```
# Sample code to do Apriori in Python
import apyori

# Creating Sample Transactions
transactions = [
     ['Milk', 'Bread', 'Saffron'],
     ['Milk', 'Saffron'],
     ['Bread', 'Saffron','Wafer'],
     ['Bread','Wafer'],
        ]

# Generating association rules
Rules = list(apyori.apriori(transactions, min_support=0.5, min_confidence=0.5))

# Extracting rules from the object
for i in range(len(Rules)):
    LHS=list(Rules[i][2][0][0])
    RHS=list(Rules[i][2][0][1])
    support=Rules[i][1]
    confidence=Rules[i][2][0][2]
    lift=Rules[i][2][0][3]
    print("LHS:",LHS,"--","RHS:",RHS)
    print("Support:",support)
    print("Confidence:",confidence)
    print("Lift:",lift)
    print(10*"----")
```

In [2]:

```
!pip install apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py): started
  Building wheel for apyori (setup.py): finished with status 'done'
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5975
sha256=d04528a72b0d447d4bc3541737a38aa9b12fe7a3b267e41b19a32497e9a065b2
  Stored in directory: c:\users\91920\appdata\local\pip\cache\wheels\cb\f6\e
1\57973c631d27efd1a2f375bd6a83b2a616c4021f24aab84080
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2

WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
```

In [171]:

```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [172]:

```
Data = pd.read_csv('E:/CDAC/Market_Basket_Optimisation.csv', header = None)
```

In [173]:

```
Data
```

Out[173]:

	0	1	2	3	4	5	6	7	8	9
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN

7501 rows × 20 columns

In [139]:

```
Data.fillna(0, inplace = True)
```

Since we shall be training an apriori model, which takes inputs in a list format, we need to transform our pandas' data frame into a list of transactions. To create this list, we start by initializing an empty list. We then populate this with different transactions in our pandas' data frame.

In [174]:

```
transacts = []
# populating a list of transactions
for i in range(0, 7501):
    transacts.append([str(Data.values[i,j]) for j in range(0, 20)])
```

In [141]:

```
transacts
```

```
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0'],
  ['sparkling water',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
   '0',
```

Where,

min_support: The minimum support of relations (float)

min_confidence: The minimum confidence of relations (float)

min_lift: The minimum lift of relations (float)

min_length: The minimum number of items in a rule

max_length: The maximum number of items in a rule

In [163]:

```
from apyori import apriori
rule = apriori(transactions = transacts, min_support = 0.003, min_confidence = 0.2, min_lif
```

In [143]:

```
for i in rule:
    print(i)
```

```
RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.0045
32728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset
({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829
059829057, lift=4.84395061728395)])
RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}), supp
ort=0.005732568990801226, ordered_statistics=[OrderedStatistic(items_base=
frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), co
nfidence=0.3006993006993007, lift=3.790832696715049)])
RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884
548726837, ordered_statistics=[OrderedStatistic(items_base=frozenset({'pas
ta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, l
ift=4.700811850163794)])
RelationRecord(items=frozenset({'fromage blanc', 'honey'}), support=0.0033
32888948140248, ordered_statistics=[OrderedStatistic(items_base=frozenset
({'fromage blanc'}), items_add=frozenset({'honey'}), confidence=0.24509803
92156863, lift=5.164270764485569)])
RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=
0.015997866951073192, ordered_statistics=[OrderedStatistic(items_base=froz
enset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence
```

In [156]:

```
a=list(rule)
print(a)
```

```
[]
```

In [144]:

```
i[0]
```

Out[144]:

```
frozenset({'pasta', 'shrimp'})
```

In [145]:

```
i[1]  #support
```

Out[145]:

```
0.005065991201173177
```

In [146]:

```
i[2]
```

Out[146]:

```
[OrderedStatistic(items_base=frozenset({'pasta'}), items_add=frozenset({'shr
imp'}), confidence=0.3220338983050847, lift=4.506672147735896)]
```

In [147]:

```
i[2][0][0]   #lhs
```

Out[147]:

```
frozenset({'pasta'})
```

In [148]:

```
i[2][0][1]  #rhs
```

Out[148]:

```
frozenset({'shrimp'})
```

In [149]:

```
i[2][0][2] #confidence
```

Out[149]:

```
0.3220338983050847
```

In [150]:

```
i[2][0][3]  # Lift
```

Out[150]:

```
4.506672147735896
```

In [153]:

```
a1=next(rule)
a2=next(rule)
a3=next(rule)
a4=next(rule)
a5=next(rule)
a6=next(rule)
a7=next(rule)
a8=next(rule)
a9=next(rule)
print(a1,a2,a3,a4,a5,a6,a7,a8,a9)
```

```
RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532
728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset({'li
ght cream'}), items_add=frozenset({'chicken'}), confidence=0.290598290598290
57, lift=4.84395061728395)]) RelationRecord(items=frozenset({'escalope', 'mu
shroom cream sauce'}), support=0.005732568990801226, ordered_statistics=[Ord
eredStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=froz
enset({'escalope'}), confidence=0.3006993006993007, lift=3.79083269671504
9)]) RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865
884548726837, ordered_statistics=[OrderedStatistic(items_base=frozenset({'pa
sta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, li
ft=4.700811850163794)]) RelationRecord(items=frozenset({'fromage blanc', 'ho
ney'}), support=0.003332888948140248, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'fromage blanc'}), items_add=frozenset({'honey'}), conf
idence=0.2450980392156863, lift=5.164270764485569)]) RelationRecord(items=fr
ozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, ord
ered_statistics=[OrderedStatistic(items_base=frozenset({'herb & pepper'}), i
tems_add=frozenset({'ground beef'}), confidence=0.3234501347708895, lift=3.2
919938411349285)]) RelationRecord(items=frozenset({'tomato sauce', 'ground b
eef'}), support=0.005332622317024397, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}),
confidence=0.3773584905660377, lift=3.840659481324083)]) RelationRecord(item
s=frozenset({'light cream', 'olive oil'}), support=0.003199573390214638, ord
ered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), ite
ms_add=frozenset({'olive oil'}), confidence=0.20512820512820515, lift=3.1147
098515519573)]) RelationRecord(items=frozenset({'olive oil', 'whole wheat pa
sta'}), support=0.007998933475536596, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'whole wheat pasta'}), items_add=frozenset({'olive oi
l'}), confidence=0.2714932126696833, lift=4.122410097642296)]) RelationRecor
d(items=frozenset({'shrimp', 'pasta'}), support=0.005065991201173177, ordere
d_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}), items_add=fr
ozenset({'shrimp'}), confidence=0.3220338983050847, lift=4.50667214773589
6)])
```

In [155]:

```
for result in (a1,a2,a3,a4,a5,a6,a7,a8,a9):
    lhs         = [tuple(result[2][0][0])]
    rhs         = [tuple(result[2][0][1])]
    support    = [result[1] ]
    confidence = [result[2][0][2]]
    lift       = [result[2][0][3]]
    print("LHS:",lhs,"--","RHS:",rhs)
    print("Support:",support)
    print("Confidence:",confidence)
    print("Lift:",lift)
    print(10*"----")
```

```
LHS: [('light cream',)] -- RHS: [('chicken',)]
Support: [0.004532728969470737]
Confidence: [0.29059829059829057]
Lift: [4.84395061728395]
----------------------------------------
LHS: [('mushroom cream sauce',)] -- RHS: [('escalope',)]
Support: [0.005732568990801226]
Confidence: [0.3006993006993007]
Lift: [3.790832696715049]
----------------------------------------
LHS: [('pasta',)] -- RHS: [('escalope',)]
Support: [0.005865884548726837]
Confidence: [0.3728813559322034]
Lift: [4.700811850163794]
----------------------------------------
LHS: [('fromage blanc',)] -- RHS: [('honey',)]
Support: [0.003332888948140248]
Confidence: [0.2450980392156863]
Lift: [5.164270764485569]
----------------------------------------
LHS: [('herb & pepper',)] -- RHS: [('ground beef',)]
Support: [0.015997866951073192]
Confidence: [0.3234501347708895]
Lift: [3.2919938411349285]
----------------------------------------
LHS: [('tomato sauce',)] -- RHS: [('ground beef',)]
Support: [0.005332622317024397]
Confidence: [0.3773584905660377]
Lift: [3.840659481324083]
----------------------------------------
LHS: [('light cream',)] -- RHS: [('olive oil',)]
Support: [0.003199573390214638]
Confidence: [0.20512820512820515]
Lift: [3.1147098515519573]
----------------------------------------
LHS: [('whole wheat pasta',)] -- RHS: [('olive oil',)]
Support: [0.007998933475536596]
Confidence: [0.2714932126696833]
Lift: [4.122410097642296]
----------------------------------------
LHS: [('pasta',)] -- RHS: [('shrimp',)]
Support: [0.005065991201173177]
Confidence: [0.3220338983050847]
Lift: [4.506672147735896]
----------------------------------------
```

In [164]:

```
lst = []
for i in rule:
    new_lst = []
    new_lst.append(i)
    lst.append(new_lst)
```

In [166]:

```
fro_set = []
sup = []
lft = []
con = []
for i in lst:
    for j in i:
        fro_set.append(j[0])
        sup.append(j[1])
        lft.append(j[2][0][3])
        con.append(j[2][0][2])
print(fro_set)
print(sup)
print(lft)
print(con)

d = {'set':fro_set, 'support':sup, 'confidence':con, 'lift':lft}
df = pd.DataFrame(d)
df
```

```
k', 'frozen vegetables'}), frozenset({'mineral water', '0', 'spaghetti',
'milk', 'frozen vegetables'}), frozenset({'mineral water', '0', 'spaghett
i', 'shrimp', 'frozen vegetables'}), frozenset({'mineral water', '0', 'tom
atoes', 'spaghetti', 'frozen vegetables'}), frozenset({'mineral water',
'0', 'ground beef', 'spaghetti', 'milk'}), frozenset({'mineral water', 'ol
ive oil', '0', 'ground beef', 'spaghetti'}), frozenset({'mineral water',
'0', 'ground beef', 'spaghetti', 'pancakes'}), frozenset({'mineral water',
'0', 'ground beef', 'spaghetti', 'tomatoes'}), frozenset({'mineral water',
'olive oil', '0', 'spaghetti', 'milk'}), frozenset({'mineral water', '0',
'tomatoes', 'spaghetti', 'milk'})]
[0.004532728969470737, 0.005732568990801226, 0.005865884548726837, 0.00333
2888948140248, 0.015997866951073192, 0.005332622317024397, 0.0031995733902
14638, 0.007998933475536596, 0.005065991201173177, 0.004532728969470737,
0.005732568990801226, 0.005865884548726837, 0.003332888948140248, 0.015997
866951073192, 0.005332622317024397, 0.003199573390214638, 0.00799893347553
6596, 0.005065991201173177, 0.003332888948140248, 0.0037328356219170776,
0.0030662578322890282, 0.003199573390214638, 0.0030662578322890282, 0.0030
662578322890282, 0.0038661511798426876, 0.0035995200639914677, 0.003466204
5060658577, 0.005332622317024397, 0.003999466737768298, 0.0039994667377682
98, 0.004799360085321957, 0.0041327822956939075, 0.0037328356219170776, 0.
```

In [167]:

```
df.nlargest(n = 10, columns = 'Lift')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, k
ey, method, tolerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

KeyError: 'Lift'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-167-ab4bb0ef22bf> in <module>
----> 1 df.nlargest(n = 10, columns = 'Lift')

~\anaconda3\lib\site-packages\pandas\core\frame.py in nlargest(self, n, colu
mns, keep)
   5123         Brunei      434000    12128      BN
   5124         """
-> 5125         return algorithms.SelectNFrame(self, n=n, keep=keep, columns
=columns).nlargest()
   5126
   5127     def nsmallest(self, n, columns, keep="first") -> "DataFrame":

~\anaconda3\lib\site-packages\pandas\core\algorithms.py in nlargest(self)
   1081
   1082     def nlargest(self):
-> 1083         return self.compute("nlargest")
   1084
   1085     def nsmallest(self):

