VIT-AP UNIVERSITY, ANDHRA PRADESH

Lab Sheet 9: Apriori Implementation

Academic year: 2021-2022	Branch/ Class: B.Tech
Semester: Fall	Date: 14-11-22
Faculty Name: Dr Aravapalli Rama Satish	School: SCOPE

NAME: Majjiga Jaswanth REGNO:20BCD7171

****

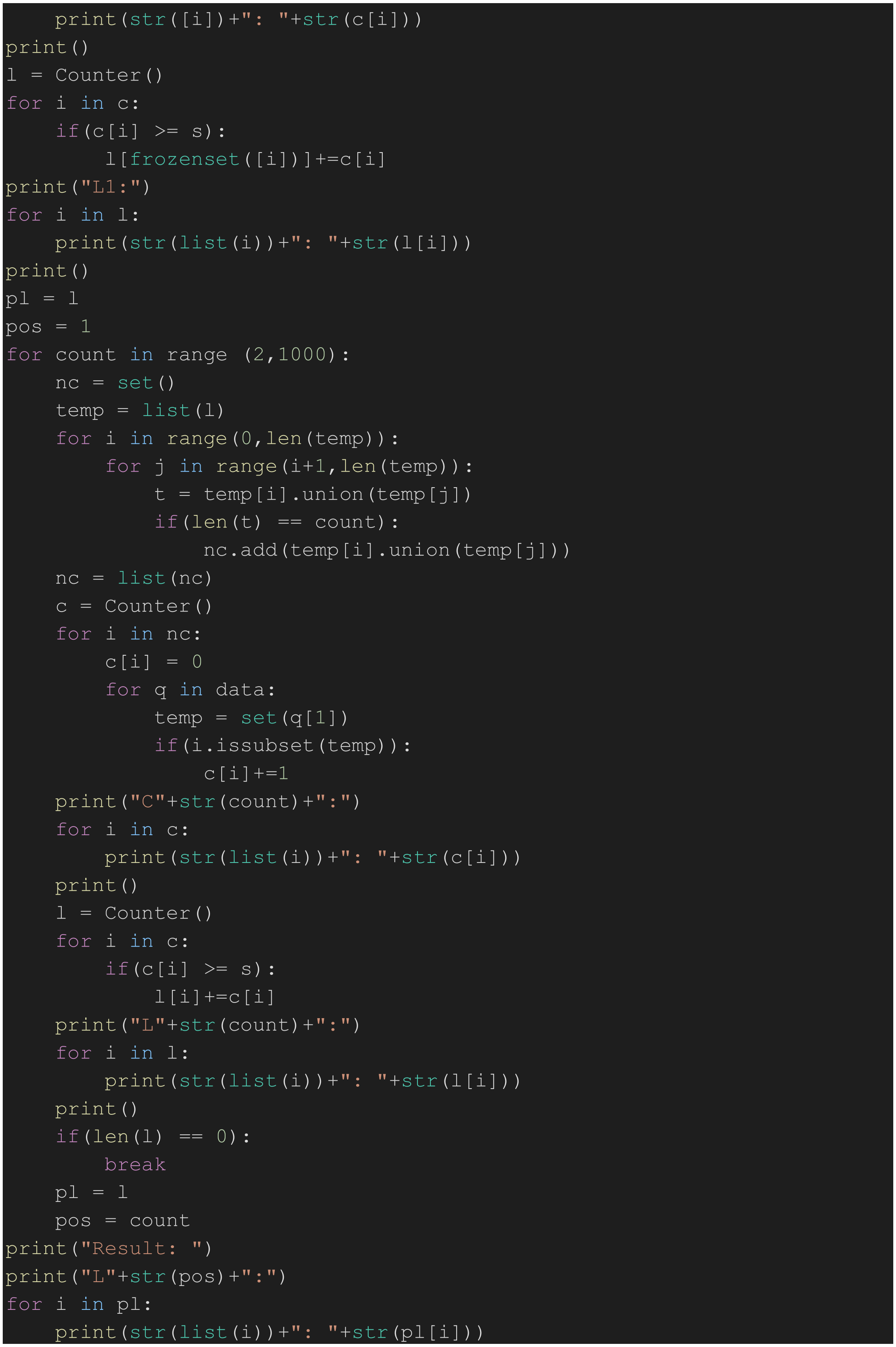
1. Develop a python code to generate frequent item sets using Apriori algorithm with minimum support is 3. Consider the following transactions/data:

Data 1:

(("a","b","c"), ("a","b"), ("a","b","d"), ("b","e"), ("b","c","e"), ("a","d","e"), ("a","c"),
("a","b","d"), ("c","e"), ("a","b","d","e"), ("a",'b','e','c'))

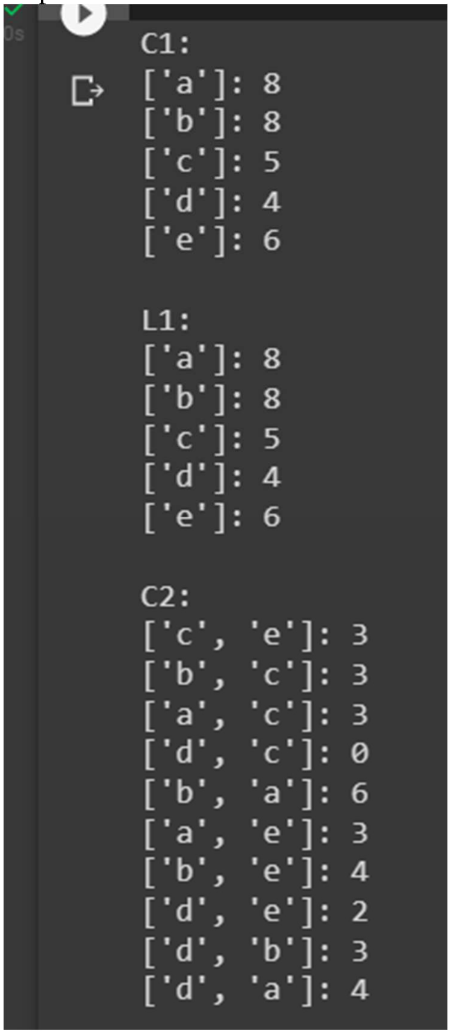
Code:

```
import numpy as np
import pandas as pd
from collections import Counter
from itertools import combinations
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
import random

data = [["T1",["a","b","c"]],["T2",["a","b"]],["T3",["a","b","d"]],
        ["T4",["b","e"]], ["T5",["b","c","e"]],["T6",["a","d","e"]],
        ["T7",["a","c"]], ["T8",["a","b","d"]], ["T9",["c","e"]],
        ["T10",["a","b","d","e"]], ["T11",["a", "b", "c","e"]]]
init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)
sp = 0.6
s = int(sp*len(init))
print("s =",s)



c = Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
```

