In [ ]:

```
pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\hp\anaconda3\lib\site-packa
ges (0.19.0)
Requirement already satisfied: setuptools in c:\users\hp\anaconda3\lib\site-pa
ckages (from mlxtend) (52.0.0.post20210125)
Requirement already satisfied: pandas>=0.24.2 in c:\users\hp\anaconda3\lib\sit
e-packages (from mlxtend) (1.2.4)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\hp\anaconda3\l
ib\site-packages (from mlxtend) (0.24.1)
Requirement already satisfied: scipy>=1.2.1 in c:\users\hp\anaconda3\lib\site-
packages (from mlxtend) (1.6.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\hp\anaconda3\lib\
site-packages (from mlxtend) (3.3.4)
Requirement already satisfied: joblib>=0.13.2 in c:\users\hp\anaconda3\lib\sit
e-packages (from mlxtend) (1.0.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\hp\anaconda3\lib\site
-packages (from mlxtend) (1.20.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\hp\anaconda3\l
ib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\
site-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\
users\hp\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site
-packages (from matplotlib>=3.0.0->mlxtend) (8.2.0)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages
(from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\hp\anaconda3\lib\site-
packages (from pandas>=0.24.2->mlxtend) (2021.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\l
ib\site-packages (from scikit-learn>=0.20.3->mlxtend) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

```
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
from mlxtend.preprocessing import TransactionEncoder
import matplotlib.pyplot as plt
from time import time
```

In [ ]:

```
import pandas as pd
from sklearn.preprocessing import Binarizer, OneHotEncoder
```

In [ ]:

```
df = pd.read_csv('Avocado Dataset.csv')
df1 = pd.read_csv('Trail.csv')
```

Question 11 - Test drive the basic version of Apriori algorithms for Frequent Itemset Mining using the package / library support in the platform of your choice. Test it with various support and confidence measures and generate a time comparison for varied data set sizes. To do the performance comparison you may use benchmark datasets provided for FIM such as the FIMI workshop or other sources.

In [ ]:

```
def apriori_function(df):
    df_out = df.apply(lambda x: list(x.dropna().values), axis=1).tolist()
    te = TransactionEncoder()
    out = te.fit(df_out).transform(df_out)
    final = pd.DataFrame(out, columns=te.columns_)
    frequent_itemsets = apriori(final, min_support=0.1, max_len=3, use_colname
    rules = association_rules(frequent_itemsets, metric="confidence", min_thre
    return frequent_itemsets, rules
```

Question 12 - Test drive the basic version of FP Growth algorithms for Frequent Itemset Mining using the package / library support in the platform of your choice. Test it with various support and confidence measures and generate a time comparison for varied data set sizes. To do the performance comparison you may use benchmark datasets provided for FIM such as the FIMI workshop or other sources.

In [ ]:

```
def fp_growth_function(df):
    df_out = df.apply(lambda x: list(x.dropna().values), axis=1).tolist()
    te = TransactionEncoder()
    out = te.fit(df_out).transform(df_out)
    chess = pd.DataFrame(out, columns=te.columns_)
    frequent_itemsets = fpgrowth(chess, min_support=0.1, max_len=3, use_colnar
    rules = association_rules(frequent_itemsets, metric="confidence", min_thre
    return frequent_itemsets, rules
```

Question 14 - Mine frequent itemsets using FP-Growth* algorithm.

In [ ]:

```
df = pd.read_csv('connect.dat', header=None, sep='\n')
df = df[0].str.split(' ', expand=True)
freq, rules = fp_growth_function(df)

print(freq)
print(rules)
```

```
        support       itemsets
0      1.000000          (109)
1      1.000000           (67)
2      1.000000           (16)
3      1.000000           (34)
4      1.000000           (37)
...         ...            ...
25905  0.319783   (44, 97, 72)
25906  0.319783    (7, 44, 72)
25907  0.308943  (44, 72, 115)
25908  0.360434   (44, 82, 72)
25909  0.325203   (44, 79, 72)

[25910 rows x 2 columns]
       antecedents consequents  antecedent support  consequent support  \
0            (109)        (67)            1.000000            1.000000
1             (67)       (109)            1.000000            1.000000
2             (16)        (67)            1.000000            1.000000
3             (67)        (16)            1.000000            1.000000
4             (16)       (109)            1.000000            1.000000
...            ...         ...                 ...                 ...
104276    (82, 72)        (44)            0.493225            0.791328
104277        (72)    (44, 82)            0.498645            0.785908
```

```
104278    (44, 72)        (79)            0.365854            0.940379
104279    (79, 72)        (44)            0.452575            0.791328
104280        (72)    (44, 79)            0.498645            0.737127

         support  confidence      lift  leverage  conviction
0       1.000000    1.000000  1.000000  0.000000         inf
1       1.000000    1.000000  1.000000  0.000000         inf
2       1.000000    1.000000  1.000000  0.000000         inf
3       1.000000    1.000000  1.000000  0.000000         inf
4       1.000000    1.000000  1.000000  0.000000         inf
...          ...         ...       ...       ...         ...
104276  0.360434    0.730769  0.923472 -0.029869    0.775068
104277  0.360434    0.722826  0.919734 -0.031455    0.772411
104278  0.325203    0.888889  0.945245 -0.018838    0.536585
104279  0.325203    0.718563  0.908047 -0.032932    0.741452
104280  0.325203    0.652174  0.884751 -0.042362    0.755759

[104281 rows x 9 columns]
```