```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import seaborn as sns

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

# Merge customer and transaction data
merged_df = pd.merge(customers_df, transactions_df, on='CustomerID', how='left')

# Feature Engineering
merged_df['TotalPurchases'] = merged_df.groupby('CustomerID')['TransactionID'].transform('count')
merged_df['TotalRevenue'] = merged_df.groupby('CustomerID')['TotalPrice'].transform('sum')
merged_df['AveragePurchaseValue'] = merged_df['TotalRevenue'] / merged_df['TotalPurchases']

# Select relevant features for clustering
features = ['TotalPurchases', 'TotalRevenue', 'AveragePurchaseValue']
X = merged_df.groupby('CustomerID')[features].mean().reset_index()

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X[features])
```

```
# Determine optimal number of clusters using the Elbow method
inertia = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 6))
plt.plot(range(2, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.show()

# Based on the Elbow method, choose the optimal number of clusters (e.g., 4)
n_clusters = 4

# Perform K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(X_scaled)

# Add cluster labels to the DataFrame
X['Cluster'] = kmeans.labels_

# Calculate DB Index
db_index = davies_bouldin_score(X_scaled, kmeans.labels_)
print(f'Davies-Bouldin Index: {db_index}')
```

```
# Visualize clusters
sns.scatterplot(x='TotalPurchases', y='TotalRevenue', hue='Cluster', data=X)
plt.title('Customer Segments')
plt.xlabel('Total Purchases')
plt.ylabel('Total Revenue')
plt.show()

# Analyze cluster characteristics
for i in range(n_clusters):
    print(f'Cluster {i}:')
    print(X[X['Cluster'] == i][features].describe())
```