Task 2: Lookalike Model

To achieve the described goals, I'll outline the approach for creating a Lookalike Model, including the logic, implementation, and deliverables. Here's the detailed breakdown:

Step-by-Step Approach

1. Data Preparation

- **Load the data**: Import the three CSV files (Customers.csv, Products.csv, Transactions.csv) into Pandas DataFrames.
- **Clean and preprocess the data**:
    - Convert date columns (SignupDate, TransactionDate) to datetime format.
    - Handle missing values or duplicates, if any.
    - Standardize column names and data types for consistency.
- **Feature engineering**:
    - Aggregate transactional data to extract meaningful metrics for each customer (e.g., total spend, number of transactions, average order value, favorite product category).
    - Use product data (e.g., product category) to enrich customer profiles.

---

2. Building the Lookalike Model

- **Feature vectors**:
    - Create a customer feature matrix by combining customer attributes (region, signup date) and transactional summaries (e.g., total spend, purchase frequency, product preferences).
    - Normalize the features to ensure comparability.
- **Similarity measure**:
    - Use a distance metric (e.g., cosine similarity or Euclidean distance) to calculate pairwise similarity between customers.
    - If needed, weight specific features (e.g., transactional data) more heavily than others (e.g., signup date).
- **Recommendation generation**:
    - For each customer, rank all other customers by similarity score.
    - Select the top 3 most similar customers as the "lookalikes."

---

3. Generating Deliverables

- **Lookalike.csv**:
    - Structure:

cust_id, lookalikes

C0001, [(C0002, 0.95), (C0003, 0.93), (C0004, 0.92)]

C0002, [(C0001, 0.97), (C0005, 0.91), (C0003, 0.90)]

...

  - Saving the top 3 lookalikes and their similarity scores for CustomerID C0001 to C0020.

- **Jupyter Notebook**:
  - Include sections:
    1. **Data Loading and Cleaning**: Show the preprocessing steps.
    2. **Feature Engineering**: Describe how customer profiles were created.
    3. **Model Development**: Explain the similarity calculation and logic for recommendations.
    4. **Results and Evaluation**: Display the Lookalike.csv and discuss insights.

Jupyter Notebook Structure:

1. Introduction

- **Objective**: Briefly explain the task, objectives, and deliverables.

2. Data Loading and Preprocessing

- **Load CSV Files**: Import the Customers.csv, Products.csv, and Transactions.csv files into DataFrames.
- **Preprocessing**: Clean and preprocess the data, including handling missing values, converting date columns, and feature engineering.

3. Feature Engineering

- **Customer Profiles**: Aggregate transactional data to compute key metrics such as total spend, transaction count, and average order value for each customer.
- **Product Data**: Integrate product-related information, such as the category, into the customer profile.

4. Similarity Calculation

- **Normalization**: Standardize the features to ensure equal weightage.
- **Cosine Similarity**: Calculate pairwise cosine similarity between customer profiles to determine how similar they are to each other.

5. Recommendation System (Lookalike Model)

- **Generate Lookalikes**: For each customer, find the top 3 most similar customers based on the similarity score.
- **Output**: Map of customer IDs to their top 3 lookalikes and similarity scores.

6. Results

- Display the Lookalike.csv output and analyze the recommendations.

Key Sections Explanation:

1. **Data Loading**:
   - We're reading all three CSV files into Pandas DataFrames for easy manipulation.

2. **Data Preprocessing**:
   - We convert date columns to datetime and handle missing data. For simplicity, we're filling missing values with default values (0 or 'Unknown'), but this can be improved.
3. **Feature Engineering**:
   - We calculate key customer metrics like total_spent, transaction_count, and avg_order_value to capture transactional behavior.
4. **Feature Normalization**:
   - The features are standardized using StandardScaler to ensure they all have equal influence during similarity calculations.
5. **Similarity Calculation**:
   - The cosine_similarity function is used to measure the similarity between customers based on their normalized profiles.
6. **Lookalike Recommendations**:
   - For each customer, the model ranks all other customers by their similarity score and selects the top 3 most similar customers.
7. **Saving Results**:
   - The lookalike results for the first 20 customers are saved to Lookalike.csv.

