Data Science Major Projectsystem.

Online Retail Recommendation System

If you have tried online shopping, you must have noticed that when you are checking out a product on an e-Commerce site, there is a list of suggested products that you are presented with. In this project, you will develop a recommendation system.

For this, we are attaching a dataset containing information about recommendation systems for online retail data, so that we can understand what type of product can be recommended.

We are providing a dataset from Kaggle, which contains historical information about online retail data which can be usedto detect which product is highly recommended. Below are all the columns from the dataset we are using here

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.

Language Used: Python

```
# import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

pip install mlxtend
# Mlxtend stands for Machine Learning Extensions.
# It is a third-party Python library which contains many utilities and
tools for machine learning and Data Science tasks,
# including feature selection, ensemble methods, visualization, and
model evaluation.

from mlxtend.frequent_patterns import apriori, association_rules
```

1. Data Preprocessing

```
df = pd.read_excel("vnd.openxmlformats-
officedocument.spreadsheetml.sheet&rendition=1.xlsx",sheet_name=
"OnlineRetail")

df.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
```

2. Exploratory Data Analysis (EDA)

```
df.shape
```

```
(541909, 8)
```

```
df.columns
```

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity',
'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

```
df.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
```

```
df.isnull().sum()
```

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

```
df.describe().T
```

```
                 count                           mean
min  \
Quantity     541909.0                        9.55225              -
80995.0
InvoiceDate    541909  2011-07-04 13:34:57.156386048  2010-12-01
08:26:00
UnitPrice    541909.0                       4.611114             -
11062.06
CustomerID   406829.0                    15287.69057
12346.0

                             25%                 50%
75%  \
Quantity                     1.0                 3.0
10.0
InvoiceDate  2011-03-28 11:34:00  2011-07-19 17:17:00  2011-10-19
11:27:00
UnitPrice                   1.25                2.08
4.13
CustomerID                13953.0             15152.0
16791.0

                             max          std
Quantity                 80995.0   218.081158
InvoiceDate  2011-12-09 12:50:00          NaN
UnitPrice                38970.0    96.759853
CustomerID               18287.0  1713.600303
```

Handle Null Values

```
# Deleting features we dont need.

df.drop(columns=['CustomerID'], axis=1, inplace=True)

df.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


          InvoiceDate  UnitPrice         Country
0 2010-12-01 08:26:00       2.55  United Kingdom
1 2010-12-01 08:26:00       3.39  United Kingdom
2 2010-12-01 08:26:00       2.75  United Kingdom
3 2010-12-01 08:26:00       3.39  United Kingdom
4 2010-12-01 08:26:00       3.39  United Kingdom
```

```
df.isnull().sum()
```

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

```
# We have enough data so we can delete them. We could fill them with
mean, mode etc. It depends on our strategy

df.dropna(subset='Description',axis=0, inplace=True)

df.reset_index(drop=True, inplace=True)
df.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6
```

```
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


          InvoiceDate  UnitPrice         Country
0 2010-12-01 08:26:00       2.55  United Kingdom
1 2010-12-01 08:26:00       3.39  United Kingdom
2 2010-12-01 08:26:00       2.75  United Kingdom
3 2010-12-01 08:26:00       3.39  United Kingdom
4 2010-12-01 08:26:00       3.39  United Kingdom
```

```
df.isnull().sum()
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 540455 entries, 0 to 540454
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    540455 non-null  object
 1   StockCode    540455 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     540455 non-null  int64
 4   InvoiceDate  540455 non-null  datetime64[ns]
 5   UnitPrice    540455 non-null  float64
 6   Country      540455 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 28.9+ MB
```

Handling Outliers

```
df.describe().T
```

```
                count                           mean
min  \
Quantity     540455.0                       9.603129             -
80995.0
```

```
InvoiceDate    540455  2011-07-04 16:20:42.947035392  2010-12-01
08:26:00
UnitPrice    540455.0                        4.623519             -
11062.06

                                25%                   50%
75%  \
Quantity                        1.0                   3.0
10.0
InvoiceDate  2011-03-28 11:49:00  2011-07-20 11:38:00  2011-10-19
11:49:00
UnitPrice                     1.25                  2.08
4.13

                                max          std
Quantity                    80995.0   218.007598
InvoiceDate  2011-12-09 12:50:00          NaN
UnitPrice                   38970.0    96.889628
```

There is negative value when we look at the quantity and unit price, so let's examine and deal with it first

```
inv = df["InvoiceNo"].str.contains("C",na=False).sum()
print(f"The number of INVOICES containing 'C' : {inv}")
```

```
The number of INVOICES containing 'C' : 9288
```

```
def cancelledInvoice(x):
    if ('C' in str(x)):
        return np.nan
    else:
        return x

# Deleting datas that includes C in Invoice. Because if it includes
'C', it means the product returned.

df ['InvoiceNo'] = df['InvoiceNo'].apply(lambda x:cancelledInvoice(x))

df.isnull().sum()
```

```
InvoiceNo      9288
StockCode         0
Description       0
Quantity          0
InvoiceDate       0
UnitPrice         0
Country           0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 540455 entries, 0 to 540454
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531167 non-null  object
 1   StockCode    540455 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     540455 non-null  int64
 4   InvoiceDate  540455 non-null  datetime64[ns]
 5   UnitPrice    540455 non-null  float64
 6   Country      540455 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 28.9+ MB
```

```
df.dropna(subset='InvoiceNo', axis=0, inplace=True)
```

```
df.isnull().sum()
```

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

```
def cleanInvoice (x):
    if(str(x).isdigit()!=True or len(str(x)) !=6):
        return np.nan
    else:
        return x

# Invoice data cant be 6 digit and it must be numeric.

df['InvoiceNo']=df['InvoiceNo'].apply(lambda x: cleanInvoice(x))
```

```
df.isnull().sum()
```

```
InvoiceNo      3
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
Country        0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 531167 entries, 0 to 540454
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531164 non-null  float64
 1   StockCode    531167 non-null  object
 2   Description  531167 non-null  object
 3   Quantity     531167 non-null  int64
 4   InvoiceDate  531167 non-null  datetime64[ns]
 5   UnitPrice    531167 non-null  float64
 6   Country      531167 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 32.4+ MB
```

```
df.dropna(subset='InvoiceNo',axis=0, inplace=True)
df.reset_index(drop=True, inplace=True)
df.isnull().sum()
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 531164 entries, 0 to 531163
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531164 non-null  float64
 1   StockCode    531164 non-null  object
 2   Description  531164 non-null  object
 3   Quantity     531164 non-null  int64
 4   InvoiceDate  531164 non-null  datetime64[ns]
 5   UnitPrice    531164 non-null  float64
 6   Country      531164 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 28.4+ MB
```

```
df["InvoiceNo"] = df["InvoiceNo"].astype('int64').astype(str)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 531164 entries, 0 to 531163
```

```
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531164 non-null  object
 1   StockCode    531164 non-null  object
 2   Description  531164 non-null  object
 3   Quantity     531164 non-null  int64
 4   InvoiceDate  531164 non-null  datetime64[ns]
 5   UnitPrice    531164 non-null  float64
 6   Country      531164 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 28.4+ MB

df.describe().T

                 count                            mean
min  \
Quantity     531164.0                       10.293676               -
9600.0
InvoiceDate    531164  2011-07-04 19:55:06.271509248  2010-12-01
08:26:00
UnitPrice    531164.0                        3.879001
0.0

                              25%                   50%
75%  \
Quantity                      1.0                   3.0
10.0
InvoiceDate  2011-03-28 12:13:00  2011-07-20 12:41:30  2011-10-19
12:54:00
UnitPrice                    1.25                  2.08
4.13

                              max         std
Quantity                  80995.0  159.301807
InvoiceDate  2011-12-09 12:50:00         NaN
UnitPrice                13541.33   32.514222

df.head()

  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
```

```
          InvoiceDate  UnitPrice         Country
0 2010-12-01 08:26:00       2.55  United Kingdom
1 2010-12-01 08:26:00       3.39  United Kingdom
2 2010-12-01 08:26:00       2.75  United Kingdom
3 2010-12-01 08:26:00       3.39  United Kingdom
4 2010-12-01 08:26:00       3.39  United Kingdom
```

```
qtt = df.loc[df["Quantity"]<0,"Quantity"].count()
print(f"The number of negative QUANTITY values: {qtt}")
```

```
The number of negative QUANTITY values: 474
```

```
up=df.loc[df["UnitPrice"]<0,"UnitPrice"].count()
up
```

```
0
```

```
# Eliminate Quantity data that are less then 0
```

```
df=df[(df['Quantity'] > 0)]
```

```
df.reset_index(drop=True, inplace=True)
```

```
df.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
```

```
          InvoiceDate  UnitPrice         Country
0 2010-12-01 08:26:00       2.55  United Kingdom
1 2010-12-01 08:26:00       3.39  United Kingdom
2 2010-12-01 08:26:00       2.75  United Kingdom
3 2010-12-01 08:26:00       3.39  United Kingdom
4 2010-12-01 08:26:00       3.39  United Kingdom
```

```
df
```

```
       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
```

```
6
1         536365     71053                  WHITE METAL LANTERN
6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER
8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE
6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.
6
...          ...       ...                                 ...
...
530685    581587     22613          PACK OF 20 SPACEBOY NAPKINS
12
530686    581587     22899          CHILDREN'S APRON DOLLY GIRL
6
530687    581587     23254         CHILDRENS CUTLERY DOLLY GIRL
4
530688    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4
530689    581587     22138         BAKING SET 9 PIECE RETROSPOT
3

                InvoiceDate  UnitPrice         Country
0       2010-12-01 08:26:00       2.55  United Kingdom
1       2010-12-01 08:26:00       3.39  United Kingdom
2       2010-12-01 08:26:00       2.75  United Kingdom
3       2010-12-01 08:26:00       3.39  United Kingdom
4       2010-12-01 08:26:00       3.39  United Kingdom
...                     ...        ...             ...
530685 2011-12-09 12:50:00       0.85          France
530686 2011-12-09 12:50:00       2.10          France
530687 2011-12-09 12:50:00       4.15          France
530688 2011-12-09 12:50:00       4.15          France
530689 2011-12-09 12:50:00       4.95          France

[530690 rows x 7 columns]
```

```
qtt = df.loc[df["Quantity"]<0,"Quantity"].count()
print(f"The number of negative QUANTITY values: {qtt}")
```

```
The number of negative QUANTITY values: 0
```

```
df.describe().T
```

```
                 count                           mean
min  \
Quantity     530690.0                      10.605873
1.0
InvoiceDate    530690  2011-07-04 19:01:04.928526848  2010-12-01
08:26:00
```

```
UnitPrice    530690.0                          3.882466
0.0

                                25%                   50%
75%  \
Quantity                        1.0                   3.0
10.0
InvoiceDate  2011-03-28 11:59:00  2011-07-20 12:14:00  2011-10-19
12:35:00
UnitPrice                      1.25                  2.08
4.13

                                max          std
Quantity                    80995.0   156.638294
InvoiceDate  2011-12-09 12:50:00          NaN
UnitPrice                13541.33    32.528533
```

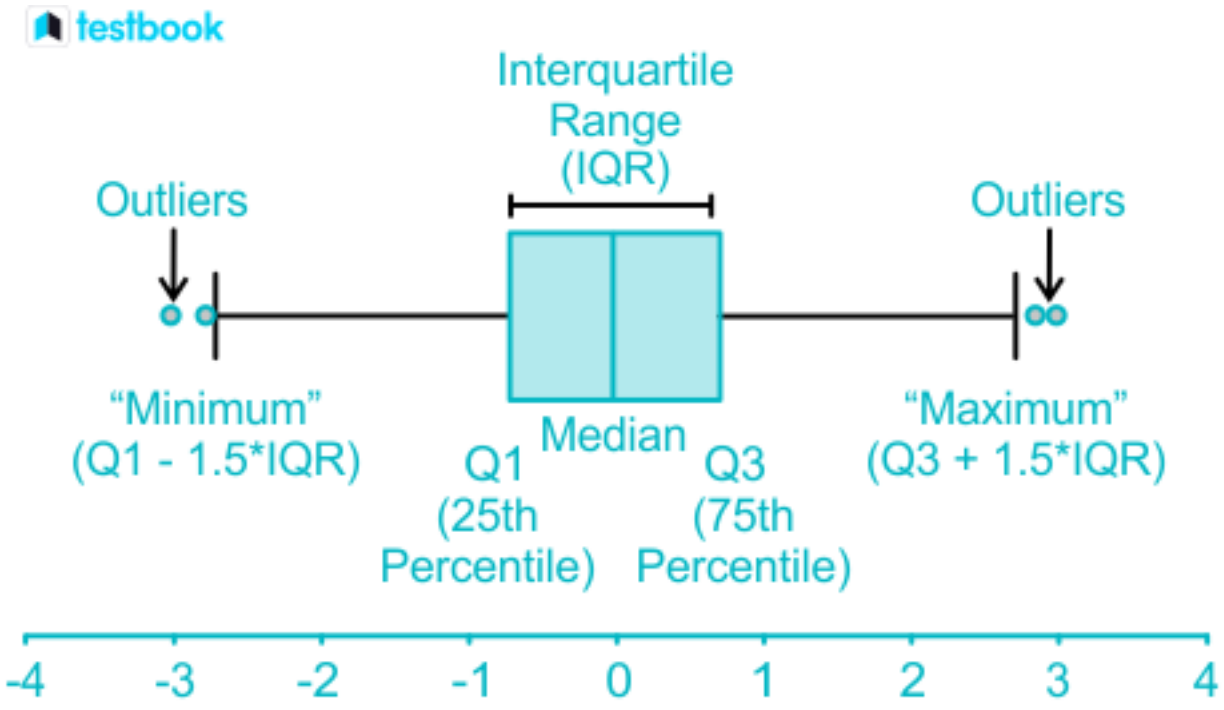
Negative price values also improved

Second, lets deal with outlier values using interquantile range

INTERQUARTILE RANGE

What is interquantile range?

In descriptive statistics, the interquartile range (IQR) is a measure of statistical dispersion, which is the spread of the data. The IQR may also be called the midspread, middle 50%, fourth spread, or H-spread. It is defined as the difference between the 75th and 25th percentiles of the data



.

```
df['Country'].unique()
```

```
array(['United Kingdom', 'France', 'Australia', 'Netherlands',
'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Finland',
       'Austria', 'Bahrain', 'Israel', 'Greece', 'Hong Kong',
'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

```
df.columns
```

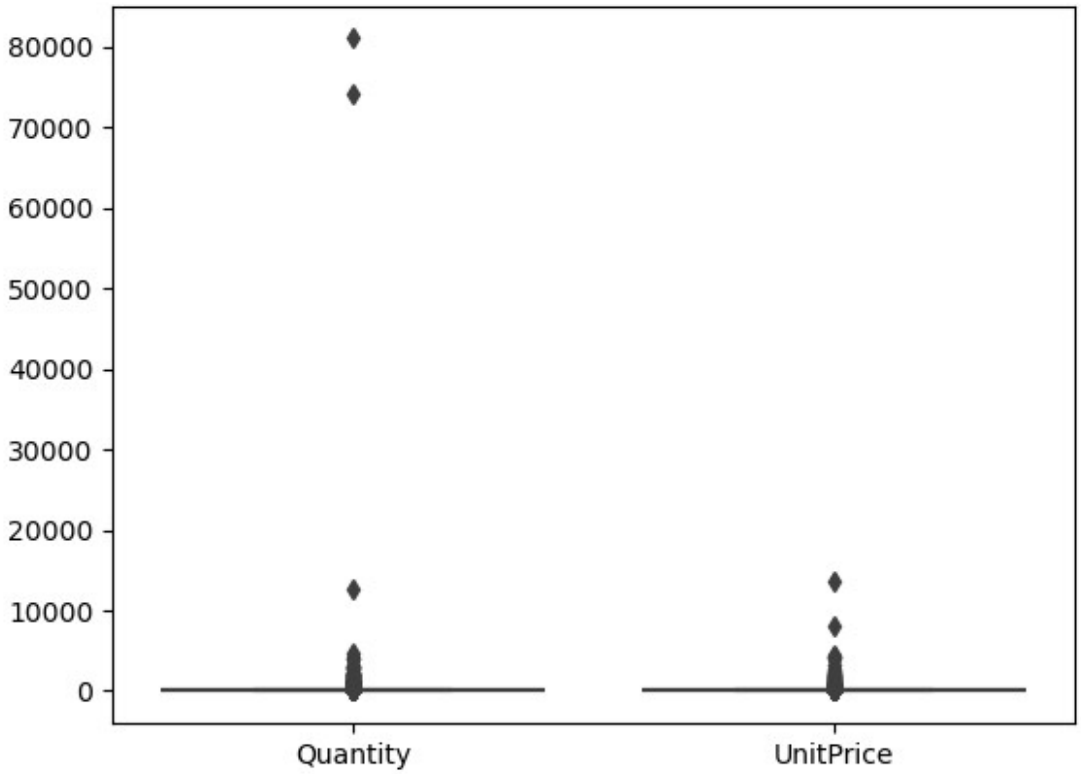
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity',
'InvoiceDate',
       'UnitPrice', 'Country'],
      dtype='object')
```

```
# We can see is there outliers in our dataset by using boxplot.
```

```
sns.boxplot(df[['Quantity','UnitPrice']])
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
<Axes: >
```




```
def handling_outlier(df,variable):
    quartile1 = df[variable].quantile(0.01)
    quartile3 = df[variable].quantile(0.99)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    df.loc[df[variable] < low_limit, variable] = low_limit
    df.loc[df[variable] > up_limit, variable] = up_limit

handling_outlier(df,"Quantity")
handling_outlier(df,"UnitPrice")

df.reset_index(drop=True, inplace=True)
df
```

```
       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6.0
1         536365     71053                  WHITE METAL LANTERN
6.0
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER
8.0
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE
6.0
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.
```

```
6.0
...          ...       ...                                  ...
...
530685    581587     22613          PACK OF 20 SPACEBOY NAPKINS
12.0
530686    581587     22899          CHILDREN'S APRON DOLLY GIRL
6.0
530687    581587     23254         CHILDRENS CUTLERY DOLLY GIRL
4.0
530688    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4.0
530689    581587     22138         BAKING SET 9 PIECE RETROSPOT
3.0

                InvoiceDate  UnitPrice         Country
0       2010-12-01 08:26:00       2.55  United Kingdom
1       2010-12-01 08:26:00       3.39  United Kingdom
2       2010-12-01 08:26:00       2.75  United Kingdom
3       2010-12-01 08:26:00       3.39  United Kingdom
4       2010-12-01 08:26:00       3.39  United Kingdom
...                     ...        ...             ...
530685 2011-12-09 12:50:00       0.85          France
530686 2011-12-09 12:50:00       2.10          France
530687 2011-12-09 12:50:00       4.15          France
530688 2011-12-09 12:50:00       4.15          France
530689 2011-12-09 12:50:00       4.95          France

[530690 rows x 7 columns]

df.describe().T

                count                           mean
min  \
Quantity     13870.0                      24.662581
1.0
InvoiceDate    13870  2011-07-08 00:36:02.080749824  2010-12-01
08:45:00
UnitPrice    13870.0                       3.210857
0.0

                                25%                  50%
75%  \
Quantity                       6.0                 12.0
24.0
InvoiceDate  2011-03-31 10:27:00  2011-07-29 13:28:00  2011-10-12
11:22:00
UnitPrice                    1.06                 1.65
3.75

                                max        std
```

```
Quantity                    248.5  40.976859
InvoiceDate  2011-12-09 12:50:00        NaN
UnitPrice                 42.015   4.430928
```

they look good

```
# Selecting some countries from the data set
list_cntry =
["Greece","Singapore","Netherlands","Switzerland","Cyprus","France","K
orea","Canada"]
for number,country in enumerate(list_cntry):
    list_cntry[number] = df[df['Country'] == country]

del df
df = pd.concat(list_cntry,axis=0)
df = df.sort_index()

df = df.reset_index(drop=True)
df.shape
```

```
(13870, 7)
```

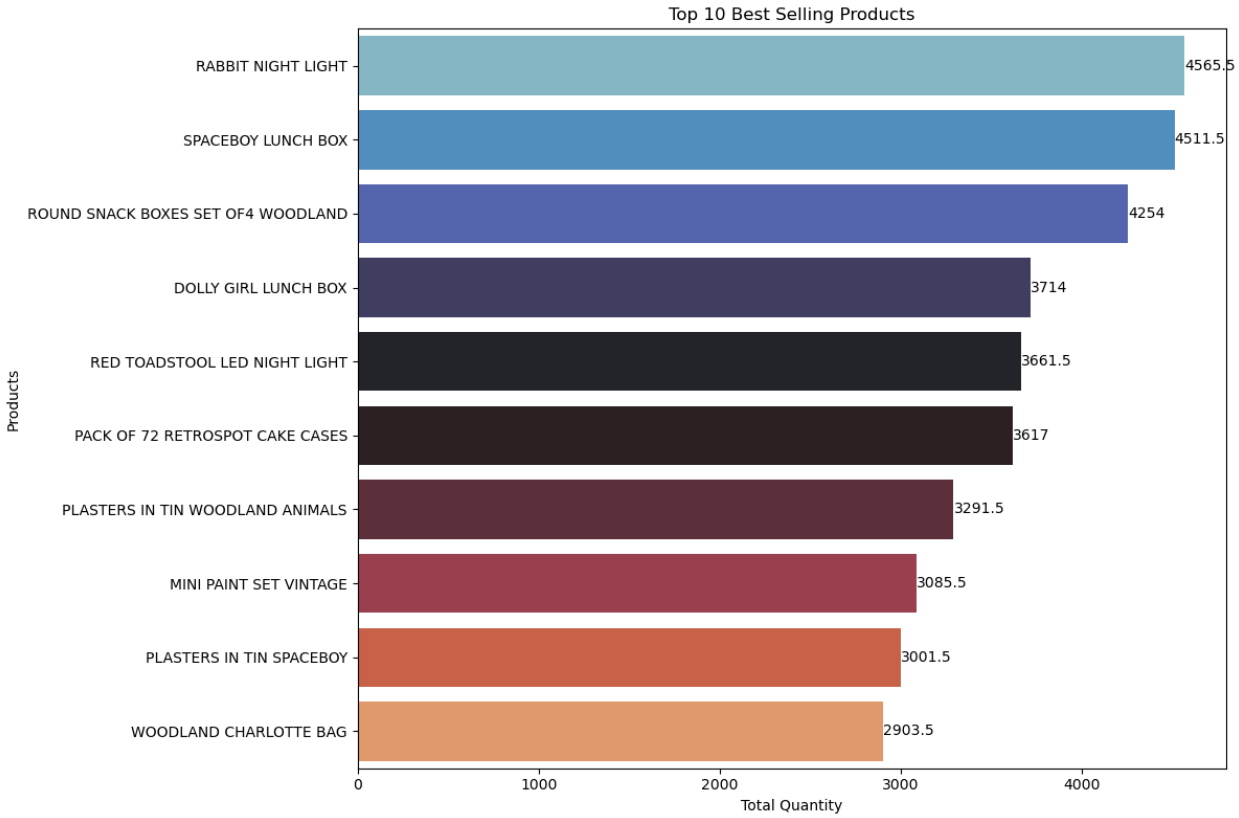
```
df
```

```
      InvoiceNo StockCode                          Description
Quantity  \
0        536370     22728         ALARM CLOCK BAKELIKE PINK      24.0

1        536370     22727           ALARM CLOCK BAKELIKE RED      24.0

2        536370     22726        ALARM CLOCK BAKELIKE GREEN      12.0

3        536370     21724  PANDA AND BUNNIES STICKER SHEET      12.0

4        536370     21883                   STARS GIFT TAPE      24.0

...         ...       ...                               ...       ...

13865    581587     22613      PACK OF 20 SPACEBOY NAPKINS      12.0

13866    581587     22899      CHILDREN'S APRON DOLLY GIRL       6.0

13867    581587     23254     CHILDRENS CUTLERY DOLLY GIRL       4.0

13868    581587     23255  CHILDRENS CUTLERY CIRCUS PARADE       4.0

13869    581587     22138     BAKING SET 9 PIECE RETROSPOT       3.0


              InvoiceDate  UnitPrice Country
0     2010-12-01 08:45:00       3.75  France
1     2010-12-01 08:45:00       3.75  France
```

```
2     2010-12-01 08:45:00       3.75  France
3     2010-12-01 08:45:00       0.85  France
4     2010-12-01 08:45:00       0.65  France
...                   ...        ...     ...
13865 2011-12-09 12:50:00       0.85  France
13866 2011-12-09 12:50:00       2.10  France
13867 2011-12-09 12:50:00       4.15  France
13868 2011-12-09 12:50:00       4.15  France
13869 2011-12-09 12:50:00       4.95  France

[13870 rows x 7 columns]
```

2.1 Data Analysis & Visualization

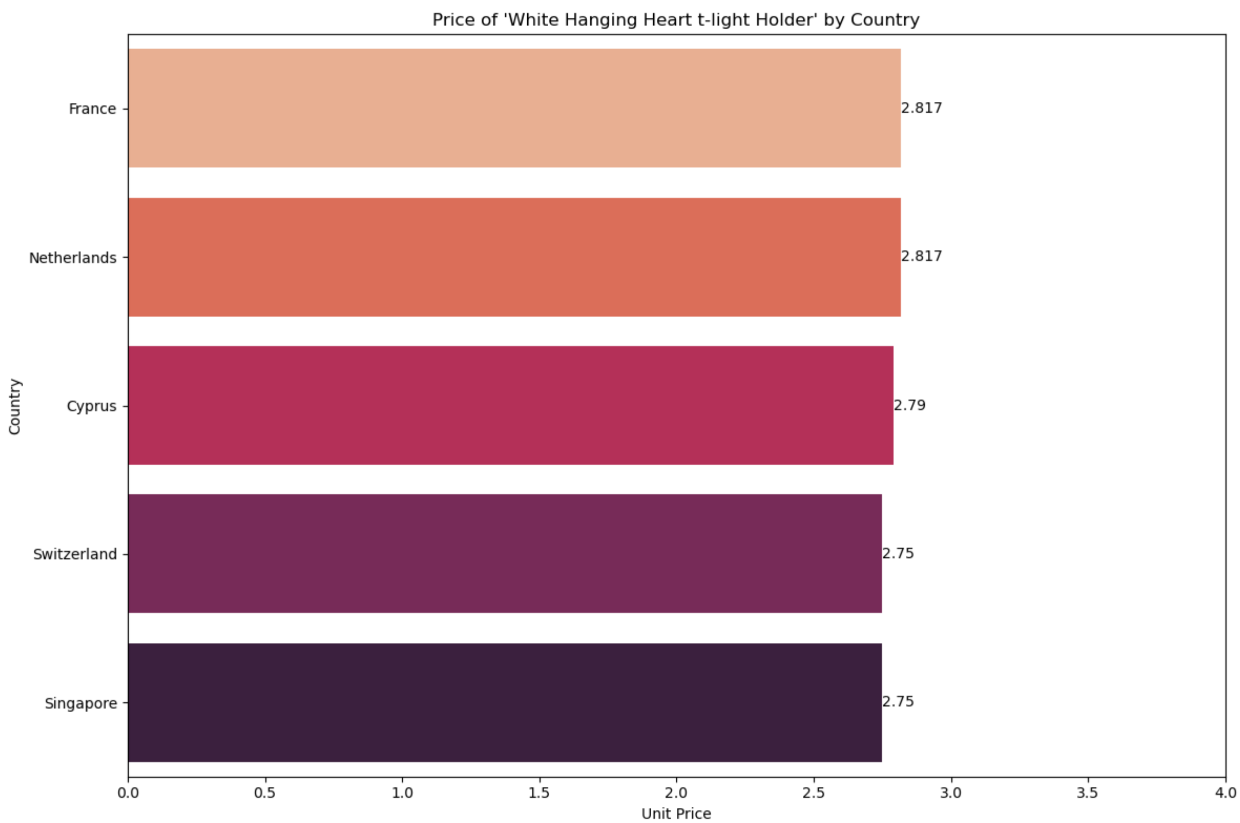
```
# Top 10 best selling products

product_count = df.groupby("Description")
["Quantity"].sum().nlargest(10)
product_count=product_count.reset_index()

plt.figure(figsize=(12, 8))

ax =
sns.barplot(data=product_count,y="Description",x="Quantity",palette="i
cefire")

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title("Top 10 Best Selling Products")
plt.xlabel("Total Quantity")
plt.ylabel("Products")
plt.tight_layout()
plt.show()

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```




```
# Price of any product by country

list_country, list_price = [], []
for col in df["Country"].unique():
    price = df.loc[(df["Country"] == col) & (df["Description"] ==
"WHITE HANGING HEART T-LIGHT HOLDER"), "UnitPrice"].mean()

    list_country.append(col)
    list_price.append(round(price,3))

df_price =
pd.DataFrame(columns=["Country"],data=list_price,index=list_country)
df_price.dropna(inplace=True)
df_price = df_price.sort_values(by="Country",ascending=False)

plt.figure(figsize=(12, 8))

ax =
sns.barplot(data=df_price,y=df_price.index,x="Country",palette="rocket
_r")

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title("Price of 'White Hanging Heart t-light Holder' by
```

```
Country")
plt.xlim(0, 4)
plt.xlabel("Unit Price")
plt.ylabel("Country")
plt.tight_layout()
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



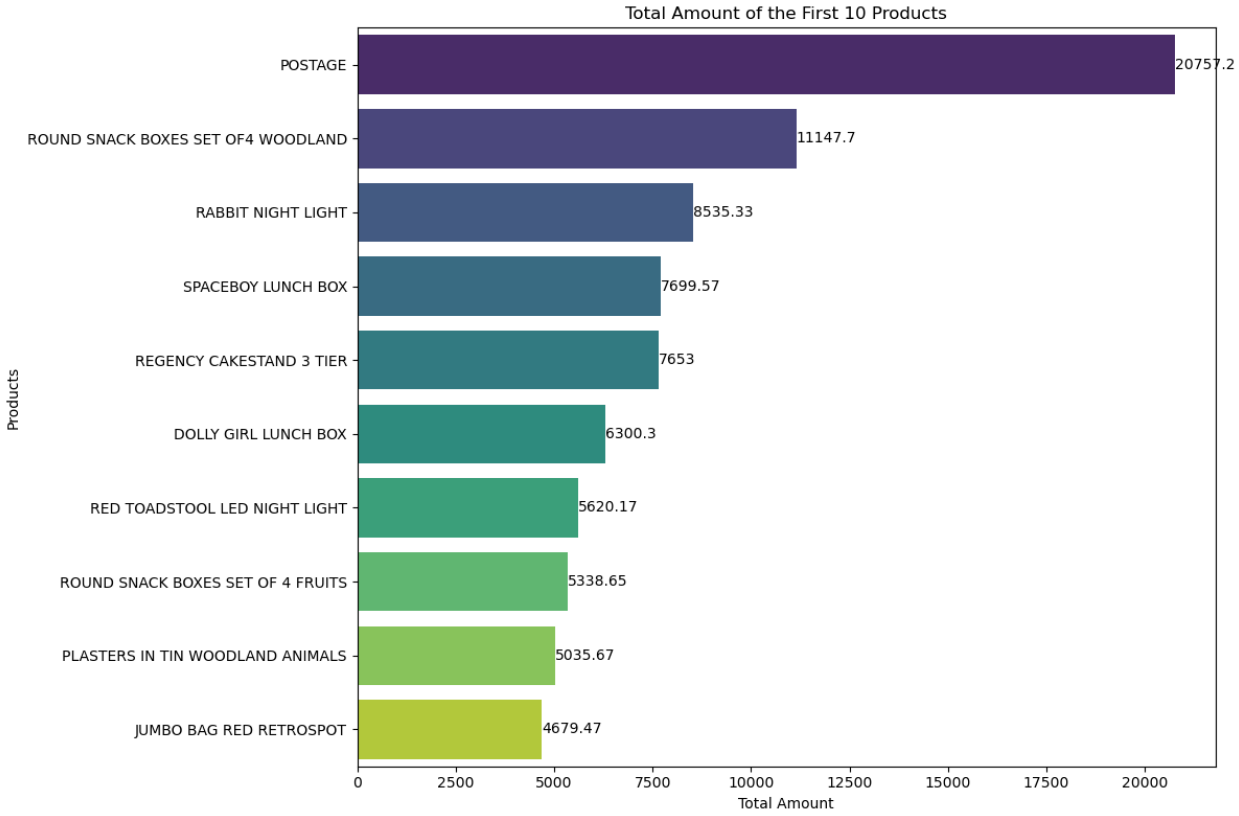
```
# Total amount of the first 10 products

df["TOTAL_AMOUNT"] = df["Quantity"] * df["UnitPrice"]
df.head()
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536370     22728       ALARM CLOCK BAKELIKE PINK      24.0
1    536370     22727        ALARM CLOCK BAKELIKE RED      24.0
2    536370     22726      ALARM CLOCK BAKELIKE GREEN      12.0
3    536370     21724  PANDA AND BUNNIES STICKER SHEET      12.0
4    536370     21883                 STARS GIFT TAPE      24.0

          InvoiceDate  UnitPrice Country  TOTAL_AMOUNT
0 2010-12-01 08:45:00       3.75  France          90.0
1 2010-12-01 08:45:00       3.75  France          90.0
2 2010-12-01 08:45:00       3.75  France          45.0
3 2010-12-01 08:45:00       0.85  France          10.2
4 2010-12-01 08:45:00       0.65  France          15.6
```

```
total_amount = df.groupby("Description")
["TOTAL_AMOUNT"].sum().nlargest(10)
total_amount=total_amount.reset_index()

plt.figure(figsize=(12, 8))

ax =
sns.barplot(data=total_amount,y="Description",x="TOTAL_AMOUNT",palette
="viridis")

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title("Total Amount of the First 10 Products")
plt.xlabel("Total Amount")
plt.ylabel("Products")
plt.tight_layout()
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498:
DeprecationWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype,
```

```
pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



3. Preparing the ARL Data Structure0.

Invoice-Product Matrix

Setting it so that there are invoices in the rows and products in the columns. If there are products, 1, otherwise 0.

