ONLINE RETAIL RECOMMENDATION SYSTEM

1. INTRODUCTION:

In the era of digital commerce, online retail recommendation system Play a crucial role in enhancing customer experience and driving sales. These systems analyze customer behavior, purchase history, and product features to provide personalized recommendations. By leveraging data-driven techniques, business can improve customer engagement, increase conversion rates, and optimize inventory management. This document outlines the development of an online retail recommendation system, detailing data preprocessing, existing methodologies, and the implementation of an effective recommendation model.

2. OVERVIEW:

The online retail recommendation system is built using a structured approach that includes data loading, preprocessing, model training, and visualization. The dataset, 'onlineRetail.csv', contains transaction records from e-commerce platform. The workflow consists of the following steps:

- Data Loading & Cleaning: Handling missing values, removing duplicates, and standardizing data formats.
- Exploratory Data Analysis (EDA): Understanding customer purchasing patterns and product popularity.
- Model Selection & Training: Implementing recommendation algorithms such as collaborative filtering and content-based filtering.
- Evaluating & visualization: Assessing model performance and visualizing recommendation insights.

3. EXISTING METHOD:

Several recommendation techniques have been used in online retail systems, including:

Collaborative Filtering: Suggests items based on user interactions and preferences. Methods include:

- User-based collaborative filtering
- Item-based collaborative filtering

Content-Based Filtering: Recommends items based on product attributes and user profiles.

Hybrid Methods: Combines collaborative and content-based approaches for improved accuracy.

4. PRESENT WORK:

This project implements a recommendation system based on collaborative filtering. The key contributions include:

- Data Preprocessing: Cleaning and structuring the ‘OnlineRetail.csv’ dataset to handle inconsistencies.
- Recommendation Model: Implementing user-based and item-based collaborative filtering techniques.
- Model Training & Evaluation: Training the recommendation model and assessing its accuracy using metrics like RMSE and precision-recall.
- Visualization & Insights: Plotting recommendation distributions, customer segmentation, and sales trends to enhance interpretability.

This work aims to provide an efficient and scalable recommendation system for online retail platforms, improving user satisfaction and business growth.

5. HARDWARE / SOFTWARE TOOLS

REQUIREMENT SPECIFICATIONS (S/W & H/W)

Hardware Requirements:

System : Pentium 4, Intel Core i3, i5, i7 and 2GHz Minium

RAM : 4GB or above

Hard Disk : 10 GB or above

Input : Keyboard and Mouse

Output : Monitor or PC

Software Requirements:

OS : Windows or Higher Versions

Platform : Jupiter Notebook

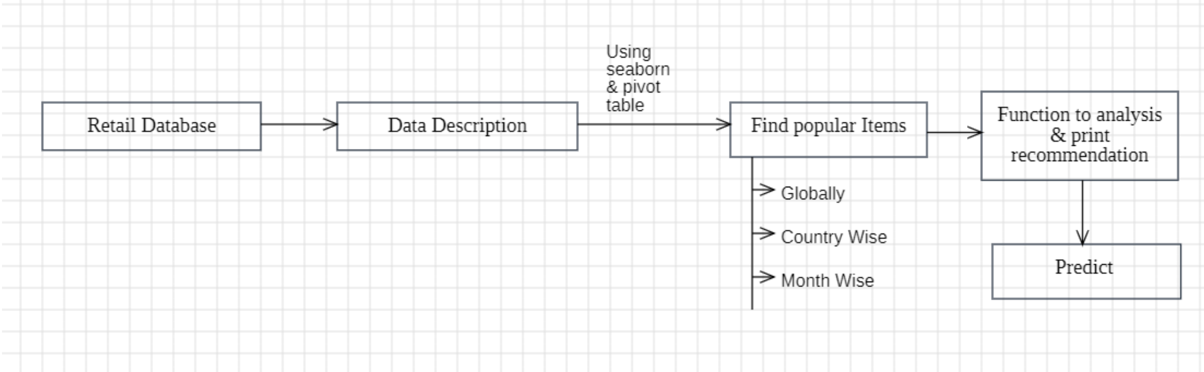
Programming Language : Python

Libraries : NumPy, Pandas, Scikit-Learn, Matplotlib, Seaborn

6. PROPOSED METHOD

The proposed method Leverages collaborative filtering to generate personalized recommendations for online retail customers. This involves analysing purchase history and user behaviour to predict future purchases.

ARCHITECTURE DIAGRAM:



1. **Retail Database**
   - Acts as the Primary data source
   - Contains transactional data
   - Could be a CSV, SQL database or other structured data source
2. **Data Description**
   - Performs basic Exploratory Data Analysis (EDA)
   - Summarizes the dataset using
     - Descriptive Statistics
     - Datatypes, null values etc.
     - Initial visualization
3. **Find popular Items**
   - Conducted using Pivot tables for summarization and seaborn for visualization
   - Items are categorized
     - Globally
     - Country-Wise
     - Month-Wise
4. **Recommendation Engine**
   - A Custom function that analyses identified trends, print recommendation for business decision (example stocking, marketing)
5. **Can Output Insight like**
   - Top products focus on
   - Seasonal trends
   - Country specific Favourities

7. DATASET

Providing a dataset from Kaggle, which contains historical information about Online Retail data which can be used to detect which product is highly recommended. The dataset used in this project is OnlineRetail.csv, which contains:

- Invoice Number : This is the number that identifies a transaction.
- Stock Code : This refers to the product ID.
- Description : This describes the product that a user purchased.
- Quantity : It specified the quantity of the item purchased.
- Invoice Date : The date on which the transaction took place.
- Unit Price : Price of one product.
- Customer ID : It identifies the customer.

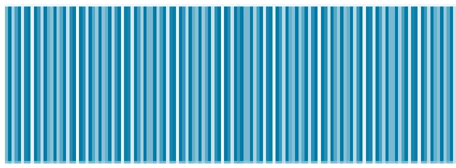
  Country : The country where the transaction was performed.

	A	B	C	D	E	F	G	H
1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
2	536365	85123A	WHITE HANGING HEART T-LIGHT HOLD	6	12/1/2010 8:26	2.55	17850	United Kingdom
3	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
4	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
5	536365	84029G	KNITTED UNION FLAG HOT WATER BOT	6	12/1/2010 8:26	3.39	17850	United Kingdom
6	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
7	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
8	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDE	6	12/1/2010 8:26	4.25	17850	United Kingdom
9	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
10	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
11	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom
12	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom
13	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom
14	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DC	8	12/1/2010 8:34	3.75	13047	United Kingdom
15	536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047	United Kingdom
16	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPC	6	12/1/2010 8:34	4.25	13047	United Kingdom
17	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047	United Kingdom
18	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047	United Kingdom
19	536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
20	536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
21	536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010 8:34	7.95	13047	United Kingdom
22	536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010 8:34	7.95	13047	United Kingdom
23	536368	22960	JAM MAKING SET WITH JARS	6	12/1/2010 8:34	4.25	13047	United Kingdom

Fig 1. Attributes and labels of the Online Retail Recommendation System

index

Index



0 542k

InvoiceNo

Invoice No

25900
unique values

StockCode

Stock Code

4070
unique values

Description

Description

4224
unique values

Quantity

Quantity

-80995 81.0k

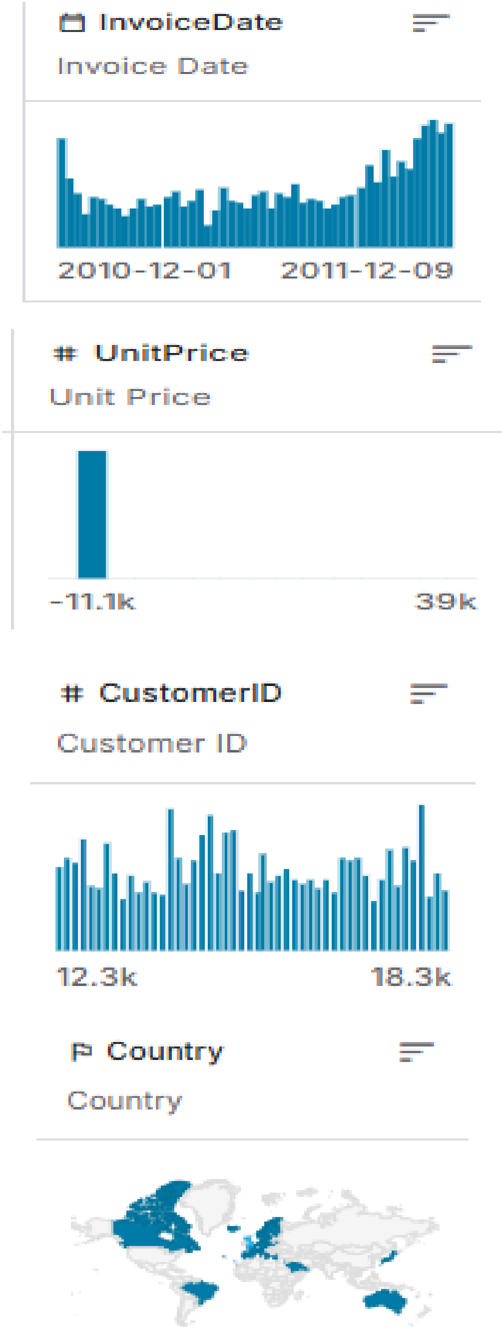


Fig 2. Visualizing all attributes

8. METHODOLOGY

Data Preprocessing:

- Load the dataset and handle missing values.
- Remove duplicate records and correct inconsistencies.
- Convert categorical variables to numerical representations where necessary.
- Normalize data to ensure consistency.

Exploratory Data Analysis (EDA):

- Analyse purchase trends and customer segmentation.
- Visualize product popularity and sales distributions.

Model Implementation:

- Apply user-based and item-based collaborative filtering techniques.
- Utilize similarity metrics such as cosine similarity and Pearson correlation.

Training & Evaluation:

- Train the recommendation model using historical transaction data.
- Evaluate the model using RMSE, precision-recall, and other relevant metrics.

Visualization & Insights:

- Generate recommendation visualizations.
- Display customer purchase trends and segment behaviour insights.

