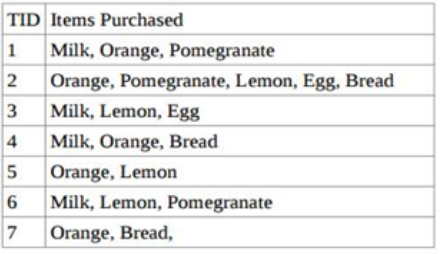
**EX NO: 3 DATA MINING LABORATORY**

**DATE: 18/09/2020 EXERCISE 3 - POSTLAB**

**Aim:**

To learn about the concepts of association mining and to implement it using Weka and Java.

**Implementation:**



**1. Let min sup =0.2 and min conf =0.8. Using weka, obtain the frequent item-sets and**

**association rules from the above transaction database using Apriori algorithm**

**satisfying the given confidence measure.**

**Step 1:** Generate the arff file according to the given transactions.

@relation supermarket

@attribute 'milk' {t}

@attribute 'orange' {t}

@attribute 'pomegranate' {t}

@attribute 'lemon' {t}

@attribute 'egg' {t}

@attribute 'bread' {t}

@data

t,t,t,?,?,?

?,t,t,t,t,t

t,?,?,t,t,?

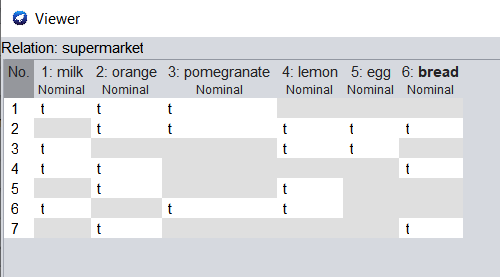
t,t,?,?,?,t

?,t,?,t,?,?

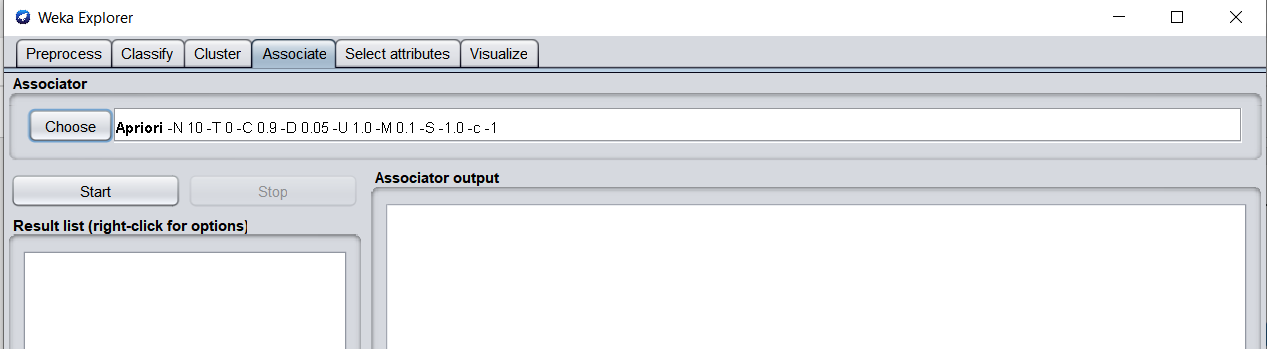
t,?,t,t,?,?

