```
import pandas as pd
import numpy as np
import os
from mlxtend.frequent_patterns import apriori, fpgrowth
from mlxtend.frequent_patterns import association_rules
from sklearn.preprocessing import OneHotEncoder

os.chdir('C:\\Users\\user\\Downloads')

# Read the Excel file
df = pd.read_excel('bread_basket.xlsx',names = ['Tx', 'products'])

# Display the first few rows of the DataFrame
print(df.head(5))
```

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

```
df.shape
```

```
(21, 2)
```

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

```
[['products'],
 ['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']]
```

APRIORI ALGORITHM

```
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ary = te.fit(df).transform(df)
te_ary
```

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

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

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

    SUGER    TEA  products
0   False  False      True
```

```
1   False  False     False
2   False  False     False
3   False   True     False
4   False  False     False
5   False   True     False
6   False   True     False
7   False   True     False
8   False   True     False
9   False   True     False
10  False  False     False
11  False  False     False
12  False  False     False
13   True  False     False
14  False  False     False
15   True  False     False
16   True  False     False
17   True  False     False
18   True  False     False
19   True  False     False
20  False   True     False
```

```
from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.2)
```

```
    support itemsets
0  0.333333      (0)
1  0.619048      (2)
2  0.380952      (4)
3  0.285714      (5)
4  0.238095      (7)
5  0.238095      (8)
6  0.285714      (9)
7  0.333333     (10)
```

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

```
    support      itemsets
0  0.333333     (BISCUIT)
1  0.619048       (BREAD)
2  0.380952      (COFFEE)
3  0.285714  (CORNFLAKES)
4  0.238095       (MAGGI)
5  0.238095        (MILK)
6  0.285714       (SUGER)
7  0.333333         (TEA)
```

```
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
frequent_itemsets['length'] =
```

```
frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

```
    support      itemsets  length
0  0.333333     (BISCUIT)       1
1  0.619048       (BREAD)       1
2  0.380952      (COFFEE)       1
3  0.285714  (CORNFLAKES)       1
4  0.238095       (MAGGI)       1
5  0.238095        (MILK)       1
6  0.285714       (SUGER)       1
7  0.333333         (TEA)       1
```

```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                (frequent_itemsets['support'] >= 0.2) ]
```

```
Empty DataFrame
Columns: [support, itemsets, length]
Index: []
```

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'BREAD',
'MILK'} ]
```

```
Empty DataFrame
Columns: [support, itemsets, length]
Index: []
```

```
from mlxtend.frequent_patterns import association_rules

association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.6)
```

```
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent
support, support, confidence, lift, leverage, conviction,
zhangs_metric]
Index: []
```

FP GROWTH ALGORITHM

```
from mlxtend.frequent_patterns import fpgrowth

fpgrowth(df, min_support=0.3)
```

```
    support itemsets
0  0.619048      (2)
1  0.333333      (0)
2  0.333333     (10)
3  0.380952      (4)
```

```
fpgrowth(df, min_support=0.3, use_colnames=True)
```

```
    support   itemsets
0  0.619048    (BREAD)
1  0.333333  (BISCUIT)
2  0.333333      (TEA)
3  0.380952   (COFFEE)
```

```
from mlxtend.frequent_patterns import association_rules

association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.8)
```

```
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent
support, support, confidence, lift, leverage, conviction,
zhangs_metric]
Index: []
```

Comparision of Apriori and FP Growth Algorithms

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder


te = TransactionEncoder()
te_ary = te.fit(df).transform(df)
df = pd.DataFrame(te_ary, columns=te.columns_)

from mlxtend.frequent_patterns import apriori

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

```
1.25 ms ± 69 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

```
%timeit -n 100 -r 10 apriori(df, min_support=0.6, low_memory=True)
```

```
1.25 ms ± 57 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

```
from mlxtend.frequent_patterns import fpgrowth

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

```
694 µs ± 101 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```