PROBLEM STATEMENT :

Customer Segmentation in E-commerce

What This Report Explains:

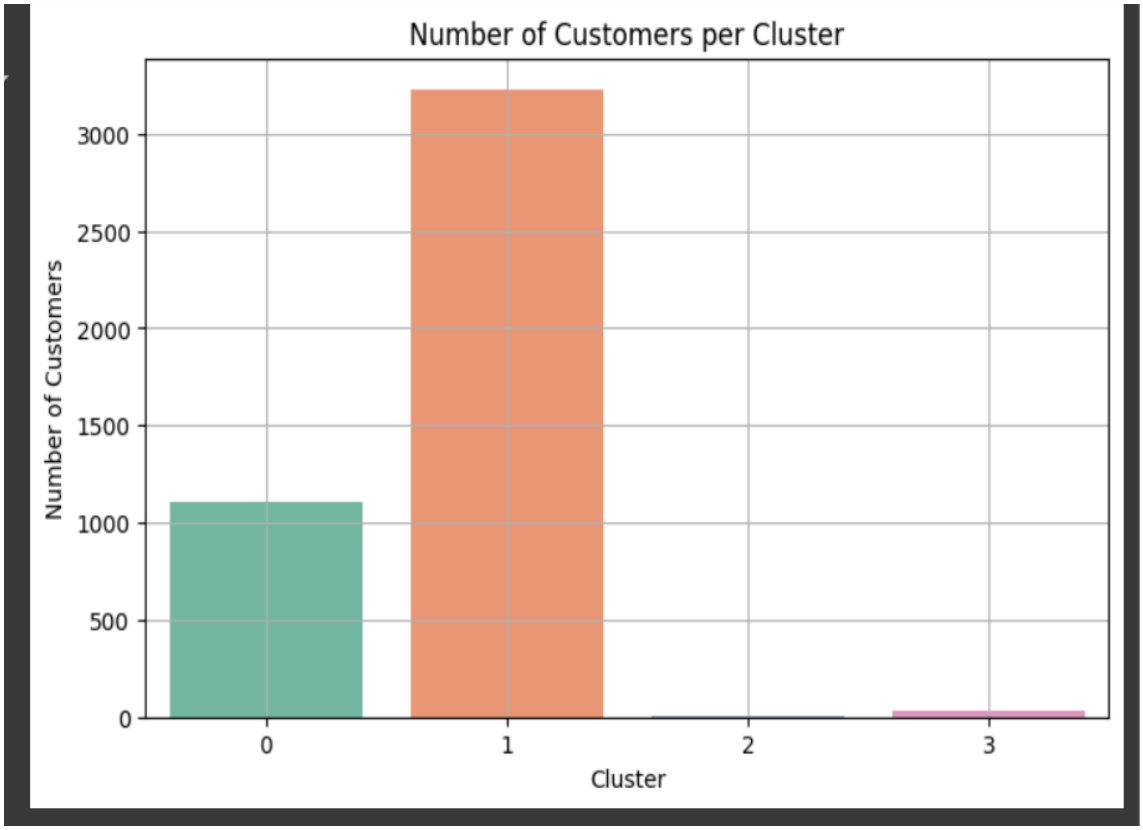
Identify customer clusters based on purchasing habits and browsing behavior.

Name: HARSH JONWAL

Roll No.: 2024011004000092(19)

Introduction :

Customer segmentation is a vital technique in e-commerce for understanding customer preferences and tailoring marketing strategies. By analyzing purchasing habits and browsing behavior, businesses can categorize customers into meaningful clusters, leading to more personalized services and improved satisfaction.



Methodology :

1.Data Collection: Gather data from e-commerce platforms, including purchase history and browsing patterns.

2. Data Preprocessing: Clean the data by handling missing values, normalizing numerical attributes, and encoding categorical variables.

3. Feature Selection: Select features such as frequency of purchase, average order value, and time spent on site.

4. Clustering Algorithm: Apply K-Means clustering to identify distinct customer segments.

5. Evaluation: Analyze the resulting clusters for meaningful business insights.

CODE :

```
# Step 1: Upload file from desktop
from google.colab import files
uploaded = files.upload()
# Step 2: Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
import numpy as np

# Step 3: Load and clean data
df = pd.read_csv("9. Customer Segmentation in E-commerce.csv")
df = df[df['CustomerID'].notnull()]
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']

# Step 4: Prepare RFM manually
customers = df['CustomerID'].unique()
recency_list = []
frequency_list = []
```

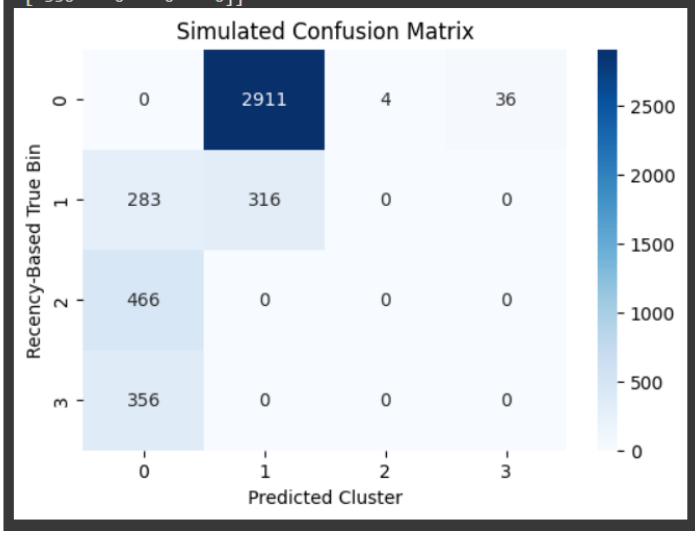
```
monetary_list = []

last_date = df['InvoiceDate'].max()

for customer in customers:
    cust_data = df[df['CustomerID'] == customer]
    recency = (last_date - cust_data['InvoiceDate'].max()).days
    frequency = len(cust_data['InvoiceNo'].unique())
    monetary = cust_data['TotalPrice'].sum()

    recency_list.append(recency)
    frequency_list.append(frequency)
    monetary_list.append(monetary)

rfm = pd.DataFrame({
    'CustomerID': customers,
    'Recency': recency_list,
    'Frequency': frequency_list,
    'Monetary': monetary_list
})

# Step 5: Normalize and cluster
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm[['Recency', 'Frequency', 'Monetary']])
```

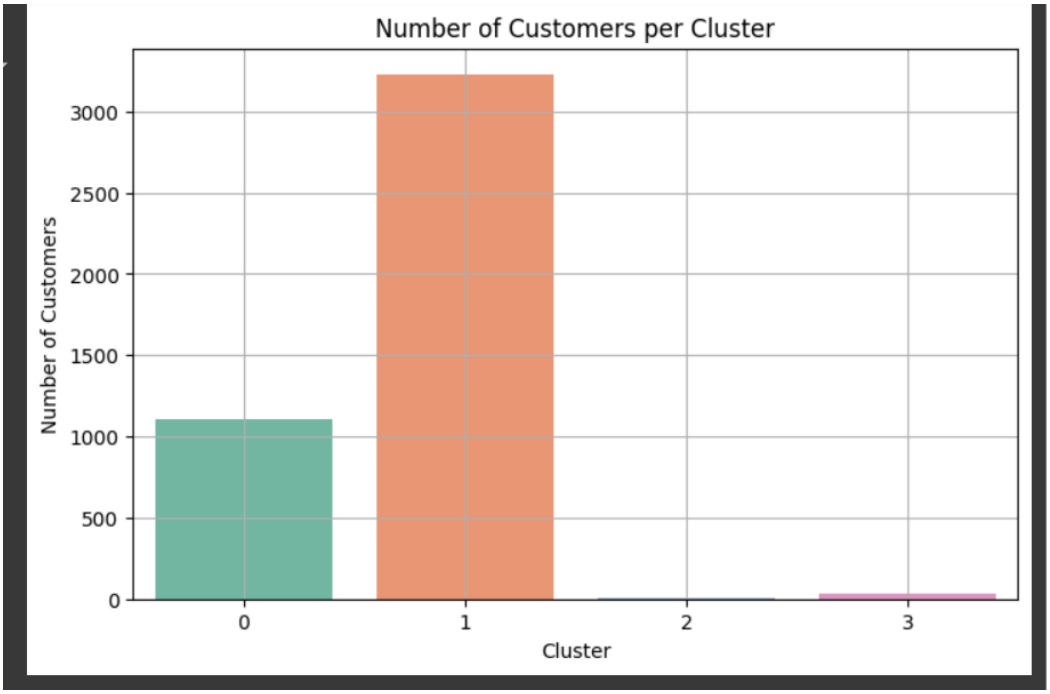
```
kmeans = KMeans(n_clusters=4, random_state=0)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)

# Step 6: Bar chart of cluster sizes
plt.figure(figsize=(8, 5))
cluster_counts = rfm['Cluster'].value_counts().sort_index()
sns.barplot(x=cluster_counts.index, y=cluster_counts.values, palette='Set2')
plt.title("Number of Customers per Cluster")
plt.xlabel("Cluster")
plt.ylabel("Number of Customers")
plt.grid(True)
plt.show()

# Step 7: Designer HashMap Matrix (manual)
print("\n Designer HashMap Matrix:")
for i in sorted(rfm['Cluster'].unique()):
    cl = rfm[rfm['Cluster'] == i]
    print(f"\n Cluster {i}")
    print(f"   - Avg Recency  : {round(cl['Recency'].mean(), 2)} days")
    print(f"   - Avg Frequency: {round(cl['Frequency'].mean(), 2)} orders")
    print(f"   - Avg Monetary : ${round(cl['Monetary'].mean(), 2)}")
    print(f"   - Total Customers: {len(cl)}")

# Step 8: Confusion Matrix (simulated, since we don't have true labels)
```

```
# We'll simulate "true labels" for educational purposes by cutting Recency into 4 bins

true_labels = pd.cut(rfm['Recency'], bins=4, labels=[0, 1, 2, 3]).astype(int)

predicted = rfm['Cluster']


cm = confusion_matrix(true_labels, predicted)


# Print confusion matrix

print("\n📉 Simulated Confusion Matrix (based on Recency bins vs predicted cluster):")

print(cm)


# Optional: visualize confusion matrix

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=[0,1,2,3], yticklabels=[0,1,2,3])

plt.xlabel('Predicted Cluster')

plt.ylabel('Recency-Based True Bin')

plt.title('Simulated Confusion Matrix')

plt.show()
```

Outputs/Result :

```
Simulated Confusion Matrix (based on Recency bins vs predicted cluster):
[[   0 2911    4   36]
 [ 283  316    0    0]
 [ 466    0    0    0]
 [ 356    0    0    0]]
```





Refrences:

- Google
- Chat Gpt
- Google Colab
- Python Libraries

CONCLUSION:

In conclusion, customer segmentation based on purchasing habits and browsing behavior enables e-commerce businesses to better understand their customers and cater to their specific needs. By applying machine learning techniques like K-Means clustering, we can uncover patterns and group similar customers, allowing for more targeted marketing, improved customer retention, and enhanced overall business strategy. This approach not only maximizes profitability but also boosts customer satisfaction and loyalty.