Apriori algorithm

Course: CS634—Data Mining
Professor: Jason Wang
Student: Shih-Chuan Weng
UCID: sw464
Email: sw464@njit.edu
Subject: Midterm Project Report
Topic: Apriori algorithm

1. Introduction

Apriori algorithm is popular in Data Mining area. We can implement this algorithm with a database to see which pairs of goods have high frequency being bought by customers.

With this algorithm, we can make the goods more convenience to be took from a store. For example, the result of the algorithm presents toothbrush and toothpaste often being took by customers. Therefore, in a shop we can put these two goods on the same shelf where customers can easily pick them.

2. Description

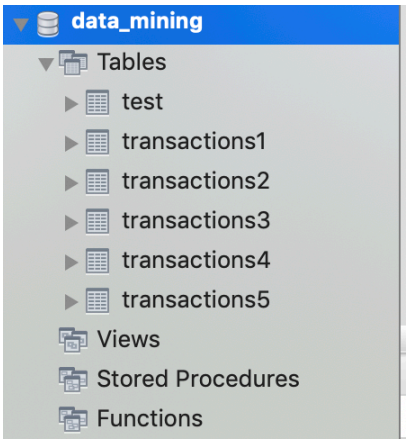
I took Amazon as my data resources, creating 5 transactions database with 10 items and 20 records in each. Gave different combination of records which will be closer to real life. This project can choose one out of five databases each time to see which support_value and confidence_value are fit to users' anticipation.

3. Requirements

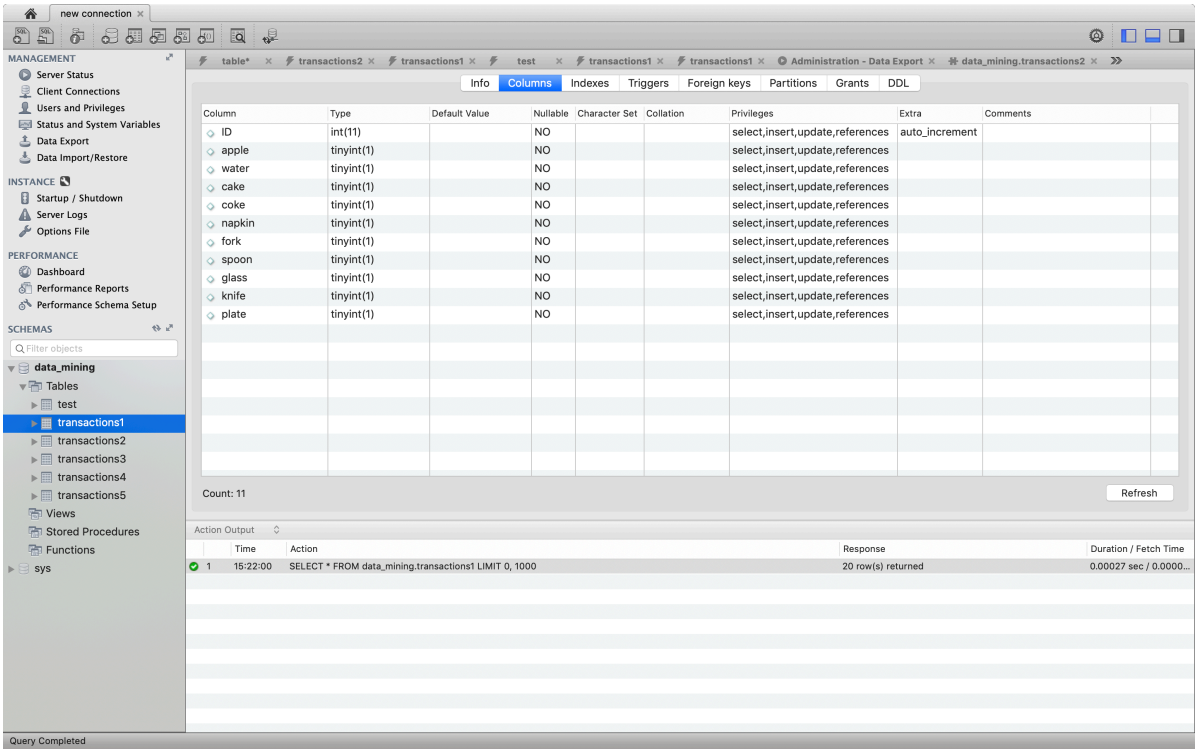
Operating System: MacOS
Programming Language: Python, SQL
DataBase: MySQL

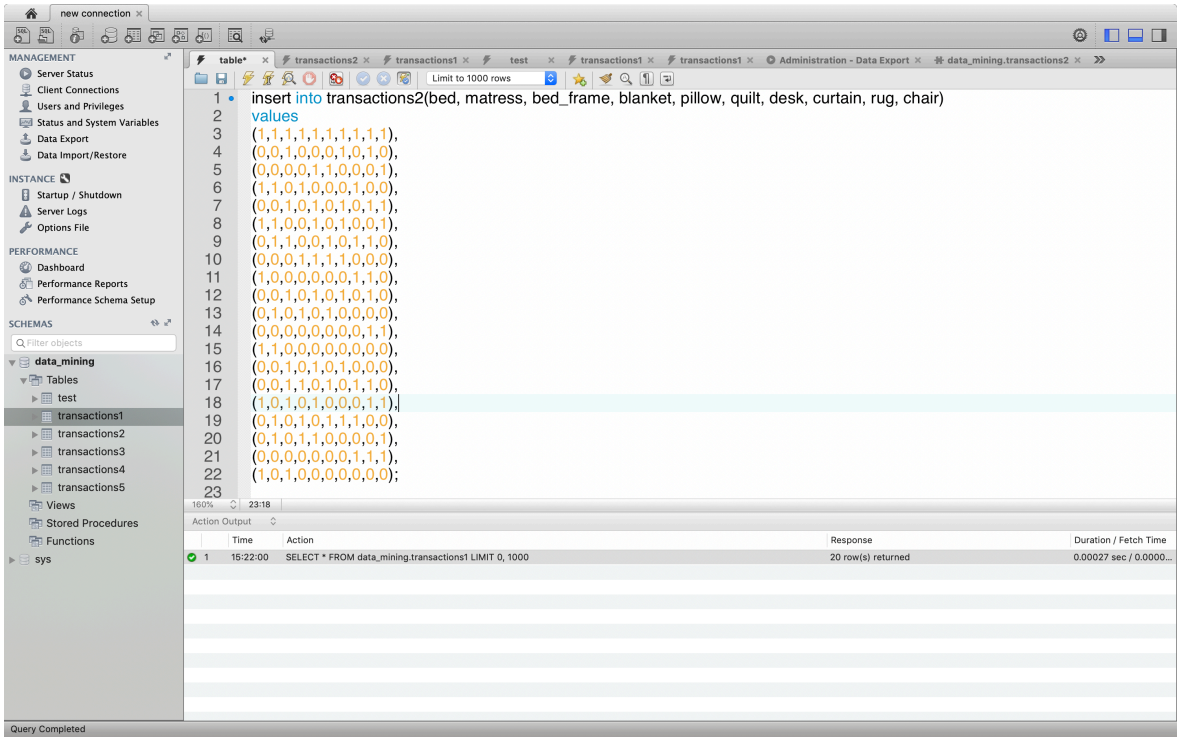
4. Create DataBase

Used MySQL workbench as my GUI. Created “data_mining” Database and transaction1~transactions5 tables.

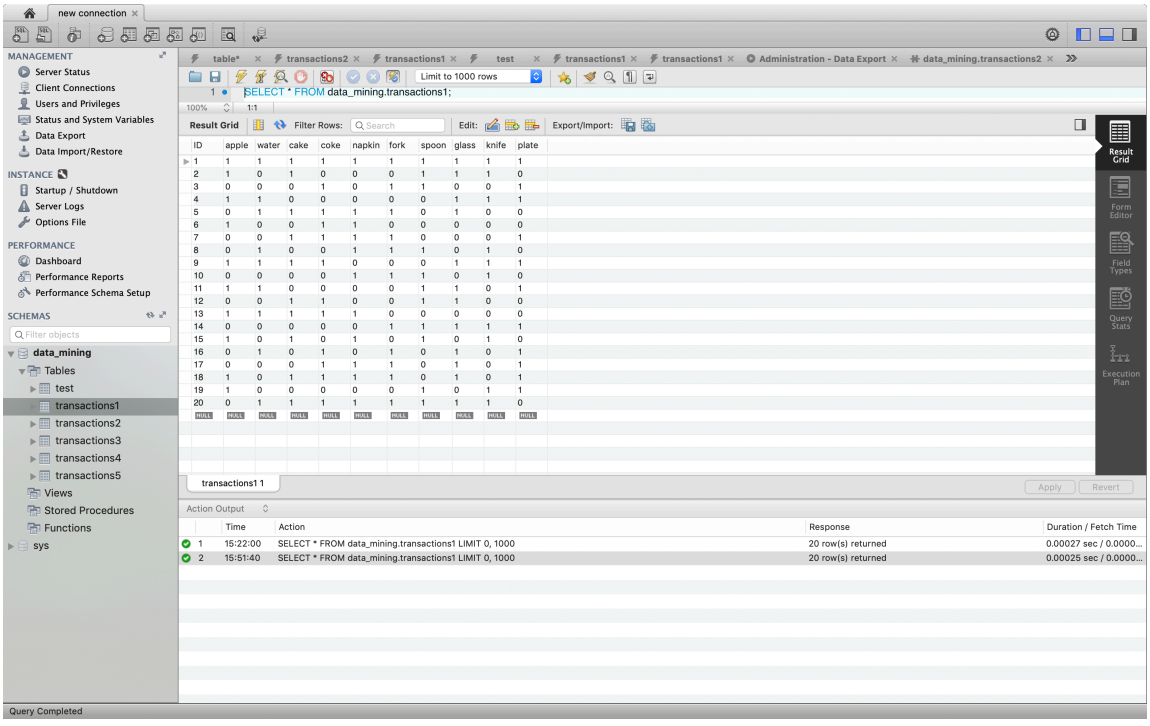


Structure of each tables, which I store the second to end column in boolean except ID stored in int type. The other four tables have the same structure.





Used this SQL syntax to generate my value in a table. For this picture is transactions2. The others used the same way to insert values. However, the values are different.
The look of my table transactions1



5. Source Code Explaination

a) Import library and connect DataBase

Import mysql.connector to connect MySQL with Python.
Import sys to exit the following process if there is no support_val and confidence_val exist.
From itertools import permutations to make the keys do permutations, which is for calculate confidence_val.

```
1  import mysql.connector
2  import sys
3  from itertools import permutations
4
5  mydb = mysql.connector.connect(
6    host="localhost",
7    database="data_mining",
8    user="",
9    passwd=""
10 )
11 mycursor = mydb.cursor()
12
```

b) Apriori Function for First Round

Line 17 - 21 are to fetch columns' names and sort them in DataBase. First Round means we only have to call this function once and on the first round. Line 22 - 27 are to calculate the amounts of good customers buy, then divide by 20, total amounts of customers. If the value larger than or equal to the support_val user enter then it will be stored.

```
13 def Apriori(columns, select_db, rows):
14     count = 0
15     columns_name = []
16     every_support = {}
17     for i in columns:
18         if(i[0]=='ID'):
19             continue
20         columns_name.append(i[0])
21     columns_name = sorted(columns_name)
22     for i in columns_name:
23         mycursor.execute("select count(ID) from "+select_db+" where "+i+" = 1")
24         result = mycursor.fetchall()
25         if float(result[0][0]/rows)>=support_val:
26             every_support[i] = float(result[0][0]/rows)
27     all_support.update(every_support)
28     return every_support, columns_name, count
29
```

C) Aprioir1 Function

Line 32 grab the key name in every_support, includes calculated support_val from the first time to the last time. Line 34 SQL syntax for my later combination. Line 36 - 37 initiates sql for sql syntax and split the name, the key type in the dictionary is 'A B' for example, into list. Line 38 and Line 40 are used to differentiate second round and larger than second round. Line 39 and 42 are for combination the SQL syntax, for example, if the key is 'A' then the sql will be 'A = 1 and '. Line 43 - 48 are to combine the rest SQL syntax and calculate support_val, same as the above process.

```
29
30  def Apriori1(every_support, columns_name, count, select_db, rows):
31      count+=1
32      key = [i for i in every_support.keys()]
33      every_support = {}
34      a, b, c, d = "select count(ID) from ", " where ", " = 1 and ", " = 1"
35      for i in range(len(key)):
36          sql = ''
37          x = key[i].split(" ")
38          if len(x)==1:
39              sql+=x[0] + c
40          else:
41              for k in range(len(x)):
42                  sql += x[k] + c
43          for j in range((columns_name.index(x[count-1])+1), len(columns_name)):
44              sql_last = a + select_db + b + sql + columns_name[j] + d
45              mycursor.execute(str(sql_last))
46              result = float(mycursor.fetchall()[0][0]/rows)
47              if result>=support_val:
48                  every_support[key[i]+" "+columns_name[j]] = result
49      all_support.update(every_support)
50      return every_support, count
51
```

D) Key names do Permutations

This function is preprocess for calculating confidence_val. If a key contains three words, it will have 3^2-3 amounts of combination. For example, 'A B C' can be 'A C B', 'B A C', 'B C A, 'C A B' and 'C B A'.

```
51
52  def Permutation(key):
53      d, ans = [], []
54      for i in key:
55          d2 = []
56          d2+=i.split(' ')
57          d.append(d2)
58      for i in d:
59          for j in permutations(i):
60              ans.append(j)
61      return ans
62
```

E) Confidence Function

Line 68 is the same function as in the Apriori1 function. Line 69 call the Permutation function. Line 70 is to calculate the space in the key name, for example, 'A B C' the arrow will 2. Line 72 - 77 sort the permutationed name, which is convenience for the later process. Line 78 removes duplicate name then sort it. Line 79 and 86 are differentiate second round and larger than second round. Line 80 - 85 and Line 87 - 98 are used the formula which is used to calculate confidence_val. The value counted by the formula will be stored if the value larger than or equal to the confidence_val entered by user.

```
64  def Confidence(every_support, all_support, count):
65      confidence_element = {}
66      if not every_support:
67          return
68      key = [i for i in every_support.keys()]
69      permutate_name = Permutation(key)
70      arrow = len(permutate_name[0])-1
71      name = []
72      for j in permutate_name:
73          b = []
74          for k in range(arrow):
75              b+=[j[k]]
76              c = sorted(b)
77              name.append(str(' '.join(c)))
78      name = sorted(set(name))
79      if len(key[0])-arrow==2:
80          for i in name:
81              for j in key:
82                  if i in j:
83                      result = every_support[j]/all_support[i]
84                      if result>=confidence_val:
85                          confidence_element[i+"->"+j] = result
86      else:
87          for j in key:
88              for i in name:
89                  amount = 0
90                  for k in i:
91                      if k in j:
92                          amount+=1
93                      else:
94                          break
95                  if amount == len(i):
96                      result = every_support[j]/all_support[i]
97                      if result>=confidence_val:
98                          confidence_element[i+"->"+j] = result
99      return confidence_element
```

F) The functions for entering

setSupport function is for user entering a minimum support value. setConfidence function is for user entering a minimum confidence value. setInput function is for user entering the DataBase, containing 1 - 5, user wants to check. The db function is to return the DataBase which selected by user, the same as switch-case in other language.

```
101 def setSupport():
102     n = input("Enter a minimum support value \n")
103     return n
104 
105 def setConfidence():
106     n = input("Enter a minimum confidence value\n")
107     return n
108 
109 def setInput():
110     n = input("Enter a number from 1 to 5 to choose which database you want to see\n")
111     return n
112 
113 def db(x):
114     return{1:'transactions1', 2:'transactions2', 3:'transactions3', 4:'transactions4', 5:'transactions5', 6:'test'}[x]
```

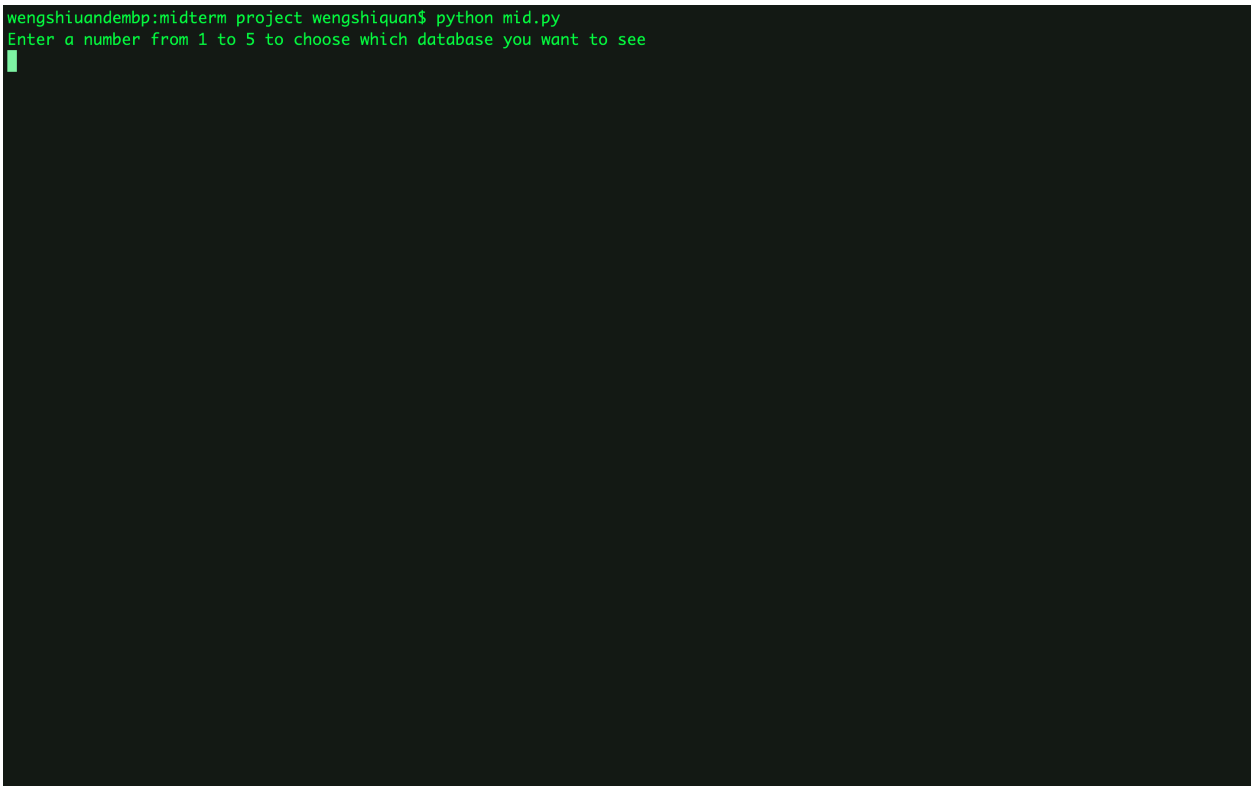
G) Main Function

It's to call the above functions and execute SQL syntax. Line 130 - 133 and Line 140 - 144 are detecting if no support_val larger than or equal to the minimum support value.

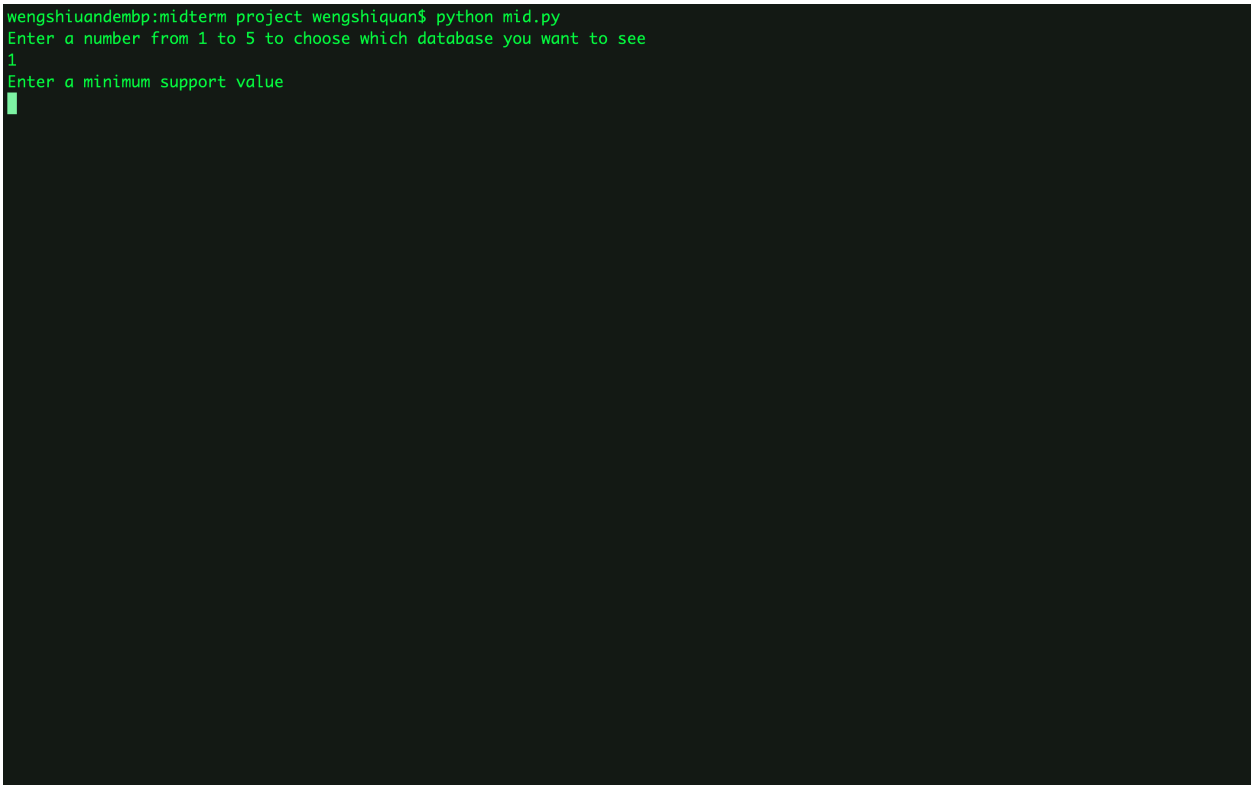
```
116 n = setInput()
117 support_val = float(setSupport())
118 confidence_val = float(setConfidence())
119 select_db = db(int(n))
120 
121 mycursor.execute("select count(ID) from "+select_db)
122 rows = int(mycursor.fetchall()[0][0])
123 
124 mycursor.execute("SHOW columns FROM "+select_db)
125 columns = mycursor.fetchall()
126 all_support = {}
127 
128 every_support, columns_name, count = Apriori(columns, select_db, rows)
129 print("1. Support_Val\n", every_support)
130 if not every_support:
131     sys.exit("1 end ----------Your support value is too high so no output from the first for loop----------")
132 else:
133     print("1 end ----------\n")
134 
135 for _ in range(9):
136     every_support, count = Apriori1(every_support, columns_name, count, select_db, rows)
137     confidence_element = Confidence(every_support, all_support, count)
138     print(str(count+1)+". Support_Val\n", every_support)
139     print(str(count+1)+". Confidence_Val\n", confidence_element)
140     if not every_support:
141         info = (str(count+1)+". end ----------There is no output from the "+str(count+1)+" for loop----------")
142         sys.exit(info)
143     else:
144         print(str(count+1)+". end ----------\n")
```

7. Output

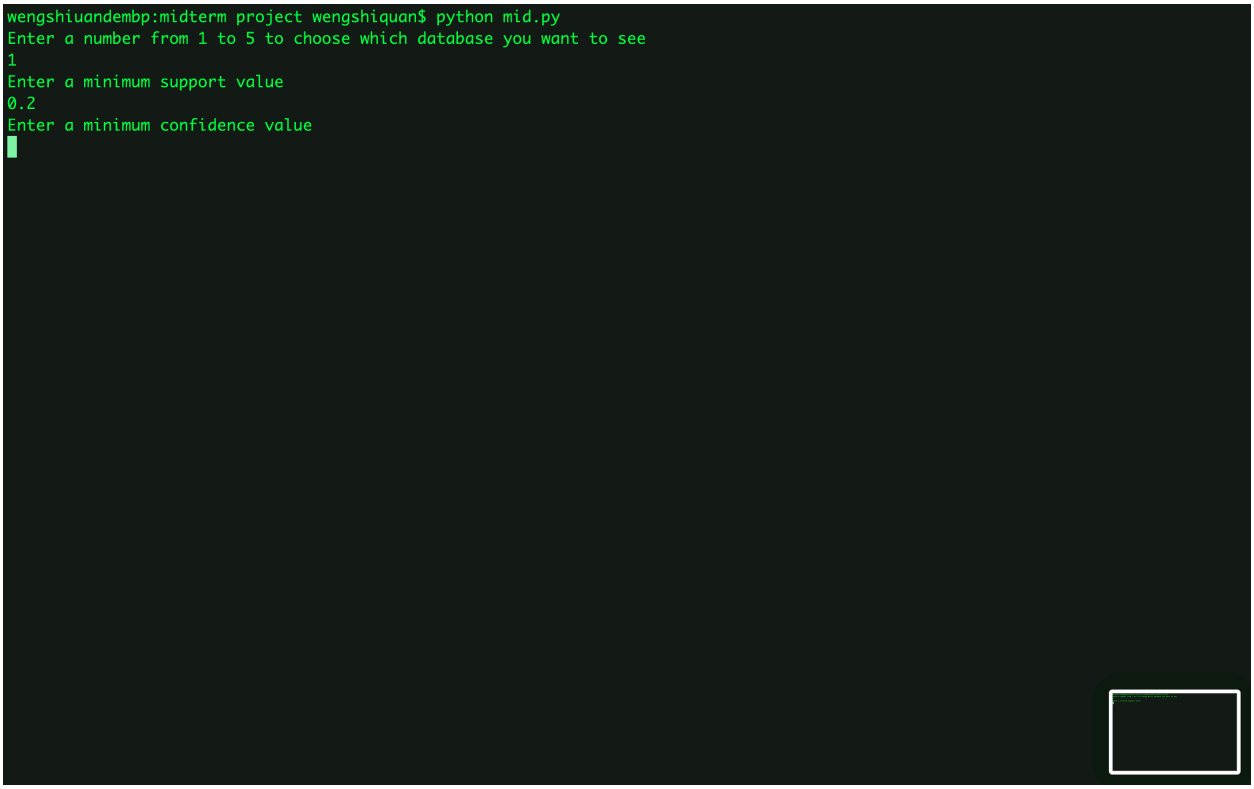
Run the mid.py which is my name of source code. The first step is select database. I entered 1.

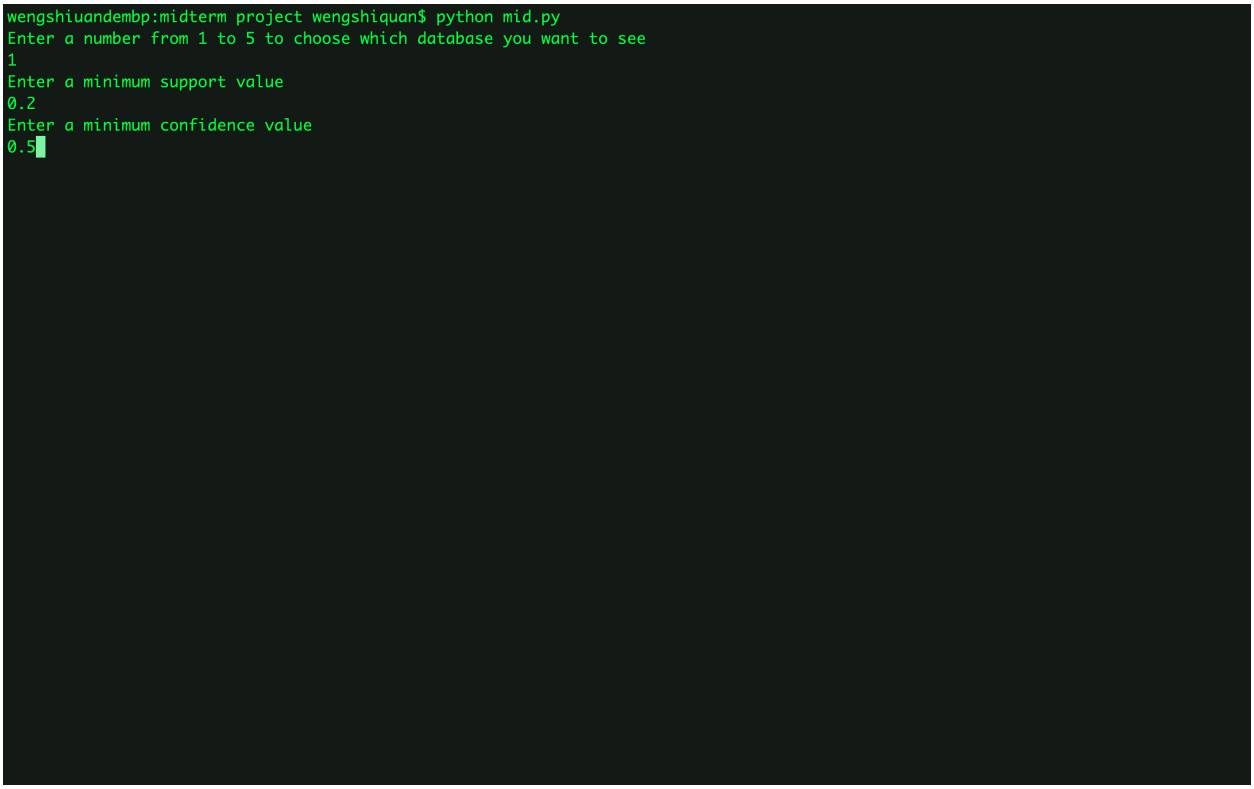


Secondly, enter the minimum support value. I entered 0.2.



Finally, enter the minimum confidence value. I entered 0.5.





```
wengshiuandembp:midterm project wengshiquan$ python mid.py
Enter a number from 1 to 5 to choose which database you want to see
1
Enter a minimum support value
0.2
Enter a minimum confidence value
0.5
1. Support_Val
 {'apple': 0.5, 'cake': 0.5, 'coke': 0.6, 'fork': 0.55, 'glass': 0.6, 'knife': 0.5, 'napkin': 0.55, 'plate': 0.55, 'spoon': 0.55,
'water': 0.45}
1 end ------------------------------------------------

2. Support_Val
 {'apple cake': 0.3, 'apple coke': 0.25, 'apple glass': 0.3, 'apple knife': 0.3, 'apple napkin': 0.25, 'apple plate': 0.3, 'apple
spoon': 0.25, 'apple water': 0.25, 'cake coke': 0.4, 'cake fork': 0.25, 'cake glass': 0.35, 'cake knife': 0.25, 'cake napkin': 0.3
5, 'cake plate': 0.2, 'cake spoon': 0.25, 'cake water': 0.25, 'coke fork': 0.4, 'coke glass': 0.4, 'coke napkin': 0.4, 'coke plate
': 0.35, 'coke spoon': 0.2, 'coke water': 0.3, 'fork glass': 0.35, 'fork knife': 0.25, 'fork napkin': 0.4, 'fork plate': 0.35, 'fo
rk spoon': 0.3, 'fork water': 0.25, 'glass knife': 0.3, 'glass napkin': 0.25, 'glass plate': 0.4, 'glass spoon': 0.3, 'glass water
': 0.35, 'knife napkin': 0.25, 'knife plate': 0.25, 'knife spoon': 0.4, 'knife water': 0.25, 'napkin plate': 0.2, 'napkin spoon':
0.25, 'napkin water': 0.25, 'plate spoon': 0.25, 'plate water': 0.25, 'spoon water': 0.2}
2. Confidence_Val
 {'apple->apple cake': 0.6, 'cake->apple cake': 0.6, 'apple->apple coke': 0.5, 'cake->apple coke': 0.5, 'apple->apple glass': 0.6,
 'glass->apple glass': 0.5, 'apple->apple knife': 0.6, 'knife->apple knife': 0.6, 'napkin->apple knife': 0.5454545454545454, 'appl
e->apple napkin': 0.5, 'apple->apple plate': 0.6, 'plate->apple plate': 0.5454545454545454, 'apple->apple spoon': 0.5, 'apple->app
le water': 0.5, 'water->apple water': 0.5555555555555556, 'cake->cake coke': 0.8, 'coke->cake coke': 0.6666666666666667, 'cake->ca
ke fork': 0.5, 'cake->cake glass': 0.7, 'glass->cake glass': 0.5833333333333334, 'cake->cake knife': 0.5, 'knife->cake knife': 0.5
, 'cake->cake napkin': 0.7, 'napkin->cake napkin': 0.6363636363636362, 'cake->cake spoon': 0.5, 'cake->cake water': 0.5, 'water->c
ake water': 0.5555555555555556, 'coke->coke fork': 0.6666666666666667, 'fork->coke fork': 0.7272727272727273, 'cake->coke glass':
0.8, 'coke->coke glass': 0.6666666666666667, 'glass->coke glass': 0.6666666666666667, 'cake->coke napkin': 0.8, 'coke->coke napkin
': 0.6666666666666667, 'napkin->coke napkin': 0.7272727272727273, 'apple->coke plate': 0.7, 'cake->coke plate': 0.7, 'coke->coke p
late': 0.5833333333333334, 'plate->coke plate': 0.6363636363636362, 'cake->coke water': 0.6, 'coke->coke water': 0.5, 'water->coke
 water': 0.6666666666666666, 'fork->fork glass': 0.6363636363636362, 'glass->fork glass': 0.5833333333333334, 'knife->fork knife':
 0.5, 'fork->fork napkin': 0.7272727272727273, 'napkin->fork napkin': 0.7272727272727273, 'apple->fork plate': 0.7, 'fork->fork pl
ate': 0.6363636363636362, 'plate->fork plate': 0.6363636363636362, 'fork->fork spoon': 0.5454545454545454, 'spoon->fork spoon': 0.
5454545454545454, 'water->fork water': 0.5555555555555556, 'glass->glass knife': 0.5, 'knife->glass knife': 0.6, 'apple->glass pla
```

The first round will not have confidence_val so there is no presentation. See the '2. Confidence_Val' which is messy. Take 'apple->apple cake' : 0.6 for example, it means 'apple -> cake'. 'cake->apple cake : 0.6' means 'cake->apple:0.6'.

The following picture 'apple cake->apple cake coke:0.666666' means 'apple cake->coke: 0.666666'.

```
3. Support_Val
 {'apple cake coke': 0.2, 'apple cake glass': 0.2, 'apple cake knife': 0.2, 'apple cake napkin': 0.2, 'apple coke napkin': 0.2, 'a
pple glass knife': 0.2, 'apple glass plate': 0.25, 'apple glass water': 0.2, 'apple knife plate': 0.2, 'apple knife spoon': 0.2, '
apple plate water': 0.2, 'cake coke fork': 0.25, 'cake coke glass': 0.3, 'cake coke napkin': 0.3, 'cake coke plate': 0.2, 'cake co
ke water': 0.25, 'cake fork glass': 0.2, 'cake fork napkin': 0.25, 'cake glass knife': 0.2, 'cake glass napkin': 0.2, 'cake glass
spoon': 0.2, 'cake glass water': 0.2, 'cake knife spoon': 0.2, 'cake napkin water': 0.2, 'coke fork glass': 0.3, 'coke fork napkin
': 0.3, 'coke fork plate': 0.3, 'coke fork water': 0.2, 'coke glass napkin': 0.25, 'coke glass plate': 0.25, 'coke glass water': 0
.25, 'coke napkin plate': 0.2, 'coke napkin water': 0.2, 'fork glass napkin': 0.25, 'fork glass plate': 0.25, 'fork glass water':
0.2, 'fork knife napkin': 0.2, 'fork knife spoon': 0.25, 'fork napkin plate': 0.2, 'fork napkin spoon': 0.2, 'fork napkin water':
0.2, 'glass knife plate': 0.2, 'glass knife spoon': 0.2, 'glass knife water': 0.2, 'glass plate water': 0.25, 'knife napkin spoon'
: 0.25}
3. Confidence_Val
 {'apple cake->apple cake coke': 0.6666666666666667, 'apple coke->apple cake coke': 0.8, 'cake coke->apple cake coke': 0.5, 'apple
 cake->apple cake glass': 0.6666666666666667, 'apple glass->apple cake glass': 0.6666666666666667, 'cake glass->apple cake glass':
 0.5714285714285715, 'apple cake->apple cake knife': 0.6666666666666667, 'apple knife->apple cake knife': 0.6666666666666667, 'app
le napkin->apple cake knife': 0.8, 'cake knife->apple cake knife': 0.8, 'cake napkin->apple cake knife': 0.5714285714285715, 'knif
e napkin->apple cake knife': 0.8, 'apple cake->apple cake napkin': 0.6666666666666667, 'apple napkin->apple cake napkin': 0.8, 'ca
ke napkin->apple cake napkin': 0.5714285714285715, 'apple cake->apple coke napkin': 0.6666666666666667, 'apple coke->apple coke na
pkin': 0.8, 'apple napkin->apple coke napkin': 0.8, 'cake coke->apple coke napkin': 0.5, 'cake napkin->apple coke napkin': 0.57142
85714285715, 'coke napkin->apple coke napkin': 0.5, 'apple glass->apple glass knife': 0.6666666666666667, 'apple knife->apple glas
s knife': 0.6666666666666667, 'apple napkin->apple glass knife': 0.8, 'glass knife->apple glass knife': 0.6666666666666667, 'glass
 napkin->apple glass knife': 0.8, 'knife napkin->apple glass knife': 0.8, 'apple->apple glass plate': 0.5, 'apple glass->apple gla
ss plate': 0.8333333333333334, 'apple plate->apple glass plate': 0.8333333333333334, 'glass plate->apple glass plate': 0.625, 'app
le glass->apple glass water': 0.6666666666666667, 'apple plate->apple glass water': 0.6666666666666667, 'apple water->apple glass
water': 0.8, 'glass plate->apple glass water': 0.5, 'glass water->apple glass water': 0.5714285714285715, 'plate water->apple glas
s water': 0.8, 'apple knife->apple knife plate': 0.6666666666666667, 'apple napkin->apple knife plate': 0.8, 'apple plate->apple k
nife plate': 0.6666666666666667, 'knife napkin->apple knife plate': 0.8, 'knife plate->apple knife plate': 0.8, 'napkin plate->app
le knife plate': 1.0, 'apple knife->apple knife spoon': 0.6666666666666667, 'apple napkin->apple knife spoon': 0.8, 'apple spoon->
apple knife spoon': 0.8, 'knife napkin->apple knife spoon': 0.8, 'knife spoon->apple knife spoon': 0.5, 'napkin spoon->apple knife
 spoon': 0.8, 'apple plate->apple plate water': 0.6666666666666667, 'apple water->apple plate water': 0.8, 'plate water->apple pla
te water': 0.8, 'cake->cake coke fork': 0.5, 'cake coke->cake coke fork': 0.625, 'cake fork->cake coke fork': 1.0, 'coke fork->cak
e coke fork': 0.625, 'cake->cake coke glass': 0.6, 'cake coke->cake coke glass': 0.7499999999999999, 'cake glass->cake coke glass'
: 0.8571428571428572, 'coke->cake coke glass': 0.5, 'coke glass->cake coke glass': 0.7499999999999999, 'glass->cake coke glass': 0
.5, 'cake->cake coke napkin': 0.6, 'cake coke->cake coke napkin': 0.7499999999999999, 'cake napkin->cake coke napkin': 0.857142857
1428572, 'coke->cake coke napkin': 0.5, 'coke napkin->cake coke napkin': 0.7499999999999999, 'napkin->cake coke napkin': 0.5454545
```

```
4. Support_Val
 {'apple glass plate water': 0.2, 'cake coke fork glass': 0.2, 'cake coke fork napkin': 0.25, 'cake coke glass napkin': 0.2, 'cake
 coke glass water': 0.2, 'cake coke napkin water': 0.2, 'cake fork glass napkin': 0.2, 'coke fork glass napkin': 0.25, 'coke fork
glass plate': 0.2, 'coke fork glass water': 0.2, 'coke fork napkin plate': 0.2, 'fork knife napkin spoon': 0.2}
4. Confidence_Val
 {'apple glass->apple glass plate water': 0.6666666666666667, 'apple glass plate->apple glass plate water': 0.8, 'apple glass wate
r->apple glass plate water': 1.0, 'apple plate->apple glass plate water': 0.6666666666666667, 'apple plate water->apple glass plat
e water': 1.0, 'apple water->apple glass plate water': 0.8, 'glass plate->apple glass plate water': 0.5, 'glass plate water->apple
 glass plate water': 0.8, 'glass water->apple glass plate water': 0.5714285714285715, 'plate water->apple glass plate water': 0.8,
 'cake coke->cake coke fork glass': 0.5, 'cake coke fork->cake coke fork glass': 0.8, 'cake coke glass->cake coke fork glass': 0.6
666666666666667, 'cake fork->cake coke fork glass': 0.8, 'cake fork glass->cake coke fork glass': 1.0, 'cake glass->cake coke fork
 glass': 0.5714285714285715, 'coke fork->cake coke fork glass': 0.5, 'coke fork glass->cake coke fork glass': 0.6666666666666667,
'coke glass->cake coke fork glass': 0.5, 'fork glass->cake coke fork glass': 0.5714285714285715, 'cake->cake coke fork napkin': 0.
5, 'cake coke->cake coke fork napkin': 0.625, 'cake coke fork->cake coke fork napkin': 1.0, 'cake coke napkin->cake coke fork napk
in': 0.8333333333333334, 'cake fork->cake coke fork napkin': 1.0, 'cake fork napkin->cake coke fork napkin': 1.0, 'cake napkin->ca
ke coke fork napkin': 0.7142857142857143, 'coke fork->cake coke fork napkin': 0.625, 'coke fork napkin->cake coke fork napkin': 0.
8333333333333334, 'coke napkin->cake coke fork napkin': 0.625, 'fork knife->cake coke fork napkin': 1.0, 'fork knife napkin->cake
coke fork napkin': 1.25, 'fork napkin->cake coke fork napkin': 0.625, 'knife->cake coke fork napkin': 0.5, 'knife napkin->cake cok
e fork napkin': 1.0, 'apple glass->cake coke glass napkin': 0.6666666666666667, 'cake coke->cake coke glass napkin': 0.5, 'cake co
ke glass->cake coke glass napkin': 0.6666666666666667, 'cake coke napkin->cake coke glass napkin': 0.6666666666666667, 'cake glass
->cake coke glass napkin': 0.5714285714285715, 'cake glass napkin->cake coke glass napkin': 1.0, 'cake napkin->cake coke glass nap
kin': 0.5714285714285715, 'coke glass->cake coke glass napkin': 0.5, 'coke glass napkin->cake coke glass napkin': 0.8, 'coke napki
n->cake coke glass napkin': 0.5, 'glass napkin->cake coke glass napkin': 0.8, 'napkin spoon->cake coke glass napkin': 0.8, 'cake c
oke->cake coke glass water': 0.5, 'cake coke glass->cake coke glass water': 0.6666666666666667, 'cake coke water->cake coke glass
water': 0.8, 'cake glass->cake coke glass water': 0.5714285714285715, 'cake glass water->cake coke glass water': 1.0, 'cake water-
>cake coke glass water': 0.8, 'coke glass->cake coke glass water': 0.5, 'coke glass water->cake coke glass water': 0.8, 'coke wate
r->cake coke glass water': 0.6666666666666667, 'glass water->cake coke glass water': 0.5714285714285715, 'cake coke->cake coke nap
kin water': 0.5, 'cake coke napkin->cake coke napkin water': 0.6666666666666667, 'cake coke water->cake coke napkin water': 0.8, '
cake napkin->cake coke napkin water': 0.5714285714285715, 'cake napkin water->cake coke napkin water': 1.0, 'cake water->cake coke
 napkin water': 0.8, 'coke napkin->cake coke napkin water': 0.5, 'coke napkin water->cake coke napkin water': 1.0, 'coke water->ca
ke coke napkin water': 0.6666666666666667, 'napkin water->cake coke napkin water': 0.8, 'apple glass->cake fork glass napkin': 0.6
666666666666667, 'cake coke->cake fork glass napkin': 0.5, 'cake coke fork->cake fork glass napkin': 0.8, 'cake coke glass->cake f
ork glass napkin': 0.6666666666666667, 'cake coke napkin->cake fork glass napkin': 0.6666666666666667, 'cake fork->c                s
 napkin': 0.8, 'cake fork glass->cake fork glass napkin': 1.0, 'cake fork napkin->cake fork glass napkin': 0.8, 'cak
fork glass napkin': 0.5714285714285715, 'cake glass napkin->cake fork glass napkin': 1.0, 'cake napkin->cake fork gl                0
```

```
>coke fork napkin plate': 0.6666666666666667, 'cake fork->coke fork napkin plate': 0.8, 'cake fork napkin->coke fork napkin plate'
: 0.8, 'cake napkin->coke fork napkin plate': 0.5714285714285715, 'coke fork->coke fork napkin plate': 0.5, 'coke fork napkin->cok
e fork napkin plate': 0.6666666666666667, 'coke fork plate->coke fork napkin plate': 0.6666666666666667, 'coke napkin->coke fork n
apkin plate': 0.5, 'coke napkin plate->coke fork napkin plate': 1.0, 'coke plate->coke fork napkin plate': 0.5714285714285715, 'fo
rk knife->coke fork napkin plate': 0.8, 'fork knife napkin->coke fork napkin plate': 1.0, 'fork napkin->coke fork napkin plate': 0
.5, 'fork napkin plate->coke fork napkin plate': 1.0, 'fork plate->coke fork napkin plate': 0.5714285714285715, 'knife napkin->cok
e fork napkin plate': 0.8, 'napkin plate->coke fork napkin plate': 1.0, 'fork knife->fork knife napkin spoon': 0.8, 'fork knife na
pkin->fork knife napkin spoon': 1.0, 'fork knife spoon->fork knife napkin spoon': 0.8, 'fork napkin->fork knife napkin spoon': 0.5
, 'fork napkin spoon->fork knife napkin spoon': 1.0, 'fork spoon->fork knife napkin spoon': 0.6666666666666667, 'knife napkin->for
k knife napkin spoon': 0.8, 'knife napkin spoon->fork knife napkin spoon': 0.8, 'knife spoon->fork knife napkin spoon': 0.5, 'napk
in spoon->fork knife napkin spoon': 0.8}
4. end ------------------------------------------------

5. Support_Val
 {'cake coke fork glass napkin': 0.2}
5. Confidence_Val
 {'cake coke->cake coke fork glass napkin': 0.5, 'cake coke fork->cake coke fork glass napkin': 0.8, 'cake coke fork glass->cake c
oke fork glass napkin': 1.0, 'cake coke fork napkin->cake coke fork glass napkin': 0.8, 'cake coke glass->cake coke fork glass nap
kin': 0.6666666666666667, 'cake coke glass napkin->cake coke fork glass napkin': 1.0, 'cake coke napkin->cake coke fork glass napk
in': 0.6666666666666667, 'cake fork->cake coke fork glass napkin': 0.8, 'cake fork glass->cake coke fork glass napkin': 1.0, 'cake
 fork glass napkin->cake coke fork glass napkin': 1.0, 'cake fork napkin->cake coke fork glass napkin': 0.8, 'cake glass->cake cok
e fork glass napkin': 0.5714285714285715, 'cake glass napkin->cake coke fork glass napkin': 1.0, 'cake napkin->cake coke fork glas
s napkin': 0.5714285714285715, 'coke fork->cake coke fork glass napkin': 0.5, 'coke fork glass->cake coke fork glass napkin': 0.66
66666666666667, 'coke fork glass napkin->cake coke fork glass napkin': 0.8, 'coke fork napkin->cake coke fork glass napkin': 0.666
6666666666667, 'coke glass->cake coke fork glass napkin': 0.5, 'coke glass napkin->cake coke fork glass napkin': 0.8, 'coke napkin
->cake coke fork glass napkin': 0.5, 'fork glass->cake coke fork glass napkin': 0.5714285714285715, 'fork glass napkin->cake coke
fork glass napkin': 0.8, 'fork napkin->cake coke fork glass napkin': 0.5, 'glass napkin->cake coke fork glass napkin': 0.8}
5. end ------------------------------------------------

6. Support_Val
 {}
6. Confidence_Val
 None
6. end ----------There is no output from the 6 for loop------------------
wengshiuandembp:midterm project wengshiquan$
```

```
wengshiuandembp:midterm project wengshiquan$ python mid.py
Enter a number from 1 to 5 to choose which database you want to see
6
Enter a minimum support value
0.2
Enter a minimum confidence value
0.5
1. Support_Val
 {'A': 0.42857142857142855, 'B': 0.5714285714285714, 'C': 0.7142857142857143, 'D': 0.7142857142857143}
1 end ----------------------------------------------------

2. Support_Val
 {'A B': 0.2857142857142857, 'A D': 0.2857142857142857, 'B C': 0.42857142857142855, 'B D': 0.2857142857142857, 'C D': 0.5714285714
285714}
2. Confidence_Val
 {'A->A B': 0.6666666666666666, 'A->A D': 0.6666666666666666, 'B->A B': 0.5, 'B->B C': 0.75, 'B->B D': 0.5, 'C->B C': 0.6, 'C->C D
': 0.7999999999999999, 'D->C D': 0.7999999999999999}
2. end ----------------------------------------------------

3. Support_Val
 {'B C D': 0.2857142857142857}
3. Confidence_Val
 {'B->B C D': 0.5, 'B C->B C D': 0.6666666666666666, 'B D->B C D': 1.0, 'C D->B C D': 0.5}
3. end ----------------------------------------------------

4. Support_Val
 {}
4. Confidence_Val
 None
4. end ----------There is no output from the 4 for loop------------------
wengshiuandembp:midterm project wengshiquan$
```

The 6 DataBase is not transactions DataBase. It is the data from the *Course Notes - Module 2B Association Rule Mining Part B*, which I used to test if my code is correct or not. The result appears my code is right.

```
wengshiuandembp:midterm project wengshiquan$ python mid.py
Enter a number from 1 to 5 to choose which database you want to see
1
Enter a minimum support value
0.4
Enter a minimum confidence value
0.4
1. Support_Val
 {'apple': 0.5, 'cake': 0.5, 'coke': 0.6, 'fork': 0.55, 'glass': 0.6, 'knife': 0.5, 'napkin': 0.55, 'plate': 0.55, 'spoon': 0.55,
'water': 0.45}
1 end ----------------------------------------------------

2. Support_Val
 {'cake coke': 0.4, 'coke fork': 0.4, 'coke glass': 0.4, 'coke napkin': 0.4, 'fork napkin': 0.4, 'glass plate': 0.4, 'knife spoon'
: 0.4}
2. Confidence_Val
 {'cake->cake coke': 0.8, 'coke->cake coke': 0.6666666666666667, 'coke->coke fork': 0.6666666666666667, 'fork->coke fork': 0.72727
27272727273, 'cake->coke glass': 0.8, 'coke->coke glass': 0.6666666666666667, 'glass->coke glass': 0.6666666666666667, 'cake->coke
 napkin': 0.8, 'coke->coke napkin': 0.6666666666666667, 'napkin->coke napkin': 0.7272727272727273, 'fork->fork napkin': 0.72727272
72727273, 'napkin->fork napkin': 0.7272727272727273, 'glass->glass plate': 0.6666666666666667, 'plate->glass plate': 0.72727272727
27273, 'knife->knife spoon': 0.8, 'spoon->knife spoon': 0.7272727272727273}
2. end ----------------------------------------------------

3. Support_Val
 {}
3. Confidence_Val
 None
3. end ----------There is no output from the 3 for loop------------------
wengshiuandembp:midterm project wengshiquan$
```

```
wengshiuandembp:midterm project wengshiquan$ python mid.py
Enter a number from 1 to 5 to choose which database you want to see
3
Enter a minimum support value
0.4
Enter a minimum confidence value
0.4
1. Support_Val
 {'TV': 0.45, 'coffee_table': 0.4, 'lantern': 0.5, 'laptop': 0.45, 'mug': 0.45, 'pillow': 0.45, 'sofa': 0.4, 'speaker': 0.4, 'tiss
ue': 0.4, 'video_player': 0.5}
1 end ---------------------------------------------------

2. Support_Val
 {}
2. Confidence_Val
 None
2. end ----------There is no output from the 2 for loop-----------------
wengshiuandembp:midterm project wengshiquan$
```

```
wengshiuandembp:midterm project wengshiquan$ python mid.py
Enter a number from 1 to 5 to choose which database you want to see
6
Enter a minimum support value
0.2
Enter a minimum confidence value
0.5
1. Support_Val
 {'A': 0.42857142857142855, 'B': 0.5714285714285714, 'C': 0.7142857142857143, 'D': 0.7142857142857143}
1 end ---------------------------------------------------

2. Support_Val
 {'A B': 0.2857142857142857, 'A D': 0.2857142857142857, 'B C': 0.42857142857142855, 'B D': 0.2857142857142857, 'C D': 0.5714285714
285714}
2. Confidence_Val
 {'A->A B': 0.6666666666666666, 'A->A D': 0.6666666666666666, 'B->A B': 0.5, 'B->B C': 0.75, 'B->B D': 0.5, 'C->B C': 0.6, 'C->C D
': 0.7999999999999999, 'D->C D': 0.7999999999999999}
2. end ---------------------------------------------------

3. Support_Val
 {'B C D': 0.2857142857142857}
3. Confidence_Val
 {'B->B C D': 0.5, 'B C->B C D': 0.6666666666666666, 'B D->B C D': 1.0, 'C D->B C D': 0.5}
3. end ---------------------------------------------------

4. Support_Val
 {}
4. Confidence_Val
 None
4. end ----------There is no output from the 4 for loop-----------------
wengshiuandembp:midterm project wengshiquan$
```

8. Full Source Code

```
import mysql.connector
import sys
from itertools import permutations

mydb = mysql.connector.connect(
  host="localhost",
  database="data_mining",
  user="root",
  passwd="andy123"
)
mycursor = mydb.cursor()

def Apriori(columns, select_db, rows):
        count = 0
        columns_name = []
        every_support = {}
        for i in columns:
                if(i[0]=='ID'):
                        continue
                columns_name.append(i[0])
        columns_name = sorted(columns_name)
        for i in columns_name:
                mycursor.execute("select count(ID) from "+select_db+" where "+i+" = 1")
                result = mycursor.fetchall()
                if float(result[0][0]/rows)>=support_val:
                        every_support[i] = float(result[0][0]/rows)
        all_support.update(every_support)
        return every_support, columns_name, count

def Apriori1(every_support, columns_name, count, select_db, rows):
        count+=1
        key = [i for i in every_support.keys()]
        every_support = {}
        a, b, c, d = "select count(ID) from ", " where ", " = 1 and ", " = 1"
        for i in range(len(key)):
                sql = ''
                x = key[i].split(" ")
                if len(x)==1:
                        sql+=x[0] + c
                else:
                        for k in range(len(x)):
                                sql += x[k] + c
                for j in range((columns_name.index(x[count-1])+1), len(columns_name)):
```

```
                        sql_last = a + select_db + b + sql + columns_name[j] + d
                        mycursor.execute(str(sql_last))
                        result = float(mycursor.fetchall()[0][0]/rows)
                        if result>=support_val:
                                every_support[key[i]+" "+columns_name[j]] = result
        all_support.update(every_support)
        return every_support, count

def Permutation(key):
        d, ans = [], []
        for i in key:
                d2 = []
                d2+=i.split(' ')
                d.append(d2)
        for i in d:
                for j in permutations(i):
                        ans.append(j)
        return ans


def Confidence(every_support, all_support, count):
        confidence_element = {}
        if not every_support:
                return
        key = [i for i in every_support.keys()]
        permutate_name = Permutation(key)
        arrow = len(permutate_name[0])-1
        name = []
        for j in permutate_name:
                b = []
                for k in range(arrow):
                        b+=[j[k]]
                        c = sorted(b)
                        name.append(str(' '.join(c)))
        name = sorted(set(name))
        if len(key[0])-arrow==2:
                for i in name:
                        for j in key:
                                if i in j:
                                        result = every_support[j]/all_support[i]
                                        if result>=confidence_val:
                                                confidence_element[i+"->"+j] = result
        else:
                for j in key:
                        for i in name:
                                amount = 0
```

```
                                        for k in i:
                                                if k in j:
                                                        amount+=1
                                                else:
                                                        break
                                        if amount == len(i):
                                                result = every_support[j]/all_support[i]
                                                if result>=confidence_val:
                                                        confidence_element[i+"->"+j] = result
        return confidence_element

def setSupport():
        n = input("Enter a minimum support value \n")
        return n

def setConfidence():
        n = input("Enter a minimum confidence value\n")
        return n

def setInput():
        n = input("Enter a number from 1 to 5 to choose which database you want to see\n")
        return n

def db(x):
        return{1:'transactions1', 2:'transactions2', 3:'transactions3', 4:'transactions4', 5:'transactions5', 6:'test'}[x]

n = setInput()
support_val = float(setSupport())
confidence_val = float(setConfidence())
select_db = db(int(n))

mycursor.execute("select count(ID) from "+select_db)
rows = int(mycursor.fetchall()[0][0])

mycursor.execute("SHOW columns FROM "+select_db)
columns = mycursor.fetchall()
all_support = {}

every_support, columns_name, count = Apriori(columns, select_db, rows)
print("1. Support_Val\n", every_support)
if not every_support:
        sys.exit("1 end ----------Your support value is too high so no output from the first for loop----------")
else:
        print("1 end ----------\n")

for _ in range(9):
```

```
        every_support, count = Apriori1(every_support, columns_name, count, select_db, rows)
        confidence_element = Confidence(every_support, all_support, count)
        print(str(count+1)+". Support_Val\n", every_support)
        print(str(count+1)+". Confidence_Val\n", confidence_element)
        if not every_support:
                info = (str(count+1)+". end ----------There is no output from the "+str(count+1)+" for
loop----------")
                sys.exit(info)
        else:
                print(str(count+1)+". end ————————————————————————————————————\n")
```

Thank You