?,t,?,?,?,t

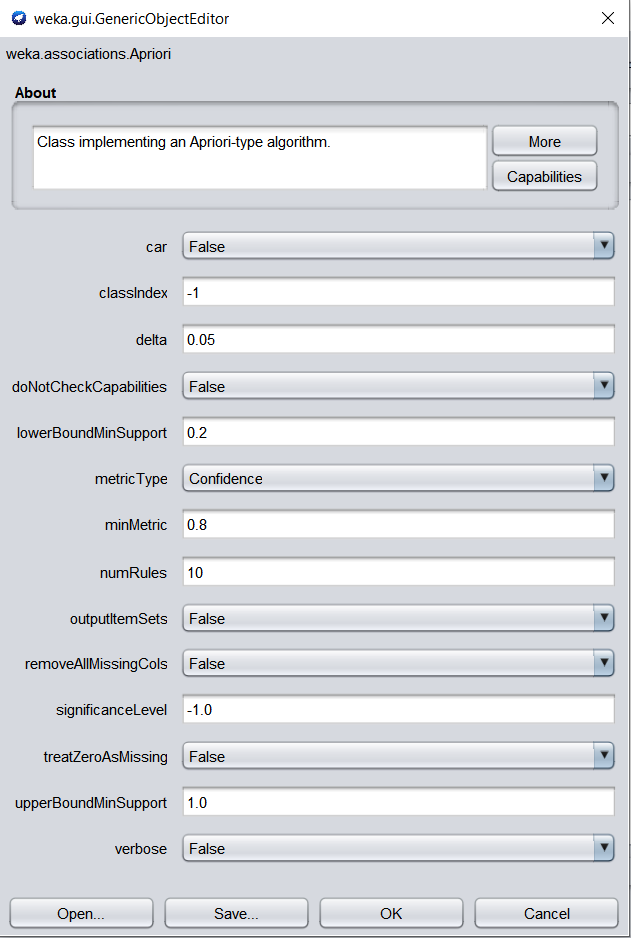
**Step 2:** Open the arff file in Weka



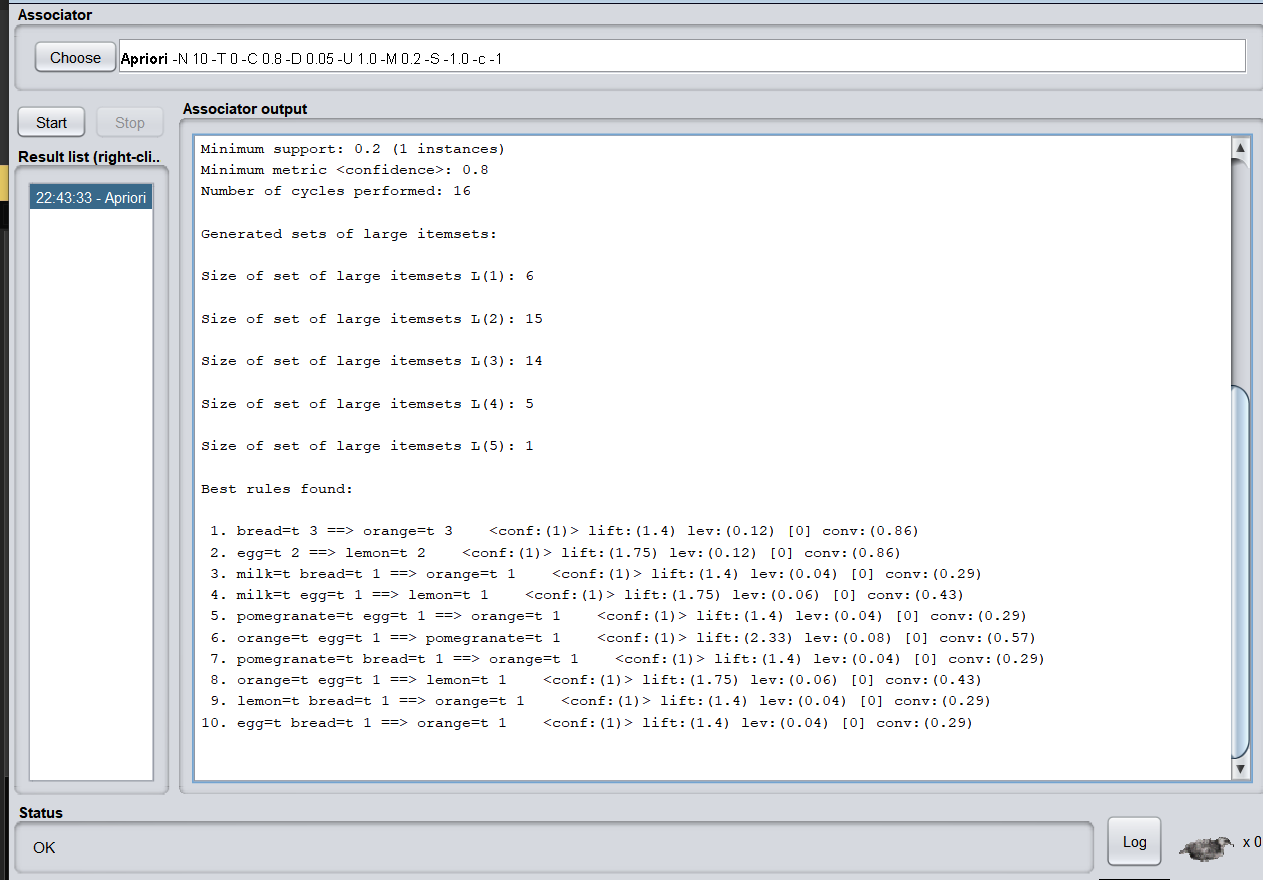
**Step 3:** Under the associations tab, select apriori

****

**Step 4:** The min support and min conf are modified according to the given values.

****

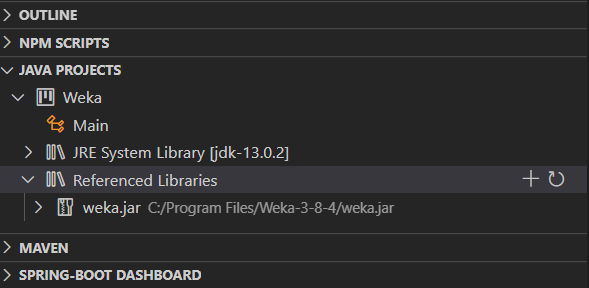
**Step 5:** Click on Start to view the output of Apriori algorithm



**2. In the java IDE Eclipse along with WEKA generate the association rules for the**

**above transaction database.**

**Step 1:** The Weka jar is included in the current project as a Referenced Libraray.

****

**Step 2:** The code for Apriori is written and compiled

import weka.associations.\*;

import weka.core.Instances;

import java.io.\*;

public class Main {

    public static void main(String args[]) {

        try {

            FileReader input = new FileReader("file.arff");

            BufferedReader bf = new BufferedReader(input);

            Instances data = new Instances(bf);

            Apriori model = new Apriori();

            model.setMinMetric(0.8);

            model.setLowerBoundMinSupport(0.2);

            model.buildAssociations(data);

            System.out.println(model);

        }

        catch(Exception e) {

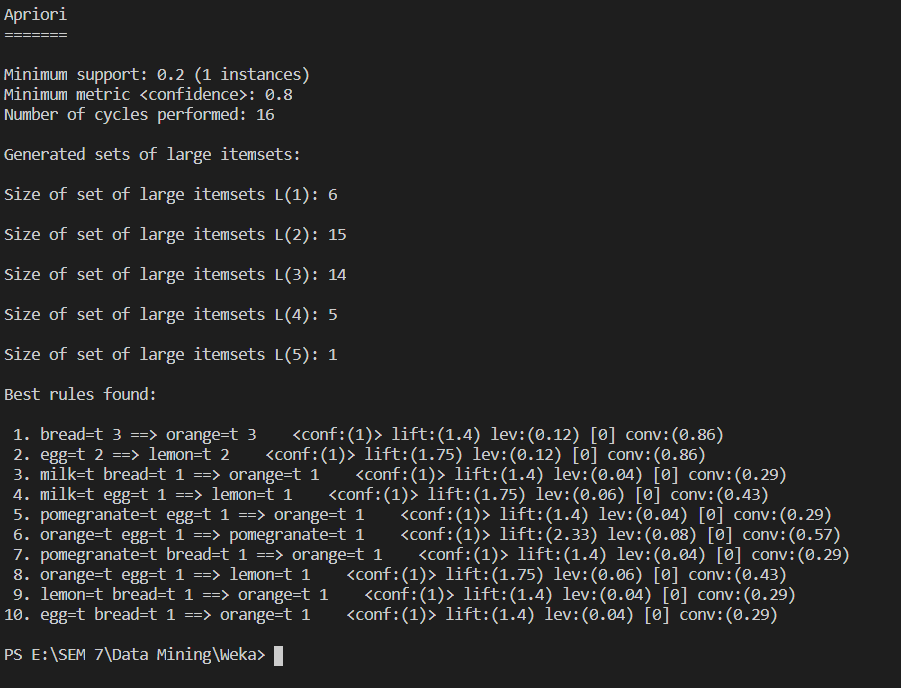
            System.out.println("Some error occurred!");

        }

    }

}

**Step 3:** The code is executed and the output is obtained.



**Interpretation:**

The output is same as that of the output obtained in Weka.

**3. Implement the Apriori algorithm to generate the two frequent item-set.**

The Apriori algorithm is implemented in Python.

**Code:**

def search(list1,list2):

    for i in list2:

        if i not in list1:

            return False

    return True

n = int(input())

m = float(input())

m = int(m\*n)

items = list()

step0 = dict()

step1 = dict()

step2 = dict()

step3 = dict()

step4 = dict()

for i in range(n):

    line = list(map(str,input().split()))

    step0[line[0]] = line[1:]

for i in step0:

    for j in step0[i]:

        items.append(j)

for i in items:

    if i in step1: step1[i] += 1

    else: step1[i] = 1

print('The min support is ',m)

print('Step 1: Finding occurrences of 1 items\n')

for i in step1:

    print(i,step1[i])

print('\nStep 2: Removing items less than min support\n')

for i in step1:

    if step1[i] >= m:

        step2[i] = step1[i]

for i in step2:

    print(i,step2[i])

items.clear()

for i in step2:

    items.append(i)

for i in range(len(items)):

    for j in range(i+1,len(items)):

        p = (items[i],items[j])

        step3[p] = 0

print('\nStep 3: Finding occurrences of 2 items\n')

for i in step3:

    for j in step0:

        if search(step0[j],list(i)):

            if i in step3:

                step3[i] += 1

            else:

                step3[i] = 1

for i in step3:

    print(i,step3[i])

for i in step3:

    if step3[i]>=m:

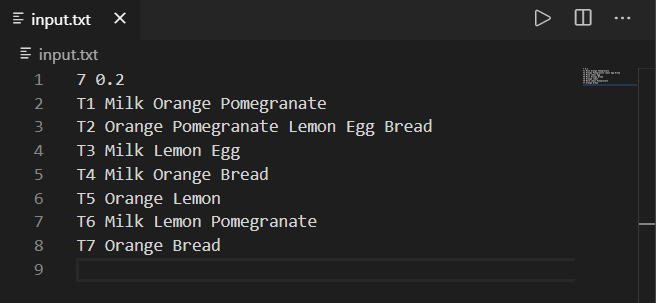
        step4[i] = step3[i]

print('\nStep 4: Removing items less than min support\n')

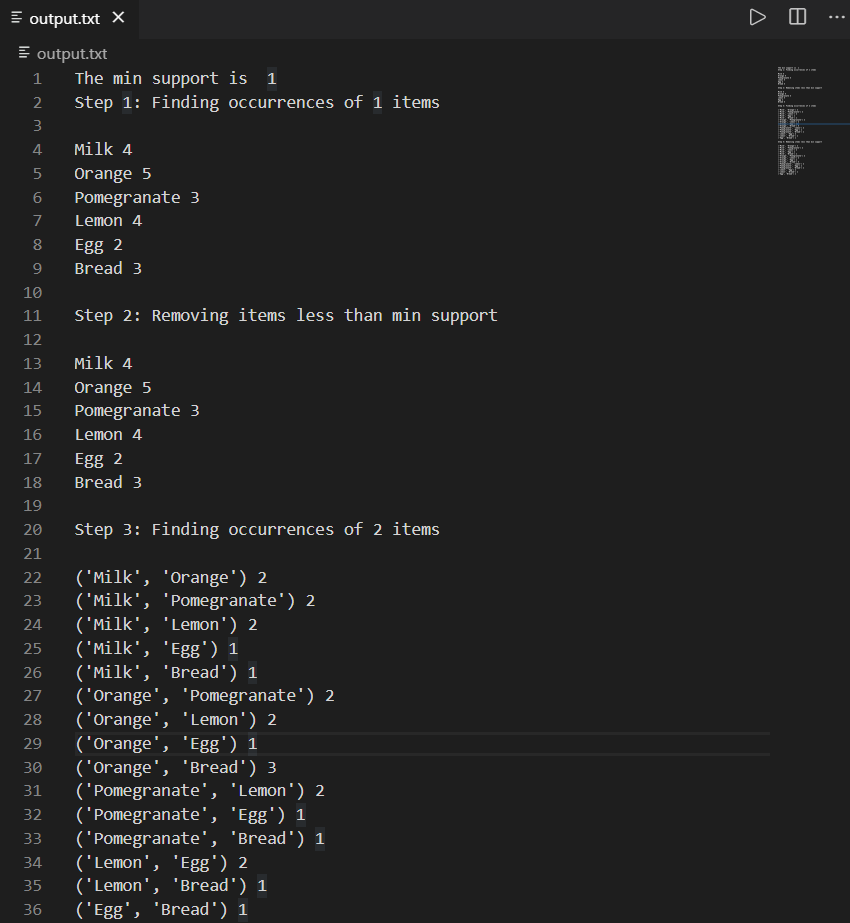
for i in step4:

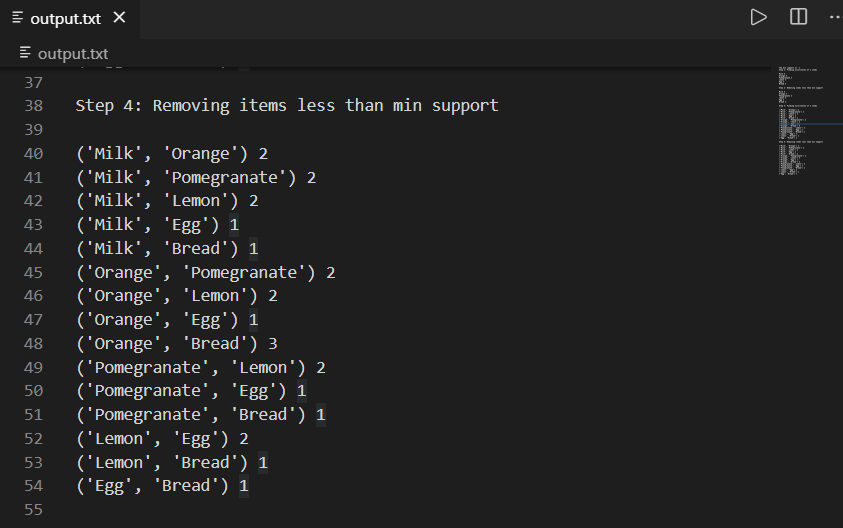
    print(i,step4[i])

**Input:**



**Output:**





**Result:**

The association mining concepts have been learnt and it is implemented using Weka, Java and Python successfully.