```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Load the datasets
customers = pd.read_csv('Customers.csv')
products = pd.read_csv('Products.csv')
transactions = pd.read_csv('Transactions.csv')

# Display the first few rows of each dataset
print("Customers Data:")
print(customers.head())
print("\nProducts Data:")
print(products.head())
print("\nTransactions Data:")
print(transactions.head())
```

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

Products Data:
  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

Transactions Data:
  TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0        T00001      C0199      P067  2024-08-25 12:38:23         1
1        T00112      C0146      P067  2024-05-27 22:23:54         1
2        T00166      C0127      P067  2024-04-25 07:38:55         1
3        T00272      C0087      P067  2024-03-26 22:55:37         2
4        T00363      C0070      P067  2024-03-21 15:10:10         3

   TotalValue   Price
0      300.68  300.68
1      300.68  300.68
2      300.68  300.68
3      601.36  300.68
4      902.04  300.68
```

```
# Strip whitespace from customer IDs
customers['CustomerID'] = customers['CustomerID'].str.strip()
transactions['CustomerID'] = transactions['CustomerID'].str.strip()

# Create a summary table for each customer
customer_summary = transactions.groupby('CustomerID').agg(
    TotalSpent=('TotalValue', 'sum'),
    PurchaseFrequency=('TransactionID', 'count'),
    LastPurchaseDate=('TransactionDate', 'max')
).reset_index()

# Calculate recency (in days)
customer_summary['LastPurchaseDate'] = pd.to_datetime(customer_summary['LastPurchaseDate'])
customer_summary['Recency'] = (pd.to_datetime('now') - customer_summary['LastPurchaseDate']).dt.days

# Merge with customer profile information
customer_summary = customer_summary.merge(customers, on='CustomerID')

# Display the customer summary
print("\nCustomer Summary:")
print(customer_summary.head())
```

```

Customer Summary:
  CustomerID  TotalSpent  PurchaseFrequency    LastPurchaseDate  Recency  \
0      C0001     3354.52                  5 2024-11-02 17:04:16       84
```

```
1      C0002     1862.74                 4 2024-12-03 01:41:41       54
2      C0003     2725.38                 4 2024-08-24 18:54:04      154
3      C0004     5354.88                 8 2024-12-23 14:13:52       33
4      C0005     2034.24                 3 2024-11-04 00:30:22       83

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

```
# Strip whitespace from customer IDs
customers['CustomerID'] = customers['CustomerID'].str.strip()
transactions['CustomerID'] = transactions['CustomerID'].str.strip()

# Create a summary table for each customer
customer_summary = transactions.groupby('CustomerID').agg(
    TotalSpent=('TotalValue', 'sum'),
    PurchaseFrequency=('TransactionID', 'count'),
    LastPurchaseDate=('TransactionDate', 'max')
).reset_index()

# Calculate recency (in days)
customer_summary['LastPurchaseDate'] = pd.to_datetime(customer_summary['LastPurchaseDate'])
customer_summary['Recency'] = (pd.to_datetime('now') - customer_summary['LastPurchaseDate']).dt.days

# Merge with customer profile information
customer_summary = customer_summary.merge(customers, on='CustomerID')

# Display the customer summary
print("\nCustomer Summary:")
print(customer_summary.head())
```

```
Customer Summary:
  CustomerID  TotalSpent  PurchaseFrequency    LastPurchaseDate  Recency  \
0      C0001     3354.52                 5 2024-11-02 17:04:16       84
1      C0002     1862.74                 4 2024-12-03 01:41:41       54
2      C0003     2725.38                 4 2024-08-24 18:54:04      154
3      C0004     5354.88                 8 2024-12-23 14:13:52       33
4      C0005     2034.24                 3 2024-11-04 00:30:22       83

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

```
# Select relevant features for similarity
features = customer_summary[['TotalSpent', 'PurchaseFrequency', 'Recency']]

# Standardize the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Calculate cosine similarity
similarity_matrix = cosine_similarity(features_scaled)

# Convert to DataFrame for easier handling
similarity_df = pd.DataFrame(similarity_matrix, index=customer_summary['CustomerID'], columns=customer_summary['CustomerID'])
```

```
# Function to get top N lookalikes for a given customer
def get_top_lookalikes(customer_id, n=3):
    if customer_id not in similarity_df.index:
        print(f"Customer ID {customer_id} not found in similarity matrix.")
        return []

