```
import pandas as pd

# Load the datasets
customers_df = pd.read_csv('Customers.csv')
transactions_df = pd.read_csv('Transactions.csv')
products_df = pd.read_csv('Products.csv')

# --- 1. Region (Encoded) ---
# One-hot encode the 'Region' column without dropping the first category
customers_df = pd.get_dummies(customers_df, columns=['Region'], drop_first=False)

# --- 2. Total Number of Transactions ---
# Count the total number of transactions for each customer
total_transactions = transactions_df.groupby('CustomerID').size().reset_index(name='Total No of Transactions')
customers_df = customers_df.merge(total_transactions, on='CustomerID', how='left')

# --- 3. Total Spend ---
# Calculate the total spend for each customer
total_spend = transactions_df.groupby('CustomerID')['TotalValue'].sum().reset_index(name='Total Spend')
customers_df = customers_df.merge(total_spend, on='CustomerID', how='left')

# --- 4. Average Transaction Value ---
# Calculate the average transaction value
customers_df['Average Transaction Value'] = customers_df['Total Spend'] / customers_df['Total No of Transactions']

# --- 5. Category Spend ---
# Merge transactions with products to get the product category
transactions_with_category = transactions_df.merge(products_df[['ProductID', 'Category']], on='ProductID', how='left')

# Calculate total spend per category for each customer
category_spend = transactions_with_category.groupby(['CustomerID', 'Category'])['TotalValue'].sum().reset_index(name='Categor

# Pivot the table to get each category as a separate column for spend
category_spend_pivot = category_spend.pivot(index='CustomerID', columns='Category', values='Category Spend').fillna(0)

# Merge with the customers dataframe
customers_df = customers_df.merge(category_spend_pivot, on='CustomerID', how='left')

# --- 6. Category Frequency ---
# Count the number of transactions per category for each customer
category_frequency = transactions_with_category.groupby(['CustomerID', 'Category']).size().reset_index(name='Category Frequen

# Pivot the table to get each category as a separate column for frequency
category_frequency_pivot = category_frequency.pivot(index='CustomerID', columns='Category', values='Category Frequency').fill

# Merge with the customers dataframe
customers_df = customers_df.merge(category_frequency_pivot, on='CustomerID', how='left')


# Save the final dataframe with features
customers_df.to_csv('Customer_Features_Keep_All_Regions_.csv', index=False)

print("Customer features with all regions saved to 'Customer_Features_Keep_All_Regions.csv'")
```

```
Customer features with all regions saved to 'Customer_Features_Keep_All_Regions.csv'
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity

# Step 1: Load the customer features CSV
customers_df = pd.read_csv('Customer_Features_Keep_All_Regions.csv')

# Step 2: Check for missing values
print(customers_df.isnull().sum())

# Fill missing values with 0
customers_df.fillna(0, inplace=True)

# Step 3: Data Preprocessing
# Extract relevant columns for similarity calculation
features = customers_df.drop(columns=['CustomerID', 'CustomerName', 'SignupDate'])

# Convert Region columns (TRUE/FALSE) to numerical values (1/0)
region_columns = ['Region_Asia', 'Region_Europe', 'Region_North America', 'Region_South America']
features[region_columns] = features[region_columns].astype(int)

# Standardize the numerical features
scaler = StandardScaler()
```

```
scaled_features = scaler.fit_transform(features)

# Step 4: Compute Cosine Similarity
# Compute cosine similarity between all customers
similarity_matrix = cosine_similarity(scaled_features)

# Step 5: Generate Lookalike Recommendations
lookalikes = {}

# For customers C0001 to C0020, we want to find the top 3 similar customers
for i in range(20):  # For customers C0001 to C0020
    cust_id = customers_df.loc[i, 'CustomerID']

    # Get the similarity scores for the current customer (row in the matrix)
    similarity_scores = similarity_matrix[i]

    # Exclude the similarity with the customer itself (i.e., 1.0)
    similarity_scores[i] = -1  # Assign a very low score to itself to exclude it

    # Get the indices of the top 3 most similar customers (excluding itself)
    top_3_indices = similarity_scores.argsort()[-3:][::-1]

    # Get the customer IDs and similarity scores for the top 3
    top_3_lookalikes = [(customers_df.loc[j, 'CustomerID'], similarity_scores[j]) for j in top_3_indices]

    # Create a list of lookalikes and scores
    lookalikes[cust_id] = [(lookalike_cust_id, score) for lookalike_cust_id, score in top_3_lookalikes]

# Step 6: Create the Lookalike CSV
lookalike_data = []

# Convert the lookalikes dictionary to the required format
for cust_id, lookalike_list in lookalikes.items():
    lookalike_data.append({
        'cust_id': cust_id,
        'lookalike_cust_ids_and_scores': str(lookalike_list)  # Convert the list of lookalikes to a string
    })

lookalike_df = pd.DataFrame(lookalike_data)

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

print("Lookalike.csv has been saved.")
```

```
CustomerID                 0
CustomerName               0
SignupDate                 0
Region_Asia                0
Region_Europe              0
Region_North America       0
Region_South America       0
Total No of Transactions   1
Total Spend                1
Average Transaction Value  1
Books_x                    1
Clothing_x                 1
Electronics_x              1
Home Decor_x               1
Books_y                    1
Clothing_y                 1
Electronics_y              1
Home Decor_y               1
dtype: int64
Lookalike.csv has been saved.
```

```
lookalike_df = pd.read_csv('Chirag_Gupta_Lookalike.csv')
lookalike_df
```

	cust_id	lookalike_cust_ids_and_scores
0	C0001	[('C0091', 0.8908422994906), ('C0120', 0.88625...
1	C0002	[('C0134', 0.9330356886590646), ('C0159', 0.91...
2	C0003	[('C0031', 0.9195362040251337), ('C0152', 0.80...
3	C0004	[('C0113', 0.8681637620310363), ('C0012', 0.83...
4	C0005	[('C0007', 0.9595673401635022), ('C0146', 0.88...
5	C0006	[('C0187', 0.7669275518781463), ('C0169', 0.76...
6	C0007	[('C0005', 0.9595673401635022), ('C0140', 0.87...
7	C0008	[('C0098', 0.7927078262500883), ('C0194', 0.77...
8	C0009	[('C0198', 0.9338276912431526), ('C0010', 0.84...
9	C0010	[('C0111', 0.9017605958313126), ('C0062', 0.89...
10	C0011	[('C0153', 0.8697689581026931), ('C0126', 0.77...
11	C0012	[('C0104', 0.9139567967558033), ('C0152', 0.89...
12	C0013	[('C0188', 0.8763216361016894), ('C0099', 0.86...
13	C0014	[('C0060', 0.9814851639655328), ('C0198', 0.88...
14	C0015	[('C0036', 0.9133874291050558), ('C0144', 0.90...
15	C0016	[('C0183', 0.8171306746865663), ('C0117', 0.77...
16	C0017	[('C0075', 0.9063812623664969), ('C0057', 0.78...
17	C0018	[('C0125', 0.8328130980174581), ('C0050', 0.82...
18	C0019	[('C0070', 0.787728756527267), ('C0121', 0.736...
19	C0020	[('C0050', 0.7949734014471683), ('C0058', 0.78...