The proposed system aims to enhance user experience by delivering relevant product recommendations based on purchasing behaviour, ultimately improving sales and customer engagement.

9. CODE

STEP-1:

Loading and cleaning the dataset for better result

File: Data Loading & Cleaning File.ipynb

Purpose: Loads and cleans the raw dataset (OnlineRetail.csv).

- Handles missing values
- Removes duplicates
- Filters invalid transactions
- Prepares the data for modelling

```
#Step 1: Importing necessary libraries
import pandas as pd

#Step 2: Load the dataset into jupyter notebook
data = pd.read_csv('OnlineRetail.csv')

#Step 3: Viewing 5 rows of data
data.head(5)
```

[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-1-2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-1-2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-1-2010 8:26	3.39	17850.0	United Kingdom

```
#Step 4: getting the dataset information and the total count
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

#Step 5: Checking for empty columns in dataset
data.isnull().sum()

```
InvoiceNo            0
StockCode            0
Description       1454
Quantity             0
InvoiceDate          0
UnitPrice            0
CustomerID      135080
Country              0
dtype: int64
```

#Step 6: Dropping the empty columns in dataset
data.dropna(inplace=True)

#Step 7: Dropping the duplicates values from dataset
data.drop_duplicates(inplace=True)

data.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  object
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
```

#Step 8: displaying the unique product and customer count
#Here we use "stock code" for unique products and "CustomerID" for unique customers.
print(f'Number of unique products : {data['StockCode'].nunique()}')
print(f'Number of unique customers : {data['CustomerID'].nunique()}')

```
Number of unique products : 3684
Number of unique customers : 4372
```

```
#Displaying the top 10 products from the dataset
top_products = data['Description'].value_counts().head(10)
print (top_products)
```

```
Description
WHITE HANGING HEART T-LIGHT HOLDER    2070
REGENCY CAKESTAND 3 TIER              1905
JUMBO BAG RED RETROSPOT               1662
ASSORTED COLOUR BIRD ORNAMENT         1418
PARTY BUNTING                         1416
LUNCH BAG RED RETROSPOT               1358
SET OF 3 CAKE TINS PANTRY DESIGN      1232
POSTAGE                               1196
LUNCH BAG  BLACK SKULL.               1126
PACK OF 72 RETROSPOT CAKE CASES       1080
Name: count, dtype: int64
```

STEP-2:

Let's visualize using plot for better analyzing Exploratory Data Analysis

File: Plotted Data File.ipynb

Purpose: Visualizes cleaned data to understand customer behaviour and product patterns.

- Time-based plots
- Top products
- Top customers

```
#Step 1: Importing necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt


#Step 2: Load the dataset into jupyter notebook

data = pd.read_csv('OnlineRetail.csv')


#Step 3: Viewing 5 rows of data

data.head(5)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

#Step 4: Creating and displaying the pivot table to find the total quantity of each product bought by each customer

total_product_quantity = data.pivot_table(

index = 'CustomerID',

columns = 'Description',

values = 'Quantity',

aggfunc = 'sum',

fill_value = 0

)

display(total_product_quantity)

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 DAISY PEGS IN WOOD BOX	12 EGG HOUSE PAINTED WOOD	12 HANGING EGGS HAND PAINTED	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	...	ZINC STAR T-LIGHT HOLDER	ZINC SWEETHEART SOAP DISH	ZI SWEETHEA WIRE LETT RA
CustomerID														
12346.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12347.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12348.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12349.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12350.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18280.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18281.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18282.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18283.0	46	0	0	0	0	0	0	1	0	1	...	0	0	
18287.0	0	0	0	0	0	0	0	0	0	0	...	0	0	

4372 rows × 3885 columns

#Plot the distribution of the number of products bought by each customer using histogram

plt.figure(figsize=(10, 6))

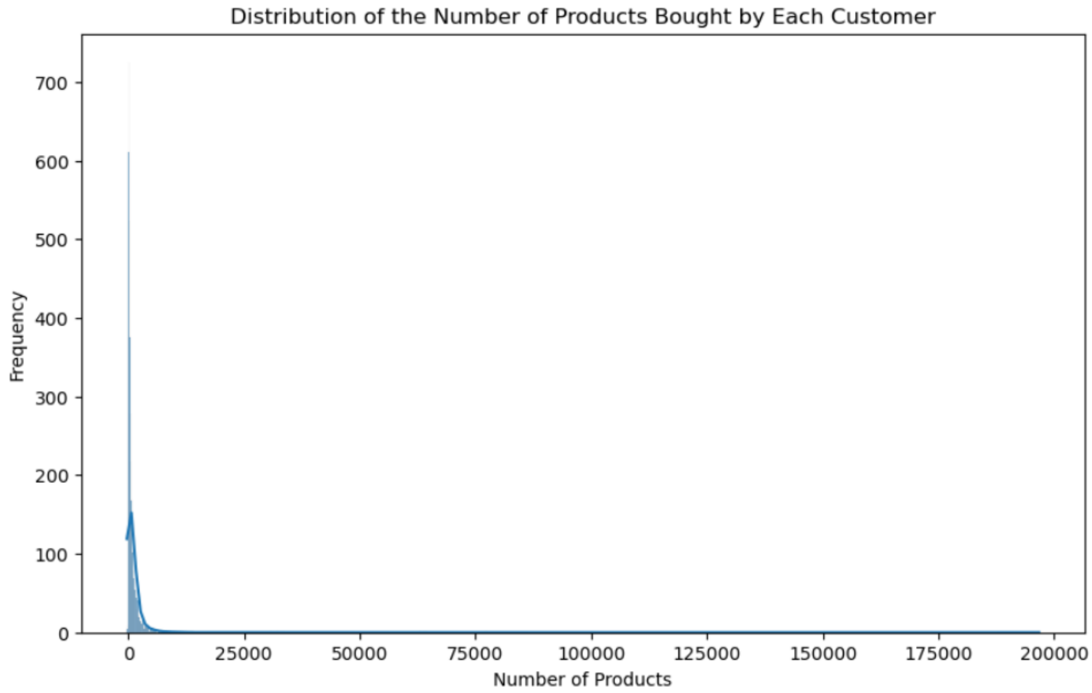
sns.histplot(data=total_product_quantity.sum(axis=1), kde=True)

plt.title('Distribution of the Number of Products Bought by Each Customer')

plt.xlabel('Number of Products')

plt.ylabel('Frequency')

plt.show()



#Step 5: Identifying Globally Popular Products

globally_popular_products = data['Description'].value_counts().head(10)

print("Globally Popular Products:")

print(globally_popular_products)

