```python
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity

import pandas as pd

# Load the dataframes
transactions_df = pd.read_csv(r'C:\Users\siddh\OneDrive\Desktop\
internship\zeotap\Transactions.csv')
customers_df = pd.read_csv(r'C:\Users\siddh\OneDrive\Desktop\
internship\zeotap\Customers.csv')
products_df = pd.read_csv(r'C:\Users\siddh\OneDrive\Desktop\
internship\zeotap\Products.csv')

# Merge the dataframes
merged_df = transactions_df.merge(customers_df,
on='CustomerID').merge(products_df, on='ProductID')

# Aggregating transactional data per customer
customer_features = merged_df.groupby('CustomerID').agg(
    TotalSpending=('TotalValue', 'sum'),
    AvgTransactionValue=('TotalValue', 'mean'),
    TotalQuantity=('Quantity', 'sum'),
    TransactionCount=('TransactionID', 'nunique')
).reset_index()

# One-hot encoding for product categories to capture preferences
category_preferences = pd.get_dummies(merged_df[['CustomerID',
'Category']], columns=['Category'])
category_features =
category_preferences.groupby('CustomerID').sum().reset_index()

# Combine transactional features with category preferences
customer_data = customer_features.merge(category_features,
on='CustomerID')

# Standardizing features for similarity calculation
scaler = StandardScaler()
scaled_features =
scaler.fit_transform(customer_data.drop(columns=['CustomerID']))

# Calculate cosine similarity
similarity_matrix = cosine_similarity(scaled_features)

# Mapping customers for lookup
customer_ids = customer_data['CustomerID'].tolist()

# Generating top 3 lookalike recommendations for customers C0001 to
C0020
lookalike_results = {}
for idx, customer_id in enumerate(customer_ids[:20]):
```

```
    # First 20 customers

    similarity_scores = list(enumerate(similarity_matrix[idx]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1],
reverse=True)
    top_3 = [(customer_ids[i], score) for i, score in
similarity_scores[1:4]]

    # Exclude self (first entry)

    lookalike_results[customer_id] = top_3

# Convert results to a dataframe
lookalike_df = pd.DataFrame([
    {"CustomerID": cust_id, "Lookalikes": lookalikes}
    for cust_id, lookalikes in lookalike_results.items()
])

# Correct file path with filename
lookalike_csv_path = 'C:\\Users\\siddh\\OneDrive\\Desktop\\
internship\\zeotap\\Siddharth_Deshwal_Lookalike.csv'

# Save the dataframe to a CSV file
lookalike_df.to_csv(lookalike_csv_path, index=False)
```

For checking the accuracy of the above model which is "Lookalike Model "

```
# Define the first 20 target customers explicitly
target_customers = [f"C{i:04d}" for i in range(1, 21)]

# Merging datasets for complete analysis
transactions = transactions_df.merge(customers_df, on="CustomerID",
how="left")
transactions = transactions_df.merge(products_df, on="ProductID",
how="left")

# Define function to validate similarity logic and quality
def evaluate_recommendations(lookalikes, transactions,
target_customers):
    # Results for evaluation
    evaluation_metrics = {
        "Average_Overlap": [],
        "Average_Similarity_Score": [],
        "Relevant_Recommendations": [],
        "Precision_At_3": [],
    }

    for cust_id, similar_list in lookalikes.items():
        # Check if this is a target customer
        if cust_id not in target_customers:
```

```
            continue

        # Get transaction history of target customer
        target_products = set(transactions[transactions["CustomerID"]
== cust_id]["ProductID"])

        # Evaluate quality of each recommended customer
        overlap_scores = []
        sim_scores = []
        relevant_count = 0

        for sim_cust, score in similar_list:
            sim_products = set(transactions[transactions["CustomerID"]
== sim_cust]["ProductID"])
            overlap = len(target_products & sim_products) /
max(len(target_products), 1)  # Avoid division by zero

            overlap_scores.append(overlap)
            sim_scores.append(score)

            # Count as relevant if overlap > 0
            if overlap > 0.1:
                relevant_count += 1

        # Store metrics

evaluation_metrics["Average_Overlap"].append(sum(overlap_scores) /
len(overlap_scores))

evaluation_metrics["Average_Similarity_Score"].append(sum(sim_scores)
/ len(sim_scores))

evaluation_metrics["Relevant_Recommendations"].append(relevant_count)
        evaluation_metrics["Precision_At_3"].append(relevant_count /
3)  # Precision @3

    # Compute overall metrics
    overall_metrics = {key: sum(value) / len(value) for key, value in
evaluation_metrics.items()}
    return evaluation_metrics, overall_metrics

# Assume lookalike_dict from earlier
lookalike_dict = {key: [(val[0], val[1]) for val in value] for key,
value in lookalike_results.items()}

# Perform evaluation
target_customers = [f"C{i:04d}" for i in range(1, 21)]
detailed_metrics, overall_metrics =
evaluate_recommendations(lookalike_dict, transactions,
target_customers)
```

```
overall_metrics
```

```
{'Average_Overlap': 0.07753968253968253,
 'Average_Similarity_Score': np.float64(0.9050367686735129),
 'Relevant_Recommendations': 0.95,
 'Precision_At_3': 0.31666666666666665}
```