```python
import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine_similarity

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import pairwise_distances
```

Load Datasets

```python
import os
os.getcwd()

'C:\\Users\\Vivek'

os.chdir('C:\\Users\\Vivek\Desktop')

customers = pd.read_csv("Customers.csv")

products = pd.read_csv("Products.csv")

transactions = pd.read_csv("Transactions.csv")

merged_data = transactions.merge(customers,
on='CustomerID').merge(products, on='ProductID')
```

Creating a pivot table

```python
product_customer_matrix = merged_data.pivot_table(
    index='CustomerID', columns='ProductName', values='Quantity',
aggfunc='sum'
).fillna(0)

# Create customer profiles based on spending behavior
customer_profile = merged_data.groupby('CustomerID').agg({
    'TotalValue': 'sum',
    'Quantity': 'sum'
}).reset_index()

# Normalize customer profiles
customer_profile['TotalValue'] = customer_profile['TotalValue'] /
customer_profile['TotalValue'].max()
customer_profile['Quantity'] = customer_profile['Quantity'] /
customer_profile['Quantity'].max()

# Compute similarity matrix using cosine similarity
profile_matrix = customer_profile[['TotalValue', 'Quantity']].values
similarity_matrix = cosine_similarity(profile_matrix)
```

```
# Function to get top 3 lookalikes for each customer
def get_top_lookalikes(customer_id, similarity_matrix,
customer_profile):
    customer_index = customer_profile[customer_profile['CustomerID']
== customer_id].index[0]
    similarities = similarity_matrix[customer_index]
    similar_customer_indices = similarities.argsort()[-4:-1]
    similar_customer_indices = similarities.argsort()[-4:-1]
    similar_customers =
customer_profile.iloc[similar_customer_indices]
    top_lookalikes = [(similar_customer[0], similarities[i]) for i,
similar_customer in enumerate(similar_customers.itertuples())]
    return top_lookalikes
```

Function to recommend similar products

```
def recommend_products(product_name, top_n=5):
    if product_name not in product_similarity_df.index:
        return f"Product '{product_name}' not found in the data."
    # Sort similar products
    similar_products =
product_similarity_df[product_name].sort_values(ascending=False)
    return similar_products[1:top_n + 1]

# List all unique product names in the dataset
print("Available product names in the data:")
print(merged_data['ProductName'].unique())
```

```
Available product names in the data:
['ComfortLiving Bluetooth Speaker' 'HomeSense Wall Art' 'ActiveWear
Rug'
 'BookWorld Bluetooth Speaker' 'TechPro Rug' 'SoundWave Textbook'
 'TechPro Headphones' 'ActiveWear Wall Art' 'BookWorld Cookbook'
 'SoundWave Headphones' 'HomeSense Desk Lamp' 'SoundWave Laptop'
 'ActiveWear Smartphone' 'TechPro Textbook' 'HomeSense Novel'
 'ActiveWear Cookbook' 'SoundWave Jeans' 'SoundWave Mystery Book'
 'ActiveWear Jeans' 'BookWorld Sweater' 'ComfortLiving Cookware Set'
 'TechPro T-Shirt' 'BookWorld Cookware Set' 'TechPro Vase' 'TechPro
Novel'
 'SoundWave Jacket' 'ActiveWear T-Shirt' 'BookWorld Rug'
 'HomeSense Cookware Set' 'SoundWave Bluetooth Speaker'
 'ComfortLiving Desk Lamp' 'BookWorld Smartwatch' 'SoundWave Rug'
 'ComfortLiving Biography' 'ActiveWear Running Shoes'
 'ActiveWear Textbook' 'ActiveWear Smartwatch' 'HomeSense Rug'
 'SoundWave Cookbook' 'TechPro Cookbook' 'SoundWave Novel'
 'HomeSense T-Shirt' 'BookWorld Jacket' 'ComfortLiving Smartwatch'
 'TechPro Smartwatch' 'BookWorld Biography' 'HomeSense Sweater'
 'HomeSense Bluetooth Speaker' 'ComfortLiving Sweater'
 'ActiveWear Cookware Set' 'SoundWave Desk Lamp'
 'ComfortLiving Smartphone' 'HomeSense Headphones'
 'HomeSense Running Shoes' 'ComfortLiving Laptop' 'SoundWave
Smartwatch'
 'BookWorld Running Shoes' 'ActiveWear Jacket' 'ActiveWear Biography'
 'ComfortLiving Rug' 'SoundWave T-Shirt' 'ActiveWear Headphones'
 'ComfortLiving Mystery Book' 'TechPro Running Shoes'
 'ComfortLiving Headphones' 'BookWorld Wall Art']

product_to_recommend = 'ActiveWear T-Shirt'
recommendations = recommend_products(product_to_recommend, top_n=5)

print(f"Top recommendations for '{product_to_recommend}':")
print(recommendations)

Top recommendations for 'ActiveWear T-Shirt':
ProductName
SoundWave Jacket           0.262445
SoundWave Desk Lamp        0.205971
TechPro Running Shoes      0.192785
ActiveWear Cookware Set    0.171980
ActiveWear Rug             0.169464
Name: ActiveWear T-Shirt, dtype: float64
```

Quality of recommendations

```
def evaluate_recommendations(product_name):
    recommended_products = recommend_products(product_name, top_n=5)
    if isinstance(recommended_products, str):  # If the product is not
found
        return recommended_products
    scores = recommended_products.values
    quality_score = sum(scores) / len(scores)  # Average similarity
score
    return quality_score
```

Example evaluation

```
evaluation_score = evaluate_recommendations(product_to_recommend)

print(f"Quality score for recommendations of '{product_to_recommend}':
{evaluation_score}")

Quality score for recommendations of 'ActiveWear T-Shirt':
0.2005290732312462
```

Function to calculate accuracy

```
def evaluate_recommendations(target_product_name, top_recommendations,
data):
    # Check if the target product exists
    if target_product_name not in data['ProductName'].values:
        return f"Product '{target_product_name}' not found in the
data."

    # Get the feature vector for the target product
    target_features = data.loc[data['ProductName'] ==
target_product_name, ['TotalValue', 'Quantity', 'Price_y']]

    # Compute similarity scores between target product and others
    similarities = cosine_similarity(target_features,
data[['TotalValue', 'Quantity', 'Price_y']])

    # Get the indices of the top recommendations
    recommended_indices = [data.index[data['ProductName'] ==
rec].tolist()[0] for rec in top_recommendations]

    # Calculate the average similarity score for the recommendations
    avg_similarity_score = similarities[0, recommended_indices].mean()
    return avg_similarity_score

# Example Usage
target_product = "ActiveWear T-Shirt"
recommended_products = ["ActiveWear Wall Art", "HomeSense Wall Art",
"ActiveWear T-Shirt"]
accuracy = evaluate_recommendations(target_product,
recommended_products, merged_data)
print(f"Average Similarity Score for Recommendations: {accuracy}")
```

```
Average Similarity Score for Recommendations: 0.9522742227549711
```

```
# Create the Lookalike CSV
lookalike_data = {}
for customer_id in customer_profile['CustomerID'][:20]:  # For first
20 customers
    lookalike_data[customer_id] = get_top_lookalikes(customer_id,
similarity_matrix, customer_profile)

# Save to CSV
import csv
with open('Lookalike.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['CustomerID', 'Lookalikes'])
    for customer_id, lookalikes in lookalike_data.items():
        lookalikes_str = "; ".join([f"{cust_id}: {score:.2f}" for
cust_id, score in lookalikes])
        writer.writerow([customer_id, lookalikes_str])
```