```
Globally Popular Products:
Description
WHITE HANGING HEART T-LIGHT HOLDER    2369
REGENCY CAKESTAND 3 TIER              2200
JUMBO BAG RED RETROSPOT               2159
PARTY BUNTING                         1727
LUNCH BAG RED RETROSPOT               1638
ASSORTED COLOUR BIRD ORNAMENT         1501
SET OF 3 CAKE TINS PANTRY DESIGN      1473
PACK OF 72 RETROSPOT CAKE CASES       1385
LUNCH BAG  BLACK SKULL.               1350
NATURAL SLATE HEART CHALKBOARD        1280
Name: count, dtype: int64
```

#Plotting globally popular products

plt.figure(figsize=(10, 6))

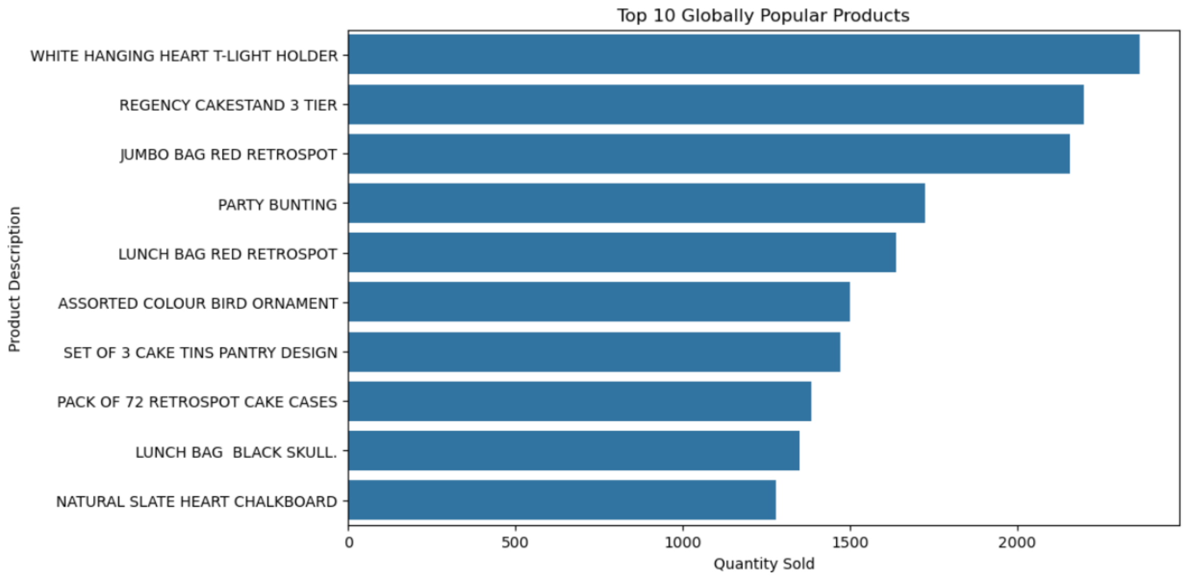
sns.barplot(x=globally_popular_products.values, y=globally_popular_products.index)

plt.title('Top 10 Globally Popular Products')

plt.xlabel('Quantity Sold')

plt.ylabel('Product Description')

plt.show()



#Step 6: Identify Country-wise Popular Products

country_popular_products=data.groupby('Country')['Description'].value_counts().groupby(level=0).nlargest(10).reset_index(level=0, drop=True).reset_index()

print("Country-wise Popular Products:")

print(country_popular_products.head(20))

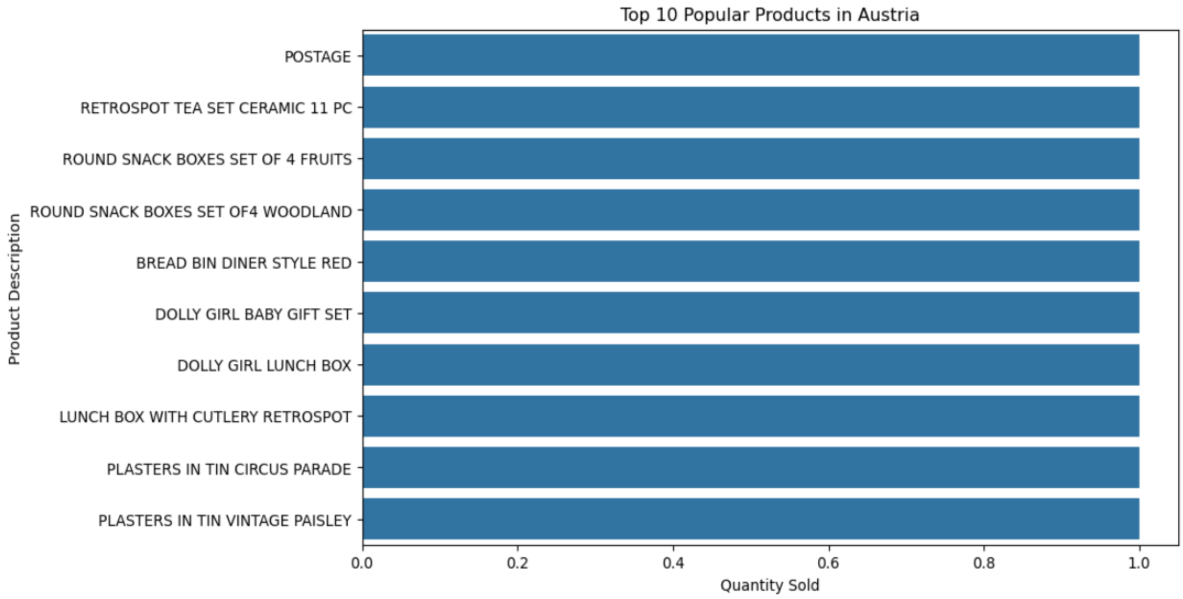
```
Country-wise Popular Products:
      Country                         Description  count
0   Australia     SET OF 3 CAKE TINS PANTRY DESIGN     10
1   Australia              LUNCH BAG RED RETROSPOT      9
2   Australia         RED TOADSTOOL LED NIGHT LIGHT      9
3   Australia          BAKING SET 9 PIECE RETROSPOT      8
4   Australia            BAKING SET SPACEBOY DESIGN      8
5   Australia     HANGING HEART JAR T-LIGHT HOLDER      8
6   Australia             LUNCH BAG SPACEBOY DESIGN      8
7   Australia              PAPER BUNTING RETROSPOT      8
8   Australia                        PARTY BUNTING      8
9   Australia      ROSES REGENCY TEACUP AND SAUCER      8
10    Austria                              POSTAGE     14
11    Austria      RETROSPOT TEA SET CERAMIC 11 PC      4
12    Austria    ROUND SNACK BOXES SET OF 4 FRUITS      4
13    Austria   ROUND SNACK BOXES SET OF4 WOODLAND      4
14    Austria             BREAD BIN DINER STYLE RED      3
15    Austria              DOLLY GIRL BABY GIFT SET      3
16    Austria                  DOLLY GIRL LUNCH BOX      3
17    Austria      LUNCH BOX WITH CUTLERY RETROSPOT      3
18    Austria          PLASTERS IN TIN CIRCUS PARADE      3
19    Austria        PLASTERS IN TIN VINTAGE PAISLEY      3
```

```
#Plot country-wise popular products for a specific country
country = input('Enter country name: ').capitalize()  # enter country from dataset
if country not in data['Country'].unique():
   print( f"Country name '{country}' not found in the data. check the country name.")
else:
   popular_products = country_popular_products[country_popular_products['Country'] == country]
   plt.figure(figsize=(10, 6))
   sns.barplot(x=popular_products['Description'].value_counts().values,
            y=popular_products['Description'].value_counts().index )
   plt.title(f'Top 10 Popular Products in {country}')
   plt.xlabel('Quantity Sold')
   plt.ylabel('Product Description')
   plt.show()
```

```
Enter country name:  mumbai
Country name 'Mumbai' not found in the data. check the country name.
```


```
#Plot country-wise popular products for a specific country
country = input('Enter country name: ').capitalize()  # enter country from dataset
if country not in data['Country'].unique():
   print( f"Country name '{country}' not found in the data. check the country name.")
else:
   popular_products = country_popular_products[country_popular_products['Country'] == country]
   plt.figure(figsize=(10, 6))
   sns.barplot(x=popular_products['Description'].value_counts().values,
            y=popular_products['Description'].value_counts().index )
   plt.title(f'Top 10 Popular Products in {country}')
   plt.xlabel('Quantity Sold')
   plt.ylabel('Product Description')
   plt.show()
```

```
Enter country name:  austria
```



#Step 7: Identifying Month-wise Popular Products

#Convert InvoiceDate to datetime

data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])

#Extract month and year from InvoiceDate

data['MonthYear'] = data['InvoiceDate'].dt.to_period('M')

month_popular_products = data.groupby('MonthYear')['Description'].value_counts().groupby(level=0).nlargest(10).reset_index(level=0, drop=True).reset_index()

print("Month-wise Popular Products:")

print(month_popular_products.head(20))

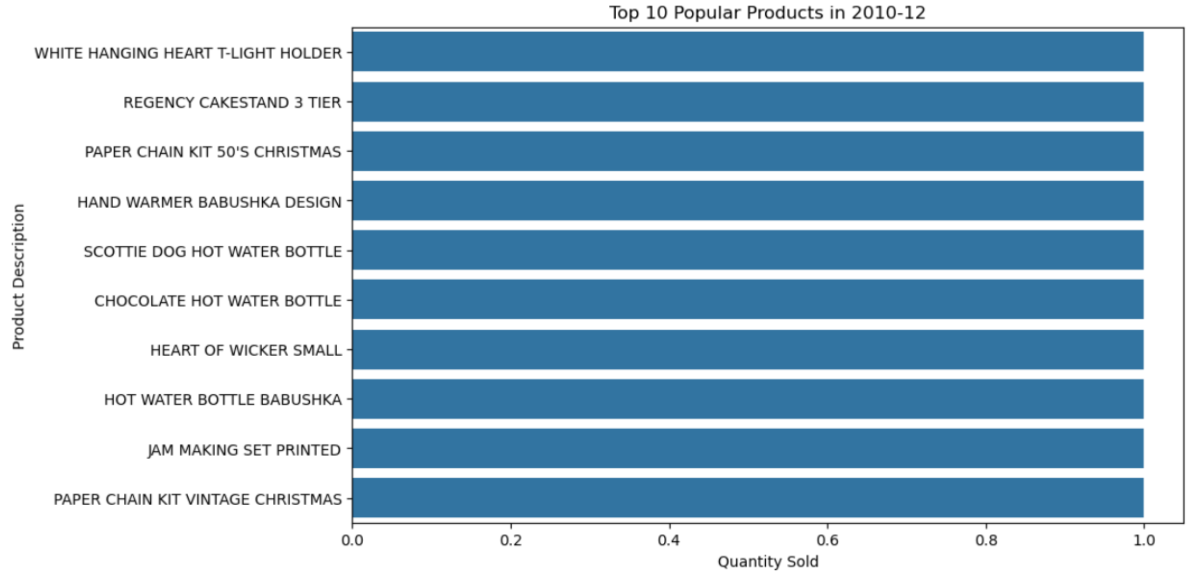
```
Month-wise Popular Products:
   MonthYear                         Description  count
0    2010-12  WHITE HANGING HEART T-LIGHT HOLDER    241
1    2010-12            REGENCY CAKESTAND 3 TIER    191
2    2010-12        PAPER CHAIN KIT 50'S CHRISTMAS    175
3    2010-12         HAND WARMER BABUSHKA DESIGN    170
4    2010-12          SCOTTIE DOG HOT WATER BOTTLE    162
5    2010-12           CHOCOLATE HOT WATER BOTTLE    155
6    2010-12               HEART OF WICKER SMALL    144
7    2010-12             HOT WATER BOTTLE BABUSHKA    142
8    2010-12              JAM MAKING SET PRINTED    141
9    2010-12     PAPER CHAIN KIT VINTAGE CHRISTMAS    139
10   2011-01  WHITE HANGING HEART T-LIGHT HOLDER    185
11   2011-01            REGENCY CAKESTAND 3 TIER    156
12   2011-01     SET OF 3 CAKE TINS PANTRY DESIGN    156
13   2011-01               HEART OF WICKER SMALL    145
14   2011-01        NATURAL SLATE HEART CHALKBOARD    126
15   2011-01             JUMBO BAG RED RETROSPOT    122
16   2011-01          SET OF 3 HEART COOKIE CUTTERS    110
17   2011-01            JAM MAKING SET WITH JARS    109
18   2011-01     SET OF 6 SPICE TINS PANTRY DESIGN    103
19   2011-01               HEART OF WICKER LARGE    100
```

```
#Plot month-wise popular products for a specific month
specific_month = input('Enter year and month (yyyy-mm): ')
if specific_month not in data['MonthYear'].unique():
        print( f"Year and month '{specific_month}' not found in the data. check the year and month.")
else:
    monthly_popular_products = month_popular_products[month_popular_products['MonthYear'] == specific_month]
    plt.figure(figsize=(10, 6))
    sns.barplot(x=monthly_popular_products['Description'].value_counts().values,
                y=monthly_popular_products['Description'].value_counts().index)
    plt.title(f'Top 10 Popular Products in {specific_month}')
    plt.xlabel('Quantity Sold')
    plt.ylabel('Product Description')
    plt.show()
```

```
Enter year and month (yyyy-mm):  2024-12
Year and month '2024-12' not found in the data. check the year and month.
```