    # Get similarity scores for the given customer
    scores = similarity_df[customer_id]

    # Sort scores in descending order and get the top N lookalikes
    top_lookalikes = scores.sort_values(ascending=False).head(n + 1)  # +1 to exclude the customer themselves
    top_lookalikes = top_lookalikes.iloc[1:]  # Exclude the customer themselves
    return top_lookalikes
```

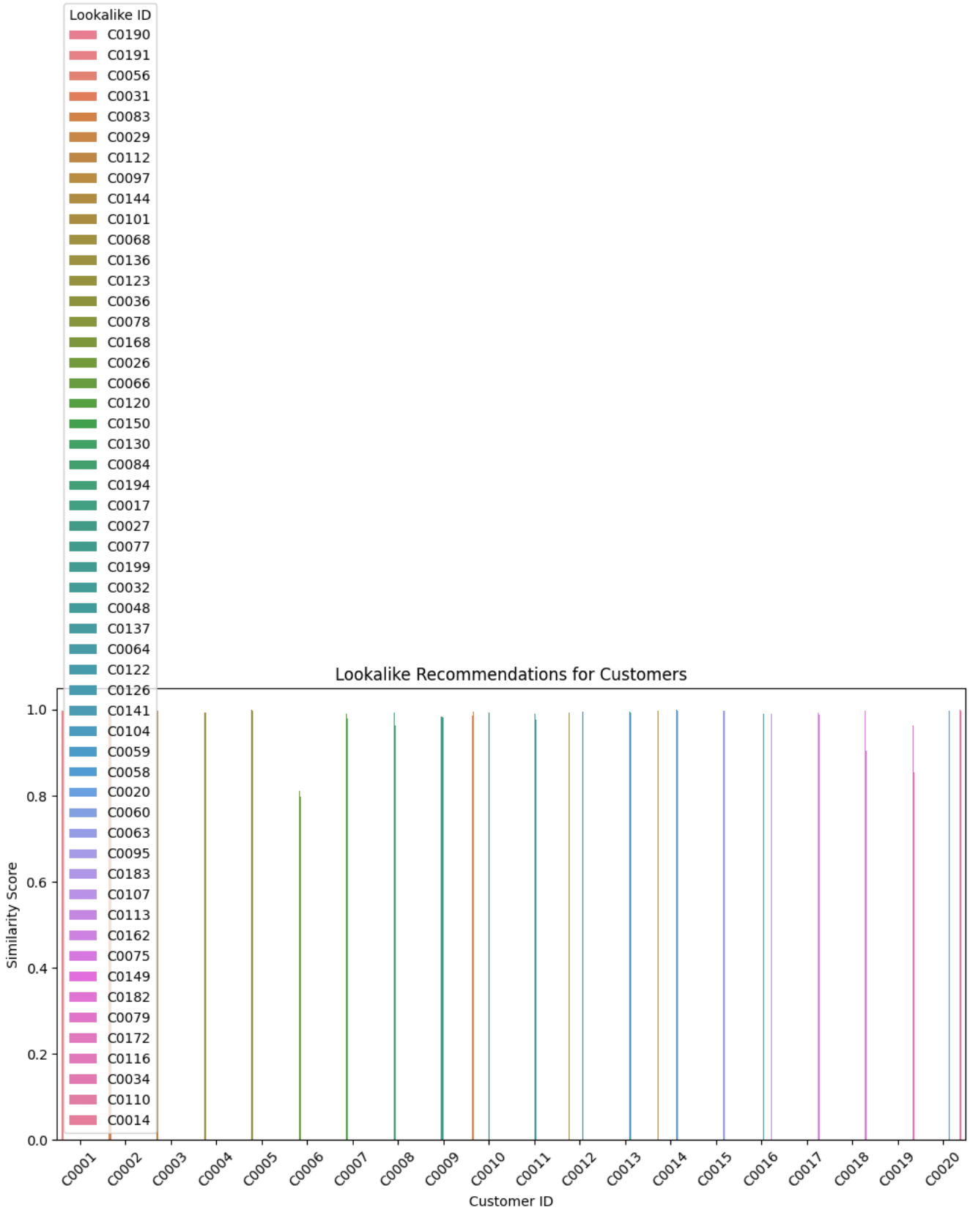
```
# Create a list to store lookalike records
lookalike_records = []

# Get lookalikes for the first 20 customers
for customer in customer_summary['CustomerID'][:20]:
    lookalikes = get_top_lookalikes(customer)
    for lookalike_id, score in zip(lookalikes.index, lookalikes.values):
        lookalike_records.append((customer, lookalike_id, score))

