6) Implement any 2 Association Rule Mining techniques using any data analytics tool.

In [14]:
```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
import numpy as np
import pandas as pd
```

In [15]:
```
data = pd.read_csv(r"C:\Users\admin\Downloads\data.csv", encoding='unicode_escape')
```

In [16]:
```
from mlxtend.frequent_patterns import apriori, association_rules
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

In [17]:
```
data.columns
```

Out[17]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [18]:
```
data.Country.unique()
```

Out[18]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

In [19]:
```
data['Description'] = data['Description'].str.strip()
```

In [20]:
```
data.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
```

In [21]:
```
data['InvoiceNo'] = data['InvoiceNo'].astype('str')
```

In [22]:
```
data = data[~data['InvoiceNo'].str.contains('C')]
```

In [23]:
```
# Dropping all transactions which were done on credit
data = data[~data['InvoiceNo'].str.contains('C')]
basket_France = (data[data['Country'] == "France"]
                 .groupby(['InvoiceNo', 'Description'])['Quantity']
                 .sum().unstack().reset_index().fillna(0)
                 .set_index('InvoiceNo'))
```

In [24]:
```

# Transactions done in the United Kingdom
basket_UK = (data[data['Country'] =="United Kingdom"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
```

```
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [25]:
```
# Transactions done in Portugal
basket_Por = (data[data['Country'] =="Portugal"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [26]:
```
basket_Sweden = (data[data['Country'] =="Sweden"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [27]:
```
basket_France = basket_France.applymap(lambda x: 1 if x > 0 else 0)

frq_items = apriori(basket_France, min_support=0.05, use_colnames=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_5340\4136799206.py:1: FutureWarning: Data
Frame.applymap has been deprecated. Use DataFrame.map instead.
  basket_France = basket_France.applymap(lambda x: 1 if x > 0 else 0)
C:\Users\admin\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110:
DeprecationWarning: DataFrames with non-bool types result in worse computationalperfo
rmance and their support might be discontinued in the future.Please use a DataFrame w
ith bool type
  warnings.warn(
```

In [28]:
```
# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
```

In [29]:
```
print(rules.head())
```

```
                                        antecedents  \
45                       (JUMBO BAG WOODLAND ANIMALS)
258  (PLASTERS IN TIN CIRCUS PARADE, RED TOADSTOOL ...
271  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
301  (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...
302  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...

                      consequents  antecedent support  consequent support  \
45                        (POSTAGE)            0.076531            0.765306
258                       (POSTAGE)            0.051020            0.765306
271                       (POSTAGE)            0.053571            0.765306
301  (SET/6 RED SPOTTY PAPER PLATES)            0.102041            0.127551
302    (SET/6 RED SPOTTY PAPER CUPS)            0.102041            0.137755

      support  confidence      lift  leverage  conviction  zhangs_metric
45   0.076531       1.000  1.306667  0.017961         inf       0.254144
258  0.051020       1.000  1.306667  0.011974         inf       0.247312
271  0.053571       1.000  1.306667  0.012573         inf       0.247978
301  0.099490       0.975  7.644000  0.086474   34.897959       0.967949
302  0.099490       0.975  7.077778  0.085433   34.489796       0.956294
```

In [30]:
```
from mlxtend.frequent_patterns import apriori, association_rules
```

In [31]:
```
data.columns
```

Out[31]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [32]:
```
data.Country.unique()
```

Out[32]:
```
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 [33]:
```
data['Description'] = data['Description'].str.strip()
```

In [34]:
```
# Dropping the rows without any invoice number
data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)
data['InvoiceNo'] = data['InvoiceNo'].astype('str')
```

In [35]:
```
# Dropping all transactions which were done on credit
data = data[~data['InvoiceNo'].str.contains('C')]
basket_France = (data[data['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [36]:
```
# Transactions done in the United Kingdom
basket_UK = (data[data['Country'] =="United Kingdom"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [37]:
```
# Transactions done in Portugal
basket_Por = (data[data['Country'] =="Portugal"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [38]:
```
basket_Sweden = (data[data['Country'] =="Sweden"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

In [39]:
```
basket_France = basket_France.applymap(lambda x: 1 if x > 0 else 0)

frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_5340\420125865.py:1: FutureWarning: DataF
rame.applymap has been deprecated. Use DataFrame.map instead.
  basket_France = basket_France.applymap(lambda x: 1 if x > 0 else 0)
C:\Users\admin\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110:
DeprecationWarning: DataFrames with non-bool types result in worse computationalperfo
rmance and their support might be discontinued in the future.Please use a DataFrame w
ith bool type
  warnings.warn(
```

In [40]:
```
# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                                     antecedents  \
45                      (JUMBO BAG WOODLAND ANIMALS)
258  (PLASTERS IN TIN CIRCUS PARADE, RED TOADSTOOL ...
271  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
301  (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...
302  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...

                       consequents  antecedent support  consequent support  \
45                        (POSTAGE)            0.076531            0.765306
258                       (POSTAGE)            0.051020            0.765306
271                       (POSTAGE)            0.053571            0.765306
301  (SET/6 RED SPOTTY PAPER PLATES)            0.102041            0.127551
302    (SET/6 RED SPOTTY PAPER CUPS)            0.102041            0.137755

      support  confidence      lift  leverage  conviction  zhangs_metric
45   0.076531       1.000  1.306667  0.017961         inf       0.254144
258  0.051020       1.000  1.306667  0.011974         inf       0.247312
271  0.053571       1.000  1.306667  0.012573         inf       0.247978
301  0.099490       0.975  7.644000  0.086474   34.897959       0.967949
302  0.099490       0.975  7.077778  0.085433   34.489796       0.956294
```