```
#Plot month-wise popular products for a specific month
specific_month = input('Enter year and month (yyyy-mm): ')
if specific_month not in data['MonthYear'].unique():
        print( f"Year and month not'{specific_month}' found in the data. check the year and month.")
else:
    monthly_popular_products = month_popular_products[month_popular_products['MonthYear'] == specific_month]
    plt.figure(figsize=(10, 6))
    sns.barplot(x=monthly_popular_products['Description'].value_counts().values,
                y=monthly_popular_products['Description'].value_counts().index)
    plt.title(f'Top 10 Popular Products in {specific_month}')
    plt.xlabel('Quantity Sold')
    plt.ylabel('Product Description')
    plt.show()
```

Enter year and month (yyyy-mm): 2010-12



STEP-3: Item-to-Item Product Recommendation by Collaborative Filtering

Load & Prepare the Data

Model Building – Item-to-Item Collaborative Filtering

File: item_to_item_by_collaborative_filtering.ipynb

Purpose:

- Builds a recommender based on product similarity
- Uses co-purchases to recommend similar products to users

```
# Import necessary libraries.
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# Read data source Excel files.
data = pd.read_csv('OnlineRetail.csv')

# Check dataframe information.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Read header of dataframe.

data.head()

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-1-2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-1-2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-1-2010 8:26	3.39	17850.0	United Kingdom

Check any column containing the null value.

data.isnull().any()

```
InvoiceNo      False
StockCode      False
Description     True
Quantity       False
InvoiceDate    False
UnitPrice      False
CustomerID      True
Country        False
dtype: bool
```

Count the number of null value records in the CustomerID column.

data['CustomerID'].isna().sum()

```
np.int64(135080)
```

dataa = data.dropna(subset=['CustomerID'])

Check dataframe information.

dataa.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    406829 non-null  object 
 1   StockCode    406829 non-null  object 
 2   Description  406829 non-null  object 
 3   Quantity     406829 non-null  int64  
 4   InvoiceDate  406829 non-null  object 
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
```

Read header of dataframe.

dataa.head()

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-1-2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-1-2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-1-2010 8:26	3.39	17850.0	United Kingdom

Create CustomerID vs Item (Purchased Items, by StockCode) matrix by pivot table function.

```
CustomerID_Item_matrix = dataa.pivot_table(
    index='CustomerID',
    columns='StockCode',
    values='Quantity',
    aggfunc='sum'
)
```

Display the shape of matrix, 4372 rows of CustomerID, 3684 columns of Item.

CustomerID_Item_matrix.shape

```
(4372, 3684)
```

Update illustration of the matrix, 1 to represent customer have purchased item, 0 to represent customer haven't purchased.

CustomerID_Item_matrix = CustomerID_Item_matrix.applymap(lambda x: 1 if x > 0 else 0)

Read header of CustomerID vs Item matrix.

CustomerID_Item_matrix.loc[12680:].head()

StockCode	10002	10080	10120	10125	10133	10135	11001	15030	15034	15036	...	90214Y	90214Z	BANK CHARGES	C2	CRUK	D	DOT	M	PADS	POST
CustomerID																					
12680.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12681.0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12682.0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12683.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12684.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	1

5 rows × 3684 columns

Calculate the Item to Item similarity by Cosine Similarity

Create Item to Item similarity matrix.

```
item_item_similarity_matrix = pd.DataFrame(
    cosine_similarity(CustomerID_Item_matrix.T)
)
```

Display header of Item to Item similarity matrix.

item_item_similarity_matrix.head()

	0	1	2	3	4	5	6	7	8	9	...	3674	3675	3676	3677	3678	3679	3680	3681	3682	368
0	1.000000	0.0	0.094868	0.091287	0.0	0.000000	0.090351	0.063246	0.098907	0.095346	...	0.0	0.0	0.0	0.029361	0.0	0.0	0.0	0.059423	0.0	0.07005
1	0.000000	1.0	0.000000	0.000000	0.0	0.000000	0.032774	0.045883	0.047836	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.017244	0.0	0.00000
2	0.094868	0.0	1.000000	0.115470	0.0	0.000000	0.057143	0.060000	0.041703	0.060302	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.075165	0.0	0.00000
3	0.091287	0.0	0.115470	1.000000	0.0	0.000000	0.164957	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.00000
4	0.000000	0.0	0.000000	0.000000	1.0	0.447214	0.063888	0.044721	0.000000	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.00000

5 rows × 3684 columns

Update index to corresponding Item Code (StockCode).

item_item_similarity_matrix.columns = CustomerID_Item_matrix.T.index

item_item_similarity_matrix['StockCode'] = CustomerID_Item_matrix.T.index

item_item_similarity_matrix = item_item_similarity_matrix.set_index('StockCode')

Display header of Item to Item similarity matrix.

item_item_similarity_matrix.head()

StockCode	10002	10080	10120	10123C	10124A	10124G	10125	10133	10135	11001	...	90214Y	90214Z	BANK CHARGES	C2	CRUK	D	DOT
StockCode																		
10002	1.000000	0.0	0.094868	0.091287	0.0	0.000000	0.090351	0.063246	0.098907	0.095346	...	0.0	0.0	0.0	0.029361	0.0	0.0	0.0
10080	0.000000	1.0	0.000000	0.000000	0.0	0.000000	0.032774	0.045883	0.047836	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0
10120	0.094868	0.0	1.000000	0.115470	0.0	0.000000	0.057143	0.060000	0.041703	0.060302	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0
10123C	0.091287	0.0	0.115470	1.000000	0.0	0.000000	0.164957	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0
10124A	0.000000	0.0	0.000000	0.000000	1.0	0.447214	0.063888	0.044721	0.000000	0.000000	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0

5 rows × 3684 columns

Randomly pick StockCode (23166) to display the most similar StockCode.

```
top_10_similar_items = list(
    item_item_similarity_matrix\
        .loc[23166]\
        .sort_values(ascending=False)\
        .iloc[:10]\
    .index
```

Display top 10 similar items of StockCode (23166).

```
top_10_similar_items
```

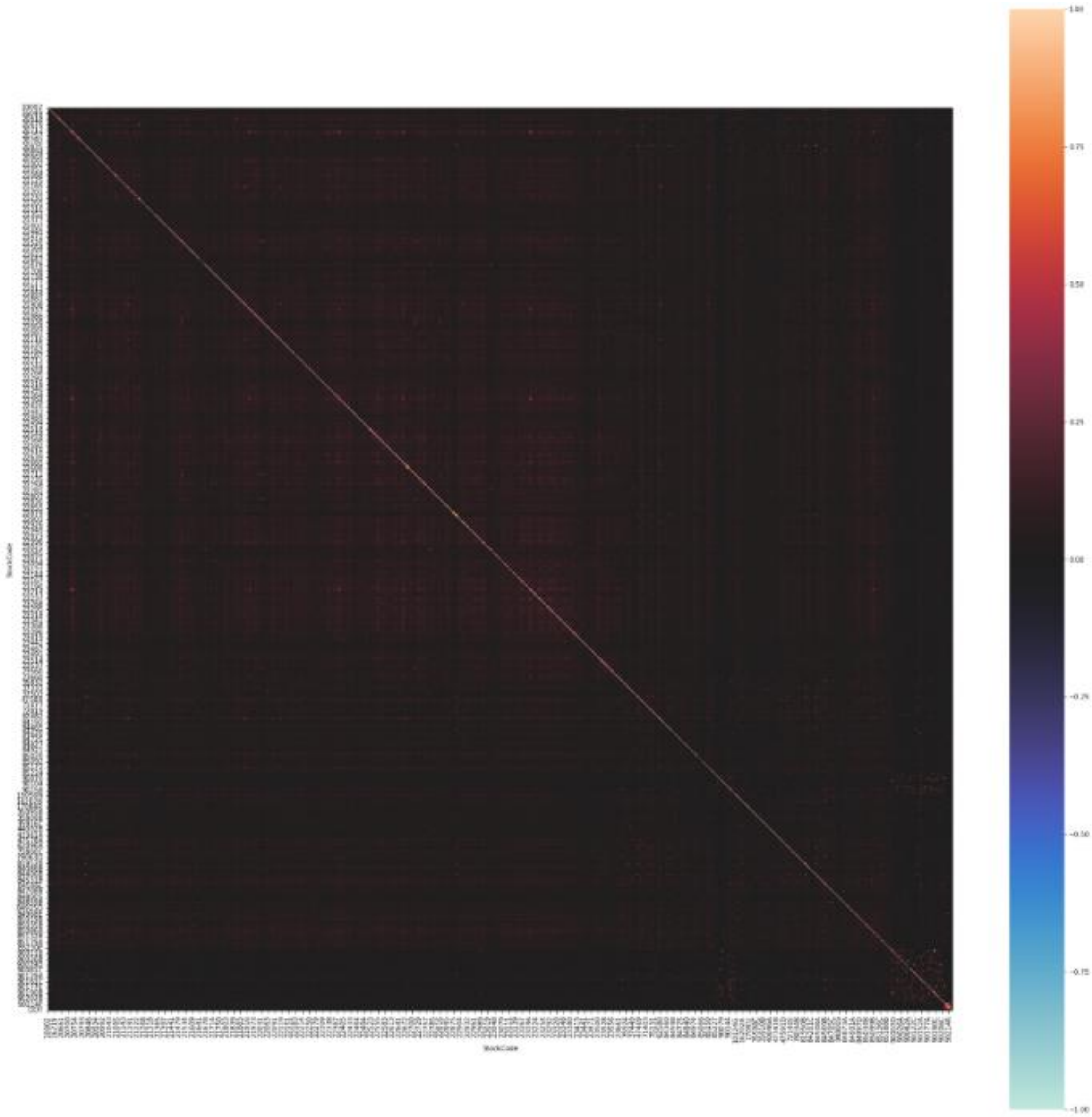
```
[23166, 23165, 23167, 22993, 23307, 22722, 23243, 22666, 22720, 22961]
```

Display the list of similar items of StockCode (23166) with item Description.