# Create a DataFrame from the records
lookalike_df = pd.DataFrame(lookalike_records, columns=['CustomerID', 'LookalikeID', 'SimilarityScore'])

# Save to CSV
lookalike_df.to_csv('Lookalike.csv', index=False)

# Display the lookalike results
print("\nLookalike Results:")
print(lookalike_df)
```

```
1       C0001       C0191         0.997061
2       C0001       C0056         0.996593
3       C0002       C0031         0.998220
4       C0002       C0083         0.989067
5       C0002       C0029         0.985330
6       C0003       C0112         0.999616
7       C0003       C0097         0.998874
8       C0003       C0144         0.998175
9       C0004       C0101         0.999047
10      C0004       C0068         0.991998
11      C0004       C0136         0.991973
12      C0005       C0123         0.999847
13      C0005       C0036         0.998530
14      C0005       C0078         0.992164
15      C0006       C0168         0.897680
16      C0006       C0026         0.812259
17      C0006       C0066         0.796630
18      C0007       C0120         0.990177
19      C0007       C0150         0.979227
20      C0007       C0130         0.975871
21      C0008       C0084         0.993624
22      C0008       C0194         0.968115
23      C0008       C0017         0.963854
24      C0009       C0027         0.983779
25      C0009       C0077         0.983589
26      C0009       C0199         0.980664
27      C0010       C0083         0.995611
28      C0010       C0032         0.993627
29      C0010       C0031         0.987139
30      C0011       C0048         0.989664
31      C0011       C0137         0.976898
32      C0011       C0064         0.967190
33      C0012       C0122         0.998723
34      C0012       C0126         0.994967
35      C0012       C0136         0.992759
36      C0013       C0141         0.998557
37      C0013       C0104         0.995125
38      C0013       C0059         0.993902
39      C0014       C0058         0.999756
40      C0014       C0020         0.998206
41      C0014       C0144         0.997501
42      C0015       C0060         0.998132
43      C0015       C0063         0.997741
44      C0015       C0095         0.997077
45      C0016       C0183         0.997540
46      C0016       C0064         0.990873
47      C0016       C0107         0.989798
48      C0017       C0113         0.999254
49      C0017       C0162         0.992220
50      C0017       C0075         0.988497
51      C0018       C0149         0.998655
52      C0018       C0182         0.966649
53      C0018       C0079         0.905506
54      C0019       C0172         0.984450
55      C0019       C0116         0.963640
56      C0019       C0034         0.854459
57      C0020       C0110         0.998992
58      C0020       C0014         0.998206
59      C0020       C0058         0.997890
```

```
# Visualization of the lookalikes
plt.figure(figsize=(12, 6))
sns.barplot(data=lookalike_df, x='CustomerID', y='SimilarityScore', hue='LookalikeID')
plt.title('Lookalike Recommendations for Customers')
```

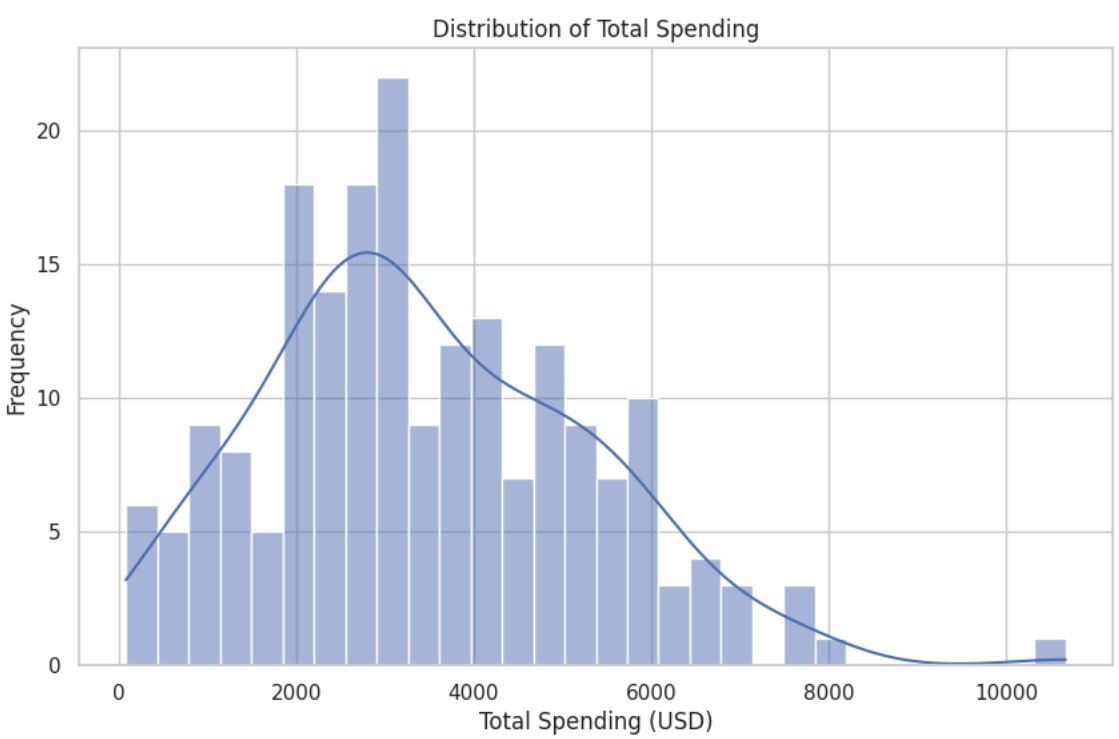
```
plt.xlabel('Customer ID')
plt.ylabel('Similarity Score')
plt.xticks(rotation=45)
plt.legend(title='Lookalike ID')
plt.tight_layout()
plt.show()
```

