In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori,association_rules
from mlxtend.preprocessing import TransactionEncoder
```

In [18]:
```
df = pd.read_csv('BreadBasket.csv')
df.head(10)
data = list(df["products"].apply(lambda x:x.split(",") ))
data
```

Out[18]:
```
[['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']]
```

In [19]:
```
df.shape
```

Out[19]:
```
(20, 2)
```

In [20]:
```
te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
te_ary
```

Out[20]:
```
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]])
```

In [21]:
```
dfm = pd.DataFrame(te_ary, columns=te.columns_)
dfm
```

Out[21]:

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

In [23]:
```
apriori(dfm, min_support=0.6)
```

Out[23]:

	support	itemsets
0	0.65	(2)

In [25]:
```
apriori(dfm, min_support=0.6, use_colnames=True)
```

Out[25]:

	support	itemsets
0	0.65	(BREAD)

In [27]:
```
frequent_itemsets = apriori(dfm, min_support=0.6, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

Out[27]:

	support	itemsets	length
0	0.65	(BREAD)	1

In [33]:
```
fpgrowth(dfm, min_support=0.6, use_colnames=True)
```

Out[33]:

	support	itemsets
0	0.65	(BREAD)

In [36]:
```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
dfm = pd.DataFrame(te_ary, columns=te.columns_)
```

In [38]:
```
from mlxtend.frequent_patterns import apriori

%timeit -n 100 -r 10 apriori(dfm, min_support=0.6)
```

```
7.92 ms ± 1.63 ms per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

In [40]:
```
%timeit -n 100 -r 10 apriori(dfm, min_support=0.6, low_memory=True)
```

```
9.46 ms ± 770 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

In [42]:
```
from mlxtend.frequent_patterns import fpgrowth

%timeit -n 100 -r 10 fpgrowth(dfm, min_support=0.6)
```

```
6.71 ms ± 747 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```