```
df1a.loc[
    df1a['StockCode'].isin(top_10_similar_items),
    ['StockCode', 'Description']
].drop_duplicates().set_index('StockCode').loc[top_10_similar_items]
```

	Description
StockCode	
23166	MEDIUM CERAMIC TOP STORAGE JAR
23165	LARGE CERAMIC TOP STORAGE JAR
23167	SMALL CERAMIC TOP STORAGE JAR
22993	SET OF 4 PANTRY JELLY MOULDS
23307	SET OF 60 PANTRY DESIGN CAKE CASES
22722	SET OF 6 SPICE TINS PANTRY DESIGN
23243	SET OF TEA COFFEE SUGAR TINS PANTRY
22666	RECIPE BOX PANTRY YELLOW DESIGN
22720	SET OF 3 CAKE TINS PANTRY DESIGN
22961	JAM MAKING SET PRINTED

```
# And optionally, use the heatmap to display the Item to Item similarity matrix.
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (30,30))
ax = sns.heatmap(
    item_item_similarity_matrix,
    vmin=-1, vmax=1, center=0,
    square=True)
```



STEP-4: User-to-User Product Recommendation by Collaborative Filtering

Load & Prepare the Data

Model Building-User-to-User Collaborative Filtering

File: user_to_user_by_collaborative_filtering.ipynb

Purpose:

- Recommends items by finding users with similar buying patterns
- Based on user-user cosine similarity

Import necessary libraries.

import pandas as pd

from sklearn.metrics.pairwise import cosine_similarity

#Step 2: Load the dataset into jupyter notebook

df1 = pd.read_csv('OnlineRetail.csv')

Check dataframe information.

df1.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

Read header of dataframe.

df1.head()

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Check any column containing the null value.

df1.isnull().any()

```
InvoiceNo      False
Collapse Output False
Description     True
Quantity       False
InvoiceDate    False
UnitPrice      False
CustomerID      True
Country        False
dtype: bool
```

Count the number of null value records in the CustomerID column.

df1['CustomerID'].isna().sum()

```
135080
```

df1a = df1.dropna(subset=['CustomerID'])

Check dataframe information.

df1a.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  datetime64[ns]
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 27.9+ MB
```

Read header of dataframe.

df1a.head()

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
# Create CustomerID vs Item (Purchased Items, by StockCode) matrix by pivot table function.
CustomerID_Item_matrix = df1a.pivot_table(
    index='CustomerID',
    columns='StockCode',
    values='Quantity',
    aggfunc='sum'
)
# Display the shape of matrix, 4372 rows of CustomerID, 3684 columns of Item.
CustomerID_Item_matrix.shape
```

```
(4372, 3684)
```

```
# Update illustration of the matrix, 1 to represent customer have purchased item, 0 to represent customer haven't purchased.
CustomerID_Item_matrix = CustomerID_Item_matrix.applymap(lambda x: 1 if x > 0 else 0)
# Read header of CustomerID vs Item matrix.
CustomerID_Item_matrix.loc[12680:].head()
```

StockCode	10002	10080	10120	10125	10133	10135	11001	15030	15034	15036	...	90214Y	90214Z	BANK CHARGES	C2	CRUK	D	DOT	M	PADS	POST
CustomerID																					
12680.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12681.0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12682.0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12683.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
12684.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	1

5 rows × 3684 columns

Calculate the User to User similarity by Cosine Similarity

Create User to User similarity matrix.

user_to_user_similarity_matrix = pd.DataFrame(

    cosine_similarity(CustomerID_Item_matrix)

)

Display header of User to User similarity matrix.

user_to_user_similarity_matrix.head()

	0	1	2	3	4	5	6	7	8	9	...	4362	4363	4364	4365	4366	4367	4368	4369	437
0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.0000(
1	0.0	1.000000	0.063022	0.046130	0.047795	0.038814	0.0	0.025876	0.136641	0.094742	...	0.0	0.0	0.054656	0.0	0.032844	0.062318	0.0	0.113776	0.1093
2	0.0	0.063022	1.000000	0.024953	0.051709	0.027995	0.0	0.027995	0.118262	0.146427	...	0.0	0.0	0.118262	0.0	0.000000	0.000000	0.0	0.000000	0.1709(
3	0.0	0.046130	0.024953	1.000000	0.056773	0.138314	0.0	0.030737	0.032461	0.144692	...	0.0	0.0	0.000000	0.0	0.039014	0.000000	0.0	0.067574	0.1371
4	0.0	0.047795	0.051709	0.056773	1.000000	0.031846	0.0	0.000000	0.000000	0.033315	...	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.0448

5 rows × 4372 columns

Update index to corresponding CustomerID.

user_to_user_similarity_matrix.columns = CustomerID_Item_matrix.index

user_to_user_similarity_matrix['CustomerID'] = CustomerID_Item_matrix.index

user_to_user_similarity_matrix = user_to_user_similarity_matrix.set_index('CustomerID')

Display header of User to User similarity matrix.

user_to_user_similarity_matrix.head()

CustomerID	12346.0	12347.0	12348.0	12349.0	12350.0	12352.0	12353.0	12354.0	12355.0	12356.0	...	18273.0	18274.0	18276.0	18277.0	18278.0	182
CustomerID																	
12346.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.0	0.0	0.000000	0.0	0.000000	0.000
12347.0	0.0	1.000000	0.063022	0.046130	0.047795	0.038814	0.0	0.025876	0.136641	0.094742	...	0.0	0.0	0.054656	0.0	0.032844	0.062
12348.0	0.0	0.063022	1.000000	0.024953	0.051709	0.027995	0.0	0.027995	0.118262	0.146427	...	0.0	0.0	0.118262	0.0	0.000000	0.000
12349.0	0.0	0.046130	0.024953	1.000000	0.056773	0.138314	0.0	0.030737	0.032461	0.144692	...	0.0	0.0	0.000000	0.0	0.039014	0.000
12350.0	0.0	0.047795	0.051709	0.056773	1.000000	0.031846	0.0	0.000000	0.000000	0.033315	...	0.0	0.0	0.000000	0.0	0.000000	0.000

5 rows × 4372 columns

Randomly pick CustomerID (12702) to display the most similar CustomerID.

The most similar CustomerID is 14608, which has 51% similarity.

user_to_user_similarity_matrix.loc[12702.0].sort_values(ascending=False)

```
CustomerID
12702.0    1.000000
14608.0    0.510310
15758.0    0.481125
18259.0    0.444444
15434.0    0.427121
             ...
14895.0    0.000000
14896.0    0.000000
14897.0    0.000000
14898.0    0.000000
15301.0    0.000000
Name: 12702.0, Length: 4372, dtype: float64
```

```
# Display CustomerID (12702) purchased items.
items_purchased_by_X = set(CustomerID_Item_matrix.loc[12702.0].iloc[
  CustomerID_Item_matrix.loc[12702.0].to_numpy().nonzero()].index)
items_purchased_by_X
```

```
{21479,
 21481,
 22111,
 22113,
 22114,
 22835,
 23355,
 23356,
 23357,
 23439,
 '84032A',
 'POST'}
```

```
# Display CustomerID (14608) purchased items.
items_purchased_by_Y = set(CustomerID_Item_matrix.loc[14608.0].iloc[
  CustomerID_Item_matrix.loc[14608.0].to_numpy().nonzero()].index)
items_purchased_by_Y
```

```
{21481, 22111, 22112, 22114, 22207, 23355, 23357, '84029E'}
```

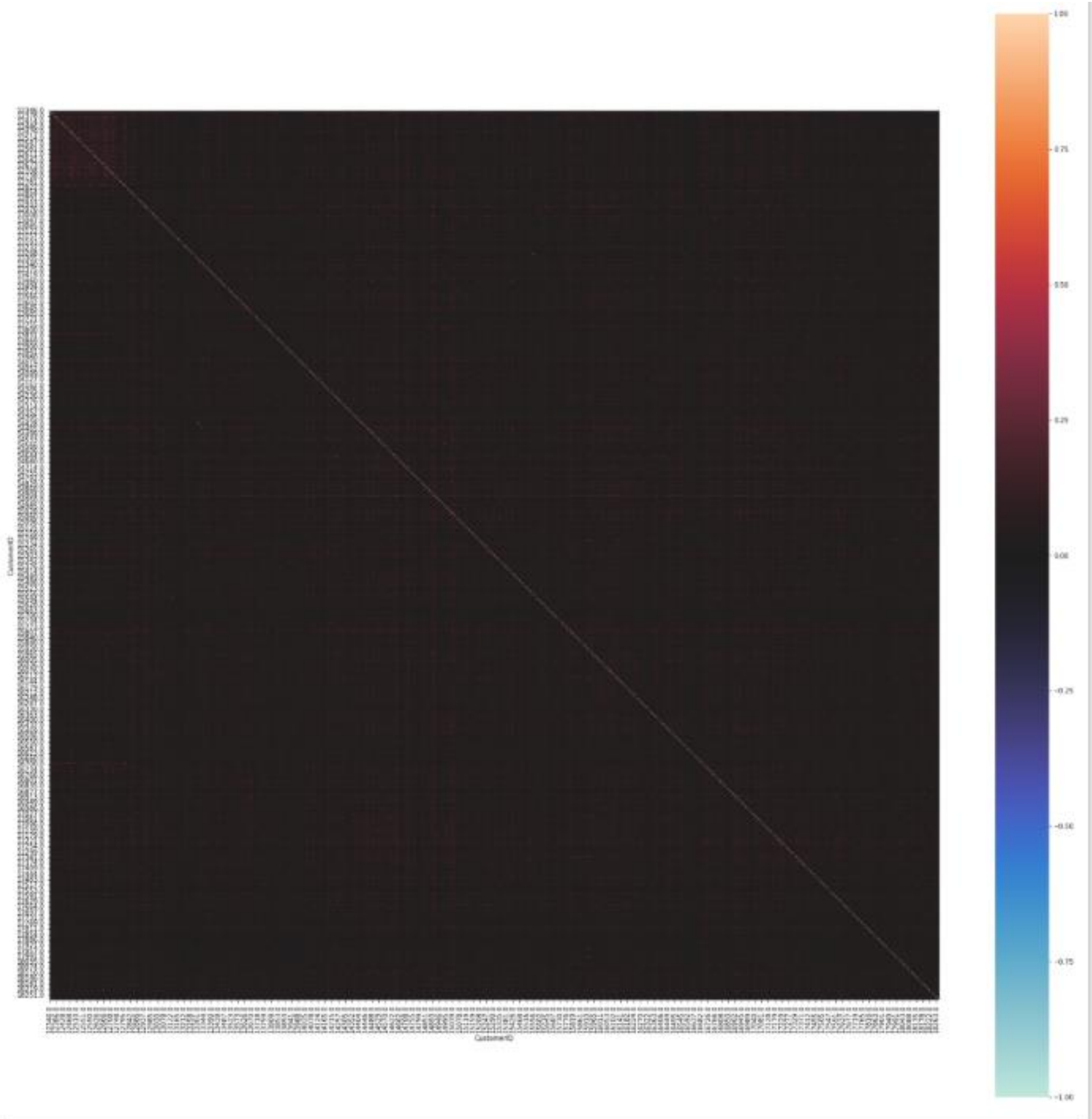
```
# Find out items which purchased by X (12702) but not yet purchased by Y (14608).
items_to_recommend_to_Y = items_purchased_by_X - items_purchased_by_Y
# Display the list of items recommended for Y (14608).
items_to_recommend_to_Y
```