~\anaconda3\lib\site-packages\pandas\core\algorithms.py in compute(self, met
hod)
   1195
   1196         for column in columns:
-> 1197             dtype = frame[column].dtype
   1198             if not self.is_valid_dtype_n_method(dtype):
   1199                 raise TypeError(

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2798             if self.columns.nlevels > 1:
   2799                 return self._getitem_multilevel(key)
-> 2800             indexer = self.columns.get_loc(key)
   2801             if is_integer(indexer):
   2802                 indexer = [indexer]
```

```
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, k
ey, method, tolerance)
   2646                 return self._engine.get_loc(key)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_indexer
(key))
   2649         indexer = self.get_indexer([key], method=method, tolerance=t
olerance)
   2650         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

KeyError: 'Lift'
```

In [ ]:

```
```

In [ ]:

```
```

In [124]:

```
b=[]
for result in (a1,a2,a3,a4,a5,a6,a7,a8,a9):
    lhs         = list([ result[2][0][0] ])
    rhs         = list([result[2][0][1]])
    support    = [result[1] ]
    confidence = [result[2][0][2]]
    lift       = [result[2][0][3]]
    a=list(zip(lhs, rhs, support, confidence, lift))

    print(a)
    b.append(a)
    #df = pd.concat([pd.Series(x)for x in a], axis=1)
```

```
[(frozenset({'light cream'}), frozenset({'chicken'}), 0.004532728969470737,
0.29059829059829057, 4.84395061728395)]
[(frozenset({'mushroom cream sauce'}), frozenset({'escalope'}), 0.0057325689
90801226, 0.3006993006993007, 3.790832696715049)]
[(frozenset({'pasta'}), frozenset({'escalope'}), 0.005865884548726837, 0.372
8813559322034, 4.700811850163794)]
[(frozenset({'fromage blanc'}), frozenset({'honey'}), 0.003332888948140248,
0.2450980392156863, 5.164270764485569)]
[(frozenset({'herb & pepper'}), frozenset({'ground beef'}), 0.01599786695107
3192, 0.3234501347708895, 3.2919938411349285)]
[(frozenset({'tomato sauce'}), frozenset({'ground beef'}), 0.005332622317024
397, 0.3773584905660377, 3.840659481324083)]
[(frozenset({'light cream'}), frozenset({'olive oil'}), 0.003199573390214638
, 0.20512820512820515, 3.1147098515519573)]
[(frozenset({'whole wheat pasta'}), frozenset({'olive oil'}), 0.007998933475
536596, 0.2714932126696833, 4.122410097642296)]
[(frozenset({'pasta'}), frozenset({'shrimp'}), 0.005065991201173177, 0.32203
38983050847, 4.506672147735896)]
```

In [132]:

```
b
```

Out[132]:

```
[[(frozenset({'light cream'}),
   frozenset({'chicken'}),
   0.004532728969470737,
   0.29059829059829057,
   4.84395061728395)],
 [(frozenset({'mushroom cream sauce'}),
   frozenset({'escalope'}),
   0.005732568990801226,
   0.3006993006993007,
   3.790832696715049)],
 [(frozenset({'pasta'}),
   frozenset({'escalope'}),
   0.005865884548726837,
   0.3728813559322034,
   4.700811850163794)],
 [(frozenset({'fromage blanc'}),
   frozenset({'honey'}),
   0.003332888948140248,
   0.2450980392156863,
   5.164270764485569)],
 [(frozenset({'herb & pepper'}),
   frozenset({'ground beef'}),
   0.015997866951073192,
   0.3234501347708895,
   3.2919938411349285)],
 [(frozenset({'tomato sauce'}),
   frozenset({'ground beef'}),
   0.005332622317024397,
   0.3773584905660377,
   3.840659481324083)],
 [(frozenset({'light cream'}),
   frozenset({'olive oil'}),
   0.003199573390214638,
   0.20512820512820515,
   3.1147098515519573)],
 [(frozenset({'whole wheat pasta'}),
   frozenset({'olive oil'}),
   0.007998933475536596,
   0.2714932126696833,
   4.122410097642296)],
 [(frozenset({'pasta'}),
   frozenset({'shrimp'}),
   0.005065991201173177,
   0.3220338983050847,
   4.506672147735896)]]
```

