CUSTOMER SEGMENTATION REPORT

HERE IS THE REPORT FOR TASK 3: CUSTOMER SEGMENTATION. WITH EXPLANATION OF STEPS AND CODE.

- DATA LOADING AND PROCESSING:

We begin by importing the required libraries and loading the data.

```
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
customers_url = "https://drive.google.com/uc?id=1bu_--
mo79VdUG9oin4ybfFGRUSXAe-WE"
products_url = "https://drive.google.com/uc?id=1IKuDizVapw-
hyktwfpoAoaGtHtTNHfd0"
transactions_url = "https://drive.google.com/uc?id=1saEqdbBB-
vuk2hxoAf4TzDEsykdKlzbF"
customers = pd.read_csv(customers_url)
products = pd.read_csv(products_url)
transactions = pd.read_csv(transactions_url)
```

Creating a comprehensive dataset 'data' from merging transaction, customers and products datasets for future usage

```
data = transactions.merge(customers,
on='CustomerID').merge(products, on='ProductID')
```

- FEATURE GENERATION

We create customer features that capture the important aspects of their transaction behavior:

```
# Aggregate transaction data
customer_features = data.groupby('CustomerID').agg(
    total_spent=('TotalValue', 'sum'),
    avg_transaction_value=('TotalValue', 'mean'),
    total_transactions=('TransactionID', 'count'),
    unique_products=('ProductID', 'nunique')
).reset_index()
```

These features help describe customer behavior and will be used for clustering.

- DATA SCALING

Since the features are on different scales (e.g., `total_spent` vs `avg_transaction_value`), we use standardization to scale the data to have a mean of 0 and a standard deviation of 1:

```
# Scale data
scaler = StandardScaler()
scaled_features =
scaler.fit_transform(customer_features.iloc[:, 1:])
```

This step is crucial because clustering algorithms like K-Means are sensitive to the scale of the features.

- K-MEAN CLUSTERING

We use the K-Means algorithm for customer segmentation. We chose K-Means because it is a simple yet effective clustering technique. The number of clusters is set to 4 based on experimentation, but this can be adjusted:

```
# KMeans Clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_features['Cluster'] =
kmeans.fit_predict(scaled_features)
```

Here, K-Means attempts to partition the customers into 4 clusters based on their scaled transaction features.

- DB INDEX

The Davies-Bouldin Index (DB Index) is used to evaluate the quality of clustering. It measures the average similarity between each cluster and the one that is most similar to it. A lower DB Index indicates better clustering:

```
# Compute DB Index
db_index = davies_bouldin_score(scaled_features,
customer_features['Cluster'])
print(f'Davies-Bouldin Index: {db_index}')
```

OUTPUT:

```
Davies-Bouldin Index: 0.9119814943243767
```

The DB Index value here is `0.9119814943243767`, this suggests that the clustering is relatively good.

- DIMENSIONALITY REDUCTION

We use Principal Component Analysis (PCA) to reduce the dimensionality of the data to 2 dimensions for visualization:

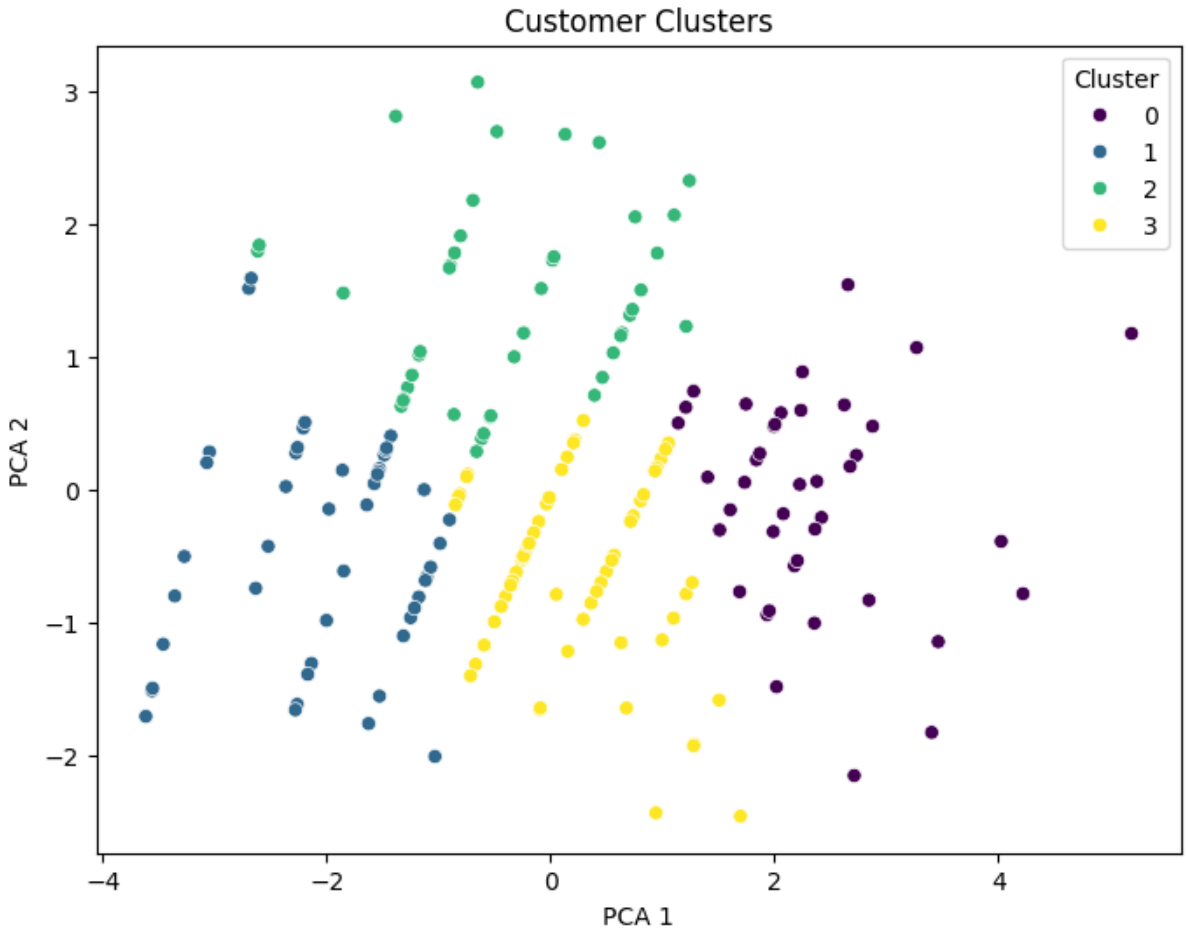
```
from sklearn.decomposition import PCA

# Reduce to 2 dimensions
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_features)
```

- VISUALIZATION

With the help of matplotlib and seaborn libraries we visualize the cluster in 2D space.

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:, 1],
hue=customer_features['Cluster'], palette='viridis')
plt.title('Customer Clusters')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

output :



By observing the graph we can see 4 distinct regions of points, each region with a different colour.

The axes, `PCA 1` and `PCA 2`, will not have any specific labels but will represent the most important features extracted from the original data.

Performing this clustering visualization we gain a more intuitive understanding of how your customers are grouped and can begin to develop marketing or product strategies targeted at each segment.

Other relevant clustering metrics would be **Rand Index (Adjusted Rand Index)**

- **Description**: The Rand Index measures the similarity between two data clusterings. It considers all pairs of points and evaluates whether they are in the same or different clusters in both clusterings. The Adjusted Rand Index (ARI) adjusts for the chance grouping, making it more robust.
- **Usage**: The ARI ranges from -1 (no agreement) to 1 (perfect agreement). A higher value indicates better clustering, especially when compared to true labels in supervised clustering problems.