```
{21479, 22113, 22835, 23356, 23439, '84032A', 'POST'}
```

```
# Display the list of items recommended for Y (14608) with item Description.
df1a.loc[
    df1a['StockCode'].isin(items_to_recommend_to_Y),
    ['StockCode', 'Description']
].drop_duplicates().set_index('StockCode')
```

	Description
StockCode	
POST	POSTAGE
22835	HOT WATER BOTTLE I AM SO POORLY
21479	WHITE SKULL HOT WATER BOTTLE
22113	GREY HEART HOT WATER BOTTLE
84032A	CHARLIE+LOLA PINK HOT WATER BOTTLE
23356	LOVE HOT WATER BOTTLE
23439	HAND WARMER RED LOVE HEART

```
# And optionally, use the heatmap to display the User to User similarity matrix.
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (30,30))
ax = sns.heatmap(
    user_to_user_similarity_matrix,
    vmin=-1, vmax=1, center=0,
    square=True)
```



STEP-5: Model Integration

File: Online Retail Recommendation System Complete FIle.ipynb

Purpose:

- Integrates all components (cleaned data, models, recommendations)
- Finalizes logic to recommend products to users
- Possibly includes evaluation and recommendation UI

#Step 1: Importing necessary libraries

import pandas as pd

#Step 2: Load the dataset into jupyter notebook

data = pd.read_csv('OnlineRetail.csv')

#Step 3: Viewing 10 rows of data

data.head(10)

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-1-2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-1-2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-1-2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12-1-2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12-1-2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12-1-2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12-1-2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12-1-2010 8:34	1.69	13047.0	United Kingdom

#Step 4: getting the dataset information and the total count

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

#Step 5: Checking for empty columns in dataset

data.isnull().sum()

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

#Step 6: Dropping the empty columns in dataset

data.dropna(inplace=True)

#Step 7: Dropping the duplicates values from dataset

data.drop_duplicates(inplace=True)

data.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 401604 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    401604 non-null  object
 1   StockCode    401604 non-null  object
 2   Description  401604 non-null  object
 3   Quantity     401604 non-null  int64
 4   InvoiceDate  401604 non-null  object
 5   UnitPrice    401604 non-null  float64
 6   CustomerID   401604 non-null  float64
 7   Country      401604 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.6+ MB
```

#Step 8: displaying the unique product and customer count

#Here we use "stock code" for unique products and "CustomerID" for unique customers.

print(f'Number of unique products : {data['StockCode'].nunique()}')

print(f'Number of unique customers : {data['CustomerID'].nunique()}')

```
Number of unique products : 3684
Number of unique customers : 4372
```

#Displaying the top 10 products from the dataset

top_products = data['Description'].value_counts().head(10)

print (top_products)

```
Description
WHITE HANGING HEART T-LIGHT HOLDER    2058
REGENCY CAKESTAND 3 TIER              1894
JUMBO BAG RED RETROSPOT               1659
PARTY BUNTING                         1409
ASSORTED COLOUR BIRD ORNAMENT         1405
LUNCH BAG RED RETROSPOT               1345
SET OF 3 CAKE TINS PANTRY DESIGN      1224
POSTAGE                               1196
LUNCH BAG  BLACK SKULL.               1099
PACK OF 72 RETROSPOT CAKE CASES       1062
Name: count, dtype: int64
```

Lets visualize using plot for better analyzing

#Step 1: importing necessary library

import seaborn as sns

import matplotlib.pyplot as plt

#Step 2: Creating and displaying the pivot table to find the total quantity of each product bought by each customer

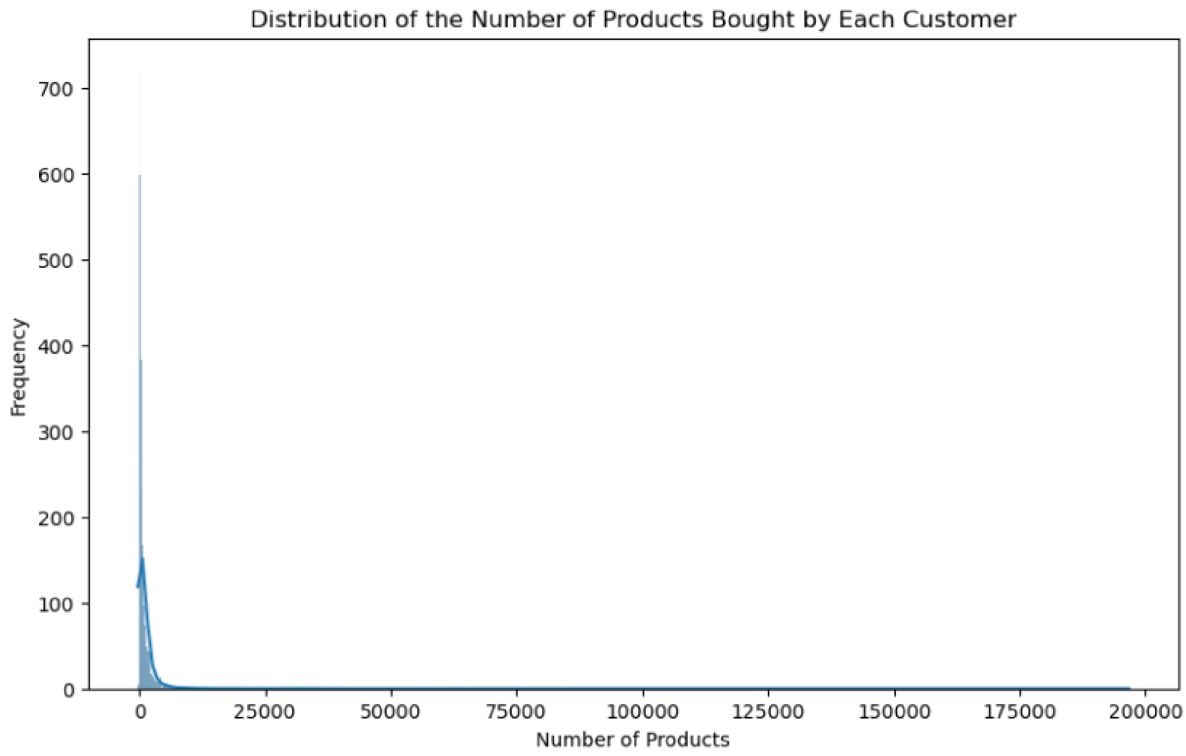
```
total_product_quantity = data.pivot_table(
    index = 'CustomerID',
    columns = 'Description',
    values = 'Quantity',
    aggfunc = 'sum',
    fill_value = 0
)
display(total_product_quantity)
```

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 DAISY PEGS IN WOOD BOX	12 EGG HOUSE PAINTED WOOD	12 HANGING EGGS HAND PAINTED	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	...	ZINC STAR T-LIGHT HOLDER	ZINC SWEETHEART SOAP DISH	ZI SWEETHEA WIRE LETT RA
CustomerID														
12346.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12347.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12348.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12349.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
12350.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18280.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18281.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18282.0	0	0	0	0	0	0	0	0	0	0	...	0	0	
18283.0	46	0	0	0	0	0	0	1	0	1	...	0	0	
18287.0	0	0	0	0	0	0	0	0	0	0	...	0	0	

4372 rows × 3885 columns

#Plot the distribution of the number of products bought by each customer using histogram

```
plt.figure(figsize=(10, 6))
sns.histplot(data=total_product_quantity.sum(axis=1), kde=True)
plt.title('Distribution of the Number of Products Bought by Each Customer')
plt.xlabel('Number of Products')
plt.ylabel('Frequency')
plt.show()
```



#Step 3: Identifying Globally Popular Products

globally_popular_products = data['Description'].value_counts().head(10)

print("Globally Popular Products:")

print(globally_popular_products)

```
Globally Popular Products:
Description
WHITE HANGING HEART T-LIGHT HOLDER    2058
REGENCY CAKESTAND 3 TIER              1894
JUMBO BAG RED RETROSPOT               1659
PARTY BUNTING                         1409
ASSORTED COLOUR BIRD ORNAMENT         1405
LUNCH BAG RED RETROSPOT               1345
SET OF 3 CAKE TINS PANTRY DESIGN      1224
POSTAGE                               1196
LUNCH BAG  BLACK SKULL.               1099
PACK OF 72 RETROSPOT CAKE CASES       1062
Name: count, dtype: int64
```

#Plotting globally popular products

plt.figure(figsize=(10, 6))

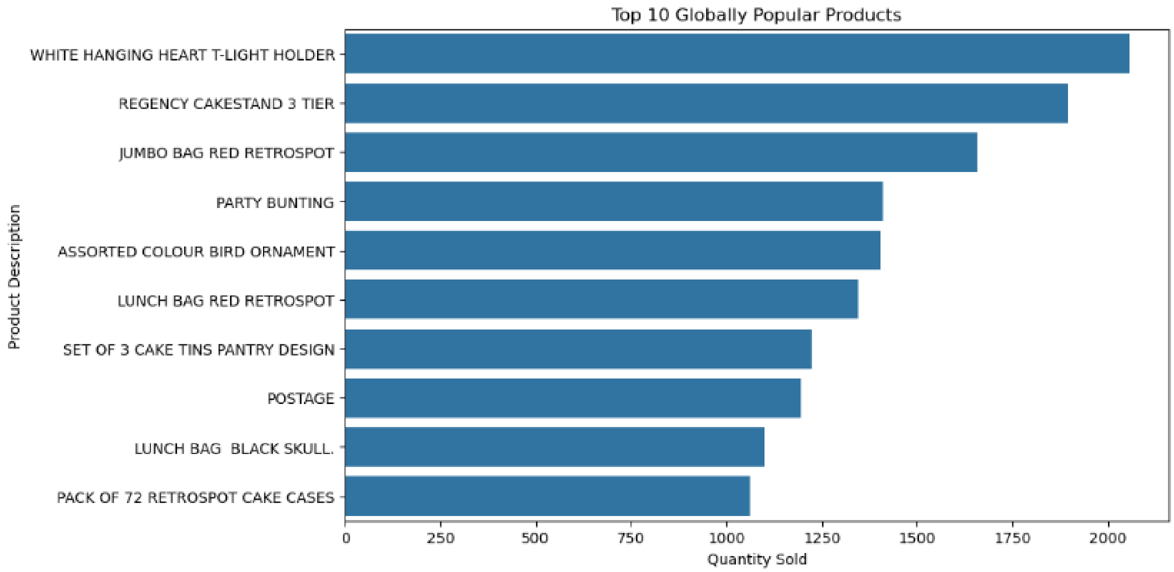
sns.barplot(x=globally_popular_products.values, y=globally_popular_products.index)

plt.title('Top 10 Globally Popular Products')

plt.xlabel('Quantity Sold')

plt.ylabel('Product Description')

plt.show()



#Step 4: Identify Country-wise Popular Products

country_popular_products = data.groupby('Country')['Description'].value_counts().groupby(level=0).nlargest(10).reset_index(level=0, drop=True).reset_index()

print("Country-wise Popular Products:")

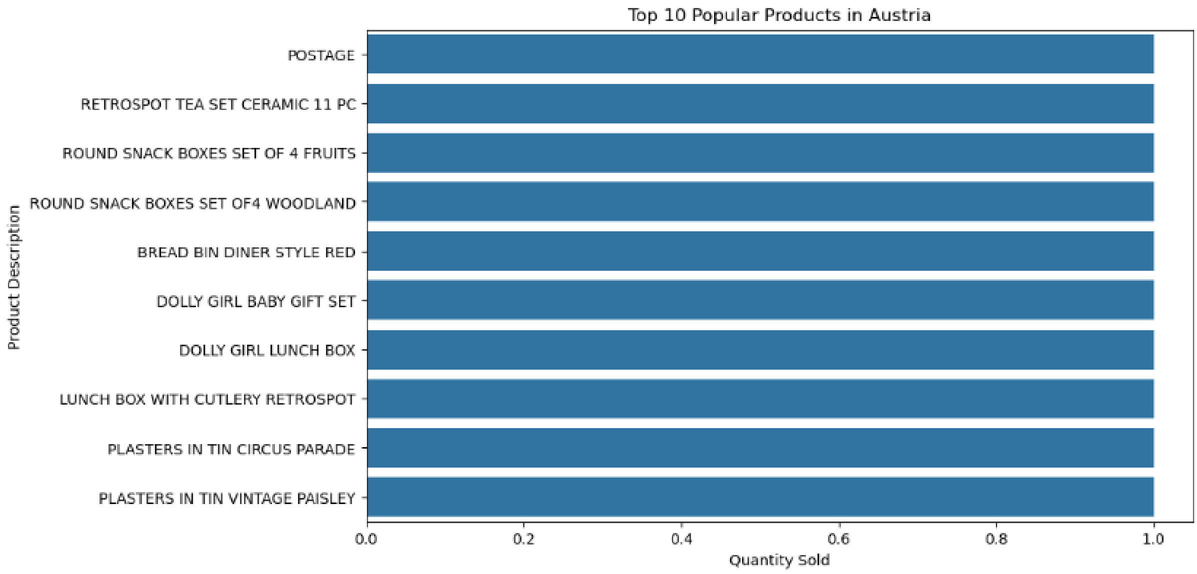
print(country_popular_products.head(20))

```
Country-wise Popular Products:
      Country                         Description  count
0   Australia    SET OF 3 CAKE TINS PANTRY DESIGN     10
1   Australia             LUNCH BAG RED RETROSPOT      9
2   Australia        RED TOADSTOOL LED NIGHT LIGHT      9
3   Australia         BAKING SET 9 PIECE RETROSPOT      8
4   Australia          BAKING SET SPACEBOY DESIGN      8
5   Australia    HANGING HEART JAR T-LIGHT HOLDER      8
6   Australia            LUNCH BAG SPACEBOY DESIGN      8
7   Australia             PAPER BUNTING RETROSPOT      8
8   Australia                       PARTY BUNTING      8
9   Australia     ROSES REGENCY TEACUP AND SAUCER      8
10    Austria                            POSTAGE     14
11    Austria      RETROSPOT TEA SET CERAMIC 11 PC      4
12    Austria     ROUND SNACK BOXES SET OF 4 FRUITS      4
13    Austria    ROUND SNACK BOXES SET OF4 WOODLAND      4
14    Austria              BREAD BIN DINER STYLE RED      3
15    Austria               DOLLY GIRL BABY GIFT SET      3
16    Austria                   DOLLY GIRL LUNCH BOX      3
17    Austria     LUNCH BOX WITH CUTLERY RETROSPOT      3
18    Austria          PLASTERS IN TIN CIRCUS PARADE      3
19    Austria      PLASTERS IN TIN VINTAGE PAISLEY      3
```

```
#Plot country-wise popular products for a specific country
country = input('Enter country name: ').capitalize()  # enter country from dataset
if country not in data['Country'].unique():
        print( f"Country name '{country}' not found in the data. check the country name.")

popular_products = country_popular_products[country_popular_products['Country'] == country]
plt.figure(figsize=(10, 6))
sns.barplot(x=popular_products['Description'].value_counts().values,
        y=popular_products['Description'].value_counts().index )
plt.title(f'Top 10 Popular Products in {country}')
plt.xlabel('Quantity Sold')
plt.ylabel('Product Description')
plt.show()
```

```
Enter country name:  austria
```




```
#Step 5: Identifying Month-wise Popular Products

#Convert InvoiceDate to datetime
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])
#Extract month and year from InvoiceDate
data['MonthYear'] = data['InvoiceDate'].dt.to_period('M')
```

month_popular_products=data.groupby('MonthYear')['Description'].value_counts().groupby(level=0).nlargest(10).reset_index(level=0, drop=True).reset_index()

print("Month-wise Popular Products:")

print(month_popular_products.head(20))

```
Month-wise Popular Products:
   MonthYear                          Description  count
0    2010-12   WHITE HANGING HEART T-LIGHT HOLDER    213
1    2010-12             REGENCY CAKESTAND 3 TIER    153
2    2010-12          HAND WARMER BABUSHKA DESIGN    141
3    2010-12        PAPER CHAIN KIT 50'S CHRISTMAS    137
4    2010-12          SCOTTIE DOG HOT WATER BOTTLE    130
5    2010-12           CHOCOLATE HOT WATER BOTTLE    123
6    2010-12                HEART OF WICKER SMALL    107
7    2010-12             HOT WATER BOTTLE BABUSHKA    107
8    2010-12               JAM MAKING SET PRINTED    107
9    2010-12   PAPER CHAIN KIT VINTAGE CHRISTMAS    106
10   2011-01   WHITE HANGING HEART T-LIGHT HOLDER    164
11   2011-01      SET OF 3 CAKE TINS PANTRY DESIGN    137
12   2011-01             REGENCY CAKESTAND 3 TIER    132
13   2011-01                HEART OF WICKER SMALL    119
14   2011-01         NATURAL SLATE HEART CHALKBOARD     96
15   2011-01        SET OF 3 HEART COOKIE CUTTERS     95
16   2011-01     RED HANGING HEART T-LIGHT HOLDER     91
17   2011-01             JAM MAKING SET WITH JARS     89
18   2011-01                HEART OF WICKER LARGE     86
19   2011-01   SET OF 6 SPICE TINS PANTRY DESIGN     84
```

#Plot month-wise popular products for a specific month

specific_month = input('Enter year and month (yyyy-mm): ')

if specific_month not in data['MonthYear'].unique():

print(f"Year and month not'{specific_month}' found in the data. check the year and month.")

monthly_popular_products = month_popular_products[month_popular_products['MonthYear'] == specific_month]

plt.figure(figsize=(10, 6))

sns.barplot(x=monthly_popular_products['Description'].value_counts().values,

y=monthly_popular_products['Description'].value_counts().index)