```
# Reaching the product quantities in each invoice.
df.groupby(["InvoiceNo","Description"])["Quantity"].sum().head(20)

# if you want you can use this code, it gives same result
#
df.groupby(["INVOICE","DESCRIPTION"]).agg({"QUANTITY":"sum"}).head(20)
```

```
InvoiceNo  Description
536370     ALARM CLOCK BAKELIKE GREEN            12.0
           ALARM CLOCK BAKELIKE PINK             24.0
           ALARM CLOCK BAKELIKE RED              24.0
           CHARLOTTE BAG DOLLY GIRL DESIGN       20.0
           CIRCUS PARADE LUNCH BOX               24.0
           INFLATABLE POLITICAL GLOBE            48.0
```

```
            LUNCH BOX I LOVE LONDON                24.0
            MINI JIGSAW CIRCUS PARADE              24.0
            MINI JIGSAW SPACEBOY                   24.0
            MINI PAINT SET VINTAGE                 36.0
            PANDA AND BUNNIES STICKER SHEET        12.0
            POSTAGE                                 3.0
            RED TOADSTOOL LED NIGHT LIGHT          24.0
            ROUND SNACK BOXES SET OF4 WOODLAND     24.0
            SET 2 TEA TOWELS I LOVE LONDON         24.0
            SET/2 RED RETROSPOT TEA TOWELS         18.0
            SPACEBOY LUNCH BOX                     24.0
            STARS GIFT TAPE                        24.0
            VINTAGE HEADS AND TAILS CARD GAME      24.0
            VINTAGE SEASIDE JIGSAW PUZZLES         12.0
Name: Quantity, dtype: float64
```

```
# Sorting descriptions by columns
df.groupby(["InvoiceNo", "Description"]).agg({"Quantity":
"sum"}).unstack().iloc[0:5, 0:5]
```

```
                            Quantity                                    \
Description 10 COLOUR SPACEBOY PEN 12 COLOURED PARTY BALLOONS
InvoiceNo
536370                           NaN                            NaN
536403                           NaN                            NaN
536852                           NaN                            NaN
536858                           NaN                            NaN
536974                           NaN                            NaN


\
Description 12 EGG HOUSE PAINTED WOOD 12 MESSAGE CARDS WITH ENVELOPES

InvoiceNo

536370                                 NaN                                 NaN

536403                                 NaN                                 NaN

536852                                 NaN                                 NaN

536858                                 NaN                                 NaN

536974                                 NaN                                 NaN



Description 12 PENCIL SMALL TUBE WOODLAND
InvoiceNo
536370                                    NaN
536403                                    NaN
```

```
536852                                    NaN
536858                                    NaN
536974                                    NaN

# Filling nan values with zero
df.groupby(['InvoiceNo', 'Description']).agg({"Quantity":
"sum"}).unstack().fillna(0).iloc[0:5, 0:5]

                             Quantity                                    \
Description 10 COLOUR SPACEBOY PEN 12 COLOURED PARTY BALLOONS
InvoiceNo
536370                              0.0                              0.0
536403                              0.0                              0.0
536852                              0.0                              0.0
536858                              0.0                              0.0
536974                              0.0                              0.0


\
Description 12 EGG HOUSE PAINTED WOOD 12 MESSAGE CARDS WITH ENVELOPES

InvoiceNo

536370                                0.0                                0.0

536403                                0.0                                0.0

536852                                0.0                                0.0

536858                                0.0                                0.0

536974                                0.0                                0.0



Description 12 PENCIL SMALL TUBE WOODLAND
InvoiceNo
536370                                   0.0
536403                                   0.0
536852                                   0.0
536858                                   0.0
536974                                   0.0

# 0,0 is converted to 0, if there is a value then it is 1
df.groupby(['InvoiceNo', 'Description']).agg({"Quantity":
"sum"}).unstack().fillna(0).applymap(lambda x: 1 if x > 0 else
0).iloc[0:5, 0:5]

                             Quantity                                    \
Description 10 COLOUR SPACEBOY PEN 12 COLOURED PARTY BALLOONS
InvoiceNo
```

```
536370                                0                                0
536403                                0                                0
536852                                0                                0
536858                                0                                0
536974                                0                                0


\
Description 12 EGG HOUSE PAINTED WOOD 12 MESSAGE CARDS WITH ENVELOPES

InvoiceNo

536370                                  0                                  0

536403                                  0                                  0

536852                                  0                                  0

536858                                  0                                  0

536974                                  0                                  0



Description 12 PENCIL SMALL TUBE WOODLAND
InvoiceNo
536370                                     0
536403                                     0
536852                                     0
536858                                     0
536974                                     0
```

```
# Changing product names with stock code
df.groupby(['InvoiceNo', 'StockCode']).agg({"Quantity":
"sum"}).unstack().fillna(0).applymap(lambda x: 1 if x > 0 else
0).iloc[0:5, 0:5]
```

```
          Quantity
StockCode    10002 10120 10125 10133 10135
InvoiceNo
536370           1     0     0     0     0
536403           0     0     0     0     0
536852           0     0     0     0     0
536858           0     0     0     0     0
536974           0     0     0     0     0
```

```
# It ready !!
df_arl = df.groupby(['InvoiceNo', 'StockCode']).agg({"Quantity":
"sum"}).unstack().fillna(0).applymap(lambda x: 1 if x > 0 else 0)
```

```
# Finding product name from stock code
def prdct_name_finder(data,stckcde):
    product_name = data[data["StockCode"] == stckcde]
[["Description"]].values[0].tolist()
    print(product_name)
prdct_name_finder(df,"85014A")
```

```
['BLACK/BLUE POLKADOT UMBRELLA']
```

it works

4. Association Rule Analysis

What is Association Rule?

Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently a itemset occurs in a transaction. A typical example is a Market Based Analysis.

```
frequent_itemsets = apriori(df_arl,min_support=0.01,use_colnames=True)
```

```
C:\Users\dshem\AppData\Roaming\Python\Python311\site-packages\mlxtend\
frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with
non-bool types result in worse computationalperformance and their
support might be discontinued in the future.Please use a DataFrame
with bool type
  warnings.warn(
```

```
frequent_itemsets.sort_values("support", ascending=False)
```

```
        support                                              itemsets
625    0.648696                                ((Quantity, POST))
206    0.182609                               ((Quantity, 22326))
274    0.165217                               ((Quantity, 22554))
118    0.161739                               ((Quantity, 21731))
276    0.154783                               ((Quantity, 22556))
...         ...                                                ...
11317  0.010435  ((Quantity, 23173), (Quantity, 22423), (Quanti...
11319  0.010435  ((Quantity, 23204), (Quantity, 22423), (Quanti...
11320  0.010435  ((Quantity, 23209), (Quantity, 22423), (Quanti...
11322  0.010435  ((Quantity, 23254), (Quantity, 22423), (Quanti...
21382  0.010435  ((Quantity, 22661), (Quantity, 22712), (Quanti...

[21383 rows x 2 columns]
```

```
rules =
association_rules(frequent_itemsets,metric="support",min_threshold=0.0
1)
```

```
# Filtering
rules[(rules["support"]>0.05) & (rules["confidence"]>0.1) &
(rules["lift"]>5)]
```

```
                                                antecedents  \
1722                                    ((Quantity, 21086))
1723                                    ((Quantity, 21080))
1724                                    ((Quantity, 21094))
1725                                    ((Quantity, 21080))
1888                                    ((Quantity, 21094))
1889                                    ((Quantity, 21086))
7018                                    ((Quantity, 22629))
7019                                    ((Quantity, 22630))
7644                                    ((Quantity, 22726))
7645                                    ((Quantity, 22727))
7646                                    ((Quantity, 22726))
7647                                    ((Quantity, 22728))
7680                                    ((Quantity, 22727))
7681                                    ((Quantity, 22728))
8596                                    ((Quantity, 23254))
8597                                    ((Quantity, 23256))
23572              ((Quantity, 21094), (Quantity, 21086))
23573              ((Quantity, 21094), (Quantity, 21080))
23574              ((Quantity, 21080), (Quantity, 21086))
23575                                   ((Quantity, 21094))
23576                                   ((Quantity, 21086))
23577                                   ((Quantity, 21080))
23800               ((Quantity, POST), (Quantity, 21086))
23802               ((Quantity, POST), (Quantity, 21080))
23803                                   ((Quantity, 21086))
23805                                   ((Quantity, 21080))
24034               ((Quantity, 21094), (Quantity, POST))
24036               ((Quantity, POST), (Quantity, 21080))
24037                                   ((Quantity, 21094))
24039                                   ((Quantity, 21080))
25288               ((Quantity, 21094), (Quantity, POST))
25290               ((Quantity, POST), (Quantity, 21086))
25291                                   ((Quantity, 21094))
25293                                   ((Quantity, 21086))
38088              ((Quantity, 22326), (Quantity, 22554))
38089                                   ((Quantity, 22551))
52714              ((Quantity, 22726), (Quantity, 22727))
52715              ((Quantity, 22726), (Quantity, 22728))
52716              ((Quantity, 22727), (Quantity, 22728))
52717                                   ((Quantity, 22726))
52718                                   ((Quantity, 22727))
52719                                   ((Quantity, 22728))
87888  ((Quantity, 21094), (Quantity, POST), (Quantit...
87890  ((Quantity, 21094), (Quantity, POST), (Quantit...
87891  ((Quantity, 21080), (Quantity, POST), (Quantit...
```

```
87892             ((Quantity, 21094), (Quantity, 21086))
87893              ((Quantity, 21094), (Quantity, POST))
87894             ((Quantity, 21094), (Quantity, 21080))
87895              ((Quantity, POST), (Quantity, 21086))
87896             ((Quantity, 21080), (Quantity, 21086))
87897              ((Quantity, POST), (Quantity, 21080))
87898                                ((Quantity, 21094))
87899                                ((Quantity, 21086))
87901                                ((Quantity, 21080))

                                                consequents  antecedent
support  \
1722                                ((Quantity, 21080))
0.104348
1723                                ((Quantity, 21086))
0.109565
1724                                ((Quantity, 21080))
0.100870
1725                                ((Quantity, 21094))
0.109565
1888                                ((Quantity, 21086))
0.100870
1889                                ((Quantity, 21094))
0.104348
7018                                ((Quantity, 22630))
0.144348
7019                                ((Quantity, 22629))
0.113043
7644                                ((Quantity, 22727))
0.076522
7645                                ((Quantity, 22726))
0.074783
7646                                ((Quantity, 22728))
0.076522
7647                                ((Quantity, 22726))
0.078261
7680                                ((Quantity, 22728))
0.074783
7681                                ((Quantity, 22727))
0.078261
8596                                ((Quantity, 23256))
0.074783
8597                                ((Quantity, 23254))
0.083478
23572                               ((Quantity, 21080))
0.092174
23573                               ((Quantity, 21086))
0.078261
23574                               ((Quantity, 21094))
```

```
0.074783
23575              ((Quantity, 21080), (Quantity, 21086))
0.100870
23576              ((Quantity, 21094), (Quantity, 21080))
0.104348
23577              ((Quantity, 21094), (Quantity, 21086))
0.109565
23800                                ((Quantity, 21080))
0.081739
23802                                ((Quantity, 21086))
0.076522
23803               ((Quantity, POST), (Quantity, 21080))
0.104348
23805               ((Quantity, POST), (Quantity, 21086))
0.109565
24034                                ((Quantity, 21080))
0.074783
24036                                ((Quantity, 21094))
0.076522
24037               ((Quantity, POST), (Quantity, 21080))
0.100870
24039               ((Quantity, 21094), (Quantity, POST))
0.109565
25288                                ((Quantity, 21086))
0.074783
25290                                ((Quantity, 21094))
0.081739
25291               ((Quantity, POST), (Quantity, 21086))
0.100870
25293               ((Quantity, 21094), (Quantity, POST))
0.104348
38088                                ((Quantity, 22551))
0.066087
38089              ((Quantity, 22326), (Quantity, 22554))
0.149565
52714                                ((Quantity, 22728))
0.060870
52715                                ((Quantity, 22727))
0.059130
52716                                ((Quantity, 22726))
0.057391
52717              ((Quantity, 22727), (Quantity, 22728))
0.076522
52718              ((Quantity, 22726), (Quantity, 22728))
0.074783
52719              ((Quantity, 22726), (Quantity, 22727))
0.078261
87888                                ((Quantity, 21080))
0.069565
```

```
87890                                  ((Quantity, 21086))
0.059130
87891                                  ((Quantity, 21094))
0.057391
87892               ((Quantity, POST), (Quantity, 21080))
0.092174
87893             ((Quantity, 21080), (Quantity, 21086))
0.074783
87894               ((Quantity, POST), (Quantity, 21086))
0.078261
87895             ((Quantity, 21094), (Quantity, 21080))
0.081739
87896               ((Quantity, 21094), (Quantity, POST))
0.074783
87897             ((Quantity, 21094), (Quantity, 21086))
0.076522
87898  ((Quantity, 21080), (Quantity, POST), (Quantit...
0.100870
87899  ((Quantity, 21094), (Quantity, POST), (Quantit...
0.104348
87901  ((Quantity, 21094), (Quantity, POST), (Quantit...
0.109565

       consequent support   support  confidence       lift
leverage  \
1722              0.109565  0.074783    0.716667   6.541005  0.063350

1723              0.104348  0.074783    0.682540   6.541005  0.063350

1724              0.109565  0.078261    0.775862   7.081281  0.067209

1725              0.100870  0.078261    0.714286   7.081281  0.067209

1888              0.104348  0.092174    0.913793   8.757184  0.081648

1889              0.100870  0.092174    0.883333   8.757184  0.081648

7018              0.113043  0.086957    0.602410   5.329008  0.070639

7019              0.144348  0.086957    0.769231   5.329008  0.070639

7644              0.074783  0.060870    0.795455  10.636892  0.055147

7645              0.076522  0.060870    0.813953  10.636892  0.055147

7646              0.078261  0.059130    0.772727   9.873737  0.053142

7647              0.076522  0.059130    0.755556   9.873737  0.053142

7680              0.078261  0.057391    0.767442   9.806202  0.051539
```

7681	0.074783	0.057391	0.733333	9.806202	0.051539
8596	0.083478	0.066087	0.883721	10.586240	0.059844
8597	0.074783	0.066087	0.791667	10.586240	0.059844
23572	0.109565	0.073043	0.792453	7.232704	0.062944
23573	0.104348	0.073043	0.933333	8.944444	0.064877
23574	0.100870	0.073043	0.976744	9.683240	0.065500
23575	0.074783	0.073043	0.724138	9.683240	0.065500
23576	0.078261	0.073043	0.700000	8.944444	0.064877
23577	0.092174	0.073043	0.666667	7.232704	0.062944
23800	0.109565	0.057391	0.702128	6.408308	0.048436
23802	0.104348	0.057391	0.750000	7.187500	0.049406
23803	0.076522	0.057391	0.550000	7.187500	0.049406
23805	0.081739	0.057391	0.523810	6.408308	0.048436
24034	0.109565	0.059130	0.790698	7.216685	0.050937
24036	0.100870	0.059130	0.772727	7.660658	0.051412
24037	0.076522	0.059130	0.586207	7.660658	0.051412
24039	0.074783	0.059130	0.539683	7.216685	0.050937
25288	0.104348	0.069565	0.930233	8.914729	0.061762
25290	0.100870	0.069565	0.851064	8.437271	0.061320
25291	0.081739	0.069565	0.689655	8.437271	0.061320
25293	0.074783	0.069565	0.666667	8.914729	0.061762
38088	0.149565	0.052174	0.789474	5.278458	0.042290
38089	0.066087	0.052174	0.348837	5.278458	0.042290
52714	0.078261	0.050435	0.828571	10.587302	0.045671
52715	0.074783	0.050435	0.852941	11.405609	0.046013
52716	0.076522	0.050435	0.878788	11.484160	0.046043

52717	0.057391	0.050435	0.659091	11.484160	0.046043
52718	0.059130	0.050435	0.674419	11.405609	0.046013
52719	0.060870	0.050435	0.644444	10.587302	0.045671
87888	0.109565	0.055652	0.800000	7.301587	0.048030
87890	0.104348	0.055652	0.941176	9.019608	0.049482
87891	0.100870	0.055652	0.969697	9.613375	0.049863
87892	0.076522	0.055652	0.603774	7.890223	0.048599
87893	0.074783	0.055652	0.744186	9.951325	0.050060
87894	0.081739	0.055652	0.711111	8.699764	0.049255
87895	0.078261	0.055652	0.680851	8.699764	0.049255
87896	0.074783	0.055652	0.744186	9.951325	0.050060
87897	0.092174	0.055652	0.727273	7.890223	0.048599
87898	0.057391	0.055652	0.551724	9.613375	0.049863
87899	0.059130	0.055652	0.533333	9.019608	0.049482
87901	0.069565	0.055652	0.507937	7.301587	0.048030

	conviction	zhangs_metric
1722	3.142711	0.945812
1723	2.821304	0.951354
1724	3.972709	0.955126
1725	3.146957	0.964453
1888	10.389565	0.985183
1889	7.706832	0.989009
7018	2.230830	0.949390
7019	3.707826	0.915882
7644	4.523285	0.981060
7645	4.963696	0.979216
7646	4.055652	0.973192
7647	3.777866	0.975028
7680	3.963478	0.970608
7681	3.469565	0.974271
8596	7.882087	0.978730
8597	4.441043	0.988016
23572	4.290277	0.949234
23573	13.434783	0.963612
23574	38.662609	0.969209

```
23575    3.353913        0.997329
23576    3.072464        0.991678
23577    2.723478        0.967773
23800    2.989317        0.919077
23802    3.582609        0.932203
23803    2.052174        0.961165
23805    1.928348        0.947798
24034    4.254300        0.931059
24036    3.956174        0.941509
24037    2.231739        0.967004
24039    2.009955        0.967429
25288   12.837681        0.959586
25290    6.037019        0.959943
25291    2.958841        0.980368
25293    2.775652        0.991262
38088    4.039565        0.867908
38089    1.434224        0.953102
52714    5.376812        0.964240
52715    6.291478        0.969660
52716    7.618696        0.968507
52717    2.764986        0.988571
52718    2.889814        0.986064
52719    2.641304        0.982433
87888    4.452174        0.927570
87890   15.226087        0.945009
87891   29.671304        0.950530
87892    2.330683        0.961925
87893    3.616759        0.972216
87894    3.178595        0.960200
87895    2.888116        0.963838
87896    3.616759        0.972216
87897    3.328696        0.945621
87898    2.102742        0.996494
87899    2.016149        0.992718
87901    1.890884        0.969238
```

```
# Filtering by confidence
rules[(rules["support"]>0.05) & (rules["confidence"]>0.1) &
(rules["lift"]>5)].sort_values("confidence", ascending=False)
```

```
                                              antecedents  \
23574             ((Quantity, 21080), (Quantity, 21086))
87891  ((Quantity, 21080), (Quantity, POST), (Quantit...
87890  ((Quantity, 21094), (Quantity, POST), (Quantit...
23573             ((Quantity, 21094), (Quantity, 21080))
25288              ((Quantity, 21094), (Quantity, POST))
1888                                ((Quantity, 21094))
8596                                ((Quantity, 23254))
1889                                ((Quantity, 21086))
52716             ((Quantity, 22727), (Quantity, 22728))
```

```
52715             ((Quantity, 22726), (Quantity, 22728))
25290              ((Quantity, POST), (Quantity, 21086))
52714             ((Quantity, 22726), (Quantity, 22727))
7645                                ((Quantity, 22727))
87888  ((Quantity, 21094), (Quantity, POST), (Quantit...
7644                                ((Quantity, 22726))
23572             ((Quantity, 21094), (Quantity, 21086))
8597                                ((Quantity, 23256))
24034              ((Quantity, 21094), (Quantity, POST))
38088             ((Quantity, 22326), (Quantity, 22554))
1724                                ((Quantity, 21094))
7646                                ((Quantity, 22726))
24036              ((Quantity, POST), (Quantity, 21080))
7019                                ((Quantity, 22630))
7680                                ((Quantity, 22727))
7647                                ((Quantity, 22728))
23802              ((Quantity, POST), (Quantity, 21080))
87893              ((Quantity, 21094), (Quantity, POST))
87896             ((Quantity, 21080), (Quantity, 21086))
7681                                ((Quantity, 22728))
87897              ((Quantity, POST), (Quantity, 21080))
23575                               ((Quantity, 21094))
1722                                ((Quantity, 21086))
1725                                ((Quantity, 21080))
87894             ((Quantity, 21094), (Quantity, 21080))
23800              ((Quantity, POST), (Quantity, 21086))
23576                               ((Quantity, 21086))
25291                               ((Quantity, 21094))
1723                                ((Quantity, 21080))
87895              ((Quantity, POST), (Quantity, 21086))
52718                               ((Quantity, 22727))
23577                               ((Quantity, 21080))
25293                               ((Quantity, 21086))
52717                               ((Quantity, 22726))
52719                               ((Quantity, 22728))
87892             ((Quantity, 21094), (Quantity, 21086))
7018                                ((Quantity, 22629))
24037                               ((Quantity, 21094))
87898                               ((Quantity, 21094))
23803                               ((Quantity, 21086))
24039                               ((Quantity, 21080))
87899                               ((Quantity, 21086))
23805                               ((Quantity, 21080))
87901                               ((Quantity, 21080))
38089                               ((Quantity, 22551))

                                          consequents  antecedent
support  \
23574                               ((Quantity, 21094))
```

```
0.074783
87891                                   ((Quantity, 21094))
0.057391
87890                                   ((Quantity, 21086))
0.059130
23573                                   ((Quantity, 21086))
0.078261
25288                                   ((Quantity, 21086))
0.074783
1888                                    ((Quantity, 21086))
0.100870
8596                                    ((Quantity, 23256))
0.074783
1889                                    ((Quantity, 21094))
0.104348
52716                                   ((Quantity, 22726))
0.057391
52715                                   ((Quantity, 22727))
0.059130
25290                                   ((Quantity, 21094))
0.081739
52714                                   ((Quantity, 22728))
0.060870
7645                                    ((Quantity, 22726))
0.074783
87888                                   ((Quantity, 21080))
0.069565
7644                                    ((Quantity, 22727))
0.076522
23572                                   ((Quantity, 21080))
0.092174
8597                                    ((Quantity, 23254))
0.083478
24034                                   ((Quantity, 21080))
0.074783
38088                                   ((Quantity, 22551))
0.066087
1724                                    ((Quantity, 21080))
0.100870
7646                                    ((Quantity, 22728))
0.076522
24036                                   ((Quantity, 21094))
0.076522
7019                                    ((Quantity, 22629))
0.113043
7680                                    ((Quantity, 22728))
0.074783
7647                                    ((Quantity, 22726))
0.078261
```

```
23802                                  ((Quantity, 21086))
0.076522
87893              ((Quantity, 21080), (Quantity, 21086))
0.074783
87896               ((Quantity, 21094), (Quantity, POST))
0.074783
7681                                   ((Quantity, 22727))
0.078261
87897              ((Quantity, 21094), (Quantity, 21086))
0.076522
23575              ((Quantity, 21080), (Quantity, 21086))
0.100870
1722                                   ((Quantity, 21080))
0.104348
1725                                   ((Quantity, 21094))
0.109565
87894               ((Quantity, POST), (Quantity, 21086))
0.078261
23800                                  ((Quantity, 21080))
0.081739
23576              ((Quantity, 21094), (Quantity, 21080))
0.104348
25291               ((Quantity, POST), (Quantity, 21086))
0.100870
1723                                   ((Quantity, 21086))
0.109565
87895              ((Quantity, 21094), (Quantity, 21080))
0.081739
52718              ((Quantity, 22726), (Quantity, 22728))
0.074783
23577              ((Quantity, 21094), (Quantity, 21086))
0.109565
25293               ((Quantity, 21094), (Quantity, POST))
0.104348
52717              ((Quantity, 22727), (Quantity, 22728))
0.076522
52719              ((Quantity, 22726), (Quantity, 22727))
0.078261
87892               ((Quantity, POST), (Quantity, 21080))
0.092174
7018                                   ((Quantity, 22630))
0.144348
24037               ((Quantity, POST), (Quantity, 21080))
0.100870
87898  ((Quantity, 21080), (Quantity, POST), (Quantit...
0.100870
23803               ((Quantity, POST), (Quantity, 21080))
0.104348
24039               ((Quantity, 21094), (Quantity, POST))
```

```
0.109565
87899  ((Quantity, 21094), (Quantity, POST), (Quantit...
0.104348
23805               ((Quantity, POST), (Quantity, 21086))
0.109565
87901  ((Quantity, 21094), (Quantity, POST), (Quantit...
0.109565
38089              ((Quantity, 22326), (Quantity, 22554))
0.149565

       consequent support   support  confidence       lift
leverage  \
23574             0.100870  0.073043    0.976744   9.683240  0.065500

87891             0.100870  0.055652    0.969697   9.613375  0.049863

87890             0.104348  0.055652    0.941176   9.019608  0.049482

23573             0.104348  0.073043    0.933333   8.944444  0.064877

25288             0.104348  0.069565    0.930233   8.914729  0.061762

1888              0.104348  0.092174    0.913793   8.757184  0.081648

8596              0.083478  0.066087    0.883721  10.586240  0.059844

1889              0.100870  0.092174    0.883333   8.757184  0.081648

52716             0.076522  0.050435    0.878788  11.484160  0.046043

52715             0.074783  0.050435    0.852941  11.405609  0.046013

25290             0.100870  0.069565    0.851064   8.437271  0.061320

52714             0.078261  0.050435    0.828571  10.587302  0.045671

7645              0.076522  0.060870    0.813953  10.636892  0.055147

87888             0.109565  0.055652    0.800000   7.301587  0.048030

7644              0.074783  0.060870    0.795455  10.636892  0.055147

23572             0.109565  0.073043    0.792453   7.232704  0.062944

8597              0.074783  0.066087    0.791667  10.586240  0.059844

24034             0.109565  0.059130    0.790698   7.216685  0.050937

38088             0.149565  0.052174    0.789474   5.278458  0.042290

1724              0.109565  0.078261    0.775862   7.081281  0.067209
```

7646	0.078261	0.059130	0.772727	9.873737	0.053142
24036	0.100870	0.059130	0.772727	7.660658	0.051412
7019	0.144348	0.086957	0.769231	5.329008	0.070639
7680	0.078261	0.057391	0.767442	9.806202	0.051539
7647	0.076522	0.059130	0.755556	9.873737	0.053142
23802	0.104348	0.057391	0.750000	7.187500	0.049406
87893	0.074783	0.055652	0.744186	9.951325	0.050060
87896	0.074783	0.055652	0.744186	9.951325	0.050060
7681	0.074783	0.057391	0.733333	9.806202	0.051539
87897	0.092174	0.055652	0.727273	7.890223	0.048599
23575	0.074783	0.073043	0.724138	9.683240	0.065500
1722	0.109565	0.074783	0.716667	6.541005	0.063350
1725	0.100870	0.078261	0.714286	7.081281	0.067209
87894	0.081739	0.055652	0.711111	8.699764	0.049255
23800	0.109565	0.057391	0.702128	6.408308	0.048436
23576	0.078261	0.073043	0.700000	8.944444	0.064877
25291	0.081739	0.069565	0.689655	8.437271	0.061320
1723	0.104348	0.074783	0.682540	6.541005	0.063350
87895	0.078261	0.055652	0.680851	8.699764	0.049255
52718	0.059130	0.050435	0.674419	11.405609	0.046013
23577	0.092174	0.073043	0.666667	7.232704	0.062944
25293	0.074783	0.069565	0.666667	8.914729	0.061762
52717	0.057391	0.050435	0.659091	11.484160	0.046043
52719	0.060870	0.050435	0.644444	10.587302	0.045671
87892	0.076522	0.055652	0.603774	7.890223	0.048599
7018	0.113043	0.086957	0.602410	5.329008	0.070639

24037	0.076522	0.059130	0.586207	7.660658	0.051412
87898	0.057391	0.055652	0.551724	9.613375	0.049863
23803	0.076522	0.057391	0.550000	7.187500	0.049406
24039	0.074783	0.059130	0.539683	7.216685	0.050937
87899	0.059130	0.055652	0.533333	9.019608	0.049482
23805	0.081739	0.057391	0.523810	6.408308	0.048436
87901	0.069565	0.055652	0.507937	7.301587	0.048030
38089	0.066087	0.052174	0.348837	5.278458	0.042290

	conviction	zhangs_metric
23574	38.662609	0.969209
87891	29.671304	0.950530
87890	15.226087	0.945009
23573	13.434783	0.963612
25288	12.837681	0.959586
1888	10.389565	0.985183
8596	7.882087	0.978730
1889	7.706832	0.989009
52716	7.618696	0.968507
52715	6.291478	0.969660
25290	6.037019	0.959943
52714	5.376812	0.964240
7645	4.963696	0.979216
87888	4.452174	0.927570
7644	4.523285	0.981060
23572	4.290277	0.949234
8597	4.441043	0.988016
24034	4.254300	0.931059
38088	4.039565	0.867908
1724	3.972709	0.955126
7646	4.055652	0.973192
24036	3.956174	0.941509
7019	3.707826	0.915882
7680	3.963478	0.970608
7647	3.777866	0.975028
23802	3.582609	0.932203
87893	3.616759	0.972216
87896	3.616759	0.972216
7681	3.469565	0.974271
87897	3.328696	0.945621
23575	3.353913	0.997329

```
1722     3.142711        0.945812
1725     3.146957        0.964453
87894    3.178595        0.960200
23800    2.989317        0.919077
23576    3.072464        0.991678
25291    2.958841        0.980368
1723     2.821304        0.951354
87895    2.888116        0.963838
52718    2.889814        0.986064
23577    2.723478        0.967773
25293    2.775652        0.991262
52717    2.764986        0.988571
52719    2.641304        0.982433
87892    2.330683        0.961925
7018     2.230830        0.949390
24037    2.231739        0.967004
87898    2.102742        0.996494
23803    2.052174        0.961165
24039    2.009955        0.967429
87899    2.016149        0.992718
23805    1.928348        0.947798
87901    1.890884        0.969238
38089    1.434224        0.953102
```

5. Application

For example, a member purchased a product with stock code 85123A ...

```
def prdct_name_finder(data,stckcde):
    product_name = data[data["StockCode"] == stckcde]
[["Description"]].values[0].tolist()
    return product_name

def arl_recommender(rules_df, product_id, rec_count):

    sorted_rules = rules_df.sort_values("lift", ascending=False)
    recommendation_list = []
    recommendation_list_name = []

    for i, product in enumerate(sorted_rules["antecedents"]):
        for j in list(product):
            if j[1] == product_id:
                for k in list(sorted_rules.iloc[i]["consequents"]):
                    if k[1] not in recommendation_list:
                        recommendation_list.append(k[1])
    added_product = prdct_name_finder(df,product_id)
    print(f"Added to Cart:           {added_product[0]}\n\n")
    print(f"Members Who Bought This Also Bought:\n\n")
    for i in range(0,rec_count):
```

```
recommendation_list_name.append(prdct_name_finder(df,recommendation_li
st[i]))
        print(f"                          {recommendation_list_name[i]
[0]}\n")

arl_recommender(rules, "84997C", 3)

Added to Cart:            BLUE 3 PIECE POLKADOT CUTLERY SET

Members Who Bought This Also Bought:

                          RED 3 PIECE RETROSPOT CUTLERY SET

                          GREEN 3 PIECE POLKADOT CUTLERY SET

                          POSTAGE

arl_recommender(rules, "15056BL", 3)

Added to Cart:            EDWARDIAN PARASOL BLACK

Members Who Bought This Also Bought:

                          EDWARDIAN PARASOL RED

                          POSTAGE

                          RED RETROSPOT UMBRELLA
```