction_summary['LastTransactionDate'] = pd.to_datetime(customer_transaction_summary['LastTransactionDate'])
merged_data = pd.merge(customers, customer_transaction_summary, on='CustomerID', how='inner')
```

```
merged_data.head(5)
```

	CustomerID	CustomerName	Region	SignupDate	TotalValue	AverageTransactionValue	TotalQuantity	LastTransactionDate
0	C0001	Lawrence Carroll	South America	2022-07-10	3354.52	670.904	12	2024-11-02 17:04:16
1	C0002	Elizabeth Lutz	Asia	2022-02-13	1862.74	465.685	10	2024-12-03 01:41:41
2	C0003	Michael Rivera	South America	2024-03-07	2725.38	681.345	14	2024-08-24 18:54:04
		Kathleen	South					

Next steps: Generate code with merged_data | View recommended plots | New interactive sheet

```
# Add derived features
merged_data['CustomerTenureDays'] = (merged_data['LastTransactionDate'] - merged_data['SignupDate']).dt.days
```

```
# Select features for clustering
features = merged_data[['TotalValue', 'AverageTransactionValue', 'TotalQuantity', 'CustomerTenureDays']]
```

```
print(features)
```

```
     TotalValue  AverageTransactionValue  TotalQuantity  CustomerTenureDays
0       3354.52               670.904000             12                 846
1       1862.74               465.685000             10                1024
2       2725.38               681.345000             14                 170
3       5354.88               669.360000             23                 806
4       2034.24               678.080000              7                 812
..          ...                      ...            ...                 ...
194     4982.88              1245.720000             12                 922
195     1928.65               642.883333              9                 647
196      931.83               465.915000              3                 950
197     1979.28               494.820000              9                 693
198     4758.60               951.720000             16                 549

[199 rows x 4 columns]
```

```
# Step 3: Feature Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

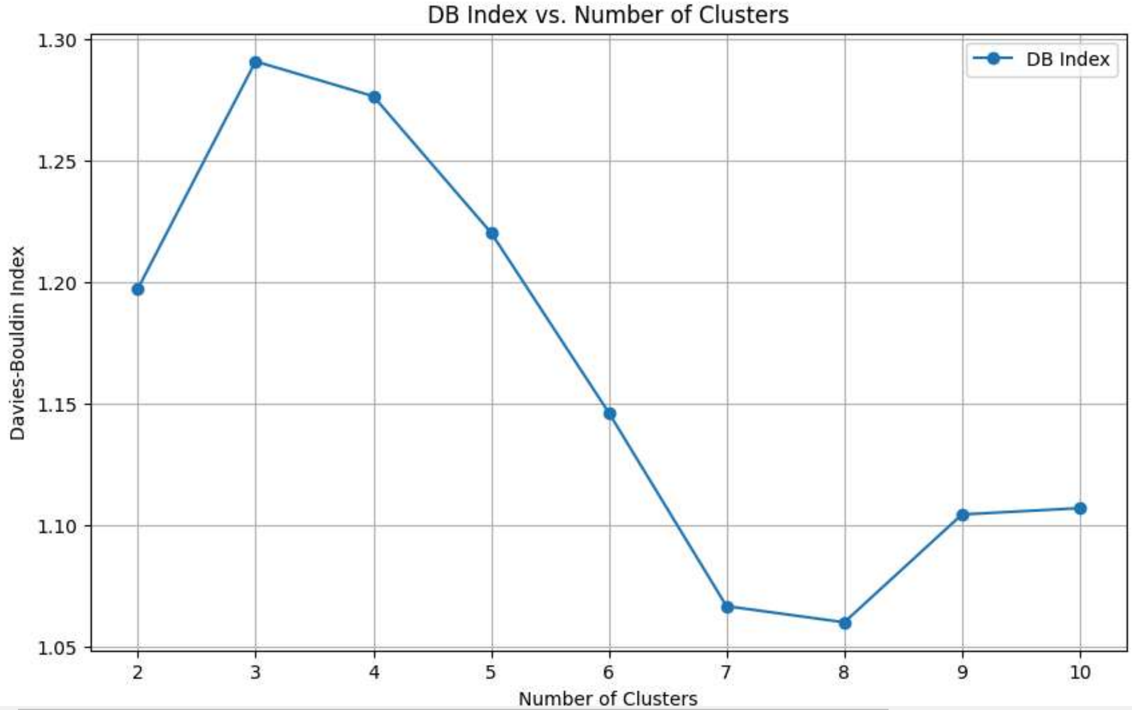
```
# Step 4: K-Means Clustering
# Try clustering with 2 to 10 clusters and calculate DB Index for each
db_scores = []
silhouette_scores = []
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(scaled_features)
    db_score = davies_bouldin_score(scaled_features, cluster_labels)
    silhouette_avg = silhouette_score(scaled_features, cluster_labels)
    db_scores.append(db_score)
    silhouette_scores.append(silhouette_avg)

# Optimal number of clusters based on DB Index
optimal_clusters = np.argmin(db_scores) + 2
print(f"Optimal number of clusters based on DB Index: {optimal_clusters}")
```

```
Optimal number of clusters based on DB Index: 8
```

```
# Final K-Means model
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
merged_data['Cluster'] = kmeans.fit_predict(scaled_features)
```

```
# Step 5: Visualizations
# Plot DB Index for different numbers of clusters
plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), db_scores, marker='o', label='DB Index')
plt.xlabel('Number of Clusters')
plt.ylabel('Davies-Bouldin Index')
plt.title('DB Index vs. Number of Clusters')
plt.legend()
```

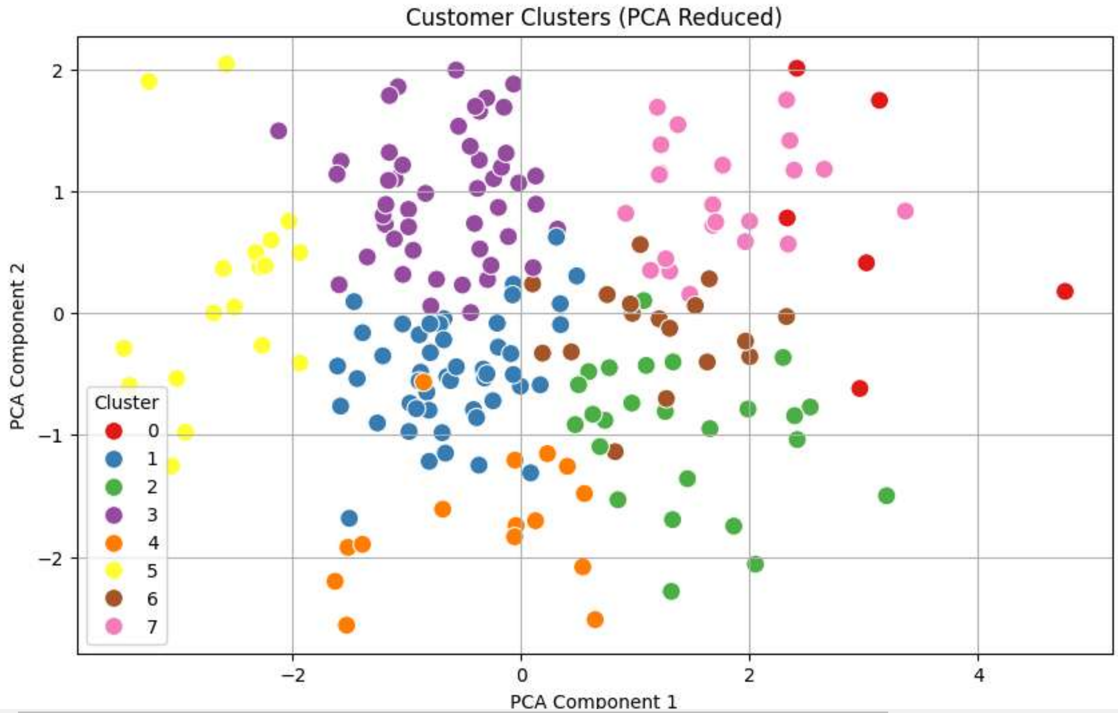
```
plt.grid(True)
plt.show()
```




```
# Plot Silhouette Score for different numbers of clusters
plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), silhouette_scores, marker='o', label='Silhouette Score')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score vs. Number of Clusters')
plt.legend()
plt.grid(True)
plt.show()
```


```
# Visualizing clusters using PCA
```

```
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(scaled_features)
merged_data['PCA1'] = reduced_features[:, 0]
merged_data['PCA2'] = reduced_features[:, 1]

plt.figure(figsize=(10, 6))
sns.scatterplot(data=merged_data, x='PCA1', y='PCA2', hue='Cluster', palette='Set1', s=100)
plt.title('Customer Clusters (PCA Reduced)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```




```
# Step 6: Report Clustering Metrics
final_db_score = davies_bouldin_score(scaled_features, merged_data['Cluster'])
print(f"Final Davies-Bouldin Index for {optimal_clusters} clusters: {final_db_score:.4f}")

# Save final clustered data to CSV
merged_data.to_csv("Customer_Segmentation_Clusters.csv", index=False)
```

```
Final Davies-Bouldin Index for 8 clusters: 1.0601
```