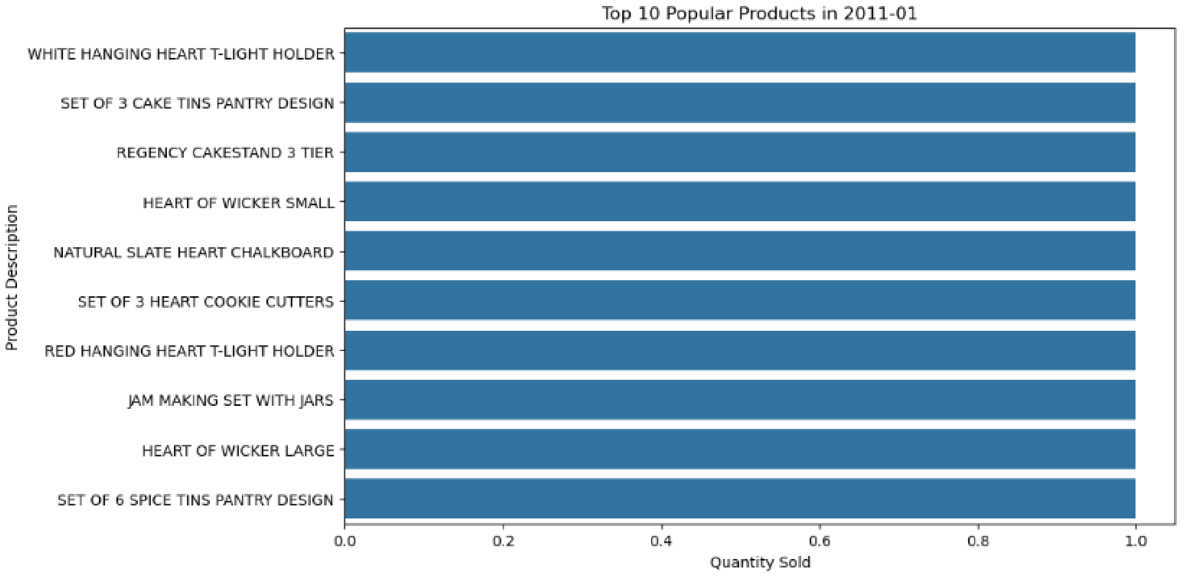
plt.title(f'Top 10 Popular Products in {specific_month}')

plt.xlabel('Quantity Sold')

plt.ylabel('Product Description')

plt.show()

Enter year and month (yyyy-mm): 2011-01



Getting product recommendations for customer

Here I use collaborative filtering using the surprise library for getting recommendations

#Step 1: Importing necessary library

from surprise import Dataset, Reader, SVD

from surprise.model_selection import cross_validate

#Creating a Reader object and specifying the rating scale

reader = Reader(rating_scale=(0, data['Quantity'].max()))

#Creating the dataset from the pandas dataframe

data_for_surprise = Dataset.load_from_df(data[['CustomerID', 'StockCode', 'Quantity']], reader)

#Using the Singular value decomposition (SVD) algorithm for collaborative filtering

algo = SVD()

#Evaluating the algorithm with cross-validation

cross_validate(algo, data_for_surprise, measures=['RMSE', 'MAE'], cv=5, verbose=True)

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    80982.261480984.513580983.805580982.470480982.966680983.20350.8438
MAE (testset)     80981.814280984.078680982.777080982.458280982.955780982.81670.7412
Fit time          2.98    3.07    3.22    3.04    3.03    3.07    0.08
Test time         0.64    0.34    0.55    0.51    0.63    0.53    0.11
```

```
{'test_rmse': array([80982.26143101, 80984.51352719, 80983.80545102, 80982.47039541,
        80982.96659102]),
 'test_mae': array([80981.81415819, 80984.07858468, 80982.77700726, 80982.45819898,
        80982.95568974]),
 'fit_time': (2.9794485569000244,
  3.067841053009033,
  3.223778247833252,
  3.0433733463287354,
  3.025991439819336),
 'test_time': (0.6426393985748291,
  0.34101009368896484,
  0.5457763671875,
  0.5076525211334229,
  0.6278979778289795)}
```

#Training the model on the entire dataset

trainset = data_for_surprise.build_full_trainset()

algo.fit(trainset)

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1b1815a4ad0>
```

#Function to get top n recommendations for a given customer

def top_recommendations(customer_id, n=15):

customer_id = float(customer_id)

#list of all products

all_products = data['Description'].unique()

#list of products the customer has already bought

purchased_products = data[data['CustomerID'] == customer_id]['Description'].unique()

#list of products the customer has not bought yet

products_to_predict = [product_description for product_description in all_products if product_description not in purchased_products]

```
    # Predict the ratings for all products the customer has not bought yet
    predictions = [algo.predict(customer_id, product_description) for product_description in products_to_predict]

    # Sort the predictions by estimated rating
    predictions.sort(key=lambda x: x.est)

    # top N recommendations
    top_recommendations = [pred.iid for pred in predictions[:n]]

    return top_recommendations
customer_id = float(input('Enter the Customer ID'))
if customer_id not in data['CustomerID'].unique():
        print(f"Customer ID {customer_id} not found in the data.")
else:
    top_product_recommendations = top_recommendations(customer_id, n=5)
    print(f'\nTop 10 recommendated products for {customer_id}:\n')

    for product in top_product_recommendations:
        print(product)
```

```
Enter the Customer ID 18283

Top 10 recommendated products for 18283.0:

WHITE METAL LANTERN
CREAM CUPID HEARTS COAT HANGER
KNITTED UNION FLAG HOT WATER BOTTLE
RED WOOLLY HOTTIE WHITE HEART.
SET 7 BABUSHKA NESTING BOXES
```

```
customer_id = float(input('Enter the Customer ID'))
if customer_id not in data['CustomerID'].unique():
        print(f"Customer ID {customer_id} not found in the data.")
else:
    top_product_recommendations = top_recommendations(customer_id, n=5)
```

```
    print(f'\nTop 10 recommendated products for {customer_id}:\n')

    for product in top_product_recommendations:
        print(product)
```

```
Enter the Customer ID 12345
Customer ID 12345.0 not found in the data.
```

STEP-6: Load Trained Model

File: Trained Model File.ipynb

Purpose:

- Loads saved models or pipelines
- Could be for inference or demonstration
- Might not need to run if you're training from scratch

Getting product recommendations for customer

Here I use collaborative filtering using the surprise library for getting recommendations

```
#Step 1: Importing necessary libraries
import pandas as pd
from surprise import Dataset, Reader, SVD
from surprise.model_selection import cross_validate

#Step 2: Load the dataset into jupyter notebook
data = pd.read_csv('OnlineRetail.csv')

#Step 3: Viewing 5 rows of data
data.head(5)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

#Step 4: Creating a Reader object and specifying the rating scale

reader = Reader(rating_scale=(0, data['Quantity'].max()))

#Step 5: Creating the dataset from the pandas dataframe

data_for_surprise = Dataset.load_from_df(data[['CustomerID', 'StockCode', 'Quantity']], reader)

#Step 6: Using the Singular value decomposition (SVD) algorithm for collaborative filtering

algo = SVD()

#Step 7: Evaluating the algorithm with cross-validation

cross_validate(algo, data_for_surprise, measures=['RMSE', 'MAE'], cv=5, verbose=True)

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    80747.188480757.405180757.478980756.594180742.277480752.18886.2903
MAE (testset)     80509.576580528.749380530.345080528.356680498.772480519.160012.7211
Fit time          5.18    5.39    5.11    4.99    5.03    5.14    0.14
Test time         1.10    0.71    0.92    1.01    0.70    0.89    0.16
```

```
{'test_rmse': array([80747.18841081, 80757.4050995 , 80757.47889524, 80756.59411981,
        80742.27740388]),
 'test_mae': array([80509.57652417, 80528.74934292, 80530.34502347, 80528.3566206 ,
        80498.77239529]),
 'fit_time': (5.1846535205841064,
  5.390363454818726,
  5.112430572509766,
  4.992536306381226,
  5.032274484634399),
 'test_time': (1.101330041885376,
  0.707421064376831,
  0.9241266250610352,
  1.0134682655334473,
  0.7025132179260254)}
```

#Step 8: Training the model on the entire dataset

trainset = data_for_surprise.build_full_trainset()

algo.fit(trainset)

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1280a8eb830>
```

#Step 9: Function to get top n recommendations for a given customer

def top_recommendations(customer_id, n=15):

    customer_id = float(customer_id)

    #list of all products

    all_products = data['Description'].unique()

    #list of products the customer has already bought

    purchased_products = data[data['CustomerID'] == customer_id]['Description'].unique()

    #list of products the customer has not bought yet

    products_to_predict = [product_description for product_description in all_products if product_description not in purchased_products]

Predict the ratings for all products the customer has not bought yet

predictions = [algo.predict(customer_id, product_description) for product_description in products_to_predict]

Sort the predictions by estimated rating

predictions.sort(key=lambda x: x.est)

top N recommendations

top_recommendations = [pred.iid for pred in predictions[:n]]

return top_recommendations

#Step 10: Getting the recommendated product list

customer_id = float(input('Enter the Customer ID'))

if customer_id not in data['CustomerID'].unique():

print(f"Customer ID {customer_id} not found in the data.")

else:

top_product_recommendations = top_recommendations(customer_id, n=5)

print(f'\nTop 10 recommendated products for {customer_id}:\n')

for product in top_product_recommendations:

print(product)

```
Enter the Customer ID 123456
Customer ID 123456.0 not found in the data.
```

customer_id = float(input('Enter the Customer ID'))

if customer_id not in data['CustomerID'].unique():

print(f"Customer ID {customer_id} not found in the data.")

else:

top_product_recommendations = top_recommendations(customer_id, n=5)

print(f'\nTop 10 recommendated products for {customer_id}:\n')

for product in top_product_recommendations:

print(product)

```
Enter the Customer ID 18283

Top 10 recommendated products for 18283.0:

WHITE METAL LANTERN
CREAM CUPID HEARTS COAT HANGER
KNITTED UNION FLAG HOT WATER BOTTLE
RED WOOLLY HOTTIE WHITE HEART.
SET 7 BABUSHKA NESTING BOXES
```

10.CONCLUSION & FUTURE SCOPE

CONCLUSION

The development and implementation of an online retail recommendation system using collaborative filtering have demonstrated promising results. The system effectively analyses customer purchase patterns and delivers personalized recommendations, improving user engagement and sales. While the model performed well, challenges such as data sparsity and cold start issues were observed, highlighting areas for further optimization.

FUTURE SCOPE

Hybrid Models: Combining collaborative filtering with content-based approaches and deep learning techniques to enhance accuracy.

Real-Time Recommendations: Implementing real-time recommendation generation to improve user experience dynamically.

Scalability: Enhancing the system to handle larger datasets efficiently using cloud-based solutions.

Customer Behaviour Analysis: Integrating advanced analytics to refine recommendations based on user behaviour trends.

A/B Testing: Conducting real-world testing to measure the effectiveness of recommendations in live retail environments.

Future advancements will ensure the recommendation system remains robust, scalable, and capable of delivering more personalized shopping experiences.