Importing Libraries

In [18]:
```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
from datetime import datetime
```

Datasets that Merged

In [19]:
```
# Load the dataset
products = pd.read_csv("Products.csv")
customer = pd.read_csv("Customers.csv")
transactions = pd.read_csv("Transactions.csv")

print("Customer Table")
customer.head(5)
```

```
Customer Table
```

Out[19]:

	CustomerID	CustomerName	Region	SignupDate
0	C0001	Lawrence Carroll	South America	10-07-2022
1	C0002	Elizabeth Lutz	Asia	13-02-2022
2	C0003	Michael Rivera	South America	07-03-2024
3	C0004	Kathleen Rodriguez	South America	09-10-2022
4	C0005	Laura Weber	Asia	15-08-2022

In [20]:
```
print("Products Table")
products.head(5)
```

```
Products Table
```

Out[20]:

	ProductID	ProductName	Category	Price
0	P001	ActiveWear Biography	Books	169.30
1	P002	ActiveWear Smartwatch	Electronics	346.30
2	P003	ComfortLiving Biography	Books	44.12
3	P004	BookWorld Rug	Home Decor	95.69
4	P005	TechPro T-Shirt	Clothing	429.31

In [21]:
```
print("Transactions Table")
transactions.head(5)
```

```
Transactions Table
```

Out[21]:

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	P
0	T00001	C0199	P067	25-08-2024 12:38	1	300.68	30
1	T00112	C0146	P067	27-05-2024 22:23	1	300.68	30
2	T00166	C0127	P067	25-04-2024 07:38	1	300.68	30
3	T00272	C0087	P067	26-03-2024 22:55	2	601.36	30
4	T00363	C0070	P067	21-03-2024 15:10	3	902.04	30

Looklike Model

In [22]:

```
# 1. Data Preprocessing

# Ensure 'SignupDate' is converted to datetime and calculate tenure (days since
df['SignupDate'] = pd.to_datetime(df['SignupDate'], format='%d-%m-%Y')
df['Tenure'] = (datetime.now() - df['SignupDate']).dt.days

# Ensure 'TransactionDate' is in datetime format
df['TransactionDate'] = pd.to_datetime(df['TransactionDate'], errors='coerce')

# Create customer-level features
agg_trans = df.groupby('CustomerID').agg(
    total_spend=('TotalValue', 'sum'),
    transaction_count=('TransactionID', 'nunique'),
    recency=('TransactionDate', lambda x: (datetime.now() - x.max()).days if pd.
    avg_price=('Price_x', 'mean')  # Use 'Price_x' instead of 'Price_y'
).reset_index()


# 2. Aggregate Product Preferences by Category for each Customer
agg_product = df.groupby(['CustomerID', 'Category']).agg(
    product_frequency=('ProductID', 'count')
).unstack(fill_value=0)

# Flatten multi-level columns (e.g., for Category)
agg_product.columns = [col[1] for col in agg_product.columns]
agg_product.reset_index(inplace=True)

# Merge customer profile and product preferences
customer_profile = pd.merge(agg_trans, agg_product, on='CustomerID', how='left')

# 3. Normalize numeric features (Total Spend, Transaction Count, Recency, Averag
scaler = StandardScaler()
num_features = ['total_spend', 'transaction_count', 'recency', 'avg_price']  # R
customer_profile[num_features] = scaler.fit_transform(customer_profile[num_featu

# 4. Calculate Similarity Scores using Cosine Similarity
# Drop CustomerID for similarity calculation
customer_features = customer_profile.drop(columns='CustomerID')

# Cosine Similarity matrix
cos_sim = cosine_similarity(customer_features)

# 5. Find Top 3 Lookalikes for each customer (C0001 to C0020)
```

```
def find_lookalikes(customer_id, top_n=3):
    # Get the index of the customer in the dataframe
    customer_index = customer_profile[customer_profile['CustomerID'] == customer

    # Get the similarity scores for this customer
    similarity_scores = cos_sim[customer_index]

    # Get the indices of the top n similar customers, excluding the customer its
    similar_customer_indices = similarity_scores.argsort()[-(top_n + 1):-1]

    # Get the corresponding customer details
    similar_customers = customer_profile.iloc[similar_customer_indices]

    return similar_customers[['CustomerID']]

# 6. Create a Lookalike Map and Save Results
lookalike_map = []

# For each customer (C0001 to C0020), find the top 3 lookalikes
for i in range(1, 21):
    customer_id = f'C{i:04d}'  # Format customer ID as C0001, C0002, ...
    lookalikes = find_lookalikes(customer_id)

    # For each lookalike, get the similarity score and add to the map
    for idx, row in lookalikes.iterrows():
        similarity_score = cos_sim[customer_profile[customer_profile['CustomerID
        lookalike_map.append([customer_id, row['CustomerID'], similarity_score])

# Save the lookalike recommendations to a CSV
lookalike_df = pd.DataFrame(lookalike_map, columns=['CustomerID', 'LookalikeCust
lookalike_df = lookalike_df.sort_values(by='Score', ascending=False).reset_index
lookalike_df.to_csv('Jofin_James_Lookalike.csv', index=False)

#print("Lookalike model created and saved to 'Jofin_James_Lookalike.csv'")
top20 =lookalike_df.head(20)
print(top20)
```

```
   CustomerID LookalikeCustomerID     Score
0       C0014                C0097  0.983845
1       C0017                C0075  0.980759
2       C0012                C0065  0.980177
3       C0011                C0126  0.973741
4       C0008                C0162  0.972486
5       C0014                C0110  0.972008
6       C0017                C0194  0.967714
7       C0019                C0191  0.963251
8       C0020                C0130  0.961294
9       C0006                C0135  0.959740
10      C0004                C0017  0.957609
11      C0017                C0004  0.957609
12      C0004                C0113  0.956201
13      C0002                C0134  0.955623
14      C0010                C0077  0.950563
15      C0014                C0144  0.948514
16      C0008                C0113  0.946371
17      C0013                C0183  0.945491
18      C0001                C0069  0.944195
19      C0012                C0104  0.943512
```

```
C:\Users\jofin\AppData\Local\Temp\ipykernel_14984\3792595445.py:8: UserWarning: P
arsing dates in %d-%m-%Y %H:%M format when dayfirst=False (the default) was speci
fied. Pass `dayfirst=True` or specify a format to silence this warning.
  df['TransactionDate'] = pd.to_datetime(df['TransactionDate'], errors='coerce')
```