**Source Code**

**Step 1: Load the Dataset**

import pandas as pd

**#** **Load the dataset**

file\_path = '/mnt/data/bank\_transactions.csv' **# Path to the uploaded dataset**

df = pd.read\_csv(file\_path)

**# Display the first few rows of the dataset**

print(df.head())

**Step 2: Explore the Dataset**

**# Summary statistics**

print(df.describe())

**# Check for missing values**

print(df.isnull().sum())

**Step 3: Data Cleaning**

**# Drop rows with missing values**

df.dropna(inplace=True)

**# Remove duplicates if any**

df.drop\_duplicates(inplace=True)

**# Display the cleaned data**

print(df.head())

**Step 4: Feature Engineering**

**# Convert TransactionDate to datetime format if necessary**

df['TransactionDate'] = pd.to\_datetime(df['TransactionDate'])

**# Calculate total amount spent by each customer**

customer\_df = df.groupby('CustomerID').agg({

'TransactionAmount (INR)': 'sum', **# Total amount spent**

'TransactionDate': 'count' **# Number of transactions**

}).rename(columns={'TransactionDate': 'TransactionCount'})

# Display the first few rows of the aggregated data

print(customer\_df.head())

**Step 5: Data Normalization**

from sklearn.preprocessing import StandardScaler

**# Normalize the data**

scaler = StandardScaler()

customer\_df\_normalized = scaler.fit\_transform(customer\_df)

**# Convert back to a DataFrame for ease of use**

customer\_df\_normalized = pd.DataFrame(customer\_df\_normalized,

columns=customer\_df.columns, index=customer\_df.index)

**Step 6: Apply Clustering Algorithm**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

**# Determine the optimal number of clusters using the Elbow method**

sse = []

for k in range(1, 11):

kmeans = KMeans(n\_clusters=k, random\_state=42, n\_init=10)

kmeans.fit(customer\_df\_normalized)

sse.append(kmeans.inertia\_)

**# Plot the SSE for each value of k**

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

plt.plot(range(1, 11), sse, marker='o')

plt.xlabel('Number of clusters')

plt.ylabel('SSE')

plt.title('Elbow Method')

plt.show()

**# Choose the optimal number of clusters (e.g., 3) and fit the KMeans model**

optimal\_k = 3

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42, n\_init=10)

customer\_df['Cluster'] = kmeans.fit\_predict(customer\_df\_normalized)

**Step 7: Analyze and Visualize the Segments**

import seaborn as sns

**# Analyze the characteristics of each cluster**

cluster\_summary = customer\_df.groupby('Cluster').agg({

'TransactionCount': 'mean',

'TransactionAmount (INR)': 'mean'

}).reset\_index()

print(cluster\_summary)

**# Visualize the segments**

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

sns.scatterplot(x='TransactionCount', y='TransactionAmount (INR)', hue='Cluster',

data=customer\_df, palette='viridis')

plt.title('Customer Segments')

plt.xlabel('Number of Transactions')

plt.ylabel('Total Amount Spent')

plt.show()

**Step 10: Save the Results**

customer\_df.to\_csv('customer\_segments.csv', index=False)

**Step 11: Interpret the Segments**

**# Profile each segment**

for cluster in customer\_df['Cluster'].unique():

print(f"Cluster {cluster} Profile:")

print(customer\_df[customer\_df['Cluster'] == cluster].describe())

print("\n")