In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Load and prepare the data
def prepare_data():
    customers_df = pd.read_csv('Customers.csv')
    products_df = pd.read_csv('Products.csv')
    transactions_df = pd.read_csv('Transactions.csv')

    # Convert dates to datetime
    customers_df['SignupDate'] = pd.to_datetime(customers_df['SignupDate'])
    transactions_df['TransactionDate'] = pd.to_datetime(transactions_df['TransactionDate'])

    return customers_df, products_df, transactions_df
```

In [3]:
```
#Create customer features
def create_customer_features(customers_df, transactions_df, products_df):
    # Customer transaction features
    customer_transactions = transactions_df.groupby('CustomerID').agg({
        'TransactionID': 'count',  # Number of transactions
        'Quantity': 'sum',         # Total items purchased
        'TotalValue': ['sum', 'mean']  # Total spend and average transaction value
    }).round(2)

    customer_transactions.columns = ['transaction_count', 'total_items', 'total_spend', 'avg_transaction_value']

    # Product category preferences
    transaction_products = pd.merge(transactions_df, products_df[['ProductID', 'Category']], on='ProductID')
    category_preferences = pd.crosstab(transaction_products['CustomerID'], transaction_products['Category'])
    category_preferences = category_preferences.div(category_preferences.sum(axis=1), axis=0)

    # Region encoding
    region_dummies = pd.get_dummies(customers_df['Region'], prefix='region')

    # Combine all features
    features_df = pd.concat([
        customer_transactions,
        category_preferences,
        region_dummies
    ], axis=1).fillna(0)

    return features_df
```

In [4]:
```
#Calculate similarity scores
def calculate_similarity(features_df):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features_df)
    similarity_matrix = cosine_similarity(scaled_features)
    return similarity_matrix
```

In [5]:
```
#Get top lookalikes
def get_top_lookalikes(customer_id, similarity_matrix, features_df, n=3):
    customer_idx = features_df.index.get_loc(customer_id)
    customer_similarities = similarity_matrix[customer_idx]

    # Get indices of top similar customers (excluding self)
    similar_indices = customer_similarities.argsort()[::-1][1:n+1]
    similar_scores = customer_similarities[similar_indices]

    similar_customers = features_df.index[similar_indices]
    return list(zip(similar_customers, similar_scores.round(4)))
```

In [6]:
```
# Main execution
customers_df, products_df, transactions_df = prepare_data()
features_df = create_customer_features(customers_df, transactions_df, products_df)
similarity_matrix = calculate_similarity(features_df)

# Generate lookalikes for first 20 customers
lookalike_results = {}
for cust_id in customers_df['CustomerID'][:20]:
    if cust_id in features_df.index:
        lookalikes = get_top_lookalikes(cust_id, similarity_matrix, features_df)
        lookalike_results[cust_id] = lookalikes
```

In [7]:
```
# Create Lookalike.csv
output_data = []
for cust_id, lookalikes in lookalike_results.items():
    lookalike_str = "|".join([f"{cust},{score}" for cust, score in lookalikes])
    output_data.append({
        'CustomerID': cust_id,
        'Lookalikes': lookalike_str
    })

output_df = pd.DataFrame(output_data)
output_df.to_csv('Lookalike.csv', index=False)
```

In [8]:
```
# Display results
print("Lookalike Results for First 20 Customers:")
print("-" * 50)
for cust_id, lookalikes in lookalike_results.items():
    print(f"\nCustomer {cust_id}:")
    for similar_cust, score in lookalikes:
        print(f"  {similar_cust}: {score}")
```

```
Lookalike Results for First 20 Customers:
--------------------------------------------------

Customer C0001:
  C0035: 0.9749
  C0146: 0.9637
  C0127: 0.9516

Customer C0002:
  C0133: 0.966
  C0144: 0.9533
  C0134: 0.945

Customer C0003:
  C0166: 0.9865
  C0031: 0.9589
  C0195: 0.9408

Customer C0004:
  C0017: 0.9743
  C0113: 0.9732
  C0075: 0.9609

Customer C0005:
  C0197: 0.9963
  C0007: 0.9877
  C0069: 0.9469

Customer C0006:
  C0135: 0.9712
  C0187: 0.949
  C0185: 0.9289

Customer C0007:
  C0005: 0.9877
  C0197: 0.9822
  C0069: 0.9328

Customer C0008:
  C0162: 0.9624
  C0113: 0.9419
  C0181: 0.9351

Customer C0009:
  C0198: 0.9177
  C0058: 0.8791
  C0033: 0.8782

Customer C0010:
  C0077: 0.9555
  C0176: 0.9475
  C0061: 0.9361

Customer C0011:
  C0126: 0.9901
  C0027: 0.9681
  C0153: 0.9429

Customer C0012:
  C0065: 0.9753
  C0104: 0.9676
  C0136: 0.9631

Customer C0013:
  C0105: 0.9783
  C0067: 0.9674
  C0183: 0.9562

Customer C0014:
  C0151: 0.9964
  C0128: 0.9944
  C0097: 0.9935

Customer C0015:
  C0123: 0.9936
  C0131: 0.9113
  C0071: 0.9101

Customer C0016:
  C0183: 0.9962
  C0107: 0.9942
  C0105: 0.9535

Customer C0017:
  C0075: 0.983
  C0090: 0.979
  C0004: 0.9743

Customer C0018:
  C0057: 0.9449
  C0023: 0.9384
  C0087: 0.9343

Customer C0019:
  C0191: 0.9618
  C0070: 0.9171
  C0174: 0.9122
```

Customer C0020:
  C0130: 0.9845
  C0120: 0.9652
  C0005: 0.8818