```
!pip install apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5974 sha256=837
  Stored in directory: /root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import pandas as pd
import numpy as np
from apyori import apriori
```

```
store_data = pd.read_csv("GroceryStoreDataSet.csv",encoding='latin-1')
```

```
store_data.head()
```

	MILK,BREAD,BISCUIT
0	BREAD,MILK,BISCUIT,CORNFLAKES
1	BREAD,TEA,BOURNVITA
2	JAM,MAGGI,BREAD,MILK
3	MAGGI,TEA,BISCUIT
4	BREAD,TEA,BOURNVITA

```
store_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19 entries, 0 to 18
Data columns (total 1 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   MILK,BREAD,BISCUIT  19 non-null     object
dtypes: object(1)
memory usage: 280.0+ bytes
```

```
#Renaming the column to PRODUCTS

store_data.rename(columns = {'MILK,BREAD,BISCUIT':'PRODUCTS'}, inplace = True)
```

```
# Creating a list of items in every transaction.
#The list will work as a training set from where we can generate the list of Association Rule

list_items = list(store_data["PRODUCTS"].apply(lambda x:x.split(',')))
list_items
```

```
[['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['JAM', 'MAGGI', 'BREAD', 'MILK'],
 ['MAGGI', 'TEA', 'BISCUIT'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['MAGGI', 'TEA', 'CORNFLAKES'],
 ['MAGGI', 'BREAD', 'TEA', 'BISCUIT'],
 ['JAM', 'MAGGI', 'BREAD', 'TEA'],
 ['BREAD', 'MILK'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'COCK'],
 ['BREAD', 'SUGER', 'BISCUIT'],
 ['COFFEE', 'SUGER', 'CORNFLAKES'],
 ['BREAD', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['TEA', 'MILK', 'COFFEE', 'CORNFLAKES']]
```

```
from mlxtend.preprocessing import TransactionEncoder

encode = TransactionEncoder()
encode_data = encode.fit(list_items).transform(list_items)
store_data = pd.DataFrame(encode_data,columns = encode.columns_)
store_data.head()
```

	BISCUIT	BOURNVITA	BREAD	COCK	COFFEE	CORNFLAKES	JAM	MAGGI	MILK	SUGER	TEA
0	True	False	True	False	False	True	False	False	True	False	False
1	False	True	True	False	False	False	False	False	False	False	True
2	False	False	True	False	False	False	True	True	True	False	False
3	True	False	False	False	False	False	False	True	False	False	True
4	False	True	True	False	False	False	False	False	False	False	True

```
#Applying apriori algorithm

association_rules = apriori(list_items, min_support=0.0045, min_confidence=0.2, min_lift=3, m
association_list = list(association_rules)
```

```
#Generating the association rule between the items

for i in range(0, len(association_list)):
    print(association_list[i][0])
```

```
frozenset({'JAM', 'MAGGI'})
frozenset({'BISCUIT', 'CORNFLAKES', 'BREAD'})
frozenset({'BISCUIT', 'COCK', 'COFFEE'})
frozenset({'BISCUIT', 'COCK', 'CORNFLAKES'})
frozenset({'BISCUIT', 'COFFEE', 'CORNFLAKES'})
frozenset({'BISCUIT', 'MILK', 'CORNFLAKES'})
frozenset({'BISCUIT', 'TEA', 'MAGGI'})
frozenset({'BOURNVITA', 'COFFEE', 'SUGER'})
frozenset({'MILK', 'CORNFLAKES', 'BREAD'})
frozenset({'JAM', 'BREAD', 'MAGGI'})
frozenset({'JAM', 'MILK', 'BREAD'})
frozenset({'COCK', 'COFFEE', 'CORNFLAKES'})
frozenset({'MILK', 'COFFEE', 'CORNFLAKES'})
frozenset({'COFFEE', 'CORNFLAKES', 'TEA'})
frozenset({'MILK', 'COFFEE', 'TEA'})
frozenset({'MILK', 'CORNFLAKES', 'TEA'})
frozenset({'JAM', 'MILK', 'MAGGI'})
frozenset({'JAM', 'TEA', 'MAGGI'})
frozenset({'BISCUIT', 'MILK', 'CORNFLAKES', 'BREAD'})
frozenset({'BISCUIT', 'TEA', 'BREAD', 'MAGGI'})
frozenset({'BISCUIT', 'COCK', 'COFFEE', 'CORNFLAKES'})
frozenset({'JAM', 'MILK', 'BREAD', 'MAGGI'})
frozenset({'JAM', 'TEA', 'BREAD', 'MAGGI'})
frozenset({'MILK', 'COFFEE', 'CORNFLAKES', 'TEA'})
```

```
#Display Rule, Support, Confidence and lift ratio for every above association rule

for item in association_list:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " --> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("-----------------------------------------------------")
```

```
Rule: JAM --> MAGGI
Support: 0.10526315789473684
Confidence: 1.0
Lift: 3.8000000000000003
-----------------------------------------------------
Rule: BISCUIT --> CORNFLAKES
Support: 0.05263157894736842
Confidence: 1.0
Lift: 3.166666666666667
-----------------------------------------------------
Rule: BISCUIT --> COCK
Support: 0.10526315789473684
Confidence: 0.6666666666666666
```

```
Lift: 6.333333333333333
-----------------------------------------------------
Rule: BISCUIT --> COCK
Support: 0.10526315789473684
Confidence: 0.3333333333333333
Lift: 3.1666666666666665
-----------------------------------------------------
Rule: BISCUIT --> COFFEE
Support: 0.10526315789473684
Confidence: 0.3333333333333333
Lift: 3.1666666666666665
-----------------------------------------------------
Rule: BISCUIT --> MILK
Support: 0.05263157894736842
Confidence: 1.0
Lift: 3.166666666666667
-----------------------------------------------------
Rule: BISCUIT --> TEA
Support: 0.10526315789473684
Confidence: 0.4
Lift: 3.8000000000000003
-----------------------------------------------------
Rule: BOURNVITA --> COFFEE
Support: 0.05263157894736842
Confidence: 1.0
Lift: 3.166666666666667
-----------------------------------------------------
Rule: MILK --> CORNFLAKES
Support: 0.05263157894736842
Confidence: 0.25
Lift: 4.75
-----------------------------------------------------
Rule: JAM --> BREAD
Support: 0.10526315789473684
Confidence: 1.0
Lift: 6.333333333333334
-----------------------------------------------------
Rule: JAM --> MILK
Support: 0.05263157894736842
Confidence: 0.5
Lift: 3.166666666666667
-----------------------------------------------------
Rule: COCK --> COFFEE
Support: 0.10526315789473684
Confidence: 0.6666666666666666
```

● ✕