In [1]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns', None)
```

In [2]:
```
df = pd.read_csv("C:\\Users\\SRI KAAVYA\\OneDrive\\Desktop\\Internship project
df.head(28)
```

Out[2]:

	Member_number	Date	itemDescription	year	month	day	day_of_week
0	1808	2015-07-21	tropical fruit	2015	7	21	1
1	2552	2015-05-01	whole milk	2015	5	1	4
2	2300	2015-09-19	pip fruit	2015	9	19	5
3	1187	2015-12-12	other vegetables	2015	12	12	5
4	3037	2015-01-02	whole milk	2015	1	2	4
5	4941	2015-02-14	rolls/buns	2015	2	14	5
6	4501	2015-08-05	other vegetables	2015	8	5	2
7	3803	2015-12-23	pot plants	2015	12	23	2
8	2762	2015-03-20	whole milk	2015	3	20	4
9	4119	2015-12-02	tropical fruit	2015	12	2	2
10	1340	2015-02-24	citrus fruit	2015	2	24	1
11	2193	2015-04-14	beef	2015	4	14	1
12	1997	2015-07-21	frankfurter	2015	7	21	1
13	4546	2015-03-09	chicken	2015	3	9	0
14	4736	2015-07-21	butter	2015	7	21	1
15	1959	2015-03-30	fruit/vegetable juice	2015	3	30	0
16	1974	2015-03-05	packaged fruit/vegetables	2015	3	5	3
17	2421	2015-02-09	chocolate	2015	2	9	0
18	1513	2015-03-08	specialty bar	2015	3	8	6
19	1905	2015-07-07	other vegetables	2015	7	7	1
20	2810	2015-08-09	butter milk	2015	8	9	6
21	2867	2015-12-11	whole milk	2015	12	11	4
22	3962	2015-09-18	tropical fruit	2015	9	18	4
23	1088	2015-11-30	tropical fruit	2015	11	30	0
24	4976	2015-07-17	bottled water	2015	7	17	4
25	4056	2015-12-06	yogurt	2015	12	6	6
26	3611	2015-02-13	sausage	2015	2	13	4
27	1420	2015-01-14	other vegetables	2015	1	14	2

In [3]:
```
df.shape
```

Out[3]:
```
(38765, 7)
```

In [4]:
```
df.dtypes
```

Out[4]:
```
Member_number       int64
Date               object
itemDescription    object
year                int64
month               int64
day                 int64
day_of_week         int64
dtype: object
```

In [5]:
```
df.describe()
```

Out[5]:

	Member_number	year	month	day	day_of_week
count	38765.000000	38765.000000	38765.000000	38765.000000	38765.000000
mean	3003.641868	2014.528518	6.477570	15.753231	3.014498
std	1153.611031	0.499193	3.431561	8.801391	1.987669
min	1000.000000	2014.000000	1.000000	1.000000	0.000000
25%	2002.000000	2014.000000	4.000000	8.000000	1.000000
50%	3005.000000	2015.000000	6.000000	16.000000	3.000000
75%	4007.000000	2015.000000	9.000000	23.000000	5.000000
max	5000.000000	2015.000000	12.000000	31.000000	6.000000

In [6]:
```

df['itemDescription']=df['itemDescription'].astype(str).str.strip()
df.dropna(axis=0,subset=['Date'],inplace=True)
df['Date']=df['Date'].astype('str')
df=df[~df['Date'].str.contains('C')]

```

In [7]:
```
df.shape
```

Out[7]:
```
(38765, 7)
```

In [8]:
```
basket = (df[df['itemDescription'] =="tropical fruit"]
          .groupby(['day', 'month'])['year']
          .sum().unstack().reset_index().fillna(0)
          .set_index('day'))
```

In [9]:
```python
basket.head()
```

Out[9]:
```
month        1        2        3       4        5        6       7        8       9      10
  day
    1  10074.0   4029.0   8059.0  2014.0   8059.0  10073.0  8060.0   2015.0  4029.0  2015.0  14
    2   8059.0   4028.0  12088.0  8057.0   6044.0      0.0  4030.0   4029.0  4030.0     0.0  60
    3   8060.0   4030.0   6043.0  4030.0  10075.0   2014.0     0.0   2015.0  6045.0  6044.0  60
    4   6044.0   6044.0   8058.0  4030.0   8060.0   6044.0  6044.0  12088.0  8060.0  8060.0  80
    5  14103.0  12087.0  10074.0  4029.0      0.0   2015.0  6045.0   6043.0  8056.0  8058.0  40
```

In [10]:
```python
basket.shape
```

Out[10]:
```
(31, 12)
```

In [11]:
```python
def encode_units(x):
    if x<=0:
        return 0
    if x>=1:
        return 1
basket_sets=basket.applymap(encode_units)
```

In [12]:
```python
basket_sets.head()
```

Out[12]:
```
month  1  2  3  4  5  6  7  8  9  10  11  12
  day
    1  1  1  1  1  1  1  1  1  1   1   1   1
    2  1  1  1  1  1  0  1  1  1   0   1   1
    3  1  1  1  1  1  1  0  1  1   1   1   1
    4  1  1  1  1  1  1  1  1  1   1   1   1
    5  1  1  1  1  0  1  1  1  1   1   1   0
```

In [13]:
```python
basket_sets = basket_sets.astype(bool)
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
```

In [14]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

Out[14]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conv
0	(1)	(2)	1.000000	0.838710	0.838710	0.838710	1.0	0.0	
1	(2)	(1)	0.838710	1.000000	0.838710	1.000000	1.0	0.0	
2	(1)	(3)	1.000000	0.903226	0.903226	0.903226	1.0	0.0	
3	(3)	(1)	0.903226	1.000000	0.903226	1.000000	1.0	0.0	
4	(1)	(4)	1.000000	0.935484	0.935484	0.935484	1.0	0.0	