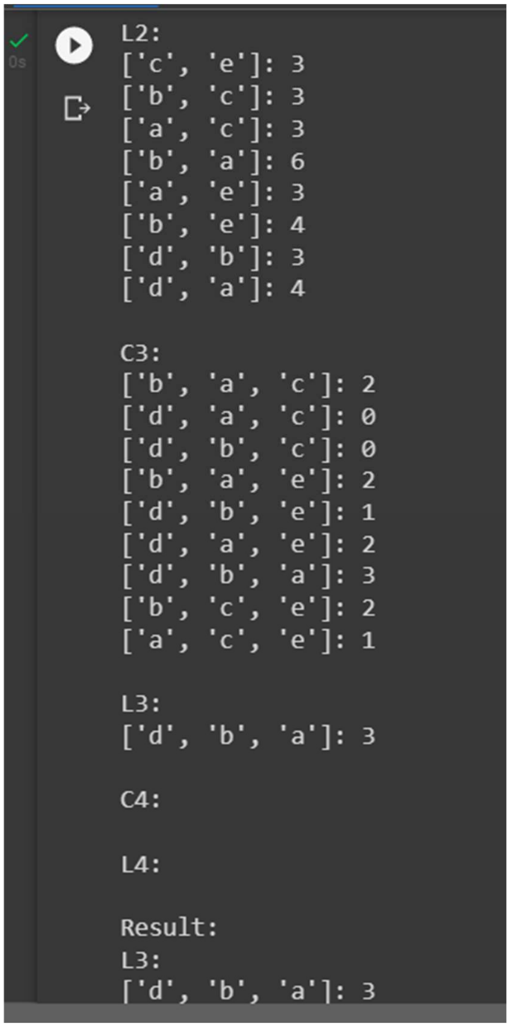


```
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = l
pos = 1
for count in range (2,1000):
    nc = set()
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for q in data:
            temp = set(q[1])
            if(i.issubset(temp)):
                c[i]+=1
    print("C"+str(count)+":")
    for i in c:
        print(str(list(i))+": "+str(c[i]))
    print()
    l = Counter()
    for i in c:
        if(c[i] >= s):
            l[i]+=c[i]
    print("L"+str(count)+":")
    for i in l:
        print(str(list(i))+": "+str(l[i]))
    print()
    if(len(l) == 0):
        break
    pl = l
    pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
    print(str(list(i))+": "+str(pl[i]))
print()
```

output:



```
C1:
['a']: 8
['b']: 8
['c']: 5
['d']: 4
['e']: 6

L1:
['a']: 8
['b']: 8
['c']: 5
['d']: 4
['e']: 6

C2:
['c', 'e']: 3
['b', 'c']: 3
['a', 'c']: 3
['d', 'c']: 0
['b', 'a']: 6
['a', 'e']: 3
['b', 'e']: 4
['d', 'e']: 2
['d', 'b']: 3
['d', 'a']: 4
```



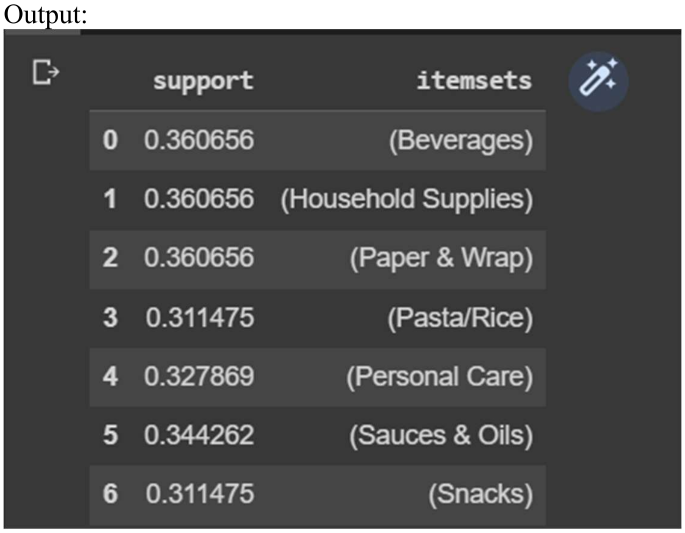
```
L2:
['c', 'e']: 3
['b', 'c']: 3
['a', 'c']: 3
['b', 'a']: 6
['a', 'e']: 3
['b', 'e']: 4
['d', 'b']: 3
['d', 'a']: 4

C3:
['b', 'a', 'c']: 2
['d', 'a', 'c']: 0
['d', 'b', 'c']: 0
['b', 'a', 'e']: 2
['d', 'b', 'e']: 1
['d', 'a', 'e']: 2
['d', 'b', 'a']: 3
['b', 'c', 'e']: 2
['a', 'c', 'e']: 1

L3:
['d', 'b', 'a']: 3

C4:

L4:

Result:
L3:
['d', 'b', 'a']: 3
```

Data 2:
Generate synthetic data transactions (#30) for a supermarket store. This store sells the following products:
Fruits, Vegetables, Canned Goods, Frozen Foods, Meat, Fish and shellfish, Deli, Condiments & Spices, Sauces & Oils, Snacks, Bread & Bakery, Beverages, Pasta/Rice, Cereal, Baking, Personal Care, Health Care, Paper & Wrap, Household Supplies, Baby Items, Other items.
Each transaction should consist of at least two products and maximum of 12 products. Each time when we run, it should generate different transactions.

