**Task 3: Customer Segmentation / Clustering**

Perform **customer segmentation** using clustering techniques. Use both **profile information** (from Customers.csv) and **transaction information** (from Transactions.csv).

Output

import pandas as pd

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

customers = pd.read\_csv('/content/drive/MyDrive/Dataset/Customers.csv')

products = pd.read\_csv('/content/drive/MyDrive/Dataset/Products.csv')

transactions = pd.read\_csv('/content/drive/MyDrive/Dataset/Transactions.csv')

customer\_transaction\_data = transactions.groupby("CustomerID").agg(

    total\_spending=("TotalValue", "sum"),

    transaction\_count=("TransactionID", "count"),

).reset\_index()

merged\_data = customers.merge(customer\_transaction\_data, on="CustomerID", how="inner")

features = transactions.groupby('CustomerID').agg({'TotalValue': 'sum', 'ProductID': 'nunique'})

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features[['TotalValue', 'ProductID']])

kmeans = KMeans(n\_clusters=5, random\_state=42)

clusters = kmeans.fit\_predict(scaled\_features)

features['Cluster'] = clusters

db\_index = davies\_bouldin\_score(scaled\_features, clusters)

print(f"Davies-Bouldin Index: {db\_index:.4f}")

**Davies-Bouldin Index: 0.8837**

pca = PCA(n\_components=2)

reduced\_data = pca.fit\_transform(scaled\_features)

plt.figure(figsize=(10, 6))

plt.scatter(reduced\_data[:, 0], reduced\_data[:, 1], c=clusters, cmap='viridis', s=50)

plt.title('Customer Clusters Visualization (PCA Reduced)')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.colorbar(label='Cluster')

plt.show()

db\_index = davies\_bouldin\_score(features, clusters)

pca = PCA(n\_components=2)

reduced\_data = pca.fit\_transform(features)

plt.scatter(reduced\_data[:, 0], reduced\_data[:, 1], c=clusters)

plt.show()

cluster\_range = range(2, 11)

db\_index\_values = []

silhouette\_scores = []

db\_scores = []

**evaluate clusters**

for n\_clusters in cluster\_range:

    kmeans = KMeans(n\_clusters=n\_clusters, random\_state=42)

    cluster\_labels = kmeans.fit\_predict(scaled\_features)

db\_index = davies\_bouldin\_score(scaled\_features, cluster\_labels)

silhouette\_avg = silhouette\_score(scaled\_features, cluster\_labels)

db\_index\_values.append(db\_index)

silhouette\_scores.append(silhouette\_avg)

optimal\_clusters = cluster\_range[db\_scores.index(min(db\_scores))]

print(f"Optimal Number of Clusters: {optimal\_clusters}")

kmeans\_optimal = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

final\_clusters = kmeans\_optimal.fit\_predict(scaled\_features)

features['Cluster'] = final\_clusters

final\_db\_index = davies\_bouldin\_score(scaled\_features, final\_clusters)

final\_silhouette\_score = silhouette\_score(scaled\_features, final\_clusters)

print(f"Davies-Bouldin Index for Optimal Clusters: {final\_db\_index:.4f}")

print(f"Silhouette Score for Optimal Clusters: {final\_silhouette\_score:.4f}")

**Optimal Number of Clusters: 2**

**Davies-Bouldin Index for Optimal Clusters: 0.7420**

**Silhouette Score for Optimal Clusters: 0.4829**features.to\_csv("Clustered\_Customers.csv", index=False)

**Plot Davies-Bouldin Index and Silhouette Score**

plt.figure(figsize=(12, 6))

plt.plot(cluster\_range, db\_scores, marker='o', label='DB Index')

plt.plot(cluster\_range, silhouette\_scores, marker='s', label='Silhouette Score')

plt.xlabel("Number of Clusters")

plt.ylabel("Score")

plt.title("Cluster Evaluation Metrics")

plt.legend()

plt.show()

**Visualize clusters with PCA**

pca = PCA(n\_components=2)

reduced\_data = pca.fit\_transform(scaled\_features)

plt.figure(figsize=(10, 6))

plt.scatter(reduced\_data[:, 0], reduced\_data[:, 1], c=final\_clusters, cmap='viridis', s=50)

plt.title('Customer Clusters Visualization (PCA Reduced)')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.colorbar(label='Cluster')

plt.show()