**Experiment- 12**

**Objective:** Implement apriori algorithm for association rules of mining.

**Theory:** The steps for Apriori algorithm are:

• In the first iteration of the algorithm, each item is a member of the set of candidate1-itemsets, *C*1. The algorithm simply scans all the transactions to count the number of occurrences of each item. Suppose that the minimum support count required is k, that is, *min sup* = k.

• The set of frequent 1-itemsets, *L*1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. Here, *Li* shows the database with *i* elements in each set.

• To discover the set of frequent 2-itemsets, *L*2, the algorithm uses the join *L*1 \* *L*1 to generate a candidate set of 2-itemsets*.*

• The set of frequent 2-itemsets, *L*2, is then determined, consisting of those candidate2-itemsets in *C*2 having minimum support.

• The generation of the set of the candidate3-itemsets, *C*3, is then calculated. It is based on the Apriori property that all subsets of a frequent itemset must also be frequent.

• Further, higher order candidate sets are generated and finally determine the maximum of such set that also satisfies the minimum support.

**Program:**

| package lab14;  public class AssociationRule {  public static void main(String[] args) {  int totalAttrs = 3, numberTransac = 4, c1 = 0, c2 = 0;  double minsup = 0.5, sup = 0.0;  // a → b Shows the association rule between two sets.  int a[] = {1, 1, 0};  int b[] = {0, 0, 0};  int aorb[] = new int[totalAttrs];  49  int aInt = 0;  int aorbInt = 0;  int dbInt[] = new int[numberTransac];  int db[][] = {{1, 0, 1},  {0, 0, 0},  {0, 0, 0},  {1, 1, 1}};  // show data set  System.out.println("The database is: ");  for(int i = 0; i < numberTransac; i++) {  for(int j = 0; j < totalAttrs; j++) {  System.out.print(db[i][j] + " ");  }  System.out.println();  }  // show item set.  System.out.println("The relation is: ");  for(int i = 0; i < totalAttrs; i++)  System.out.print(a[i] + " ");  System.out.print("-> ");  for(int i = 0; i < totalAttrs; i++)  System.out.print(b[i] + " ");  for(int i = 0; i < totalAttrs; i++) {  aorb[i] = a[i] | b[i];  }  // convert the values to integers.  for(int i = totalAttrs - 1; i >= 0; i--)  aInt = aInt + a[i] \* (int) Math.pow(2, totalAttrs - 1 - i);  for(int i = totalAttrs - 1; i >= 0; i--)  aorbInt = aorbInt + aorb[i] \* (int) Math.pow(2, totalAttrs - 1  - i);  for(int i = 0; i < numberTransac; i++)  for(int j = totalAttrs - 1; j >= 0; j--)  dbInt[i] = dbInt[i] + db[i][j] \* (int) Math.pow(2,  50  totalAttrs - 1 - j);  // 101 & 111 => 101 i.e a.b = a means the attributes of first are  present in second.  for(int i = 0; i < numberTransac; i++)  if((aInt & dbInt[i]) == aInt)  c1 = c1 + 1;  // calculate confidence.  for(int i = 0; i < numberTransac; i++)  if((aorbInt & dbInt[i]) == aorbInt)  c2 = c2 + 1;  sup = c2 / c1;  System.out.println();  // print output.  if(sup > minsup)  System.out.println("It will be included in the itemset with  confidence: " + sup);  else  System.out.println("It won't be included in itemset with  confidence: " + sup);  }  } |
| --- |

**Output**

The database is:

1 0 1

0 0 0

0 0 0

1 1 1

The relation is:

1 1 0 → 0 0 0

It will be included in the itemset with confidence: 1.0

**Conclusion:**

The program shows whether the a → b present or not using the apriori algorithm and the association rules. It uses the formula:

confidence(A → B) =

**Experiment- 24**

**Objective:** Define a LISP function to compute sum of squares.

**Theory:** SUM OF SQUARE

Sum of square means by adding the squares of two numbers.

Example:

Sum of Square of x ,y= ( x \* x ) + ( y \* y )

Now in prefix expression it should be written as,

( + ( \* x x ) ( \* y y ) )

**Code:**

| >>(defun sumsqr(x y)  (+(\* x x)(\* y y)))  SUMSQR  >>(sumsqr 2 3); |
| --- |

**Output:**

13

**Conclusion:**

We defined a LISP function to compute sum of square and test it on various values.