Code:

```
transactions= []
items = ['Fruits', 'Vegetables', 'Canned Goods', 'Frozen Foods', 'Meat'
, 'Fish and shellfish', 'Deli', 'Condiments& Spices','Sauces & Oils', '
Snacks', 'Bread & Bakery', 'Beverages', 'Pasta/Rice', 'Cereal', 'Baking
',
'Personal Care', 'Health Care', 'Paper & Wrap', 'Household Supplies', '
Baby Items', 'Other items']
for i in range(30):
  a=np.random.randint(12)
  t=random.sample(items,a)
  transactions.append(t)

te=TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary,columns=te.columns_)
apriori(df,use_colnames=True,min_support=0.3)
```

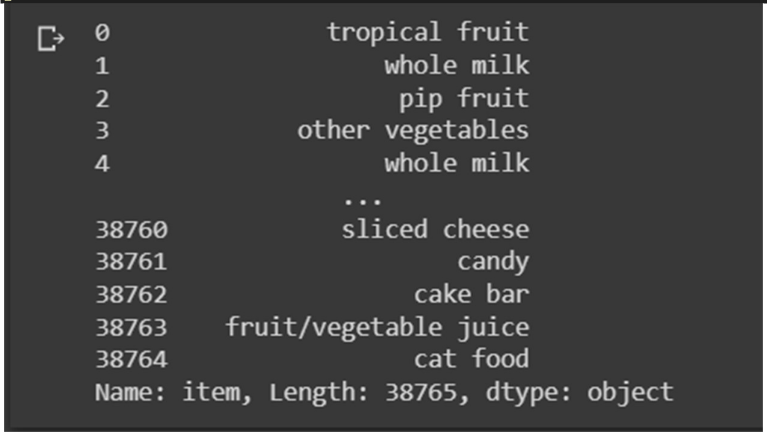
Output:



	support	itemsets
0	0.360656	(Beverages)
1	0.360656	(Household Supplies)
2	0.360656	(Paper & Wrap)
3	0.311475	(Pasta/Rice)
4	0.327869	(Personal Care)
5	0.344262	(Sauces & Oils)
6	0.311475	(Snacks)

Data 3:
Generate synthetic data transactions at least 1000 with the items provided in excel sheet “GroceryList_spreadsheet.xls”.

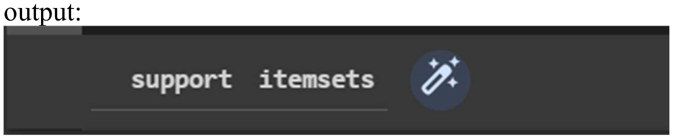
```
df = pd.read_csv('Groceries_dataset.csv')
print(df['item'])
```



```
0                tropical fruit
1                    whole milk
2                     pip fruit
3              other vegetables
4                    whole milk
                      ...
38760             sliced cheese
38761                     candy
38762                  cake bar
38763      fruit/vegetable juice
38764                  cat food
Name: item, Length: 38765, dtype: object
```

```
trans=[]
for i in range(1000):
  a = np.random.randint(1,10)
  t = random.sample(items,a)
  trans.append(t)
te=TransactionEncoder()
te_ary = te.fit(trans).transform(trans)
df = pd.DataFrame(te_ary,columns=te.columns_)
apriori(df,use_colnames=True,min_support=0.3)
```

output:



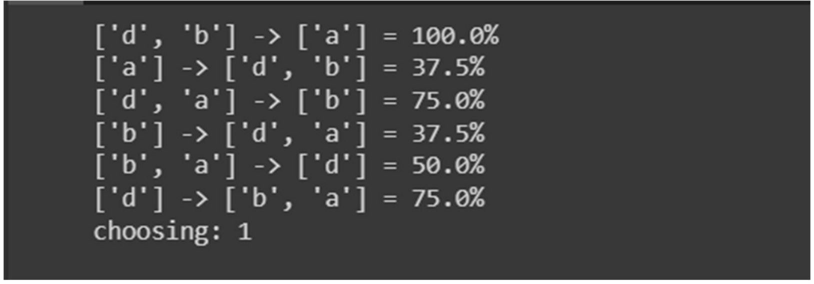
```
support  itemsets
```

2. Develop a python code to provide association rules from the generated frequent item sets in question 1 with minimum confidence of 80%. [Perform the comparison of your output with predefined packages output carried out in Lab Exercise 8.]
*** Do not use any predefined packages such as mlxtend, apyori to apply Apriori algorithm (for the questions 1 and 2).

Code:

```
for l in pl:
    c = [frozenset(q) for q in combinations(l,len(l)-1)]
    mmax = 0
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp > mmax):
            mmax = temp
        temp = sab/sb*100
        if(temp > mmax):
            mmax = temp
        print(str(list(a))+" -
> "+str(list(b))+" = "+str(sab/sa*100)+"%")
        print(str(list(b))+" -
> "+str(list(a))+" = "+str(sab/sb*100)+"%")
    curr = 1
    print("choosing:", end=' ')
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
        temp = sab/sb*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
    print()
    print()
```

output:

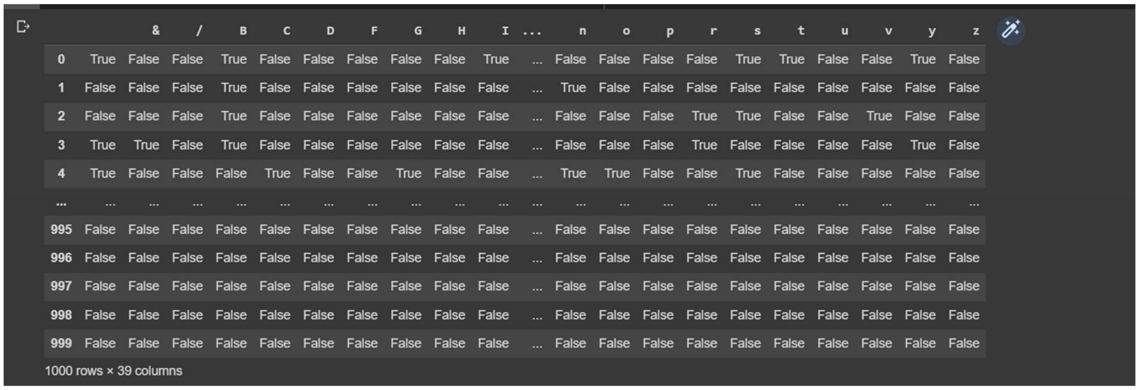


```
['d', 'b'] -> ['a'] = 100.0%
['a'] -> ['d', 'b'] = 37.5%
['d', 'a'] -> ['b'] = 75.0%
['b'] -> ['d', 'a'] = 37.5%
['b', 'a'] -> ['d'] = 50.0%
['d'] -> ['b', 'a'] = 75.0%
choosing: 1
```

3. Use mlxtend and pyfpgrowth packages to apply fpgrowth algorithm on the above Data sets provided in Question 1.

Code:

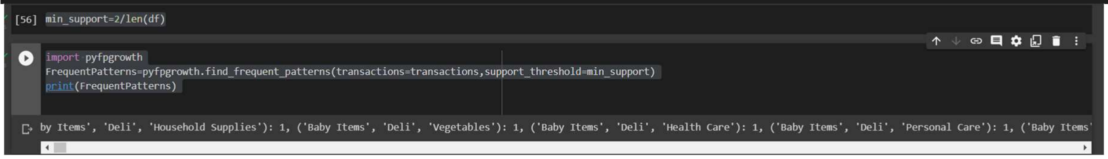
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
my_transactionencoder=TransactionEncoder()
my_transactionencoder.fit(df)
encoded_transactions = my_transactionencoder.transform(df)
encoded_transactions_data=pd.DataFrame(encoded_transactions,columns=my_
transactionencoder.columns_)
encoded_transactions_data
```



		&	/	B	C	D	F	G	H	I	...	n	o	p	r	s	t	u	v	y	z
0	True	False	False	True	False	False	False	False	False	True	...	False	False	False	False	True	True	False	False	True	False
1	False	False	False	True	False	False	False	False	False	False	...	True	False	False	False	False	False	False	False	False	False
2	False	False	False	True	False	False	False	False	False	False	...	False	False	False	True	True	False	False	True	False	False
3	True	True	False	True	False	False	False	False	False	False	...	False	False	False	True	False	False	False	False	True	False
4	True	False	False	False	True	False	False	True	False	False	...	True	True	False	False	True	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

1000 rows × 39 columns

```
min_support=2/len(df)
import pyfpgrowth
FrequentPatterns=pyfpgrowth.find_frequent_patterns(transactions=transac
tions,support_threshold=min_support)
print(FrequentPatterns)
```

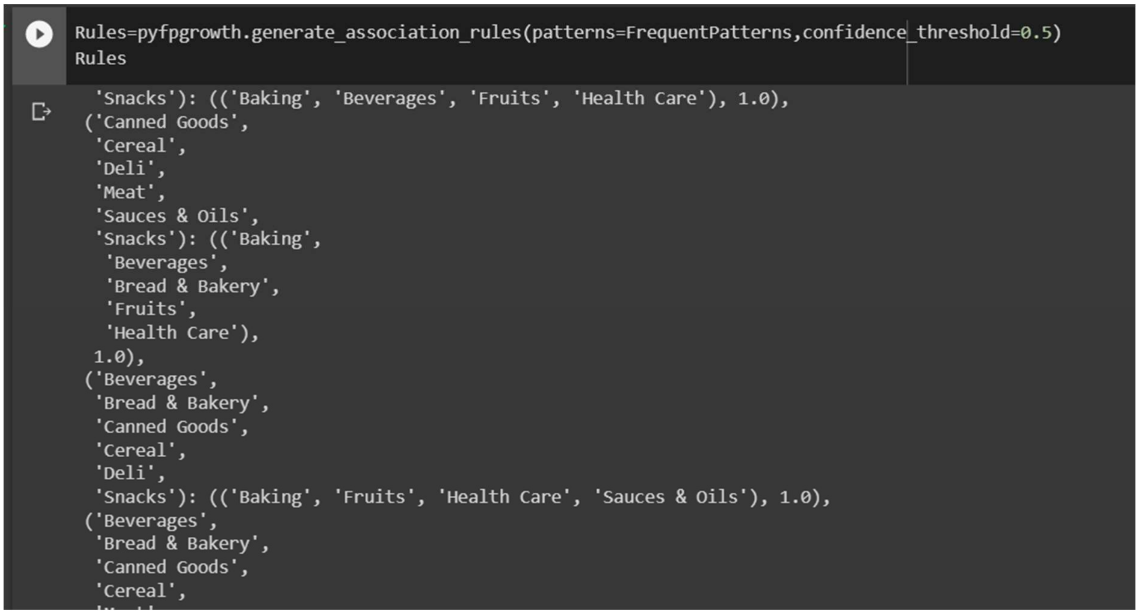


```
[56] min_support=2/len(df)
```

```
import pyfpgrowth
FrequentPatterns=pyfpgrowth.find_frequent_patterns(transactions=transactions,support_threshold=min_support)
print(FrequentPatterns)
```

```
by Items', 'Deli', 'Household Supplies'): 1, ('Baby Items', 'Deli', 'Vegetables'): 1, ('Baby Items', 'Deli', 'Health Care'): 1, ('Baby Items', 'Deli', 'Personal Care'): 1, ('Baby Items'
```

```
Rules=pyfpgrowth.generate_association_rules(patterns=FrequentPatterns,c
onfidence_threshold=0.5)
Rules
```



```
Rules=pyfpgrowth.generate_association_rules(patterns=FrequentPatterns,confidence_threshold=0.5)
Rules
```

```
  'Snacks'): (('Baking', 'Beverages', 'Fruits', 'Health Care'), 1.0),
 ('Canned Goods',
  'Cereal',
  'Deli',
  'Meat',
  'Sauces & Oils',
  'Snacks'): (('Baking',
   'Beverages',
   'Bread & Bakery',
   'Fruits',
   'Health Care'),
  1.0),
 ('Beverages',
  'Bread & Bakery',
  'Canned Goods',
  'Cereal',
  'Deli',
  'Snacks'): (('Baking', 'Fruits', 'Health Care', 'Sauces & Oils'), 1.0),
 ('Beverages',
  'Bread & Bakery',
  'Canned Goods',
  'Cereal',
```

Output:

```
{('Baby Items', 'Deli'): 1, ('Baby Items', 'Deli', 'Pasta/Rice'): 1,
('Baby Items', 'Deli', 'Household Supplies'): 1, ('Baby Items', 'Deli',
'Vegetables'): 1, ('Baby Items', 'Deli', 'Health Care'): 1, ('Baby
```

```
Items', 'Deli', 'Personal Care'): 1, ('Baby Items', 'Deli', 'Fruits'):
1, ('Baby Items', 'Deli', 'Sauces & Oils'): 1, ('Baby Items', 'Deli',
'Household Supplies', 'Pasta/Rice'): 1, ('Baby Items', 'Deli',
'Pasta/Rice', 'Vegetables'): 1, ('Baby Items', 'Deli', 'Health Care',
'Pasta/Rice'): 1, ('Baby Items', 'Deli', 'Pasta/Rice', 'Personal
Care'): 1, ('Baby Items', 'Deli', 'Fruits', 'Pasta/Rice'): 1, ('Baby
Items', 'Deli', 'Pasta/Rice', 'Sauces & Oils'): 1, ('Baby Items',
'Deli', 'Household Supplies', 'Vegetables'): 1, ('Baby Items', 'Deli',
'Health Care', 'Household Supplies'): 1, ('Baby Items', 'Deli',
'Household Supplies', 'Personal Care'): 1, ('Baby Items', 'Deli',
'Fruits', 'Household Supplies'): 1, ('Baby Items', 'Deli', 'Household
Supplies', 'Sauces & Oils'): 1, ('Baby Items', 'Deli', 'Health Care',
'Vegetables'): 1, ('Baby Items', 'Deli', 'Personal Care',
'Vegetables'): 1, ('Baby Items', 'Deli', 'Fruits', 'Vegetables'): 1,
('Baby Items', 'Deli', 'Sauces & Oils', 'Vegetables'): 1, ('Baby
Items', 'Deli', 'Health Care', 'Personal Care'): 1, ('Baby Items',
'Deli', 'Fruits', 'Health Care'): 1, ('Baby Items', 'Deli', 'Health
Care', 'Sauces & Oils'): 1, ('Baby Items', 'Deli', 'Fruits', 'Personal
Care'): 1, ('Baby Items', 'Deli', 'Personal Care', 'Sauces & Oils'): 1,
('Baby Items', 'Deli', 'Fruits', 'Sauces & Oils'): 1, ('Baby Items',
'Deli', 'Household Supplies', 'Pasta/Rice', 'Vegetables'): 1, ('Baby
Items', 'Deli', 'Health Care', 'Household Supplies', 'Pasta/Rice'): 1,
('Baby Items', 'Deli', 'Household Supplies', 'Pasta/Rice', 'Personal
Care'): 1, ('Baby Items', 'Deli', 'Fruits', 'Household Supplies',
'Pasta/Rice'): 1, ('Baby Items', 'Deli', 'Household Supplies',
'Pasta/Rice', 'Sauces & Oils'): 1, ('Baby Items', 'Deli', 'Health
```