# Customer Segmentation / Clustering

1.Combine Customers.csv and Transactions.csv into a single dataset for clustering.
2.Use relevant features for segmentation:

- Customer profile (e.g., region, signup date).
- Transaction behavior (e.g., total spending, quantity purchased, etc.).

3.Normalize the data and use clustering techniques (e.g., K-Means or hierarchical clustering).
4.Evaluate clusters using the DB Index and other metrics.
5.Visualize the clusters.

```
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

#Combine Customers.csv and Transactions.csv for clustering
customer_transactions = transactions.groupby("CustomerID").agg({
    "TotalValue": "sum",  # Total spending
    "Quantity": "sum"     # Total quantity purchased
}).reset_index()

customer_data = customers.merge(customer_transactions, on="CustomerID", how="left").fillna(0)

# Encode categorical variables (Region)
customer_data_encoded = pd.get_dummies(customer_data, columns=["Region"])

# Normalize numerical columns
numerical_features = ["TotalValue", "Quantity"]
scaler = MinMaxScaler()
customer_data_encoded[numerical_features] =
scaler.fit_transform(customer_data_encoded[numerical_features])

# Prepare features for clustering
features = customer_data_encoded.drop(columns=["CustomerID", "CustomerName", "SignupDate"])

# Apply K-Means with different cluster counts (2 to 10) and calculate DB Index
db_index_scores = {}
kmeans_models = {}
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(features)
    db_index = davies_bouldin_score(features, labels)
    db_index_scores[k] = db_index
    kmeans_models[k] = kmeans

# Find the best cluster count based on the lowest DB Index
optimal_clusters = min(db_index_scores, key=db_index_scores.get)
```

```
best_kmeans = kmeans_models[optimal_clusters]

# Add cluster labels to customer data
customer_data["Cluster"] = best_kmeans.labels_

# Visualize the DB Index scores
plt.figure(figsize=(10, 6))
plt.plot(list(db_index_scores.keys()), list(db_index_scores.values()), marker='o')
plt.title("DB Index Scores for Different Cluster Counts")
plt.xlabel("Number of Clusters")
plt.ylabel("DB Index")
plt.xticks(range(2, 11))
plt.grid()
plt.show()

optimal_clusters, db_index_scores[optimal_clusters]
```