```
df= pd.DataFrame(b ,columns = ['Left_Hand_Side', 'Right_Hand_Side', 'Support',
'Confidence', 'Lift'])
df
```

```
output = list(rule) # returns a non-tabular output

# putting output into a pandas dataframe
```

```
def inspect(output):
    lhs         = [tuple(result[2][0][0])[0] for result in output]
    rhs         = [tuple(result[2][0][1])[0] for result in output]
    support    = [result[1] for result in output]
    confidence = [result[2][0][2] for result in output]
    lift       = [result[2][0][3] for result in output]
    return list(zip(lhs, rhs, support, confidence, lift))
output_DataFrame = pd.DataFrame(inspect(rule), columns = ['Left_Hand_Side',
'Right_Hand_Side', 'Support', 'Confidence', 'Lift'])
```

```
output_DataFrame.nlargest(n = 10, columns = 'Lift')
```

In [ ]:

In [ ]:

In [ ]:

In [158]:

```
!pip install pyfpgrowth
```

```
Collecting pyfpgrowth

WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -umpy (c:\users\91920\anaconda3\lib\s
ite-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\91920\anaconda3
\lib\site-packages)


  Downloading pyfpgrowth-1.0.tar.gz (1.6 MB)
Building wheels for collected packages: pyfpgrowth
  Building wheel for pyfpgrowth (setup.py): started
  Building wheel for pyfpgrowth (setup.py): finished with status 'done'
  Created wheel for pyfpgrowth: filename=pyfpgrowth-1.0-py2.py3-none-any.whl
size=5477 sha256=7f24c9dc0e5a61ab93d2f0759d16e4499d6c9e43b35283622d816b1bf59
a8191
  Stored in directory: c:\users\91920\appdata\local\pip\cache\wheels\73\97\4
b\f12ac994f6bbb99597396255435824c73ad3916be1e678be55
Successfully built pyfpgrowth
Installing collected packages: pyfpgrowth
Successfully installed pyfpgrowth-1.0
```

In [159]:

```
import pyfpgrowth
```

In [175]:

```
patterns = pyfpgrowth. find_frequent_patterns(transacts, 10)
```

Patterns are generated based on the parameters passed in the find_frequent_patterns() , where “transactions” are the list of items bought at each transaction(refer to the ITEMS column of the table) and 10 is the minimum threshold set for support count

and 10 is the minimum threshold set for support count.

In [176]:

```
rules = pyfpgrowth. generate_association_rules(patterns,0.8)
```

Rules are generated based on the patterns and 0.8 is the minimum threshold set for confidence.

In [177]:

```
rules
```

Out[177]:

```
{('bramble',): (('nan',), 12.5),
 ('frozen vegetables', 'tea'): (('nan',), 11.4),
 ('spaghetti', 'tea'): (('nan',), 11.0),
 ('mineral water', 'tea'): (('nan',), 10.818181818181818),
 ('nan', 'tea'): ((), 6.197260273972603),
 ('chutney', 'spaghetti'): (('nan',), 10.545454545454545),
 ('chutney', 'eggs'): (('nan',), 11.545454545454545),
 ('chutney', 'mineral water'): (('nan',), 11.153846153846153),
 ('chutney', 'nan'): ((), 6.854066985645933),
 ('mashed potato', 'mineral water'): (('nan',), 13.0),
 ('mashed potato', 'spaghetti'): (('nan',), 10.545454545454545),
 ('mashed potato', 'nan'): ((), 6.84433962264151),
 ('chocolate bread', 'mineral water'): (('nan',), 12.214285714285714),
 ('chocolate bread', 'nan'): ((), 6.6506849315068495),
 ('dessert wine', 'spaghetti'): (('nan',), 13.3),
 ('dessert wine', 'mineral water'): (('nan',), 13.0),
 ('dessert wine', 'nan'): ((), 6.737991266375546),
 ('ketchup', 'mineral water'): (('nan',), 10.6),
```

In [ ]: