Name : Chinmayee Taralkar
Roll No : BECOB262

Problem Statement : Apply a-priori algorithm to find frequently occurring items from given data and generate strong association rules using support and confidence thresholds. For Example: Market Basket Analysis

```
Code :
```

#Importing the required libraries

In [38]:
```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

#Loading the data

In [6]:
```
data = pd.read_excel("C:\\Users\\Lenovo\\Downloads\\Online Retail.xlsx")
```

In [39]:
```
data.head()
```

Out[39]:

	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

```
#Checking the columns of data
```

In [8]:
```
data.columns
```

Out[8]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [9]:

```
data.shape
```

Out[9]:

```
(541909, 8)
```

In [12]:

```
data.isnull().values.any()
```

Out[12]:

```
True
```

In [13]:

```
data.isnull().sum()
```

Out[13]:

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

```
#Cleaning data
```

In [40]:

```
data['Description']=data['Description'].str.strip()
data.dropna(axis=0,subset=['InvoiceNo'],inplace=True)
data['InvoiceNo']=data['InvoiceNo'].astype('str')
data=data[~data['InvoiceNo'].str.contains('C')]
```

In [41]:

```
data.head()
```

Out[41]:

	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

In [42]:

```
data['Country'].value_counts()
```

Out[42]:

```
United Kingdom          487622
Germany                   9042
France                    8408
EIRE                      7894
Spain                     2485
Netherlands               2363
Belgium                   2031
Switzerland               1967
Portugal                  1501
Australia                 1185
Norway                    1072
Italy                      758
Channel Islands            748
Finland                    685
Cyprus                     614
Sweden                     451
Unspecified                446
Austria                    398
Denmark                    380
Poland                     330
Japan                      321
Israel                     295
Hong Kong                  284
Singapore                  222
Iceland                    182
USA                        179
Canada                     151
Greece                     145
Malta                      112
United Arab Emirates        68
European Community          60
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              25
Bahrain                     18
Saudi Arabia                 9
Name: Country, dtype: int64
```

In [43]:

```
data.Country.unique()
```

Out[43]:

```
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)
```

In [64]:
```
#transactions done in Netherlands
basket_Nether = (data[data['Country'] == "Netherlands"]
                        .groupby(['InvoiceNo','Description'])['Quantity']
                        .sum().unstack().reset_index().fillna(0).set_index('Invoi
```

In [20]:
```
#transactions done in Germany
basket_Germany = (data[data['Country'] == "Germany"]
                        .groupby(['InvoiceNo','Description'])['Quantity']
                        .sum().unstack().reset_index().fillna(0).set_index('Invoi
```

In [21]:
```
#transactions done in France
basket_France = (data[data['Country'] == "France"]
                        .groupby(['InvoiceNo','Description'])['Quantity']
                        .sum().unstack().reset_index().fillna(0).set_index('Invoi
```

In [22]:
```
#transactions done in Spain
basket_Spain = (data[data['Country'] == "Spain"]
                        .groupby(['InvoiceNo','Description'])['Quantity']
                        .sum().unstack().reset_index().fillna(0).set_index('Invoi
```

In [65]:
```
#encoding data
def encode(x):
    if x<=0:
        return 0
    if x>=1:
        return 1
```

In [66]:
```
#encoding EIRE
basket_encoded = basket_Nether.applymap(encode)
basket_Nether = basket_encoded
```

In [67]:
```
#encoding Germany
basket_encoded = basket_Germany.applymap(encode)
basket_Germany = basket_encoded
```

In [68]:
```
#encoding France
basket_encoded = basket_France.applymap(encode)
basket_France = basket_encoded
```

In [69]:
```
#encoding Spain
basket_encoded = basket_Spain.applymap(encode)
basket_Spain = basket_encoded
```

In [70]:
```
# Building the model for EIRE
freq_Items_Nether = apriori(basket_Nether, min_support = 0.05, use_colnames = Tru
```

In [71]:

```
# Collecting the inferred rules
rulesNether = association_rules(freq_Items_Nether, metric ="lift", min_threshold
rulesNether = rulesNether.sort_values(['confidence', 'lift'], ascending =[False,
rulesNether.head()
```

Out[71]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
650	(FOLDING BUTTERFLY MIRROR HOT PINK)	(FOLDING BUTTERFLY MIRROR RED)	0.052632	0.052632	0.052632	1.0	19.0	0.049861
651	(FOLDING BUTTERFLY MIRROR RED)	(FOLDING BUTTERFLY MIRROR HOT PINK)	0.052632	0.052632	0.052632	1.0	19.0	0.049861
1351	(FOOD CONTAINER SET 3 LOVE HEART, CARD DOLLY G...	(10 COLOUR SPACEBOY PEN)	0.052632	0.052632	0.052632	1.0	19.0	0.049861
1354	(10 COLOUR SPACEBOY PEN)	(FOOD CONTAINER SET 3 LOVE HEART, CARD DOLLY G...	0.052632	0.052632	0.052632	1.0	19.0	0.049861
1381	(STRAWBERRY LUNCH BOX WITH CUTLERY, CARD DOLLY...	(10 COLOUR SPACEBOY PEN)	0.052632	0.052632	0.052632	1.0	19.0	0.049861

In [73]:

```
# Building the model for germany
freq_Items_Germany = apriori(basket_Germany, min_support = 0.05, use_colnames = T
```

In [74]:

```
# Collecting the inferred rules
rulesGermany = association_rules(freq_Items_Germany, metric ="lift", min_threshol
rulesGermany = rulesGermany.sort_values(['confidence', 'lift'], ascending =[False
rulesGermany.head()
```

Out[74]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
15	(PLASTERS IN TIN STRONGMAN)	(POSTAGE)	0.053068	0.620232	0.051410	0.968750	1.561915	0.018495
25	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE)	0.119403	0.620232	0.114428	0.958333	1.545120	0.040370
40	(ROUND SNACK BOXES SET OF4 WOODLAND , ROUND SN...	(POSTAGE)	0.099502	0.620232	0.094527	0.950000	1.531684	0.032813
13	(PLASTERS IN TIN SPACEBOY)	(POSTAGE)	0.081260	0.620232	0.076285	0.938776	1.513587	0.025885
27	(ROUND SNACK BOXES SET OF4 WOODLAND)	(POSTAGE)	0.185738	0.620232	0.170813	0.919643	1.482740	0.055612

In [75]:

```
# Building the model for France
freq_Items_France = apriori(basket_France, min_support = 0.05, use_colnames = Tru
```

In [78]:

```
# Collecting the inferred rules
rulesFrance = association_rules(freq_Items_France, metric ="lift", min_threshold
rulesFrance = rulesFrance.sort_values(['confidence', 'lift'], ascending =[False,
rulesFrance.head()
```

Out[78]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
25	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.065076	0.650759	0.065076	1.000000	1.536667	0.022727
186	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.086768	0.108460	0.084599	0.975000	8.989500	0.075188
185	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.086768	0.117137	0.084599	0.975000	8.323611	0.074435
192	(POSTAGE, SET/6 RED SPOTTY PAPER CUPS, SET/20 ...	(SET/6 RED SPOTTY PAPER PLATES)	0.071584	0.108460	0.069414	0.969697	8.940606	0.061650
191	(POSTAGE, SET/6 RED SPOTTY PAPER PLATES, SET/2...	(SET/6 RED SPOTTY PAPER CUPS)	0.071584	0.117137	0.069414	0.969697	8.278339	0.061029

In [77]:

```
# Building the model for Spain
freq_Items_Spain = apriori(basket_Spain, min_support = 0.05, use_colnames = True)
```

In [79]:

```
# Collecting the inferred rules
rulesSpain = association_rules(freq_Items_Spain, metric ="lift", min_threshold = 
rulesSpain = rulesSpain.sort_values(['confidence', 'lift'], ascending =[False, Fa
rulesSpain.head()
```

Out[79]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
48	(POPPY'S PLAYHOUSE KITCHEN)	(POPPY'S PLAYHOUSE BEDROOM)	0.066667	0.076190	0.066667	1.0	13.125000	0.06158
102	(POPPY'S PLAYHOUSE KITCHEN, POSTAGE)	(POPPY'S PLAYHOUSE BEDROOM)	0.057143	0.076190	0.057143	1.0	13.125000	0.05278
31	(GREEN REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.057143	0.104762	0.057143	1.0	9.545455	0.05115
116	(PACK OF 72 RETROSPOT CAKE CASES, 6 RIBBONS RU...	(POSTAGE, ASSORTED COLOUR BIRD ORNAMENT)	0.057143	0.104762	0.057143	1.0	9.545455	0.05115
73	(PACK OF 72 RETROSPOT CAKE CASES, 6 RIBBONS RU...	(ASSORTED COLOUR BIRD ORNAMENT)	0.057143	0.114286	0.057143	1.0	8.750000	0.05061

In [80]:
```
#extracting rules for EIRE based on condition
rulesNether[(rulesNether['lift']>=2)&(rulesNether['confidence']>=0.3)]
```

5753	(FOLDING BUTTERFLY MIRROR RED)	BUTTERFLY MIRROR HOT PINK, SPACEBOY B...	0.052632	0.052632	0.052632	1.0000	19.00000
9601	(FOOD CONTAINER SET 3 LOVE HEART, COWBOYS AND ...	(10 COLOUR SPACEBOY PEN)	0.052632	0.052632	0.052632	1.0000	19.00000
9606	(FOOD CONTAINER SET 3 LOVE HEART, CARD DOLLY G...	(COWBOYS AND INDIANS BIRTHDAY CARD, 10 COLOUR ...	0.052632	0.052632	0.052632	1.0000	19.00000
9607	(COWBOYS AND INDIANS BIRTHDAY CARD, 10 COLOUR ...	(FOOD CONTAINER SET 3 LOVE HEART, CARD DOLLY G...	0.052632	0.052632	0.052632	1.0000	19.00000

In [81]:

```
#extracting rules for Germany based on condition
rulesGermany[(rulesGermany['lift']>=2)&(rulesGermany['confidence']>=0.3)]
```

Out[81]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
35	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.119403	0.185738	0.099502	0.833333	4.486607	0.0773
39	(POSTAGE, ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.114428	0.185738	0.094527	0.826087	4.447593	0.0732
43	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE, ROUND SNACK BOXES SET OF4 WOODLAND)	0.119403	0.170813	0.094527	0.791667	4.634709	0.0741
37	(SPACEBOY LUNCH BOX)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.077944	0.185738	0.053068	0.680851	3.665653	0.0385
9	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.087894	0.104478	0.051410	0.584906	5.598383	0.0422
38	(POSTAGE, ROUND SNACK BOXES SET OF4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.170813	0.119403	0.094527	0.553398	4.634709	0.0741
19	(PLASTERS IN TIN WOODLAND ANIMALS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.104478	0.185738	0.056385	0.539683	2.905612	0.0369
34	(ROUND SNACK BOXES SET OF4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.185738	0.119403	0.099502	0.535714	4.486607	0.0773
42	(ROUND SNACK BOXES SET OF4 WOODLAND)	(POSTAGE, ROUND SNACK BOXES SET OF 4 FRUITS)	0.185738	0.114428	0.094527	0.508929	4.447593	0.0732

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
8	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.104478	0.087894	0.051410	0.492063	5.598383	0.0422
18	(ROUND SNACK BOXES SET OF4 WOODLAND)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.185738	0.104478	0.056385	0.303571	2.905612	0.0369

In [62]:

```
#extracting rules for France based on condition
rulesFrance[(rulesFrance['lift']>=2)&(rulesFrance['confidence']>=0.3)]
```

Out[62]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
186	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.086768	0.108460	0.084599	0.975000	8.989500
185	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.086768	0.117137	0.084599	0.975000	8.323611
192	(POSTAGE, SET/6 RED SPOTTY PAPER CUPS, SET/20 ...	(SET/6 RED SPOTTY PAPER PLATES)	0.071584	0.108460	0.069414	0.969697	8.940606

In [63]:
```
#extracting rules based on condition
rulesSpain[(rulesSpain['lift']>=2)&(rulesSpain['confidence']>=0.3)]
```

Out[63]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	
48	(POPPY'S PLAYHOUSE KITCHEN)	(POPPY'S PLAYHOUSE BEDROOM)	0.066667	0.076190	0.066667	1.000000	13.125000	(
102	(POPPY'S PLAYHOUSE KITCHEN, POSTAGE)	(POPPY'S PLAYHOUSE BEDROOM)	0.057143	0.076190	0.057143	1.000000	13.125000	(
31	(GREEN REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.057143	0.104762	0.057143	1.000000	9.545455	(
116	(PACK OF 72 RETROSPOT CAKE CASES, 6 RIBBONS RU...	(POSTAGE, ASSORTED COLOUR BIRD ORNAMENT)	0.057143	0.104762	0.057143	1.000000	9.545455	(