In [1]:

```
import pandas as pd
import numpy as np

from mlxtend.frequent_patterns import apriori, association_rules
```

In [2]:

```
data = pd.read_excel('BreadBasket.xlsx')
data.head(5)
```

Out[2]:

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

In [3]:

```
data.shape
```

Out[3]:

```
(23, 2)
```

In [4]:

```
dataset = list(data["products"].apply(lambda x:x.split(",") ))
dataset
```

Out[4]:

```
[['MILK', 'BREAD', 'BISCUIT'],
 ['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'],
 ['MILK', 'BREAD', 'BISCUIT'],
 ['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
 ['BREAD', 'TEA', 'BOURNVITA']]
```

Apriori algorithm

In [5]:

```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
te_ary
```

Out[5]:

```
array([[ True, False,  True, False, False, False, False, False,  True,
        False, False],
       [ True, False,  True, False, False,  True, False, False,  True,
        False, False],
       [False,  True,  True, False, False, False, False, False, False,
        False,  True],
       [False, False,  True, False, False, False,  True,  True,  True,
        False, False],
       [ True, False, False, False, False, False, False,  True, False,
        False,  True],
       [False,  True,  True, False, False, False, False, False, False,
        False,  True],
       [False, False, False, False, False,  True, False,  True, False,
        False,  True],
       [ True, False,  True, False, False, False, False,  True, False,
        False,  True],
       [False, False,  True, False, False, False,  True,  True, False,
        False,  True],
       [False, False,  True, False, False, False, False, False,  True,
        False, False],
       [ True, False, False,  True,  True,  True, False, False, False,
        False, False],
       [ True, False, False,  True,  True,  True, False, False, False,
        False, False],
       [False,  True, False, False,  True, False, False, False, False,
         True, False],
       [False, False,  True,  True,  True, False, False, False, False,
        False, False],
       [ True, False,  True, False, False, False, False, False, False,
         True, False],
       [False, False, False, False,  True,  True, False, False, False,
         True, False],
       [False,  True,  True, False, False, False, False, False, False,
         True, False],
       [False, False,  True, False,  True, False, False, False, False,
         True, False],
       [False, False,  True, False,  True, False, False, False, False,
         True, False],
       [False, False, False, False,  True,  True, False, False,  True,
        False,  True],
       [ True, False,  True, False, False, False, False, False,  True,
        False, False],
       [ True, False,  True, False, False,  True, False, False,  True,
        False, False],
       [False,  True,  True, False, False, False, False, False, False,
        False,  True]])
```

In [6]:

```
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[6]:

	BISCUIT	BOURNVITA	BREAD	COCK	COFFEE	CORNFLAKES	JAM	MAGGI	MILK	S
0	True	False	True	False	False	False	False	False	True	
1	True	False	True	False	False	True	False	False	True	
2	False	True	True	False	False	False	False	False	False	
3	False	False	True	False	False	False	True	True	True	
4	True	False	False	False	False	False	False	True	False	
5	False	True	True	False	False	False	False	False	False	
6	False	False	False	False	False	True	False	True	False	
7	True	False	True	False	False	False	False	True	False	
8	False	False	True	False	False	False	True	True	False	
9	False	False	True	False	False	False	False	False	True	
10	True	False	False	True	True	True	False	False	False	
11	True	False	False	True	True	True	False	False	False	
12	False	True	False	False	True	False	False	False	False	
13	False	False	True	True	True	False	False	False	False	
14	True	False	True	False	False	False	False	False	False	
15	False	False	False	False	True	True	False	False	False	
16	False	True	True	False	False	False	False	False	False	
17	False	False	True	False	True	False	False	False	False	
18	False	False	True	False	True	False	False	False	False	
19	False	False	False	False	True	True	False	False	True	
20	True	False	True	False	False	False	False	False	True	
21	True	False	True	False	False	True	False	False	True	
22	False	True	True	False	False	False	False	False	False	

In [7]:

```
from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.2)
```

Out[7]:

	support	itemsets
0	0.391304	(0)
1	0.217391	(1)
2	0.695652	(2)
3	0.347826	(4)
4	0.304348	(5)
5	0.217391	(7)
6	0.304348	(8)
7	0.260870	(9)
8	0.347826	(10)
9	0.260870	(0, 2)
10	0.260870	(8, 2)
11	0.217391	(2, 10)

In [8]:

```
apriori(df, min_support=0.2, use_colnames=True)
```

Out[8]:

	support	itemsets
0	0.391304	(BISCUIT)
1	0.217391	(BOURNVITA)
2	0.695652	(BREAD)
3	0.347826	(COFFEE)
4	0.304348	(CORNFLAKES)
5	0.217391	(MAGGI)
6	0.304348	(MILK)
7	0.260870	(SUGER)
8	0.347826	(TEA)
9	0.260870	(BREAD, BISCUIT)
10	0.260870	(BREAD, MILK)
11	0.217391	(BREAD, TEA)

In [9]:

```
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
```

In [10]:

```
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1.0)
```

In [11]:

```
print(frequent_itemsets)
print(rules)
```

```
     support          itemsets
0   0.391304         (BISCUIT)
1   0.217391       (BOURNVITA)
2   0.695652           (BREAD)
3   0.347826          (COFFEE)
4   0.304348      (CORNFLAKES)
5   0.217391           (MAGGI)
6   0.304348            (MILK)
7   0.260870           (SUGER)
8   0.347826             (TEA)
9   0.260870  (BREAD, BISCUIT)
10  0.260870     (BREAD, MILK)
11  0.217391      (BREAD, TEA)
  antecedents consequents  antecedent support  consequent support  support
\
0     (BREAD)      (MILK)            0.695652            0.304348  0.26087
1      (MILK)     (BREAD)            0.304348            0.695652  0.26087

   confidence      lift  leverage  conviction  zhangs_metric
0    0.375000  1.232143  0.049149    1.113043       0.619048
1    0.857143  1.232143  0.049149    2.130435       0.270833
```

FP Growth

In [12]:

```
import pandas as pd
from mlxtend.frequent_patterns import fpgrowth
```

In [13]:

```
fpgrowth(df, min_support=0.2, use_colnames=True)
```

Out[13]:

	support	itemsets
0	0.695652	(BREAD)
1	0.391304	(BISCUIT)
2	0.304348	(MILK)
3	0.304348	(CORNFLAKES)
4	0.347826	(TEA)
5	0.217391	(BOURNVITA)
6	0.217391	(MAGGI)
7	0.347826	(COFFEE)
8	0.260870	(SUGER)
9	0.260870	(BREAD, BISCUIT)
10	0.260870	(BREAD, MILK)
11	0.217391	(BREAD, TEA)

In [14]:

```
frequent_itemsets = fpgrowth(df, min_support=0.2, use_colnames=True)


from mlxtend.frequent_patterns import association_rules

rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1.0)
```

In [15]:

```
print(frequent_itemsets)
```

```
     support          itemsets
0   0.695652           (BREAD)
1   0.391304         (BISCUIT)
2   0.304348            (MILK)
3   0.304348      (CORNFLAKES)
4   0.347826             (TEA)
5   0.217391       (BOURNVITA)
6   0.217391           (MAGGI)
7   0.347826          (COFFEE)
8   0.260870           (SUGER)
9   0.260870  (BREAD, BISCUIT)
10  0.260870     (BREAD, MILK)
11  0.217391      (BREAD, TEA)
```

In [16]:

```
print(rules)
```

```
  antecedents consequents  antecedent support  consequent support  support
\
0     (BREAD)      (MILK)            0.695652            0.304348  0.26087
1      (MILK)     (BREAD)            0.304348            0.695652  0.26087

   confidence      lift  leverage  conviction  zhangs_metric
0    0.375000  1.232143  0.049149    1.113043       0.619048
1    0.857143  1.232143  0.049149    2.130435       0.270833
```