VIT-AP UNIVERSITY, ANDHRA PRADESH

DWDM

Lab Exercise –8: Functions related to Apriori Algorithm

Academic year: 2021-2022 **Branch/ Class:** B.Tech

Semester: Fall **Date:** 08-11-22

Faculty Name: Dr.Aravapalli Rama Satish **School:** SCOPE

NAME: Majjiga Jaswanth REGNO:20BCD7171

****

Question 1: Develop a python function to return count of each set in the list of element sets.
Input: Key: (a,b,c),
Data: [[a,b,c,d],[b,c,d],[b,c,d,e],[a,b,c,d,e]]
Output: (a,b,c) --> 2

Code:

```
def Myfun(key, data):
  count = 0
  for i in data:
    b = all([j in i for j in key])
    if b:
      print(i)
      count = count+1
  return count
key = ['a','b','c']
data = [['a','b','c','d'],['b','c','d'],['b','c','d','e'],['a','b','c',
'd','e']]
Myfun(key,data)
```

Output:

```
def Myfun(key, data):
  count = 0
  for i in data:
    b = all([j in i for j in key])
    if b:
      print(i)
      count = count+1
  return count
key = ['a','b','c']
data = [['a','b','c','d'],['b','c','d'],['b','c','d','e'],['a','b','c','d','e']]
Myfun(key,data)

['a', 'b', 'c', 'd']
['a', 'b', 'c', 'd', 'e']
2
```

Question-2: Develop a python function to perform self-join operation on a set of items(of size k) to yield unique set of items of size(k+1).

Input:((a,b),(a,c),(b,c),(b,d),(c,d),(c,e),(c,f))
Output:((a,b,c),(b,c,d),(c,d,e),(c,d,f),(c,e,f))
Note:It should not generate (c,d,e,f) since given k=2. we should generate sets of k=3 but not k=4.

Code:

```
def fun1(key,data):
  count=0
  for i in data:
     bool1=all([j in i for j in key])
     if bool1:
       count+=1
       return i
def func2(S):
  result=list()
  for i in S:
    for j in S:
      if i!=j and [i,j] not in result:
        result.append((i,j))
      if i==j and i not in result:
        result.append((i))
  return result
Input=[('a','b'),('a','c'),('b','c'),('b','d'),('c','d' ),('c','e'),('c
','f')]
arr=func2(Input)
arr=list(set(arr))
result=list()
for i in arr:
  b=set()
  for j in i:
    a=[ b.add(k) for k in j]
  if len(b)==3:
      b=list(b)
      b.sort()
      result.append(b)
  check=list()
  result1=list()
  for i in range(len(Input)-1):
   key=set(Input[i]+Input[i+1])
   key=list(key)
   print(key not in check)
   if key not in check:
     check.append(key)
     result1.append(fun1(key,result))
print(result1)
```

Output:

```
False
True
True
True
False
True
False
True
True
[['a', 'b', 'c'], ['b', 'c', 'd'], ['c', 'd', 'e'], ['c', 'e', 'f']]
```

Question-3:Develop a python function to generate nonempty subsets for a given list of items.
Input:[a,b,c]
Ouput:[(a),(b),(c),(a,b),(a,c),(b,c)]

Code:

```
def subset(l):
  lists = []
  for i in range(len(l)+1):
    for j in range(i):
      lists.append(l[j: i])
  return lists
l1 = ["a","b","c"]
print(subset(l1))
```

Output:

```
[ ]  def subset(l):
       lists = []
       for i in range(len(l)+1):
         for j in range(i):
           lists.append(l[j: i])
       return lists
     l1 = ["a","b","c"]
     print(subset(l1))

     [['a'], ['a', 'b'], ['b'], ['a', 'b', 'c'], ['b', 'c'], ['c']]
```

Question-4:Use mlxtend/apyori package to apply Apriori algorithm on following transactions:
(("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 pandas as pd
from mlxtend.preprocessing import TransactionEncoder
```

```
dataset=[["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']]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
from mlxtend.frequent_patterns import apriori
apriori(df, use_colnames = True)
```

output:

```
[ ] import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    dataset=[["a","b","c"], ["a","b"], ["a","b","d"], ["b","e"], ["b","c","e"], ["a","
    ["a","b","d"], ["c","e"], ["a","b","d","e"], ["a",'b','e','c']]
    te = TransactionEncoder()
    te_ary = te.fit(dataset).transform(dataset)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    from mlxtend.frequent_patterns import apriori
    apriori(df, use_colnames = True)
```

	support	itemsets
0	0.727273	(a)
1	0.727273	(b)
2	0.545455	(e)
3	0.545455	(a, b)