Task 3: Customer Segmentation / Clustering

This task focuses on segmenting customers using clustering techniques and evaluating the clustering results with relevant metrics, including the Davies-Bouldin (DB) Index. Here's a complete breakdown:

Step 1: Import Libraries

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from sklearn.cluster import KMeans

from sklearn.metrics import davies_bouldin_score

import matplotlib.pyplot as plt

import seaborn as sns
```

Step 2: Load and Merge the Datasets

Load the datasets and merge them to create a consolidated view for customer segmentation:

```
# Load datasets

customers = pd.read_csv("Customers.csv")

transactions = pd.read_csv("Transactions.csv")

# Merge Customers and Transactions

data = transactions.merge(customers, on="CustomerID")

# Display data

print(data.head())
```

Step 3: Data Preprocessing

1. Encode Categorical Data

Encode Region and other categorical columns using LabelEncoder:

le = LabelEncoder()

data["Region"] = le.fit_transform(data["Region"])

2. Aggregate Transaction and Customer Data

Aggregate the data at the customer level to create customer profiles:

```
customer_profiles = data.groupby("CustomerID").agg({
    "Region": "first",                  # Customer region
    "TotalValue": "sum",                # Total spending
    "Quantity": "sum",                  # Total quantity purchased
    "TransactionID": "count"            # Number of transactions
}).reset_index()


# Rename columns for clarity
customer_profiles.rename(columns={"TransactionID": "NumTransactions"}, inplace=True)
```

3. Normalize Numerical Features

Scale the numerical data for clustering:

```
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(customer_profiles[["TotalValue", "Quantity", "NumTransactions"]])
```

Step 4: Perform Clustering

1. Choose a Clustering Algorithm

We'll use **KMeans** for this task. You can experiment with other clustering algorithms as needed.

2. Determine the Optimal Number of Clusters

Use the **Elbow Method** to decide on the number of clusters

```
wcss = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)
# Plot the Elbow Curve
plt.plot(range(2, 11), wcss, marker="o")
plt.title("Elbow Method for Optimal Clusters")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS (Within-Cluster Sum of Squares)")
plt.show()
```

3. Fit KMeans with Optimal Clusters

Assume you choose 4 clusters based on the elbow curve:

```
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
customer_profiles["Cluster"] = kmeans.fit_predict(scaled_features)
```

Step 5: Evaluate Clustering with DB Index

Calculate the **Davies-Bouldin Index** to assess the quality of clustering:

```
db_index = davies_bouldin_score(scaled_features, customer_profiles["Cluster"])
print(f"Davies-Bouldin Index: {db_index}")
```

Step 6: Visualize Clusters

1. Visualize with Scatter Plots

Use PCA or select two features to visualize clusters:

```
from sklearn.decomposition import PCA

# Reduce dimensions to 2D
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(scaled_features)

# Add reduced dimensions to the DataFrame
customer_profiles["PCA1"] = reduced_features[:, 0]
customer_profiles["PCA2"] = reduced_features[:, 1]

# Plot clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x="PCA1", y="PCA2", hue="Cluster", data=customer_profiles,
palette="viridis", s=100
)
plt.title("Customer Clusters (PCA Visualization)")
plt.show()
```

2. Additional Visualizations

Visualize feature distributions for each cluster:

```
for feature in ["TotalValue", "Quantity", "NumTransactions"]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x="Cluster", y=feature, data=customer_profiles)
    plt.title(f"{feature} Distribution by Cluster")
```

```
plt.show()
```

Step 7: Deliverables

1. Report

Prepare a report with the following:

- **Number of Clusters Formed**: Specify the number of clusters chosen (e.g., 4).
- **Davies-Bouldin Index**: Mention the DB Index score (lower is better).
- **Cluster Characteristics**: Describe the traits of each cluster based on aggregated features (e.g., high spenders, frequent buyers).
- **Actionable Insights**: Recommend business strategies based on the clustering results.

2. Jupyter Notebook

Include:

1. Data loading and preprocessing.
2. Clustering implementation and evaluation.
3. Visualizations.

Sample Results

Cluster Characteristics

Cluster	Region	Avg. Spending	Avg. Transactions	Description
0	Asia	High	Low	High-value, infrequent
1	Europe	Medium	Medium	Average buyers
2	NA	Low	High	Low-value, frequent

Cluster	Region	Avg. Spending	Avg. Transactions	Description
3	Asia	High	High	High-value, frequent

Insights

1. **High-value customers** are concentrated in Cluster 3, mainly from Asia. Offer loyalty programs to retain them.
2. **Frequent low-value buyers** (Cluster 2) may benefit from bulk-purchase discounts.
3. Clusters with **medium spending** (Cluster 1) can be targeted with upselling strategies.
4. Regions with fewer transactions (e.g., Europe) indicate potential for growth through marketing campaigns.