1. CHAPTER-1

INTRODUCTION

1.1 HISTORY

Association and sequential rules have been widely used for many years in solving various practical issues, such as consumer behavior in marketing research, recommendation systems in e-commerce and protein sequences in biometrics . The main difference between mining association rules and sequential rules is that association rules only concern the co-occurrence relationship between items and ignore the sequential pattern between items .In general, the problems of association and sequential rules can be categorized as unsupervised or supervised, depending on whether a decision variable exists. In this project, we focus on unsupervised association and sequential rules. Since they have no decision variables, the supervised problems can be considered to be classification problems and a number of algorithms can be used to solve them. Sequential pattern mining is a data mining technique used to identify patterns of ordered events. With increasing applications on item set data mining has been applied to the distributed environment of large amount of data.

Since data are located in different places, an efficient algorithm is needed to mine the large amount of data so in this project, we use sequential pattern mining I distributed database for mining the association rules by finding the local frequent item set and then generate global frequent item set.

Sequential pattern mining is a data mining technique used to identify patterns of ordered events within a database. First introduced in 1995 by Rakesh Agrawal of IBM’s Almaden Research its original applications were in the retail industry where it can be used to predict that within a certain time period after purchasing a certain book, a customer is likely to purchase its sequel.

Data mining concepts and techniques are applicable for all the application areas. Its application is also expanded from market basket analysis to varieties of areas like customer segmentation, medicine, electronic commerce, classification, clustering. To improve the efficiency of the mining algorithms parallelism is viewed upon as an effective concept. Even though, there are some applications areas where we are unable to exploit parallelism successfully, data intensive applications can be easily parallelized when compared to computational intensive applications. However, most of the algorithms suffer from its time-consuming process. Parallel and Distributed algorithms resolve this proble--m by applying the parallelization techniques.

**1.2 SYSTEM REQUIREMENT**

* **OPERATING SYSTEMS:**
* Windows
* Linux
* Ubuntu
* **S/W APPLICATIONS:**
* Dev C++/ CodeBlocks
* MS Powerpoint
* MS Word
* Gantt Project
* SQL Server
* Database
* Ms Excel

**1.3 Main Objective**

* The main objective of the project is to propose an Effective and efficient Apriori-based algorithm to find frequent item sets in the distributed datasets. Most enterprises collect huge amounts of business data from daily transactions and store them in distributed datasets specially, for security issues and communication overhead, those distributed datasets are usually not allowed to be transmitted or joined together, therefore, in this study we are focusing on Apriori-based algorithms and discovering frequent item sets on extremely large and distributed datasets over different geographic locations also in and will present a well-adapted distributed approach for this purpose, based on both analytical and experimental approaches.
* With the help of local and global support count over the distributed datasets to find Frequent Item sets for overall dataset as a one and Frequent item sets for the individual datasets with respect to time e.g. Quarter of the year.

**1.4 Pert Chart Legend**

**CHAPTER-2**

SYSTEM ANALYSIS

2.1 **EXISTING SYSTEM**

* The most famous is the Apriori algorithm which has been brought in 1993 by Agrawal which uses association rule mining [1].

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

1. Minimum support (threshold) is applied to find all frequent item-sets in a database.

2. These frequent item-sets and the minimum confidence constraints are used to form rules.

Advantage of this algorithm, it is easy to find frequent item sets if database is small but it has two deadly bottlenecks. First, It needs great I/O load when frequently scans database and Second, It may produce overfull candidates of frequent item-sets.

R. Agrawal etal presents an efficient algorithm that generates all significant association rules between items in the database. The algorithm incorporates buffer management and novel estimation and pruning techniques. Also represent results of applying algorithm to sales data obtained from a large retailing company, which shows the effectiveness of the algorithm.

* T. Tassa proposed [1] a protocol for secure mining of association rules in horizontally distributed databases. The main ingredients in our protocol are two novel secure multi-party algorithms-one that computes the union of private subsets that each of the interacting players hold, and another that tests the inclusion of an element held by one player in a subset held by another. This protocol offers enhanced privacy with respect to the protocol. In addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.
* Liu Han-bing et al [4]. proposes a new efficient incremental updating algorithm for mining association rules (ATLUP).It resolves the problem of updating the association rules when increasing transaction database without changing the minimum support and minimum confidence. Main features of their algorithm were frequent item sets of new transaction database were produced by AprioriTidList algorithm, and candidate item sets were classified and prunedin effective ways . It uses intersection operation of item's tidlist to replace scanning database, therefore, the time of scanning database and the number of candidate item sets are reduced. Hence efficiency of updating association rules is improved.
* S.Einakian [5] discusses parallel Data Mining architecture for large volume of data which eventually scanning billions of rows of data per record. Compare the different parallel algorithms for Association Rule Mining and discuss the advantages and disadvantages of each method. Also compare the computational time of serial and parallel algorithms for Association Rule Mining.

**2.2 MOTIVATION**

Data mining is also known as Knowledge Discovery in Database (KDD). Data mining is a relative technology that’s not fully grown. Despite this, there are a number of industries that are already using it on a regular basis. The performance benefit comes only when every partition of data is run by process on different processors. Also it is important that the data is partitioned equally so that the efficiency can be maximized.