```
<ipython-input-7-cb349adfc9d8>:9: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large er
  plt.tight_layout()
```

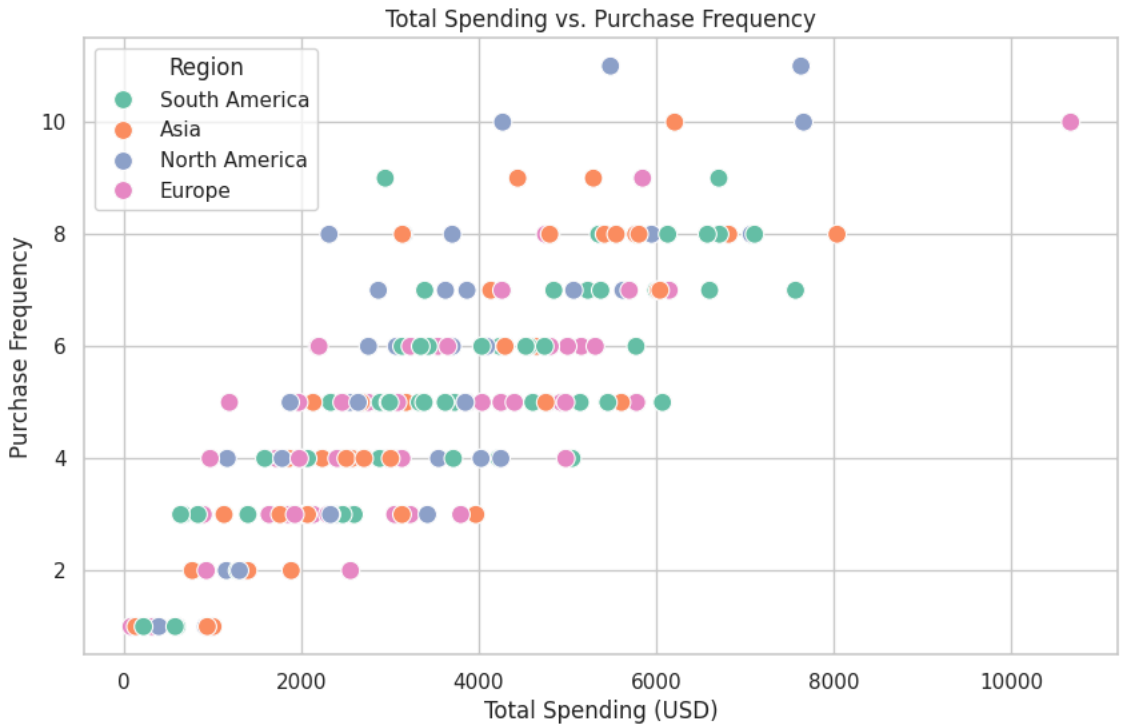


```
sns.set(style="whitegrid")
```

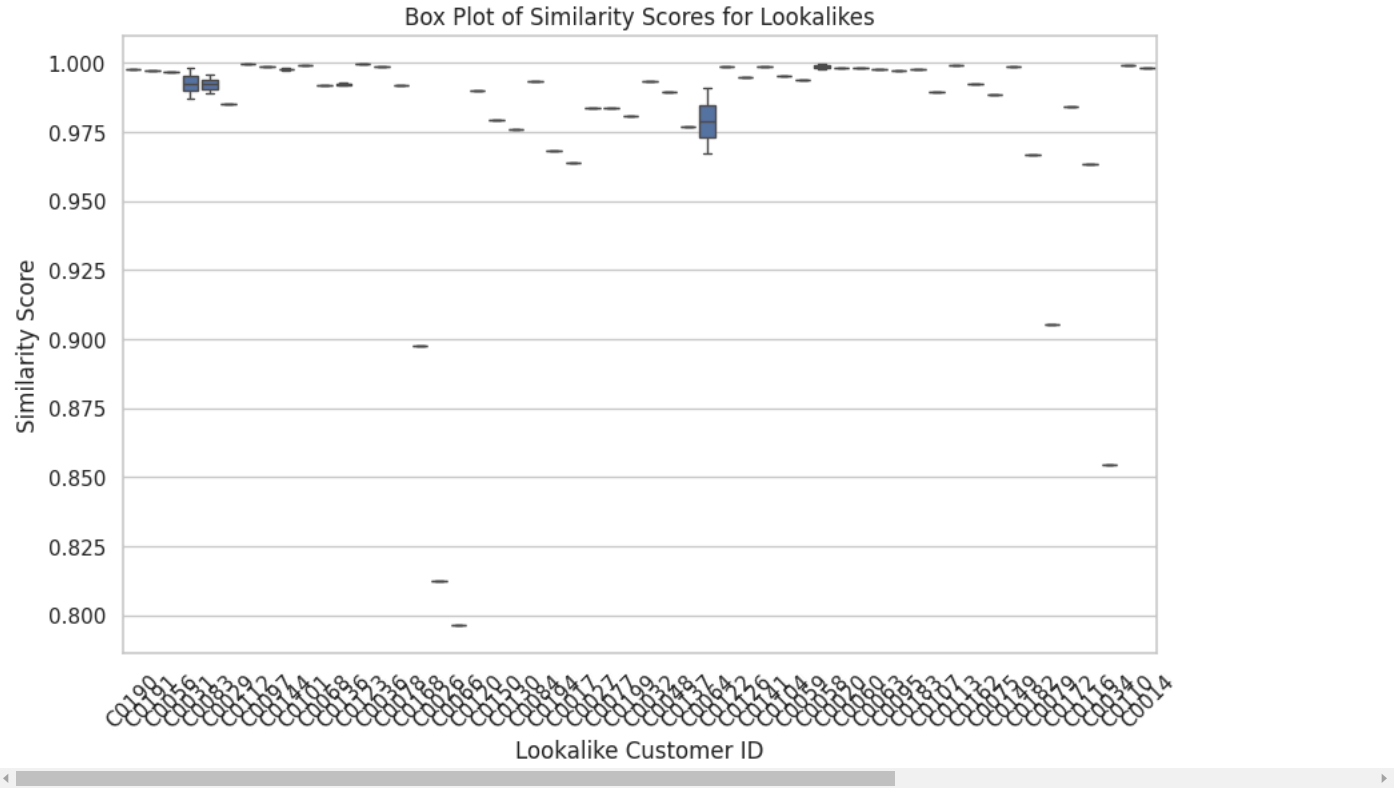
```
plt.figure(figsize=(10, 6))
sns.histplot(customer_summary['TotalSpent'], bins=30, kde=True)
plt.title('Distribution of Total Spending')
plt.xlabel('Total Spending (USD)')
plt.ylabel('Frequency')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=customer_summary, x='TotalSpent', y='PurchaseFrequency', hue='Region', palette='Set2', s=100)
plt.title('Total Spending vs. Purchase Frequency')
plt.xlabel('Total Spending (USD)')
plt.ylabel('Purchase Frequency')
plt.legend(title='Region')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.boxplot(data=lookalike_df, x='LookalikeID', y='SimilarityScore')
plt.title('Box Plot of Similarity Scores for Lookalikes')
plt.xlabel('Lookalike Customer ID')
plt.ylabel('Similarity Score')
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(12, 10))
sns.heatmap(similarity_df, cmap='coolwarm', annot=False, fmt=".2f", cbar=True)
plt.title('Heatmap of Customer Similarity Scores')
plt.xlabel('Customer ID')
```

```
plt.ylabel('Customer ID')
plt.show()
```