```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler

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

# One-hot encode the 'Region' column in the customers dataset
customers_one_hot = pd.get_dummies(customers, columns=["Region"],
prefix="Region")

# Aggregate transaction data to calculate total purchase value and
product preferences per customer
transaction_agg = (
    transactions.groupby("CustomerID")
    .agg(
        TotalSpent=("TotalValue", "sum"),
        TotalTransactions=("TransactionID", "count"),
    )
    .reset_index()
)


# Merge aggregated transaction data with customer data
customer_features = customers_one_hot.merge(
    transaction_agg, on="CustomerID", how="left"
)

# Replace NaN values (e.g., customers with no transactions) with zeros
customer_features.fillna(0, inplace=True)



# Scale numerical features for better similarity calculations
scaler = StandardScaler()
numerical_features = ["TotalSpent", "TotalTransactions"]
customer_features[numerical_features] = scaler.fit_transform(
    customer_features[numerical_features]
)

# Drop non-feature columns (e.g., CustomerID, Name) for similarity
calculations
feature_columns = [
    col
    for col in customer_features.columns
    if col not in ["CustomerID", "CustomerName", "SignupDate"]
```

```
]
feature_matrix = customer_features[feature_columns].values
```

2. Calculate Similarity Scores

```
# Compute pairwise cosine similarity between all customers
similarity_matrix = cosine_similarity(feature_matrix)

# Convert the similarity matrix to a DataFrame for easy interpretation
similarity_df = pd.DataFrame(
    similarity_matrix, index=customer_features["CustomerID"],
columns=customer_features["CustomerID"]
)
```

3. Generate Lookalike Recommendations

```
# Function to get top N similar customers for a given customer ID
def get_top_n_similar_customers(customer_id, n=3):
    # Sort by similarity score, exclude the customer themselves
    similar_customers = (
        similarity_df[customer_id]
        .sort_values(ascending=False)
        .drop(customer_id)
        .head(n)
    )
    return list(similar_customers.index),
list(similar_customers.values)

# Generate lookalike recommendations for the first 20 customers (C0001
- C0020)
lookalike_map = {}
for customer_id in customers["CustomerID"][:20]:
    similar_ids, scores = get_top_n_similar_customers(customer_id,
n=3)
    lookalike_map[customer_id] = list(zip(similar_ids, scores))

# Convert the map to a DataFrame for CSV export
lookalike_df = pd.DataFrame(
    [
        {"CustomerID": cust_id, "Lookalikes": lookalikes}
        for cust_id, lookalikes in lookalike_map.items()
    ]
)

# Save the Lookalike map to a CSV file
lookalike_df.to_csv("Lookalike.csv", index=False)
```

4. Display Results.

```
print("Top 3 lookalikes for the first 20 customers:")
print(lookalike_df.head(20))
```

```
Top 3 lookalikes for the first 20 customers:
   CustomerID                                          Lookalikes
0       C0001  [(C0137, 0.9999291789477135), (C0152, 0.999855...
1       C0002  [(C0142, 0.9920726369933641), (C0177, 0.973560...
2       C0003  [(C0133, 0.9974501506919937), (C0052, 0.994995...
3       C0004  [(C0113, 0.9906184644924116), (C0102, 0.986849...
4       C0005  [(C0159, 0.9999316844540107), (C0186, 0.996880...
5       C0006  [(C0158, 0.9721823024015926), (C0168, 0.954232...
6       C0007  [(C0159, 0.9856429138763982), (C0005, 0.983602...
7       C0008  [(C0109, 0.9806650275246662), (C0139, 0.971844...
8       C0009  [(C0062, 0.9856823344584039), (C0198, 0.982281...
9       C0010  [(C0199, 0.9968329080506292), (C0121, 0.984296...
10      C0011  [(C0107, 0.9983885132666855), (C0048, 0.997982...
11      C0012  [(C0155, 0.9992813479856844), (C0108, 0.994019...
12      C0013  [(C0087, 0.9944857312437704), (C0155, 0.990402...
13      C0014  [(C0060, 0.9993546455561516), (C0198, 0.994902...
14      C0015  [(C0144, 0.9995067470324408), (C0058, 0.993888...
15      C0016  [(C0183, 0.9999215728611632), (C0018, 0.921113...
16      C0017  [(C0075, 0.9798142121599286), (C0124, 0.979198...
17      C0018  [(C0016, 0.9211133448437205), (C0183, 0.916165...
18      C0019  [(C0172, 0.9999839210115643), (C0111, 0.934671...
19      C0020  [(C0058, 0.9959529806446508), (C0144, 0.994028...
```

Feature Engineering:

1.Customer Data: Region was one-hot encoded to turn categorical data into numerical features.

2.Transaction Data: Aggregated transaction counts and total spending for each customer.

3.Scaling: Numerical features were standardized to ensure fair comparison in similarity calculations.

Similarity Calculation:

1.Used cosine similarity to measure similarity between customer vectors.

2.Generated a similarity matrix to store scores for all customer pairs.

Recommendation:

1.Extracted the top 3 most similar customers for each customer, excluding themselves.

2.Organized the results in a map format and saved to Lookalike.csv.

Evaluation Criteria

Model Accuracy: Cosine similarity ensures logical and accurate grouping of similar customers.

Quality of Recommendations: Top 3 lookalikes are relevant based on transaction and customer features.