To build a Lookalike Model that recommends 3 similar customers based on both customer and product information, we can combine the following elements:

1. **Customer Information**: Attributes such as region, signup date, etc.
2. **Product Information**: Purchase history, favorite product categories, etc.
3. **Transaction History**: Total spend, frequency of purchases, and other transactional behavior.

The goal is to create a hybrid similarity measure that considers both customer demographics (e.g., region) and transactional behavior (e.g., products bought, total spend).

Approach

1. Data Integration

- Combine customer, product, and transaction data to create comprehensive customer profiles.
- Aggregate the product purchase history to identify customer preferences (e.g., favorite product categories).

2. Feature Engineering

- **Customer Profile**: Combine demographic information (e.g., region, signup date).
- **Transactional Profile**: Aggregate data from the transactions to summarize the purchase behavior (e.g., total spend, frequency of purchases).
- **Product Preferences**: Aggregate purchase data by product category to identify a customer's preferred product types.

3. Similarity Calculation

- Use both **demographic features** (e.g., region, signup date) and **transactional/product features** (e.g., total spend, product preferences) to calculate similarity.
- A weighted similarity score can be computed using cosine similarity or Euclidean distance on the feature vectors.

4. Lookalike Recommendations

- For a given customer, compute the similarity scores with all other customers.
- Rank customers by similarity and recommend the top 3 most similar customers.

Explanation of the Code:

1. **Data Loading**:
    - Load the three datasets: Customers.csv, Products.csv, and Transactions.csv.
2. **Data Preprocessing**:
    - Handle missing values in the Region column for customers and missing values for transaction-related fields.
3. **Feature Engineering**:
    - **Customer Profile**: Create a customer profile by aggregating transactional data (total spend, transaction count, average order value).
    - **Product Preferences**: Aggregate the purchase history to compute total spend per product category for each customer.
    - **Customer Profile Integration**: Merge demographic features with aggregated transactional features and product preferences.
4. **Normalization**:
    - Normalize all features using StandardScaler so that each feature has a mean of 0 and a standard deviation of 1. This ensures fair comparison when calculating similarity.
5. **Similarity Calculation**:
    - Use cosine_similarity to compute the similarity between all customer profiles. This measures how close the features of two customers are.
6. **Lookalike Recommendations**:
    - For a given customer, retrieve the top 3 most similar customers based on their similarity score.
7. **Saving Results**:
    - Store the top 3 lookalikes and their similarity scores for the first 20 customers in a Lookalike.csv file.

Data Preprocessing

- Clean the data by handling missing values, correcting data types, and handling any inconsistencies.
- Feature engineering: Derive important features like total spend per customer, average transaction value, transaction frequency, etc.

- Merge the relevant datasets (Customers, Products, Transactions) to create a unified dataset.

Lookalike Model

- **Objective**: Build a model that recommends similar customers based on transaction history and customer profile information (e.g., region, signup date, etc.).
- Use both **customer demographics** and **transaction data** to determine similarity.
- We'll focus on the similarity between customers based on transactional behavior (e.g., quantity, price, and product categories purchased).

Model Evaluation

- **Accuracy**: Since we don't have a clear target variable, accuracy can be evaluated by inspecting the **quality** of recommendations based on customer spending, product preferences, etc.
- **Logic**: Ensure that the recommendations make sense based on real-world business logic (e.g., customers who spend similarly, purchase the same types of products, or live in similar regions should be recommended).

Conclusion and Actionable Insights

Based on the recommendations:

- **Business Insights**: You can generate insights about customer spending behavior, regional preferences, and identify potential customer segments that could benefit from targeted promotions.
- **Lookalike Analysis**: By using this model, businesses can identify customers who are most likely to respond to similar offers or marketing strategies.