In [2]:

```
import csv
from collections import defaultdict
from itertools import chain, combinations

def readdata():
    with open('example-simple.csv','r') as csvfile:
                read=csv.reader(csvfile, delimiter=',')
                data=[]
                items=set()
                count=0
                for row in read:
                        data.append(row)
                        for x in row:
                                if(x):
                                        items.add(frozenset([x]))
                # print data
    dataset=list(frozenset(x) for x in data)
    return items,dataset

def calculateFreq(itemset, dataset, MinSupport,freqSet):
        freq=set()
        localSet=defaultdict(int)
        #print("ITEM=",itemset)
        for x in itemset:
                if(x!=set({})):
                        for trans in dataset:
                                if x.issubset(trans):
                                        freqSet[x]+=1
                                        localSet[x]+=1

        for item, count in localSet.items():
                support = count

                if support>=MinSupport:
                        freq.add(item)
        #print("freq",freq)
        return freq

def joinSet(itemSet, length): # returns all the subset of given length
        return set([i.union(j) for i in itemSet for j in itemSet if len(i.unio
n(j)) == length])

def subsets(arr):
        return chain(*[combinations(arr, i+1) for i, a in enumerate(arr)])

def Apriori(itemset, dataset, MinSupport, MinConfidence):
        freqSet=defaultdict(int)
        freq=calculateFreq(itemset,dataset,MinSupport,freqSet)
        universal=set(freq)
        largeSet=dict()

        k=2
        while (universal):
```

```
                largeSet[k-1]=universal
                newItemSet=joinSet(universal,k)

                freq=calculateFreq(newItemSet,dataset,MinSupport,freqSet)
                universal=freq
                #print("Uni",universal)
                k=k+1

        def getSupport(item):
                return freqSet[item]

        Subsets=[]
        AssocRules=[]

        for key, value in list(largeSet.items())[-1:]:
                for item in value:
                        _subset=map(frozenset,[x for x in subsets(item)])
                        Subsets.append(_subset)
                        print("Set of transactions above minimum support count
:")
                        print([x for x in subsets(item)])
                        for element in _subset:
                                remain=item.difference(element)
                                if(len(remain)>0):
                                        confidence=getSupport(item)/getSupport
(element)
                                        if(confidence>MinConfidence):
                                                AssocRules.append(((tuple(elem
ent),tuple(remain)), confidence))


        print("Association rules")
        for x in AssocRules:
                print(x)

itemset, dataset=readdata()
Apriori(itemset, dataset, 2, 0.7)
```

```
Set of transactions above minimum support count:
[('F',), ('E',), ('D',), ('B',), ('F', 'E'), ('F', 'D'), ('F', 'B'
), ('E', 'D'), ('E', 'B'), ('D', 'B'), ('F', 'E', 'D'), ('F', 'E',
'B'), ('F', 'D', 'B'), ('E', 'D', 'B'), ('F', 'E', 'D', 'B')]
Set of transactions above minimum support count:
[('F',), ('E',), ('A',), ('C',), ('F', 'E'), ('F', 'A'), ('F', 'C'
), ('E', 'A'), ('E', 'C'), ('A', 'C'), ('F', 'E', 'A'), ('F', 'E',
'C'), ('F', 'A', 'C'), ('E', 'A', 'C'), ('F', 'E', 'A', 'C')]
Association rules
((('B',), ('F', 'E', 'D')), 1.0)
((('F', 'B'), ('D', 'E')), 1.0)
((('B', 'E'), ('F', 'D')), 1.0)
((('D', 'B'), ('F', 'E')), 1.0)
((('F', 'B', 'E'), ('D',)), 1.0)
((('F', 'B', 'D'), ('E',)), 1.0)
((('D', 'B', 'E'), ('F',)), 1.0)
((('F', 'C'), ('A', 'E')), 1.0)
((('A', 'E'), ('F', 'C')), 1.0)
((('C', 'E'), ('F', 'A')), 1.0)
((('A', 'C'), ('F', 'E')), 1.0)
((('F', 'A', 'E'), ('C',)), 1.0)
((('F', 'C', 'E'), ('A',)), 1.0)
((('F', 'C', 'A'), ('E',)), 1.0)
((('A', 'C', 'E'), ('F',)), 1.0)
```

In [ ]:

```
#Conclusion: On  Given Data Aprior Algorithm Generate Above Rule's with Minimu
m Support 2 and Minimum Confidence 0.7.
```