Task 3: Customer Segmentation / Clustering Perform customer segmentation using clustering techniques. Use both profile information (from Customers.csv) and transaction information (from Transactions.csv).

● You have the flexibility to choose any clustering algorithm and any number of clusters in between(2 and 10)

● Calculate clustering metrics, including the DB Index(Evaluation will be done on this). ● Visualise your clusters using relevant plots.

Deliverables: ● A report on your clustering results, including:

The number of clusters formed.

DB Index value. ○ Other relevant clustering metrics.

● A Jupyter Notebook/Python script containing your clustering code.

```
import pandas as pd

# Load the datasets
customers_path = r'C:\Users\Dell\Downloads\Customers.csv'
transactions_path = r'C:\Users\Dell\Downloads\Transactions.csv'

customers_df = pd.read_csv(customers_path)
transactions_df = pd.read_csv(transactions_path)

# Display the first few rows of each dataset to understand their structure
customers_df.head(), transactions_df.head()

# Check for missing values and data types in both datasets
customers_info = customers_df.info(), customers_df.isnull().sum()
transactions_info = transactions_df.info(), transactions_df.isnull().sum()

customers_info, transactions_info

from datetime import datetime

# Convert date columns to datetime format
customers_df['SignupDate'] = pd.to_datetime(customers_df['SignupDate'], format='%d-%m-%Y',
errors='coerce')
transactions_df['TransactionDate'] = pd.to_datetime(transactions_df['TransactionDate'],
errors='coerce')

# Check if any conversion failed (NaT indicates failed conversion)
signup_date_issues = customers_df['SignupDate'].isna().sum()
transaction_date_issues = transactions_df['TransactionDate'].isna().sum()

signup_date_issues, transaction_date_issues
```

```
# Aggregate transactional data at the customer level
transaction_metrics = transactions_df.groupby('CustomerID').agg(
    TotalTransactions=('TransactionID', 'count'),
    TotalSpending=('TotalValue', 'sum'),
    AvgSpendingPerTransaction=('TotalValue', 'mean'),
    LastTransactionDate=('TransactionDate', 'max')
).reset_index()

# Calculate Recency (days since last transaction from the most recent date in the dataset)
most_recent_date = transactions_df['TransactionDate'].max()
transaction_metrics['Recency'] = (most_recent_date -
transaction_metrics['LastTransactionDate']).dt.days

# Merge aggregated transaction metrics with customer profile data
customer_data = pd.merge(customers_df, transaction_metrics, on='CustomerID', how='left')

# Replace NaN values in transaction-related fields with 0 (indicating no transactions for certain
customers)
customer_data.fillna({
    'TotalTransactions': 0,
    'TotalSpending': 0,
    'AvgSpendingPerTransaction': 0,
    'Recency': (most_recent_date - customers_df['SignupDate']).dt.days  # Recency based on signup
date for non-transactional customers
}, inplace=True)

customer_data.head()
```