This technology is popular with many businesses, because it allows them to learn more about their customers and make smart marketing decisions by using Association rule mining.

Rapid advancement of IT technology has resulted accumulation of tremendous amount of data for organization and therefore extracting needed information from huge amount of data has been a big challenge.

* 1. **PROPOSED SYSTEM**
* In this venture, we propose an incorporated structure to join brought together calculations, for the:-

1. Apriori
2. Frequent Pattern Growth (FP-Growth)
3. GENETIC calculations

* To mine choice guidelines in a conveyed situation. Furthermore, our strategy discovers some noteworthy tenets that different calculations can't. The tests additionally show that the proposed strategy is appropriate to finding the affiliation and consecutive principles in a conveyed situation.
* In large Distributed Dataset where association rules are implied and they concern the co-occurrence relationship between items and ignore the sequential pattern between items and the end-user is not able to mine the sequential patterns Most ventures gather enormous measures of business information from everyday exchanges and store them in appropriated datasets exceptionally.
* Øn this process we can also use Genetic Algorithm because they are quick to implement and adaptive themselves to the searching the pattern mining and We can enhance the efficiency considerably a greater amount of algorithm under consideration.
  1. **MODULES**

**2.4.1 MODULE-1**

In this module we apply the Apriori Algorithm over the dataset of the retail store to create the most successive thing set up to two iteration. CSV document is stacked into the framework over which Apriori calculation is executed.

**2.4.2 MODULE-2**

In this module we have global frequent item set for various store of Retail Store and will generate the frequent item set for the store quarter wise and which will serve as the purpose of frequent the items set for the particular quarter.

**2.4.3 MODULE 3**

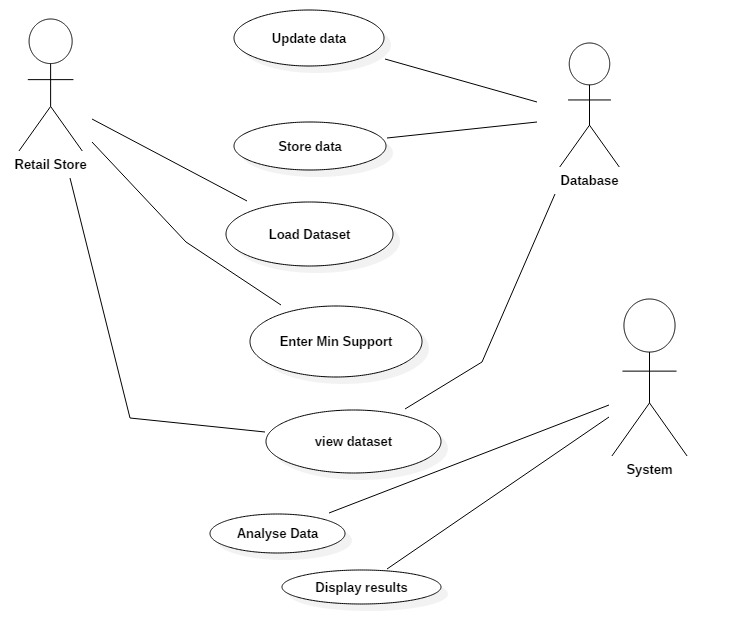
In this module we apply Genetic Algorithm since they rush to actualize and versatile themselves to the looking through the example mining and this module will likewise contrast the Apriori calculation and the Genetic Algorithm and will tell which calculation performs better.

`

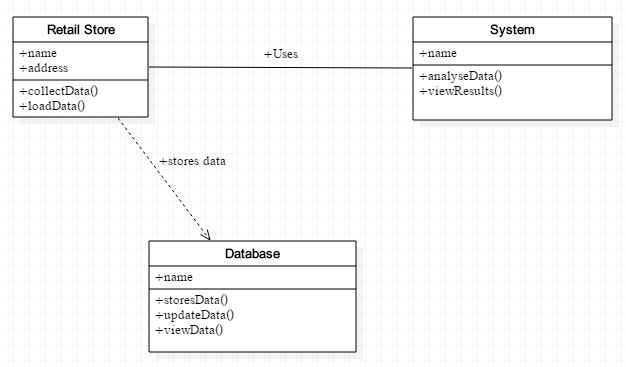
CHAPTER-3

DESIGN MODELLING

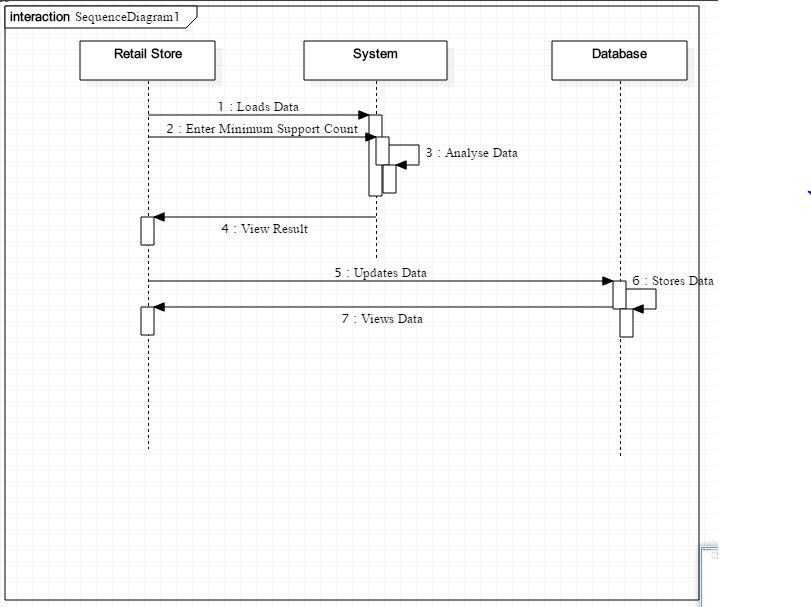
**3.1 Use Case Diagram**

****

**3.2 Class Diagram**



**3.3 Sequence Diagram**



3.4 DFD Level 0

Show Data

Updates Data

Database

Display Results

Upload File

Retail Store

System

3.4 Flowchart

**3.5 DFD LEVEL 1**

Display Results

Show Data

Updates Data

Database

Request Data

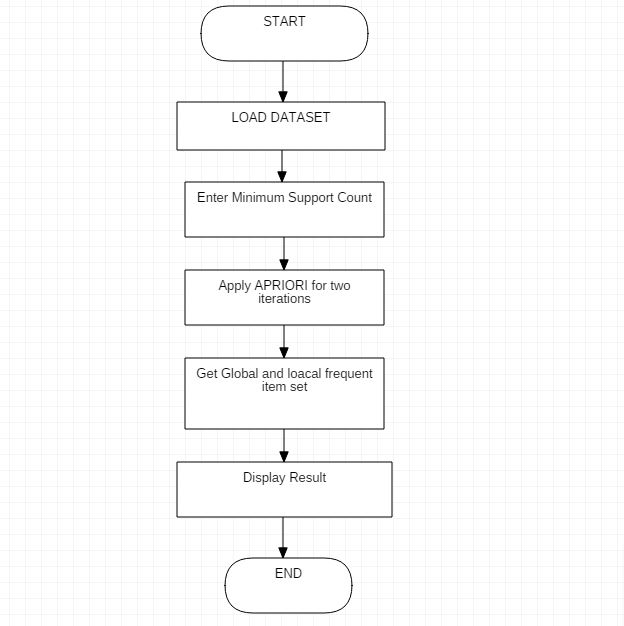
Get details

Upload File

System

Retail Store

3.6 Flowchart



CHAPTER-4

CODING AND IMPLEMENTATION

**Algorithm:**

The Algorithm can be viewed as a modified version of the Sequential Apriori Algorithm as this method can be used to find frequent item sets in the distributed environment.

**Input:**

* Data from distributed datasets
* Local and Global Support count

**Output:**

* Local and Global frequent item set
* Frequent item set with respect to time

**Method:**

1. Read the Data sets.
2. Using Modify Apriori for Distributed Dataset find most frequent item set for all datasets individually and global frequent item set as a whole database.
3. Now Find the Frequent item sets with respect to time (Quarter of the year or season).

**Genetic Algorithm:**

**Pseudo Code**

GA()

initialize population

find fitness of population

while (termination criteria is reached) do

parent selection

crossover with probability pc

mutation with probability pm

decode and fitness calculation

survivor selection

find best

return best

SOURCE CODE FOR PROJECT:

#include<iostream>

#include<iterator>

#include<fstream>

#include<sstream>

#include<vector>

#include<string>

using namespace std;

class CSVRow{

public:

std::string const& operator[](std::size\_t index) const

{

return m\_data[index];

}

std::size\_t size() const

{

return m\_data.size();

}

void readNextRow(std::istream& str)

{

std::string line;

std::getline(str, line);

std::stringstream lineStream(line);

std::string cell;

m\_data.clear();

while(std::getline(lineStream, cell, ','))

{

m\_data.push\_back(cell);

}

if(!lineStream && cell.empty())

{

m\_data.push\_back("");

}

}

private:

std::vector<std::string> m\_data;

};

std::istream& operator>>(std::istream& str, CSVRow& data)

{

data.readNextRow(str);

return str;

}

int main(){

int k,a[30][10],i=0,j=0;//size according to csv data first is rows second column

std::ifstream file("data1.csv");

CSVRow row;

for (i=0;i<30;i++){//no of rows in csv

for(j=0;j<10;j++){// no of columns in csv

a[i][j]=0;

}

}

i=0;

while(file>>row)

{

j=0;

std::string s = row[5];

cout<<s<<endl;

std::string delimiter = " ";

size\_t pos = 0;

std::string token;

while ((pos = s.find(delimiter)) != std::string::npos) {

token = s.substr(0, pos);

const char\* conv\_my\_str = token.c\_str();

sscanf(conv\_my\_str, "%d", &a[i][j]);

j++;

//std::cout << token << std::endl;

s.erase(0, pos + delimiter.length());

}

//std::cout << s << std::endl;

const char\* conv\_my\_str = s.c\_str();

sscanf(conv\_my\_str, "%d", &a[i][j]);

j++;

i++;

//i++;

//std::cout<<row[10];

//std::cout << "4th Element ("<< row[5] <<")\n";

//std::cout << "4th Element ("<< a[i] <<")\n";

}

int t1,l,m,f,f1,f2,f3;

//Initial item-purchase

//int a[5][5];

//Defining minimum level for acceptence

int min;

cout<<"\n Enter minimum acceptance level";

cin>>min;

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

cout<<a[i][j]<<" ";

}

cout<<endl;

}

//Printing initial input

//cout<<"\nInitial Input:\n";

//cout<<"\nTrasaction\tItems\n";

for(i=0;i<14;i++)

{

cout<<i+1<<":\t";

for(j=0;j<10;j++)

{

cout<<a[i][j]<<"\t";

}

cout<<"\n";

}

cout<<"\nAssume minimum support: "<<min;

//First pass

int l1[14];

for(i=0;i<10;i++)

{

t1=0;

for(j=0;j<14;j++){

for(k=0;k<10;k++){

if(a[j][k]==i+1){

t1++;

}

}

}

l1[i]=t1;

}

//Printing first pass

cout<<"\n\nGenerating C1 from data\n";

for(i=0;i<10;i++)

{

cout<<i+1<<": "<<l1[i]<<"\n";

}

//Second pass

//Counting number of possibilities for pass2

int p2pcount=0;

int p2items[5];

int p2pos=0;

for(i=0;i<5;i++)

{

if(l1[i]>=min)

{

p2pcount++;

p2items[p2pos]=i;

p2pos++;

}

}

std::ifstream file1("product.csv");

std::ifstream file2("product.csv");

CSVRow row1,row2;

int x;

int y;

cout<<"\nGenerating L1 From C1\n";

for(i=0;i<p2pos;i++)

{

//cout<<p2items[i]+1<<"\t"<<l1[p2items[i]]<<"\n";

while(file2>>row2)

{

std::string s = row2[0];

const char\* conv\_my\_str = s.c\_str();

sscanf(conv\_my\_str, "%d", &y);

//std::cout<<p2items[i]+1<<"\t"<<x<<"\t"<<row1[0]<<"\t"<<row1[1]<<"\n";

if(y==p2items[i]+1){

std::cout<<row2[1]<<"\t\t";

break;

}

}

while(file1>>row1)

{

//cout<<"running2";

std::string s = row1[0];

const char\* conv\_my\_str = s.c\_str();

sscanf(conv\_my\_str, "%d", &x);

//std::cout<<p2items[i]+1<<"\t"<<x<<"\t"<<row1[0]<<"\t"<<row1[1]<<"\n";

if(x==l1[p2items[i]]){

std::cout<<row1[1]<<"\n";

break;

}

}

}

//Printing selected items for second pass

/\*for(i=0;i<p2pos;i++)

{

cout<<p2items[i]+1<<"\t"<<l1[p2items[i]]<<"\n";

}\*/

//Joining items

int l2[5][3];

int l2t1; //will hold first item for join

int l2t2; //will hold second item for join

int l2pos1=0; //position pointer in l2 array

int l2ocount=0; //product join occruance counter

int l2jcount=0; //join counter

for(i=0;i<p2pcount;i++)

{

for(j=i+1;j<p2pcount;j++)

{

l2t1=p2items[i]+1;

l2t2=p2items[j]+1;

if(l2t1==l2t2)

{

//it is self join

continue;

}

//join the elements

l2[l2pos1][0]=l2t1;

l2[l2pos1][1]=l2t2;

l2jcount++;

//count occurances

l2ocount=0; //reset counter

for(k=0;k<5;k++)

{

f1=f2=0; //resetting flag

//scan a purcahse

for(l=0;l<5;l++)

{

if(l2t1==a[k][l])

{

//one of the element found

f1=1;

}

if(l2t2==a[k][l])

{

//second elements also found

f2=1;

}

}

//one purchase scanned

if(f1==1&&f2==1) //both items are present in

int purchase;

{

l2ocount++;

}

}

//assign count

l2[l2pos1][2]=l2ocount;

l2pos1++;

}

}

//Printing second pass

cout<<"\n\nGenerating L2\n";

std::ifstream file3("product.csv");

CSVRow row3;

int z;

for(i=0;i<l2jcount;i++)

{

for(j=0;j<3;j++)

{

cout<<l2[i][j]<<"\t";

/\*while(file3>>row3)

{

//cout<<"running2";

std::string s = row3[0];

const char\* conv\_my\_str = s.c\_str();

sscanf(conv\_my\_str, "%d", &z);

//std::cout<<p2items[i]+1<<"\t"<<x<<"\t"<<row1[0]<<"\t"<<row1[1]<<"\n";

if(z==l2[i][j]){

std::cout<<row3[1]<<"\t";

break;

}

}\*/

}

cout<<"\n";

}

//Third pass

int p3pcount=0;

int p3items[5]={-1,-1,-1,-1,-1};

int p3pos=0;

for(i=0;i<5;i++)

{

if(l2[i][2]>=min)

{

f=0;

for(j=0;j<5;j++)

{

if(p3items[j]==l2[i][0])

{

f=1;

}

}

if(f!=1)

{

p3items[p3pos]=l2[i][0];

p3pos++;

p3pcount++;

}

f=0;

for(j=0;j<5;j++)

{

if(p3items[j]==l2[i][1])

{

f=1;

}

}

if(f!=1)

{

p3items[p3pos]=l2[i][1];

p3pos++;

p3pcount++;

}

}

}

//Joining

int l3[5][4];

int l3ocount=0; //occurance counter

int l3jcount=0; //join counter

for(i=0;i<p3pcount;i++)

{

for(j=i+1;j<p3pcount;j++)

{

for(k=j+1;k<p3pcount;k++)

{

l3[i][0]=p3items[i];

l3[i][1]=p3items[j];

l3[i][2]=p3items[k];

l3jcount++;

//count occurances

l3ocount=0; //reset counter

for(k=0;k<5;k++)

{

f1=f2=f3=0; //resetting flag

//scan a purcahse

for(l=0;l<5;l++)

{

if(l3[i][0]==a[k][l])

{

//one of the element found

f1=1;

}

if(l3[i][1]==a[k][l])

{

//second elements also found

f2=1;

}

if(l3[i][2]==a[k][l])

{

//third element also found

f3=1;

}

}

//one purchase scanned

if(f1==1&&f2==1&&f3==1) //all items are present

int purchase;

{

l3ocount++;

}

}

//assign count

l3[i][3]=l3ocount;

}

}

}

//Printing second pass

cout<<"\n\nGenerating L3\n";

for(i=0;i<l3jcount;i++)

{

for(j=0;j<4;j++)

{

cout<<l3[i][j]<<"\t";

}

cout<<"\n";

}

//Ending

return 0;

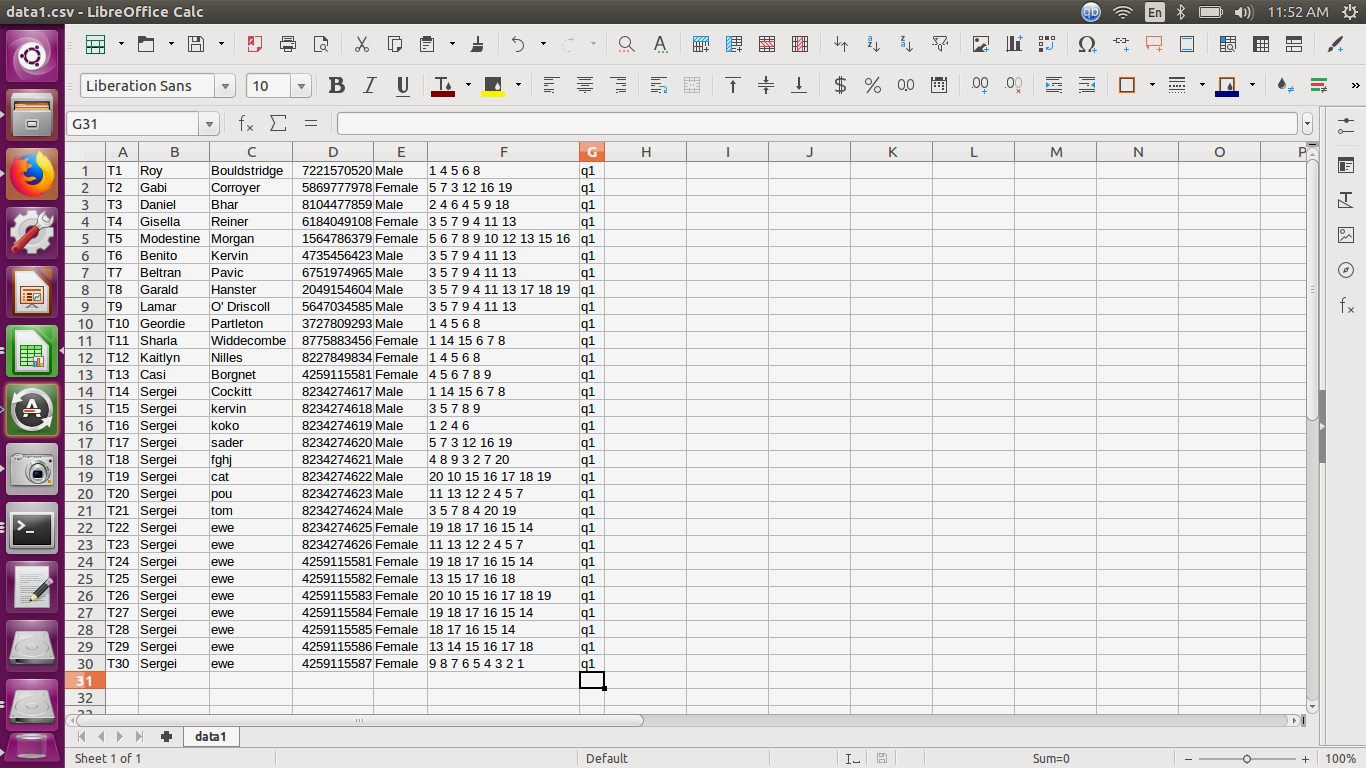
}

//}

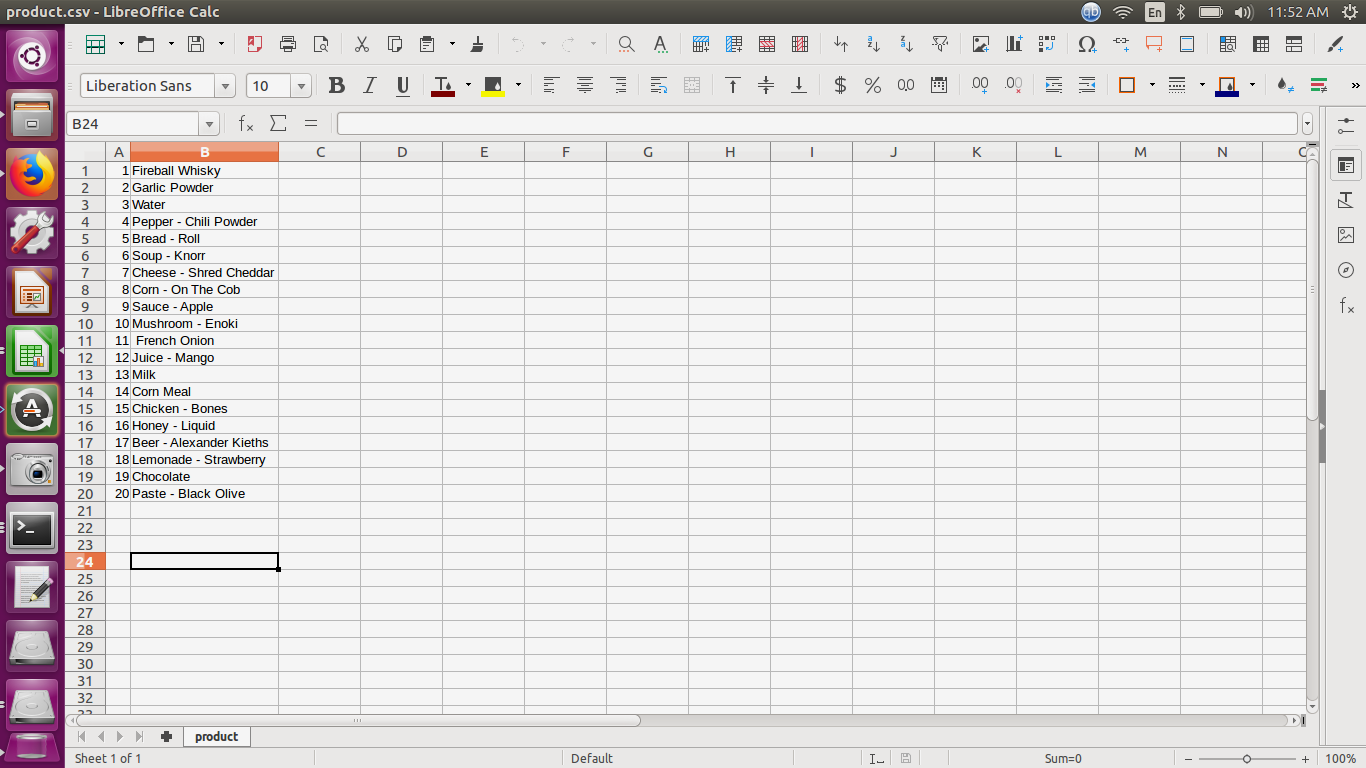
CHAPTER -5

OUTPUT SCREEN

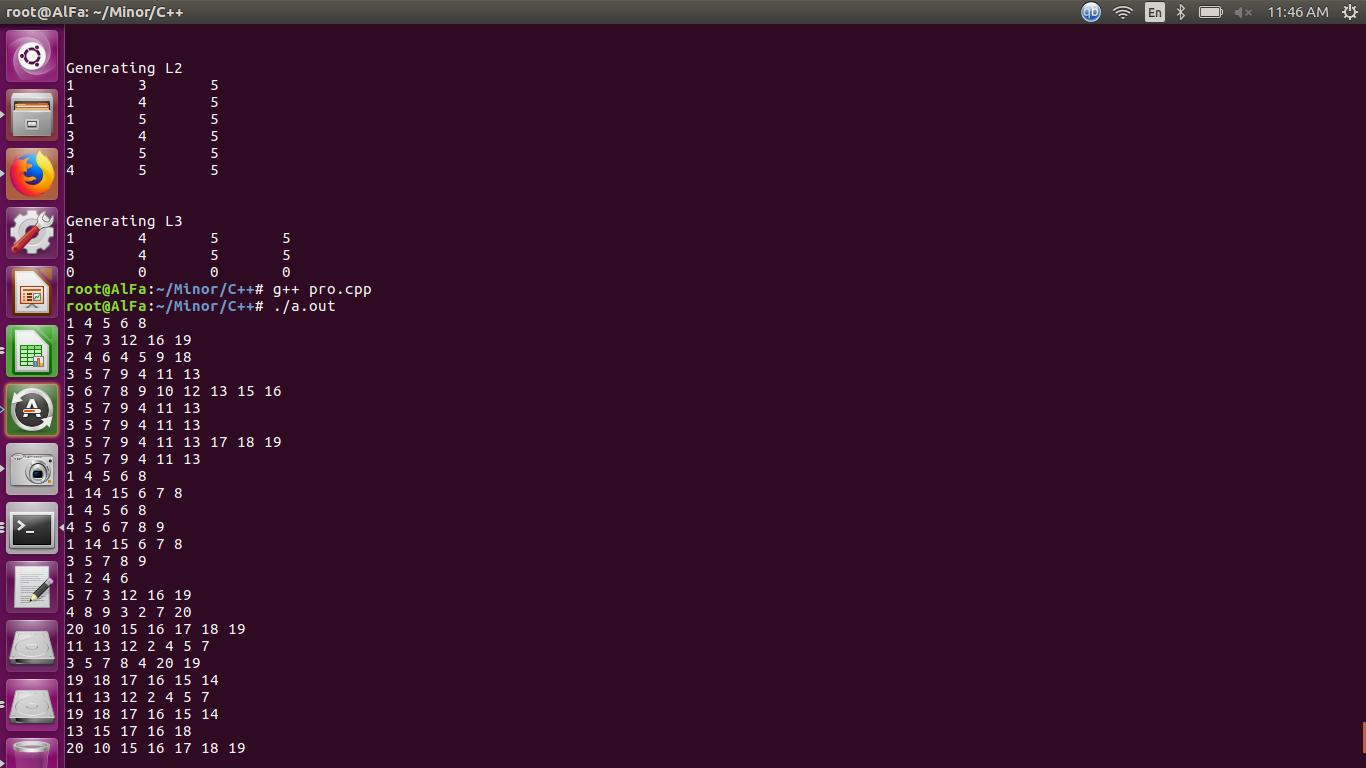
**Figure 5.1 : Transactional Database**



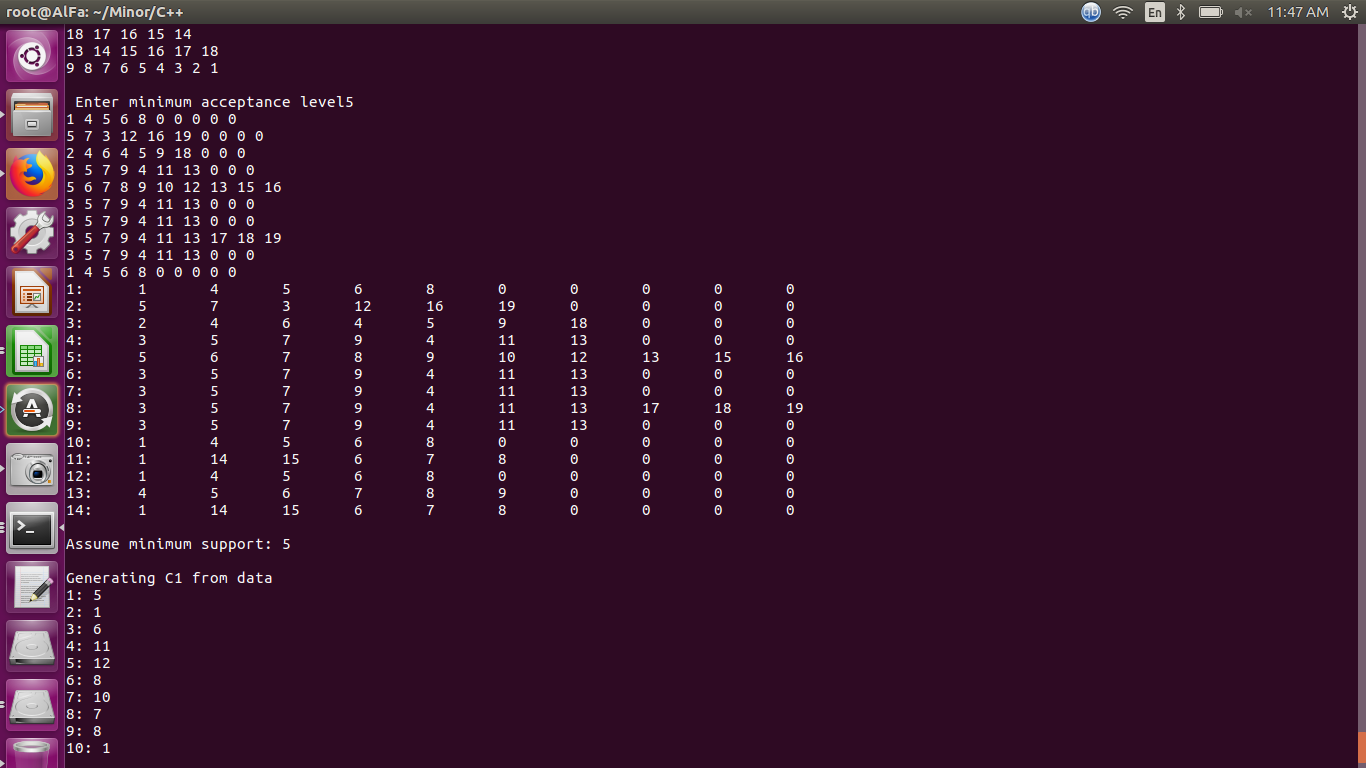
**Figure 5.2 : Product Dataset**



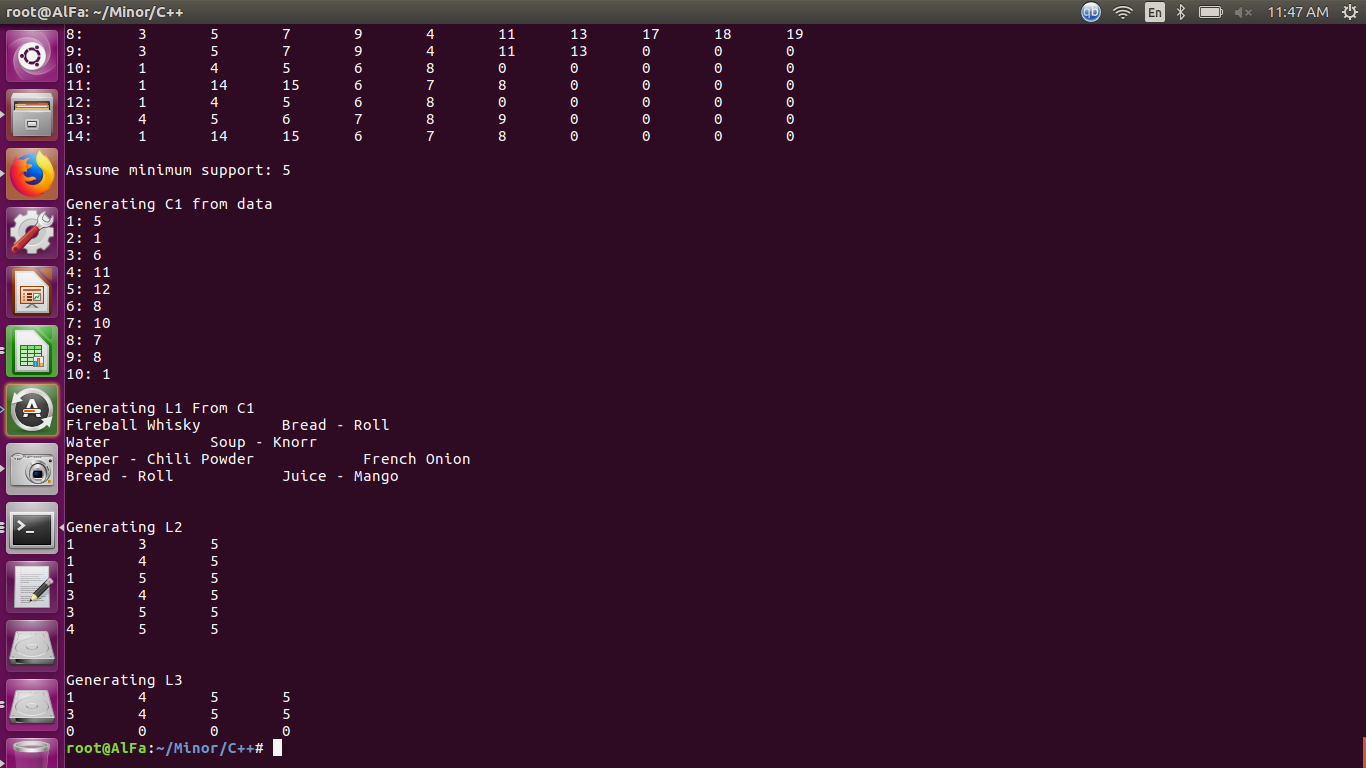
**Figure 5.2 : Apriori Output**



**Figure 5.2 : output screen 1**



**Figure 5.2 : output screen 2**



CHAPTER-6

REFERENCES

1. http://www.isaet.org/images/extraimages/IJCSEE%200101312.pdf
2. https://www.tutorialspoint.com/genetic\_algorithms/genetic\_algorithms\_fundamentals.htm
3. https://gist.github.com/ozcanovunc/e3dda9c0886321fcc6ee
4. http://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit20140503462.pdf
5. http://research.ijcaonline.org/volume48/number17/pxc3880267.pdf
6. https://www.slideshare.net/karthiksankar/genetic-algorithms-3626322
7. http://iasri.res.in/ebook/win\_school\_aa/notes/Genetic\_Algorithm.pdf
8. Software Engineering, Pearson Education; Tenth edition by Ian Sommerville
9. Let Us C++ by Yashwant Kanetkar
10. R. Agrawal, J.C. Shafer, “Parallel mining of association rules”, IEEE Transactions on Knowledge and Data Engineering, vol 8, no 6, pp. 962-969, 1996.
11. D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, Y. Fu, “A fast distributed algorithm for mining association rules”, In Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, pp. 31-42, 1996
12. Data Mining Concepts And Techniques by Jiawei Han and Micheline Kamber , Jian Pei
13. C++ Projects, BPB Publications, Fifth edition by Yashwant Kanetkar
14. M.Z. Ashrafi, D. Taniar, K.A. Smith, “ODAM: an optimized distributed association rule mining algorithm”, IEEE Distributed Systems Online, vol 5, no 3, 2004.