Reading File

In [525…
```
import csv
import time
import pandas as pd
import itertools
```

In [526…
```
def load_data(filename):
    full_transaction_list= []
    with open(filename, encoding = 'utf-8-sig') as data:
        transaction_data = csv.reader(data, delimiter = ',')
        for  row in transaction_data:
            filtered_rows = [value for value in row if value != '']
            full_transaction_list.append(filtered_rows)
        return full_transaction_list
```

In [527…
```
#Asking the user to input the file
new_list = load_data(input('please enter file name\n'))
```

```
please enter file name
Transactions_5.csv
```

Asking user for min support and confidence

In [528…
```
try:
    user_input_minsupport =int(input('Please enter the minimum support value
    user_input_minConfidence = int(input('\nPlease enter the min confidence
except:
    if user_input_minsupport == int(user_input_minsupport) or user_input_min
        print('please enter a numerical value')
```

```
Please enter the minimum support value in percentage ex: 25 is 25%:

10

Please enter the min confidence value n percentage ex: 50 is 50%:

55
```

Transaction 5

Apriori Algorithm

In [529…
```
start_time = time.time()

#function to find all the unique values with their counts
def Uniquevalues(Transactions):
    unique_items = {}
    for rows in Transactions:
        for items in rows:
            if items not in unique_items:
                unique_items[items] = 1
            else:
                unique_items[items] = unique_items[items] + 1
    uniqueitemlist = []
    for value in unique_items:
        Valuelist = []
        Valuelist.append(value)
        uniqueitemlist.append(Valuelist)
        uniqueitemlist.append(unique_items[value])
    return uniqueitemlist
```

In [530…
```
One_UniqueItems = Uniquevalues(new_list)
print('These are the unique items for Transaction 1:\n\n', One_UniqueItems)
```

```
These are the unique items for Transaction 1:

 [['ink'], 6, ['pen'], 6, ['cheese'], 12, ['bag'], 8, ['milk'], 10, ['juic
e'], 13, ['soap'], 6, ['candy'], 3]
```

In [531…
```
#function used to remove the items that do not meet the threshold
def remove_lessthansupportone(Candidates, transactions):
    Firstcandidate_list= []
    for i in range(len(Candidates)):
        if i%2 != 0:
            if (Candidates[i] / len(new_list))*100 >= user_input_minsupport:
                Firstcandidate_list.append(Candidates[i-1])
                Firstcandidate_list.append(Candidates[i])
    candidatesforcombo = []
    for i in range(len(Firstcandidate_list)):
        if i%2 == 0:
            candidatesforcombo.append(Firstcandidate_list[i])
    return candidatesforcombo
```

In [532…
```
removed_first = remove_lessthansupportone(One_UniqueItems, new_list)
print('\nThese are the candidates after the first pass\n\n',removed_first )
```

```
These are the candidates after the first pass

 [['ink'], ['pen'], ['cheese'], ['bag'], ['milk'], ['juice'], ['soap'], ['ca
ndy']]
```

In [533…
```
#function used to output all possible combinations (k itemsets)
def Allpossiblecombinations(candidatesforcombo):
    if not candidatesforcombo:
        return [[]]
    first= candidatesforcombo[0]
    Allothers = candidatesforcombo[1:]
    Withoutfirst = Allpossiblecombinations(Allothers)
    Withfirst = [combo + [first] for combo in Withoutfirst]
    Combinedlist=Withoutfirst + Withfirst
    return Combinedlist
```

In [534…
```
All_combos = Allpossiblecombinations(removed_first)
```

In [535…
```
#function used to add the number of counts to the list provided before
def allcombosunique(Combination, dataset):
    from collections import Counter
    Count = Counter()
    for row in Combination:
        for s in dataset:
            if all(item in s for item in sum(row, [])):
                Count[tuple(map(tuple, row))] += 1
    listcount = [[list(subset), count] for subset, count in Count.items()]
    return(listcount)
```

In [536…
```
All_uniquecombos = allcombosunique(All_combos, new_list)
print('\nAll unique possible combinations\n',All_uniquecombos )
```

```
All unique possible combinations
 [[[], 20], [[('candy',)], 3], [[('soap',)], 6], [[('juice',)], 13], [[('can
dy',), ('juice',)], 3], [[('soap',), ('juice',)], 3], [[('milk',)], 10],
[[('soap',), ('milk',)], 3], [[('juice',), ('milk',)], 5], [[('bag',)], 8],
[[('soap',), ('bag',)], 4], [[('juice',), ('bag',)], 2], [[('soap',), ('juic
e',), ('bag',)], 1], [[('milk',), ('bag',)], 4], [[('soap',), ('milk',), ('b
ag',)], 3], [[('cheese',)], 12], [[('candy',), ('cheese',)], 1], [[('soa
p',), ('cheese',)], 2], [[('juice',), ('cheese',)], 7], [[('candy',), ('juic
e',), ('cheese',)], 1], [[('soap',), ('juice',), ('cheese',)], 1], [[('mil
k',), ('cheese',)], 6], [[('soap',), ('milk',), ('cheese',)], 1], [[('juic
e',), ('milk',), ('cheese',)], 3], [[('bag',), ('cheese',)], 5], [[('soa
p',), ('bag',), ('cheese',)], 1], [[('juice',), ('bag',), ('cheese',)], 1],
[[('milk',), ('bag',), ('cheese',)], 2], [[('soap',), ('milk',), ('bag',),
('cheese',)], 1], [[('pen',)], 6], [[('candy',), ('pen',)], 1], [[('juic
e',), ('pen',)], 4], [[('candy',), ('juice',), ('pen',)], 1], [[('milk',),
('pen',)], 2], [[('juice',), ('milk',), ('pen',)], 2], [[('bag',), ('pe
n',)], 3], [[('juice',), ('bag',), ('pen',)], 1], [[('cheese',), ('pen',)],
5], [[('juice',), ('cheese',), ('pen',)], 3], [[('milk',), ('cheese',), ('pe
n',)], 2], [[('juice',), ('milk',), ('cheese',), ('pen',)], 2], [[('bag',),
('cheese',), ('pen',)], 3], [[('juice',), ('bag',), ('cheese',), ('pen',)],
1], [[('ink',)], 6], [[('soap',), ('ink',)], 1], [[('juice',), ('ink',)],
2], [[('milk',), ('ink',)], 3], [[('soap',), ('milk',), ('ink',)], 1], [[('j
uice',), ('milk',), ('ink',)], 1], [[('bag',), ('ink',)], 3], [[('soap',),
('bag',), ('ink',)], 1], [[('milk',), ('bag',), ('ink',)], 1], [[('soap',),
('milk',), ('bag',), ('ink',)], 1], [[('cheese',), ('ink',)], 4], [[('juic
e',), ('cheese',), ('ink',)], 1], [[('milk',), ('cheese',), ('ink',)], 1],
[[('bag',), ('cheese',), ('ink',)], 2], [[('pen',), ('ink',)], 2], [[('ba
g',), ('pen',), ('ink',)], 2], [[('cheese',), ('pen',), ('ink',)], 2], [[('b
ag',), ('cheese',), ('pen',), ('ink',)], 2]]
```

In [537…

```
#Second function to remove the items that do not meet the threshold
def remove_lessthansupporttwo(Candidates, dataset):
    list1 = []
    for outlist in Candidates:
        if len(outlist) >= 2:
            second_object = outlist[1]
            if (second_object / len(new_list))*100  >= user_input_minsupport
                list1.append(outlist[0])
                list1.append(outlist[1])
    return list1
```

In [538…

```
removed_second = remove_lessthansupporttwo(All_uniquecombos, new_list)
print('\nThese are the candidates after the next pass\n',removed_second )
```

```
These are the candidates after the next pass
 [[], 20, [('candy',)], 3, [('soap',)], 6, [('juice',)], 13, [('candy',), ('
juice',)], 3, [('soap',), ('juice',)], 3, [('milk',)], 10, [('soap',), ('mil
k',)], 3, [('juice',), ('milk',)], 5, [('bag',)], 8, [('soap',), ('bag',)],
4, [('juice',), ('bag',)], 2, [('milk',), ('bag',)], 4, [('soap',), ('mil
k',), ('bag',)], 3, [('cheese',)], 12, [('soap',), ('cheese',)], 2, [('juic
e',), ('cheese',)], 7, [('milk',), ('cheese',)], 6, [('juice',), ('milk',),
('cheese',)], 3, [('bag',), ('cheese',)], 5, [('milk',), ('bag',), ('chees
e',)], 2, [('pen',)], 6, [('juice',), ('pen',)], 4, [('milk',), ('pen',)],
2, [('juice',), ('milk',), ('pen',)], 2, [('bag',), ('pen',)], 3, [('chees
e',), ('pen',)], 5, [('juice',), ('cheese',), ('pen',)], 3, [('milk',), ('ch
eese',), ('pen',)], 2, [('juice',), ('milk',), ('cheese',), ('pen',)], 2,
[('bag',), ('cheese',), ('pen',)], 3, [('ink',)], 6, [('juice',), ('ink',)],
2, [('milk',), ('ink',)], 3, [('bag',), ('ink',)], 3, [('cheese',), ('in
k',)], 4, [('bag',), ('cheese',), ('ink',)], 2, [('pen',), ('ink',)], 2, [('
bag',), ('pen',), ('ink',)], 2, [('cheese',), ('pen',), ('ink',)], 2, [('ba
g',), ('cheese',), ('pen',), ('ink',)], 2]
```

In [539…

```python
def Rules(CandidateSet):
    CandidateRule = []
    for candidates in CandidateSet:
        if isinstance(candidates, list):
            if len(candidates) != 0:
                length_candidates = len(candidates) - 1
                while length_candidates > 0:
                    combos = list(itertools.combinations(candidates, length_
                    combolist = []
                    Left = []
                    for Right in combos:
                        Left = set(candidates) - set(Right)
                        combolist.append(list(Left))
                        combolist.append(list(Right))
                        CandidateRule.append(combolist)
                        combolist = []
                    length_candidates = length_candidates - 1

    return CandidateRule
```

