```
import pandas as pd
import numpy as np
import warnings
# Suppress specific warning
warnings.filterwarnings('ignore', category=UserWarning)



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

# Merge datasets
data = transactions.merge(customers, on='CustomerID',
how='inner').merge(products, on='ProductID', how='inner')

# Ensure TransactionDate is parsed as datetime
data['TransactionDate'] = pd.to_datetime(data['TransactionDate'],
errors='coerce')

# Compute the last transaction date for each customer
last_transaction = data.groupby('CustomerID')
['TransactionDate'].max().reset_index()
last_transaction.columns = ['CustomerID', 'LastTransactionDate']

# Compute Recency: Days since the last transaction
current_date = pd.Timestamp.today()
last_transaction['Recency'] = (current_date -
last_transaction['LastTransactionDate']).dt.days

# Aggregate other features
customer_features = data.groupby('CustomerID').agg({
    'TotalValue': ['sum', 'mean'],  # Total and average transaction
value
    'Quantity': 'sum',              # Total quantity purchased
    'TransactionID': 'count',       # Number of transactions
    'Category': lambda x: x.nunique()  # Number of unique product
categories
}).reset_index()

# Rename columns
customer_features.columns = ['CustomerID', 'TotalValue_sum',
'TotalValue_mean',
                             'Quantity_sum', 'Transaction_count',
'Unique_categories']

# Merge last transaction data with customer features
customer_features = customer_features.merge(last_transaction,
on='CustomerID', how='left')
```

```
# One-hot encode product categories
category_prefs = pd.get_dummies(data[['CustomerID', 'Category']],
columns=['Category'], prefix='Category')
category_prefs =
category_prefs.groupby('CustomerID').sum().reset_index()

# Combine aggregated features with category preferences
final_features = customer_features.merge(category_prefs,
on='CustomerID', how='inner')

import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Assuming 'final_features' is already defined, and contains customer
data
# final_features = ... (your dataset)

# Drop non-numerical columns (like CustomerID, LastTransactionDate)
for scaling
numerical_features = final_features.drop(columns=['CustomerID',
'LastTransactionDate'])

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler to the numerical features and transform the data
features_scaled = scaler.fit_transform(numerical_features)

# Define the target variable (Total spend) and features
X = features_scaled  # Features
y = final_features['TotalValue_sum']  # Target variable (Total spend)

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Now you can proceed with model training or any other operations.

from sklearn.metrics.pairwise import cosine_similarity

# Compute similarity matrix (cosine similarity)
similarity_matrix = cosine_similarity(features_scaled)

# Create a DataFrame for similarity scores (similarity_df)
similarity_df = pd.DataFrame(similarity_matrix,
index=final_features['CustomerID'],
columns=final_features['CustomerID'])

# Function to get top 3 similar customers
def get_top_similar_users(customer_id, similarity_df, top_n=3):
```

```
    similar_customers =
similarity_df[customer_id].sort_values(ascending=False).iloc[1:top_n+1
]
    return [(cust_id, score) for cust_id, score in
similar_customers.items()]

# Generate recommendations for customers C0001 to C0020
customer_ids = [f'C{str(i).zfill(4)}' for i in range(1, 21)]
recommendations = {}

for cust_id in customer_ids:
    if cust_id in similarity_df.index:
        recommendations[cust_id] = get_top_similar_users(cust_id,
similarity_df)

# Convert recommendations to a DataFrame
lookalike_data = []
for customer, similar_list in recommendations.items():
    for similar_customer, score in similar_list:
        lookalike_data.append({'cust_id': customer, 'similar_cust':
similar_customer, 'score': score})

lookalike_df = pd.DataFrame(lookalike_data)

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