Task 2: Building a Lookalike Model

To create a lookalike model, we will:

1. Preprocess and combine datasets.
2. Engineer relevant features using customer profiles and transaction history.
3. Use a similarity measure to find the top 3 similar customers for each target customer.
4. Save the results in the specified format: Map<cust_id, List<cust_id, score>>.

we will use 2 approaches to figure out and save it as lookalike models

1. **K-Nearest Neighbors (KNN) Approach**

We will now use customer and transaction features together for similarity calculation.

Steps:

1. **Combine Data:**
   - Merge Transactions with Customers and Products to form a comprehensive dataset.
   - Aggregate transaction data (e.g., total spending, total quantity purchased) for each customer.
2. **Feature Engineering:**
   - Encode categorical features like Region and Category.
   - Calculate summary statistics for transactions (e.g., total spend, average transaction value).
3. **Scale Features:** Normalize numerical columns for similarity calculation.
4. **Fit KNN:** Use the combined feature space for similarity.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import NearestNeighbors

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

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

# Aggregate transaction data for each customer
customer_agg = merged_data.groupby('CustomerID').agg({
    'TotalValue': ['sum', 'mean'],  # Total and average spending
    'Quantity': ['sum'],            # Total quantity purchased
    'Category': lambda x: x.nunique(),  # Number of unique categories bought
    'Region': 'first',              # Region of the customer
    'SignupDate': 'first'           # Signup date
}).reset_index()
```

```
customer_agg.columns = ['CustomerID', 'TotalSpend', 'AvgSpend', 'TotalQuantity', 'UniqueCategories', 'Region', 'SignupDate']

# Encode categorical variables
le_region = LabelEncoder()
customer_agg['RegionEncoded'] = le_region.fit_transform(customer_agg['Region'])

# Convert SignupDate to numerical (e.g., days since signup)
customer_agg['SignupDate'] = pd.to_datetime(customer_agg['SignupDate'])
customer_agg['DaysSinceSignup'] = (pd.Timestamp.now() - customer_agg['SignupDate']).dt.days
customer_agg = customer_agg.drop(['Region', 'SignupDate'], axis=1)

# Scale features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(customer_agg.drop('CustomerID', axis=1))

# Fit KNN model
knn_model = NearestNeighbors(n_neighbors=4, metric='euclidean')
knn_model.fit(scaled_features)

# Find top 3 similar customers for first 20 customers
lookalike_results_knn = {}
for idx, customer_id in enumerate(customer_agg['CustomerID'][:20]):
    distances, indices = knn_model.kneighbors([scaled_features[idx]], n_neighbors=4)  # Include self
    lookalikes = [(customer_agg['CustomerID'][i], 1 / (1 + distances[0][j])) for j, i in enumerate(indices[0][1:])]  # Exclude self
    lookalike_results_knn[customer_id] = lookalikes

# Save results to CSV
lookalike_df_knn = pd.DataFrame({
    "CustomerID": list(lookalike_results_knn.keys()),
    "Lookalikes": [str(v) for v in lookalike_results_knn.values()]
})
lookalike_df_knn.to_csv("Lookalike_KNN.csv", index=False)
```

2. **Collaborative Filtering (Matrix Factorization) Approach**

Use CustomerID and ProductID to create a sparse user-product matrix and apply matrix factorization.

Steps:

1. **Prepare User-Product Matrix:**
    - Create a pivot table with CustomerID as rows, ProductID as columns, and TotalValue as values.
    - Fill missing values with 0.

2. **Apply SVD:**
    - Factorize the matrix using TruncatedSVD to reduce dimensions.
3. **Find Similar Customers:**
    - Compute cosine similarity between reduced vectors.

```
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics.pairwise import cosine_similarity

# Create user-product matrix
user_product_matrix = merged_data.pivot_table(index='CustomerID', columns='ProductID', values='TotalValue', fill_value=0)

# Apply SVD
svd = TruncatedSVD(n_components=10, random_state=42)
reduced_features = svd.fit_transform(user_product_matrix)

# Compute cosine similarity
similarity_matrix = cosine_similarity(reduced_features)
similarity_df = pd.DataFrame(similarity_matrix, index=user_product_matrix.index, columns=user_product_matrix.index)

# Get top 3 similar customers for first 20 customers
lookalike_results_svd = {}
for customer_id in similarity_df.index[:20]:
    similar_customers = similarity_df[customer_id].sort_values(ascending=False).iloc[1:4]  # Exclude self
    lookalike_results_svd[customer_id] = list(zip(similar_customers.index, similar_customers.values))

# Save results to CSV
lookalike_df_svd = pd.DataFrame({
    "CustomerID": list(lookalike_results_svd.keys()),
    "Lookalikes": [str(v) for v in lookalike_results_svd.values()]
})
lookalike_df_svd.to_csv("Lookalike_SVD.csv", index=False)
```