In [540…

```python
Associationrules = Rules(removed_second)
print('\nThese are the association rules\n\n',Associationrules )
```

```
These are the association rules

 [[[('juice',)], [('candy',)]], [[('candy',)], [('juice',)]], [[('juice',)],
[('soap',)]], [[('soap',)], [('juice',)]], [[('milk',)], [('soap',)]], [[('s
oap',)], [('milk',)]], [[('milk',)], [('juice',)]], [[('juice',)], [('mil
k',)]], [[('bag',)], [('soap',)]], [[('soap',)], [('bag',)]], [[('bag',)],
[('juice',)]], [[('juice',)], [('bag',)]], [[('bag',)], [('milk',)]], [[('mi
lk',)], [('bag',)]], [[('bag',)], [('soap',), ('milk',)]], [[('milk',)], [('
soap',), ('bag',)]], [[('soap',)], [('milk',), ('bag',)]], [[('milk',), ('ba
g',)], [('soap',)]], [[('bag',), ('soap',)], [('milk',)]], [[('milk',), ('so
ap',)], [('bag',)]], [[('cheese',)], [('soap',)]], [[('soap',)], [('chees
e',)]], [[('cheese',)], [('juice',)]], [[('juice',)], [('cheese',)]], [[('ch
```

```
eese',)], [('milk',)]], [[('milk',)], [('cheese',)]], [[('cheese',)], [('jui
ce',), ('milk',)]], [[('milk',)], [('juice',), ('cheese',)]], [[('juice',)],
[('milk',), ('cheese',)]], [[('milk',), ('cheese',)], [('juice',)]], [[('jui
ce',), ('cheese',)], [('milk',)]], [[('juice',), ('milk',)], [('cheese',)]],
[[('cheese',)], [('bag',)]], [[('bag',)], [('cheese',)]], [[('cheese',)],
[('milk',), ('bag',)]], [[('bag',)], [('milk',), ('cheese',)]], [[('mil
k',)], [('bag',), ('cheese',)]], [[('bag',), ('cheese',)], [('milk',)]],
[[('milk',), ('cheese',)], [('bag',)]], [[('milk',), ('bag',)], [('chees
e',)]], [[('pen',)], [('juice',)]], [[('juice',)], [('pen',)]], [[('pen',)],
[('milk',)]], [[('milk',)], [('pen',)]], [[('pen',)], [('juice',), ('mil
k',)]], [[('milk',)], [('juice',), ('pen',)]], [[('juice',)], [('milk',), ('
pen',)]], [[('milk',), ('pen',)], [('juice',)]], [[('juice',), ('pen',)],
[('milk',)]], [[('juice',), ('milk',)], [('pen',)]], [[('pen',)], [('ba
g',)]], [[('bag',)], [('pen',)]], [[('pen',)], [('cheese',)]], [[('chees
e',)], [('pen',)]], [[('pen',)], [('juice',), ('cheese',)]], [[('cheese',)],
[('juice',), ('pen',)]], [[('juice',)], [('cheese',), ('pen',)]], [[('chees
e',), ('pen',)], [('juice',)]], [[('juice',), ('pen',)], [('cheese',)]],
[[('juice',), ('cheese',)], [('pen',)]], [[('pen',)], [('milk',), ('chees
e',)]], [[('cheese',)], [('milk',), ('pen',)]], [[('milk',)], [('cheese',),
('pen',)]], [[('cheese',), ('pen',)], [('milk',)]], [[('milk',), ('pen',)],
[('cheese',)]], [[('milk',), ('cheese',)], [('pen',)]], [[('pen',)], [('juic
e',), ('milk',), ('cheese',)]], [[('cheese',)], [('juice',), ('milk',), ('pe
n',)]], [[('milk',)], [('juice',), ('cheese',), ('pen',)]], [[('juice',)],
[('milk',), ('cheese',), ('pen',)]], [[('cheese',), ('pen',)], [('juice',),
('milk',)]], [[('milk',), ('pen',)], [('juice',), ('cheese',)]], [[('mil
k',), ('cheese',)], [('juice',), ('pen',)]], [[('juice',), ('pen',)], [('mil
k',), ('cheese',)]], [[('juice',), ('cheese',)], [('milk',), ('pen',)]],
[[('juice',), ('milk',)], [('cheese',), ('pen',)]], [[('milk',), ('chees
e',), ('pen',)], [('juice',)]], [[('juice',), ('cheese',), ('pen',)], [('mil
k',)]], [[('juice',), ('pen',), ('milk',)], [('cheese',)]], [[('juice',), ('
cheese',), ('milk',)], [('pen',)]], [[('pen',)], [('bag',), ('cheese',)]],
[[('cheese',)], [('bag',), ('pen',)]], [[('bag',)], [('cheese',), ('pe
n',)]], [[('cheese',), ('pen',)], [('bag',)]], [[('bag',), ('pen',)], [('che
ese',)]], [[('bag',), ('cheese',)], [('pen',)]], [[('ink',)], [('juice',)]],
[[('juice',)], [('ink',)]], [[('ink',)], [('milk',)]], [[('milk',)], [('in
k',)]], [[('ink',)], [('bag',)]], [[('bag',)], [('ink',)]], [[('ink',)], [('
cheese',)]], [[('cheese',)], [('ink',)]], [[('ink',)], [('bag',), ('chees
e',)]], [[('cheese',)], [('bag',), ('ink',)]], [[('bag',)], [('cheese',), ('
ink',)]], [[('cheese',), ('ink',)], [('bag',)]], [[('bag',), ('ink',)], [('c
heese',)]], [[('bag',), ('cheese',)], [('ink',)]], [[('ink',)], [('pen',)]],
[[('pen',)], [('ink',)]], [[('ink',)], [('bag',), ('pen',)]], [[('pen',)],
[('bag',), ('ink',)]], [[('bag',)], [('pen',), ('ink',)]], [[('pen',), ('in
k',)], [('bag',)]], [[('bag',), ('ink',)], [('pen',)]], [[('bag',), ('pe
n',)], [('ink',)]], [[('ink',)], [('cheese',), ('pen',)]], [[('pen',)], [('c
heese',), ('ink',)]], [[('cheese',)], [('pen',), ('ink',)]], [[('pen',), ('i
nk',)], [('cheese',)]], [[('cheese',), ('ink',)], [('pen',)]], [[('chees
e',), ('pen',)], [('ink',)]], [[('ink',)], [('bag',), ('cheese',), ('pe
n',)]], [[('pen',)], [('bag',), ('cheese',), ('ink',)]], [[('cheese',)], [('
bag',), ('pen',), ('ink',)]], [[('bag',)], [('cheese',), ('pen',), ('in
k',)]], [[('pen',), ('ink',)], [('bag',), ('cheese',)]], [[('cheese',), ('in
k',)], [('bag',), ('pen',)]], [[('cheese',), ('pen',)], [('bag',), ('in
k',)]], [[('bag',), ('ink',)], [('cheese',), ('pen',)]], [[('bag',), ('pe
n',)], [('cheese',), ('ink',)]], [[('bag',), ('cheese',)], [('pen',), ('in
```

```
k',)]], [[('cheese',), ('pen',), ('ink',)], [('bag',)]], [[('bag',), ('pe
n',), ('ink',)], [('cheese',)]], [[('bag',), ('cheese',), ('ink',)], [('pe
n',)]], [[('bag',), ('cheese',), ('pen',)], [('ink',)]]]
```

In [541…

```
def Apriori(Associationrules, new_list, user_input_minConfidence):
    AAlgorithm = []
    for rule in Associationrules:
        first = set(item[0] for item in rule[0])
        Asupport = 0
        ABsupport = 0
        for transaction in new_list:
            if first.issubset(set(transaction)):
                Asupport += 1
            if all(set(item) <= set(transaction) for each in rule for item i
                ABsupport += 1
        CalculateASupport = (Asupport * 1.0 / len(new_list)) * 100
        CalculateABSupport = (ABsupport * 1.0 / len(new_list)) * 100
        confidence = (CalculateABSupport / CalculateASupport) * 100
        if confidence >= user_input_minConfidence:
            OutputASupport = "A Support is: " + str(round(CalculateASupport,
            OutputABSupport = "\nA&B support is: " + str(CalculateABSupport)
            OutputConfidence = "\nConfidence is: " + str(round(confidence))
            AAlgorithm.append(OutputASupport)
            AAlgorithm.append(OutputABSupport)
            AAlgorithm.append(OutputConfidence)
            AAlgorithm.append(rule)
    return AAlgorithm
```

In [542…

```
Apriori = Apriori(Associationrules, new_list, user_input_minConfidence)
print('\nApriori algorithm\n', Apriori )
```

```
Apriori algorithm
 ['A Support is: 15.0', '\nA&B support is: 15.0', '\nConfidence is: 100',
[[('candy',)], [('juice',)]], 'A Support is: 30.0', '\nA&B support is: 20.
0', '\nConfidence is: 67', [[('soap',)], [('bag',)]], 'A Support is: 20.0',
'\nA&B support is: 15.0', '\nConfidence is: 75', [[('milk',), ('bag',)], [('
soap',)]], 'A Support is: 20.0', '\nA&B support is: 15.0', '\nConfidence is:
75', [[('bag',), ('soap',)], [('milk',)]], 'A Support is: 15.0', '\nA&B supp
ort is: 15.0', '\nConfidence is: 100', [[('milk',), ('soap',)], [('bag',)]],
'A Support is: 60.0', '\nA&B support is: 35.0', '\nConfidence is: 58', [[('c
heese',)], [('juice',)]], 'A Support is: 50.0', '\nA&B support is: 30.0', '\
nConfidence is: 60', [[('milk',)], [('cheese',)]], 'A Support is: 25.0', '\n
A&B support is: 15.0', '\nConfidence is: 60', [[('juice',), ('milk',)], [('c
heese',)]], 'A Support is: 40.0', '\nA&B support is: 25.0', '\nConfidence i
s: 62', [[('bag',)], [('cheese',)]], 'A Support is: 30.0', '\nA&B support i
s: 20.0', '\nConfidence is: 67', [[('pen',)], [('juice',)]], 'A Support is:
10.0', '\nA&B support is: 10.0', '\nConfidence is: 100', [[('milk',), ('pe
n',)], [('juice',)]], 'A Support is: 30.0', '\nA&B support is: 25.0', '\nCon
fidence is: 83', [[('pen',)], [('cheese',)]], 'A Support is: 25.0', '\nA&B s
upport is: 15.0', '\nConfidence is: 60', [[('cheese',), ('pen',)], [('juic
e',)]], 'A Support is: 20.0', '\nA&B support is: 15.0', '\nConfidence is: 7
5', [[('juice',), ('pen',)], [('cheese',)]], 'A Support is: 10.0', '\nA&B su
```

```
pport is: 10.0', '\nConfidence is: 100', [[('milk',), ('pen',)], [('chees
e',)]], 'A Support is: 10.0', '\nA&B support is: 10.0', '\nConfidence is: 10
0', [[('milk',), ('pen',)], [('juice',), ('cheese',)]], 'A Support is: 10.
0', '\nA&B support is: 10.0', '\nConfidence is: 100', [[('milk',), ('chees
e',), ('pen',)], [('juice',)]], 'A Support is: 15.0', '\nA&B support is: 10.
0', '\nConfidence is: 67', [[('juice',), ('cheese',), ('pen',)], [('mil
k',)]], 'A Support is: 10.0', '\nA&B support is: 10.0', '\nConfidence is: 10
0', [[('juice',), ('pen',), ('milk',)], [('cheese',)]], 'A Support is: 15.
0', '\nA&B support is: 10.0', '\nConfidence is: 67', [[('juice',), ('chees
e',), ('milk',)], [('pen',)]], 'A Support is: 25.0', '\nA&B support is: 15.
0', '\nConfidence is: 60', [[('cheese',), ('pen',)], [('bag',)]], 'A Support
is: 15.0', '\nA&B support is: 15.0', '\nConfidence is: 100', [[('bag',), ('p
en',)], [('cheese',)]], 'A Support is: 25.0', '\nA&B support is: 15.0', '\nC
onfidence is: 60', [[('bag',), ('cheese',)], [('pen',)]], 'A Support is: 30.
0', '\nA&B support is: 20.0', '\nConfidence is: 67', [[('ink',)], [('chees
e',)]], 'A Support is: 15.0', '\nA&B support is: 10.0', '\nConfidence is: 6
7', [[('bag',), ('ink',)], [('cheese',)]], 'A Support is: 10.0', '\nA&B supp
ort is: 10.0', '\nConfidence is: 100', [[('pen',), ('ink',)], [('bag',)]], '
A Support is: 15.0', '\nA&B support is: 10.0', '\nConfidence is: 67', [[('ba
g',), ('ink',)], [('pen',)]], 'A Support is: 15.0', '\nA&B support is: 10.
0', '\nConfidence is: 67', [[('bag',), ('pen',)], [('ink',)]], 'A Support i
s: 10.0', '\nA&B support is: 10.0', '\nConfidence is: 100', [[('pen',), ('in
k',)], [('cheese',)]], 'A Support is: 10.0', '\nA&B support is: 10.0', '\nCo
nfidence is: 100', [[('pen',), ('ink',)], [('bag',), ('cheese',)]], 'A Suppo
rt is: 15.0', '\nA&B support is: 10.0', '\nConfidence is: 67', [[('bag',),
('ink',)], [('cheese',), ('pen',)]], 'A Support is: 15.0', '\nA&B support i
s: 10.0', '\nConfidence is: 67', [[('bag',), ('pen',)], [('cheese',), ('in
k',)]], 'A Support is: 10.0', '\nA&B support is: 10.0', '\nConfidence is: 10
0', [[('cheese',), ('pen',), ('ink',)], [('bag',)]], 'A Support is: 10.0', '
\nA&B support is: 10.0', '\nConfidence is: 100', [[('bag',), ('pen',), ('in
k',)], [('cheese',)]], 'A Support is: 10.0', '\nA&B support is: 10.0', '\nCo
nfidence is: 100', [[('bag',), ('cheese',), ('ink',)], [('pen',)]], 'A Suppo
rt is: 15.0', '\nA&B support is: 10.0', '\nConfidence is: 67', [[('bag',),
('cheese',), ('pen',)], [('ink',)]]]
```

In [543…

```
counter = 1
for i in Apriori:
    if counter == 4:
        print("\n"+str(i[0]) + "------->" + str(i[1])+"\n")
        counter = 0
    else:
        print(i, end='  ')
    counter = counter + 1
elapsed_time = time.time() - start_time
print("--- %s seconds ---" % (elapsed_time))
```

```
A Support is: 15.0
A&B support is: 15.0
Confidence is: 100
[('candy',)]------->[('juice',)]

A Support is: 30.0
A&B support is: 20.0
```

```
Confidence is: 67
[('soap',)]------->[('bag',)]

A Support is: 20.0
A&B support is: 15.0
Confidence is: 75
[('milk',), ('bag',)]------->[('soap',)]

A Support is: 20.0
A&B support is: 15.0
Confidence is: 75
[('bag',), ('soap',)]------->[('milk',)]

A Support is: 15.0
A&B support is: 15.0
Confidence is: 100
[('milk',), ('soap',)]------->[('bag',)]

A Support is: 60.0
A&B support is: 35.0
Confidence is: 58
[('cheese',)]------->[('juice',)]

A Support is: 50.0
A&B support is: 30.0
Confidence is: 60
[('milk',)]------->[('cheese',)]

A Support is: 25.0
A&B support is: 15.0
Confidence is: 60
[('juice',), ('milk',)]------->[('cheese',)]

A Support is: 40.0
A&B support is: 25.0
Confidence is: 62
[('bag',)]------->[('cheese',)]

A Support is: 30.0
A&B support is: 20.0
Confidence is: 67
[('pen',)]------->[('juice',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('milk',), ('pen',)]------->[('juice',)]

A Support is: 30.0
A&B support is: 25.0
Confidence is: 83
[('pen',)]------->[('cheese',)]
```

```
A Support is: 25.0
A&B support is: 15.0
Confidence is: 60
[('cheese',), ('pen',)]------->[('juice',)]

A Support is: 20.0
A&B support is: 15.0
Confidence is: 75
[('juice',), ('pen',)]------->[('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('milk',), ('pen',)]------->[('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('milk',), ('pen',)]------->[('juice',), ('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('milk',), ('cheese',), ('pen',)]------->[('juice',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('juice',), ('cheese',), ('pen',)]------->[('milk',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('juice',), ('pen',), ('milk',)]------->[('cheese',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('juice',), ('cheese',), ('milk',)]------->[('pen',)]

A Support is: 25.0
A&B support is: 15.0
Confidence is: 60
[('cheese',), ('pen',)]------->[('bag',)]

A Support is: 15.0
A&B support is: 15.0
Confidence is: 100
[('bag',), ('pen',)]------->[('cheese',)]

A Support is: 25.0
A&B support is: 15.0
Confidence is: 60
```

```
[('bag',), ('cheese',)]------->[('pen',)]

A Support is: 30.0
A&B support is: 20.0
Confidence is: 67
[('ink',)]------->[('cheese',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('ink',)]------->[('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('pen',), ('ink',)]------->[('bag',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('ink',)]------->[('pen',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('pen',)]------->[('ink',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('pen',), ('ink',)]------->[('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('pen',), ('ink',)]------->[('bag',), ('cheese',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('ink',)]------->[('cheese',), ('pen',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('pen',)]------->[('cheese',), ('ink',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('cheese',), ('pen',), ('ink',)]------->[('bag',)]

A Support is: 10.0
```

```
A&B support is: 10.0
Confidence is: 100
[('bag',), ('pen',), ('ink',)]------->[('cheese',)]

A Support is: 10.0
A&B support is: 10.0
Confidence is: 100
[('bag',), ('cheese',), ('ink',)]------->[('pen',)]

A Support is: 15.0
A&B support is: 10.0
Confidence is: 67
[('bag',), ('cheese',), ('pen',)]------->[('ink',)]

--- 0.08525896072387695 seconds ---
```

Brute Force

In [544…
```
import pandas as pd
import time
from itertools import combinations
```

In [545…
```
Transactiondata = input("Enter the file name: ")
minsupport = float(input('Please enter the minimum support value'))
```

```
Enter the file name: Transactions_5.csv
Please enter the minimum support value10
```

In [546…
```
start_time = time.time()
transaction = pd.read_csv(Transactiondata, header =None)
TransactionforSum = pd.get_dummies(transaction.unstack().dropna()).groupby(l
UniqueItems = TransactionforSum.sum()
```

In [547…
```
print('\nThese are all unique one item sets:\n\n', UniqueItems)
```

```
These are all unique one item sets:

 bag        8
candy       3
cheese     12
ink         6
juice      13
milk       10
pen         6
soap        6
dtype: int64
```

In [548…
```
OneItemSets = pd.DataFrame((UniqueItems / len(transaction) * 100), columns =
OneFrequentItems = OneItemSets[OneItemSets['support'] >= minsupport]
print('These are the Frequent One Item sets:\n', OneFrequentItems)
```

```
These are the Frequent One Item sets:
         support
bag         40.0
candy       15.0
cheese      60.0
ink         30.0
juice       65.0
milk        50.0
pen         30.0
soap        30.0
```

In [549…
```
import itertools
items = UniqueItems.index
combos = list(itertools.combinations(items, 2))
combinations = []
for combo in combos:
    combinations.append(combo)
```

In [550…
```
combo_counts = {}
for i in range(len(combinations)):
    combo = combinations[i]
    count = 0
    for index, row in transaction.iterrows():
        if set(combo).issubset(row):
            count += 1
    combo_counts[i+1] = count
print('Thesea are all the two possible combinations:\n\n')
for combo_num, count in combo_counts.items():
    print(f"({combinations[combo_num-1]})  , Number of repetitions {count}."
```

Thesea are all the two possible combinations:

```
(('bag', 'candy'))  , Number of repetitions 0.
(('bag', 'cheese'))  , Number of repetitions 5.
(('bag', 'ink'))  , Number of repetitions 3.
(('bag', 'juice'))  , Number of repetitions 2.
(('bag', 'milk'))  , Number of repetitions 4.
(('bag', 'pen'))  , Number of repetitions 3.
(('bag', 'soap'))  , Number of repetitions 4.
(('candy', 'cheese'))  , Number of repetitions 1.
(('candy', 'ink'))  , Number of repetitions 0.
(('candy', 'juice'))  , Number of repetitions 3.
(('candy', 'milk'))  , Number of repetitions 0.
(('candy', 'pen'))  , Number of repetitions 1.
(('candy', 'soap'))  , Number of repetitions 0.
(('cheese', 'ink'))  , Number of repetitions 4.
(('cheese', 'juice'))  , Number of repetitions 7.
(('cheese', 'milk'))  , Number of repetitions 6.
(('cheese', 'pen'))  , Number of repetitions 5.
(('cheese', 'soap'))  , Number of repetitions 2.
(('ink', 'juice'))  , Number of repetitions 2.
(('ink', 'milk'))  , Number of repetitions 3.
(('ink', 'pen'))  , Number of repetitions 2.
(('ink', 'soap'))  , Number of repetitions 1.
(('juice', 'milk'))  , Number of repetitions 5.
(('juice', 'pen'))  , Number of repetitions 4.
(('juice', 'soap'))  , Number of repetitions 3.
(('milk', 'pen'))  , Number of repetitions 2.
(('milk', 'soap'))  , Number of repetitions 3.
(('pen', 'soap'))  , Number of repetitions 0.
```

In [551…

```
print('These are the 2 frequent itemsets:\n\n')
for combo_num, count in combo_counts.items():
    if (count / len(transaction) * 100) >= minsupport:
        print(f" ({combinations[combo_num-1]})   number of repetition: {coun
```

```
These are the 2 frequent itemsets:


 (('bag', 'cheese'))   number of repetition: 5.
 (('bag', 'ink'))   number of repetition: 3.
 (('bag', 'juice'))   number of repetition: 2.
 (('bag', 'milk'))   number of repetition: 4.
 (('bag', 'pen'))   number of repetition: 3.
 (('bag', 'soap'))   number of repetition: 4.
 (('candy', 'juice'))   number of repetition: 3.
 (('cheese', 'ink'))   number of repetition: 4.
 (('cheese', 'juice'))   number of repetition: 7.
 (('cheese', 'milk'))   number of repetition: 6.
 (('cheese', 'pen'))   number of repetition: 5.
 (('cheese', 'soap'))   number of repetition: 2.
 (('ink', 'juice'))   number of repetition: 2.
 (('ink', 'milk'))   number of repetition: 3.
 (('ink', 'pen'))   number of repetition: 2.
 (('juice', 'milk'))   number of repetition: 5.
 (('juice', 'pen'))   number of repetition: 4.
 (('juice', 'soap'))   number of repetition: 3.
 (('milk', 'pen'))   number of repetition: 2.
 (('milk', 'soap'))   number of repetition: 3.
```

In [552…

```python
combinations = []
for r in range(3,4):
    combos = list(itertools.combinations(items, r))
    combinations.extend(combos)

# Filter out empty tuples
combinations = [combo for combo in combinations if combo]
```

In [553…

```python
combo_counts = {}
for i in range(len(combinations)):
    combo = combinations[i]
    count = 0
    for index, row in transaction.iterrows():
        if set(combo).issubset(row):
            count += 1
    combo_counts[i+1] = count
print('These are the 3 possible combinations\n\n')
for combo_num, count in combo_counts.items():
    print(f"({combinations[combo_num-1]}) number of repetitions {count}\n")
```

```
These are the 3 possible combinations


(('bag', 'candy', 'cheese')) number of repetitions 0

(('bag', 'candy', 'ink')) number of repetitions 0

(('bag', 'candy', 'juice')) number of repetitions 0
```

```
(('bag', 'candy', 'milk')) number of repetitions 0

(('bag', 'candy', 'pen')) number of repetitions 0

(('bag', 'candy', 'soap')) number of repetitions 0

(('bag', 'cheese', 'ink')) number of repetitions 2

(('bag', 'cheese', 'juice')) number of repetitions 1

(('bag', 'cheese', 'milk')) number of repetitions 2

(('bag', 'cheese', 'pen')) number of repetitions 3

(('bag', 'cheese', 'soap')) number of repetitions 1

(('bag', 'ink', 'juice')) number of repetitions 0

(('bag', 'ink', 'milk')) number of repetitions 1

(('bag', 'ink', 'pen')) number of repetitions 2

(('bag', 'ink', 'soap')) number of repetitions 1

(('bag', 'juice', 'milk')) number of repetitions 0

(('bag', 'juice', 'pen')) number of repetitions 1

(('bag', 'juice', 'soap')) number of repetitions 1

(('bag', 'milk', 'pen')) number of repetitions 0

(('bag', 'milk', 'soap')) number of repetitions 3

(('bag', 'pen', 'soap')) number of repetitions 0

(('candy', 'cheese', 'ink')) number of repetitions 0

(('candy', 'cheese', 'juice')) number of repetitions 1

(('candy', 'cheese', 'milk')) number of repetitions 0

(('candy', 'cheese', 'pen')) number of repetitions 0

(('candy', 'cheese', 'soap')) number of repetitions 0

(('candy', 'ink', 'juice')) number of repetitions 0

(('candy', 'ink', 'milk')) number of repetitions 0

(('candy', 'ink', 'pen')) number of repetitions 0

(('candy', 'ink', 'soap')) number of repetitions 0
```

```
(('candy', 'juice', 'milk')) number of repetitions 0

(('candy', 'juice', 'pen')) number of repetitions 1

(('candy', 'juice', 'soap')) number of repetitions 0

(('candy', 'milk', 'pen')) number of repetitions 0

(('candy', 'milk', 'soap')) number of repetitions 0

(('candy', 'pen', 'soap')) number of repetitions 0

(('cheese', 'ink', 'juice')) number of repetitions 1

(('cheese', 'ink', 'milk')) number of repetitions 1

(('cheese', 'ink', 'pen')) number of repetitions 2

(('cheese', 'ink', 'soap')) number of repetitions 0

(('cheese', 'juice', 'milk')) number of repetitions 3

(('cheese', 'juice', 'pen')) number of repetitions 3

(('cheese', 'juice', 'soap')) number of repetitions 1

(('cheese', 'milk', 'pen')) number of repetitions 2

(('cheese', 'milk', 'soap')) number of repetitions 1

(('cheese', 'pen', 'soap')) number of repetitions 0

(('ink', 'juice', 'milk')) number of repetitions 1

(('ink', 'juice', 'pen')) number of repetitions 0

(('ink', 'juice', 'soap')) number of repetitions 0

(('ink', 'milk', 'pen')) number of repetitions 0

(('ink', 'milk', 'soap')) number of repetitions 1

(('ink', 'pen', 'soap')) number of repetitions 0

(('juice', 'milk', 'pen')) number of repetitions 2

(('juice', 'milk', 'soap')) number of repetitions 0

(('juice', 'pen', 'soap')) number of repetitions 0

(('milk', 'pen', 'soap')) number of repetitions 0
```

In [554…
```
print('These are the 3 Frequent item set:\n\n')
for combo_num, count in combo_counts.items():
    if (count / len(transaction) * 100) >= minsupport:
        print(f"({combinations[combo_num-1]}) number of repetitions {count}\
```

```
These are the 3 Frequent item set:


(('bag', 'cheese', 'ink')) number of repetitions 2

(('bag', 'cheese', 'milk')) number of repetitions 2

(('bag', 'cheese', 'pen')) number of repetitions 3

(('bag', 'ink', 'pen')) number of repetitions 2

(('bag', 'milk', 'soap')) number of repetitions 3

(('cheese', 'ink', 'pen')) number of repetitions 2

(('cheese', 'juice', 'milk')) number of repetitions 3

(('cheese', 'juice', 'pen')) number of repetitions 3

(('cheese', 'milk', 'pen')) number of repetitions 2

(('juice', 'milk', 'pen')) number of repetitions 2
```

In [555…
```
combinations = []
for r in range(4,20):
    combos = list(itertools.combinations(items, r))
    combinations.extend(combos)

# Filter out empty tuples
combinations = [combo for combo in combinations if combo]
```

In [556…
```
combo_counts = {}
for i in range(len(combinations)):
    combo = combinations[i]
    count = 0
    for index, row in transaction.iterrows():
        if set(combo).issubset(row):
            count += 1
            #print(f"Combo {i+1} ({combo}) is a subset of row {index} in the
    combo_counts[i+1] = count
print('These are the 4 and 4+ Frequent item sets:\n\n')
for combo_num, count in combo_counts.items():
    print(f"({combinations[combo_num-1]}) number of repetitions {count}.\n\n
```

```
These are the 4 and 4+ Frequent item sets:
```

```
(('bag', 'candy', 'cheese', 'ink')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'juice')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'milk')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice')) number of repetitions 0.

(('bag', 'candy', 'ink', 'milk')) number of repetitions 0.

(('bag', 'candy', 'ink', 'pen')) number of repetitions 0.

(('bag', 'candy', 'ink', 'soap')) number of repetitions 0.

(('bag', 'candy', 'juice', 'milk')) number of repetitions 0.

(('bag', 'candy', 'juice', 'pen')) number of repetitions 0.

(('bag', 'candy', 'juice', 'soap')) number of repetitions 0.

(('bag', 'candy', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'milk')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'pen')) number of repetitions 2.
```

```
(('bag', 'cheese', 'ink', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'milk')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'pen')) number of repetitions 1.

(('bag', 'cheese', 'juice', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'milk', 'pen')) number of repetitions 0.

(('bag', 'cheese', 'milk', 'soap')) number of repetitions 1.

(('bag', 'cheese', 'pen', 'soap')) number of repetitions 0.

(('bag', 'ink', 'juice', 'milk')) number of repetitions 0.

(('bag', 'ink', 'juice', 'pen')) number of repetitions 0.

(('bag', 'ink', 'juice', 'soap')) number of repetitions 0.

(('bag', 'ink', 'milk', 'pen')) number of repetitions 0.

(('bag', 'ink', 'milk', 'soap')) number of repetitions 1.

(('bag', 'ink', 'pen', 'soap')) number of repetitions 0.

(('bag', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'juice', 'pen', 'soap')) number of repetitions 0.

(('bag', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'juice')) number of repetitions 0.
```

```
(('candy', 'cheese', 'ink', 'milk')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'milk')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'milk', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'milk', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'pen', 'soap')) number of repetitions 0.

(('candy', 'ink', 'juice', 'milk')) number of repetitions 0.

(('candy', 'ink', 'juice', 'pen')) number of repetitions 0.

(('candy', 'ink', 'juice', 'soap')) number of repetitions 0.

(('candy', 'ink', 'milk', 'pen')) number of repetitions 0.

(('candy', 'ink', 'milk', 'soap')) number of repetitions 0.

(('candy', 'ink', 'pen', 'soap')) number of repetitions 0.

(('candy', 'juice', 'milk', 'pen')) number of repetitions 0.

(('candy', 'juice', 'milk', 'soap')) number of repetitions 0.
```

```
(('candy', 'juice', 'pen', 'soap')) number of repetitions 0.

(('candy', 'milk', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'milk')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'pen')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'milk', 'pen')) number of repetitions 0.

(('cheese', 'ink', 'milk', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'juice', 'milk', 'pen')) number of repetitions 2.

(('cheese', 'juice', 'milk', 'soap')) number of repetitions 0.

(('cheese', 'juice', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'milk', 'pen', 'soap')) number of repetitions 0.

(('ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('ink', 'milk', 'pen', 'soap')) number of repetitions 0.

(('juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'juice')) number of repetitions 0.
```

```
(('bag', 'candy', 'cheese', 'ink', 'milk')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'juice', 'milk')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'juice', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'juice', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'milk')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'pen')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'ink', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'juice', 'pen', 'soap')) number of repetitions 0.
```

```
(('bag', 'candy', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'milk')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'pen')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'milk', 'pen')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'milk', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('bag', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'juice', 'milk')) number of repetitions 0.
```

```
(('candy', 'cheese', 'ink', 'juice', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'juice', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'milk', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'milk', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'milk', 'pen')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'milk', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('candy', 'ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('candy', 'ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('candy', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.
```

```
(('cheese', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('ink', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'juice', 'milk')) number of repetitions
0.

(('bag', 'candy', 'cheese', 'ink', 'juice', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'juice', 'soap')) number of repetitions
0.

(('bag', 'candy', 'cheese', 'ink', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'juice', 'milk', 'pen')) number of repetitions
0.

(('bag', 'candy', 'cheese', 'juice', 'milk', 'soap')) number of repetitions
0.

(('bag', 'candy', 'cheese', 'juice', 'pen', 'soap')) number of repetitions
0.

(('bag', 'candy', 'cheese', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.
```

```
(('bag', 'candy', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'milk', 'pen')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'milk', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'juice', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'cheese', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'ink', 'juice', 'milk', 'pen')) number of repetitions
0.

(('candy', 'cheese', 'ink', 'juice', 'milk', 'soap')) number of repetitions
0.

(('candy', 'cheese', 'ink', 'juice', 'pen', 'soap')) number of repetitions
0.

(('candy', 'cheese', 'ink', 'milk', 'pen', 'soap')) number of repetitions 0.

(('candy', 'cheese', 'juice', 'milk', 'pen', 'soap')) number of repetitions
0.

(('candy', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('cheese', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repetitions 0.

(('bag', 'candy', 'cheese', 'ink', 'juice', 'milk', 'pen')) number of repeti
tions 0.

(('bag', 'candy', 'cheese', 'ink', 'juice', 'milk', 'soap')) number of repet
itions 0.
```

```
(('bag', 'candy', 'cheese', 'ink', 'juice', 'pen', 'soap')) number of repeti
tions 0.


(('bag', 'candy', 'cheese', 'ink', 'milk', 'pen', 'soap')) number of repetit
ions 0.


(('bag', 'candy', 'cheese', 'juice', 'milk', 'pen', 'soap')) number of repet
itions 0.


(('bag', 'candy', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repetiti
ons 0.


(('bag', 'cheese', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repetit
ions 0.


(('candy', 'cheese', 'ink', 'juice', 'milk', 'pen', 'soap')) number of repet
itions 0.


(('bag', 'candy', 'cheese', 'ink', 'juice', 'milk', 'pen', 'soap')) number o
f repetitions 0.
```

In [557…
```
print('These are all the other frequent item sets:\n')
for combo_num, count in combo_counts.items():
    if (count / len(transaction) * 100) >= minsupport:
        print(f"({combinations[combo_num-1]}) number of repetitions: {count}
```

```
These are all the other frequent item sets:

(('bag', 'cheese', 'ink', 'pen')) number of repetitions: 2 .
(('cheese', 'juice', 'milk', 'pen')) number of repetitions: 2 .
```

In [558…
```
elapsed_time = time.time() - start_time
print("--- %s seconds ---" % (elapsed_time))
```

```
--- 0.16816091537475586 seconds ---
```

Aplying Apriori Library

In [559…
```
import csv
import pandas as pd
import time
from itertools import combinations
from apyori import apriori
from mlxtend.frequent_patterns import apriori, fpmax
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
```

In [560…
```
#The below will allow us to read the file
def load_data(filename):
    full_transaction_list= []
    with open(filename, encoding = 'utf-8-sig') as data:
        transaction_data = csv.reader(data, delimiter = ',')
        for  row in transaction_data:
            filtered_rows = [value for value in row if value != '']
            full_transaction_list.append(filtered_rows)
        return full_transaction_list
```

In [561…
```
new_list = load_data(input('please enter file name\n\n'))
```

```
please enter file name

Transactions_5.csv
```

In [562…
```
new_list
```

Out[562]:
```
[['ink', 'pen', 'cheese', 'bag'],
 ['milk', 'pen', 'juice', 'cheese'],
 ['milk', 'juice'],
 ['juice', 'milk', 'cheese'],
 ['ink', 'pen', 'cheese', 'bag'],
 ['milk', 'pen', 'juice', 'cheese'],
 ['milk', 'soap', 'bag'],
 ['juice', 'soap', 'bag'],
 ['juice', 'soap'],
 ['juice', 'milk', 'ink'],
 ['juice', 'candy'],
 ['pen', 'juice', 'candy'],
 ['pen', 'juice', 'bag', 'cheese'],
 ['cheese', 'milk', 'bag'],
 ['cheese', 'juice', 'soap'],
 ['cheese', 'juice', 'ink'],
 ['milk', 'cheese', 'ink'],
 ['cheese', 'juice', 'candy'],
 ['cheese', 'milk', 'soap', 'bag'],
 ['milk', 'ink', 'soap', 'bag']]
```

In [563…
```
TranEn = TransactionEncoder()
TranEn_ary=TranEn.fit(new_list).transform(new_list)
```

In [564…
```
Dataframe = pd.DataFrame(TranEn_ary, columns=TranEn.columns_)
Dataframe
```

Out[564]:

	bag	candy	cheese	ink	juice	milk	pen	soap
0	True	False	True	True	False	False	True	False
1	False	False	True	False	True	True	True	False
2	False	False	False	False	True	True	False	False
3	False	False	True	False	True	True	False	False
4	True	False	True	True	False	False	True	False
5	False	False	True	False	True	True	True	False
6	True	False	False	False	False	True	False	True
7	True	False	False	False	True	False	False	True
8	False	False	False	False	True	False	False	True
9	False	False	False	True	True	True	False	False
10	False	True	False	False	True	False	False	False
11	False	True	False	False	True	False	True	False
12	True	False	True	False	True	False	True	False
13	True	False	True	False	False	True	False	False
14	False	False	True	False	True	False	False	True
15	False	False	True	True	True	False	False	False
16	False	False	True	True	False	True	False	False
17	False	True	True	False	True	False	False	False
18	True	False	True	False	False	True	False	True
19	True	False	False	True	False	True	False	True

In [567…
```
frequentItemsets = apriori(Dataframe, min_support = 0.10, use_colnames=True)
frequentItemsets
```

Out[567]:

	support	itemsets
0	0.40	(bag)
1	0.15	(candy)
2	0.60	(cheese)
3	0.30	(ink)
4	0.65	(juice)

5	0.50	(milk)
6	0.30	(pen)
7	0.30	(soap)
8	0.25	(cheese, bag)
9	0.15	(ink, bag)
10	0.10	(juice, bag)
11	0.20	(milk, bag)
12	0.15	(pen, bag)
13	0.20	(soap, bag)
14	0.15	(candy, juice)
15	0.20	(cheese, ink)
16	0.35	(cheese, juice)
17	0.30	(cheese, milk)
18	0.25	(cheese, pen)
19	0.10	(cheese, soap)
20	0.10	(ink, juice)
21	0.15	(ink, milk)
22	0.10	(ink, pen)
23	0.25	(juice, milk)
24	0.20	(juice, pen)
25	0.15	(juice, soap)
26	0.10	(pen, milk)
27	0.15	(soap, milk)
28	0.10	(cheese, ink, bag)
29	0.10	(cheese, milk, bag)
30	0.15	(cheese, pen, bag)
31	0.10	(ink, pen, bag)
32	0.15	(soap, milk, bag)
33	0.10	(cheese, ink, pen)
34	0.15	(cheese, juice, milk)
35	0.15	(cheese, juice, pen)

36	0.10	(cheese, pen, milk)
37	0.10	(pen, juice, milk)
38	0.10	(cheese, ink, pen, bag)
39	0.10	(pen, cheese, juice, milk)

In [568…
```
Rules = association_rules(frequentItemsets, metric="confidence", min_threshc
Rules
```

Out[568]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	le
0	(bag)	(cheese)	0.40	0.60	0.25	0.625000	1.041667	
1	(soap)	(bag)	0.30	0.40	0.20	0.666667	1.666667	
2	(candy)	(juice)	0.15	0.65	0.15	1.000000	1.538462	
3	(ink)	(cheese)	0.30	0.60	0.20	0.666667	1.111111	
4	(cheese)	(juice)	0.60	0.65	0.35	0.583333	0.897436	-
5	(milk)	(cheese)	0.50	0.60	0.30	0.600000	1.000000	
6	(pen)	(cheese)	0.30	0.60	0.25	0.833333	1.388889	
7	(pen)	(juice)	0.30	0.65	0.20	0.666667	1.025641	
8	(ink, bag)	(cheese)	0.15	0.60	0.10	0.666667	1.111111	
9	(cheese, pen)	(bag)	0.25	0.40	0.15	0.600000	1.500000	
10	(cheese, bag)	(pen)	0.25	0.30	0.15	0.600000	2.000000	
11	(bag, pen)	(cheese)	0.15	0.60	0.15	1.000000	1.666667	
12	(ink, pen)	(bag)	0.10	0.40	0.10	1.000000	2.500000	
13	(ink, bag)	(pen)	0.15	0.30	0.10	0.666667	2.222222	
14	(bag, pen)	(ink)	0.15	0.30	0.10	0.666667	2.222222	
15	(soap, milk)	(bag)	0.15	0.40	0.15	1.000000	2.500000	
16	(soap, bag)	(milk)	0.20	0.50	0.15	0.750000	1.500000	
17	(bag, milk)	(soap)	0.20	0.30	0.15	0.750000	2.500000	
18	(ink, pen)	(cheese)	0.10	0.60	0.10	1.000000	1.666667	
19	(juice, milk)	(cheese)	0.25	0.60	0.15	0.600000	1.000000	
20	(cheese, pen)	(juice)	0.25	0.65	0.15	0.600000	0.923077	-
21	(juice, pen)	(cheese)	0.20	0.60	0.15	0.750000	1.250000	

22	(milk, pen)	(cheese)	0.10	0.60	0.10	1.000000	1.666667
23	(milk, pen)	(juice)	0.10	0.65	0.10	1.000000	1.538462
24	(cheese, ink, pen)	(bag)	0.10	0.40	0.10	1.000000	2.500000
25	(cheese, ink, bag)	(pen)	0.10	0.30	0.10	1.000000	3.333333
26	(bag, cheese, pen)	(ink)	0.15	0.30	0.10	0.666667	2.222222
27	(bag, ink, pen)	(cheese)	0.10	0.60	0.10	1.000000	1.666667
28	(ink, pen)	(cheese, bag)	0.10	0.25	0.10	1.000000	4.000000
29	(ink, bag)	(cheese, pen)	0.15	0.25	0.10	0.666667	2.666667
30	(bag, pen)	(cheese, ink)	0.15	0.20	0.10	0.666667	3.333333
31	(cheese, juice, pen)	(milk)	0.15	0.50	0.10	0.666667	1.333333
32	(cheese, milk, pen)	(juice)	0.10	0.65	0.10	1.000000	1.538462
33	(juice, milk, pen)	(cheese)	0.10	0.60	0.10	1.000000	1.666667
34	(cheese, juice, milk)	(pen)	0.15	0.30	0.10	0.666667	2.222222
35	(milk, pen)	(cheese, juice)	0.10	0.35	0.10	1.000000	2.857143

Conclusion

In conclusion, for the Frist transaction, it was very noticible how faster Apriori (time: 0.08525896072387695 seconds) was compared to brute force method (Time: 0.16816091537475586 seconds). Also, by using an existing library, I was able to compare the results to my implementation and they both had the same number of Rules meaning that the implementation was working correctly. Also, by implementing the brute force method, I was able to see that the One, two, and three itemsets matched to ones from the apriori implementation meaning that the Brute Force even though it took more time, it returned the same results as the Apriori Algorithm

In [ ]: