**Practical 1**

**Aim: Write a program which implements – Data Summarization with data characteristics.**

**Data Summarization:**

#include <bits/stdc++.h>

using namespace std;

int dataSetMean(vector<int> d, int n) {

int sum = 0;

for(int i=0; i<n; i++) {

sum = sum + d[i];

}

return sum/n;

}

int dataSetMedian(vector<int> d, int n) {

sort(d.begin(), d.end());

int t = n/2;

if(n%2==0) {

return (d[t-1] + d[t] )/2;

} else {

return d[t];

}

}

int dataSetMode(vector<int> d, int n) {

int max = d[0];

for(int i=1; i<n; i++) {

if(max < d[i]) {

max = d[i];

}

}

int count[max+1];

for(int i=0; i<max+1; i++) {

count[i] = 0;

}

for(int i=0; i<n; i++) {

count[d[i]]++;

}

int mode = 0;

int temp = count[0];

for (int i = 1; i < max+1; i++) {

if (count[i] > temp) {

temp = count[i];

mode = i;

}

}

return mode;

}

float dataSetStandardDevaition(vector<int> d, int n) {

float sum = 0, mean, sd = 0;

int i;

for(i = 0; i < n; ++i) {

sum += d[i];

}

mean = sum / n;

for(i = 0; i < n; ++i) {

sd += pow(d[i] - mean, 2);

}

return sqrt(sd / n);

}

int main() {

vector<int> dataSet;

int n;

cout<<"How many values you want to input in the dataset: "<<endl;

cin>>n;

for(int i=0; i<n; i++) {

int temp;

cin>>temp;

dataSet.push\_back(temp);

}

int mean = dataSetMean(dataSet, n);

cout<<"Mean of the dataSet is: "<<mean<<endl;

int mode = dataSetMode(dataSet, n);

cout<<"Mode of the dataSet is: "<<mode<<endl;

float median = dataSetMedian(dataSet, n);

cout<<"Median of the dataSet is: "<<median<<endl;

float standardDeviation = dataSetStandardDevaition(dataSet, n);

cout<<"Standard Deviation of the dataSet is: "<<standardDeviation<<endl;

}

Output:

How many values you want to input in the dataset:

6

23

56

43

78

33

21

Mean of the dataSet is: 42

Mode of the dataSet is: 21

Median of the dataSet is: 38

Standard Deviation of the dataSet is: 19.897

**Practical 2**

**Aim: Write a program which implements-“Data Cleaning”  Smoothing by binning techniques mean, median and boundaries**

#include <bits/stdc++.h>

using namespace std;

float dataSetMean(vector<int> d, int n) {

float sum = 0;

for(int i=0; i<n; i++) {

sum = sum + d[i];

}

return sum/n;

}

float dataSetMedian(vector<int> d, int n) {

sort(d.begin(), d.end());

int t = n/2;

if(n%2==0) {

return float(d[t-1] + d[t] )/2;

} else {

return float(d[t]);

}

}

int dataSetMode(vector<int> d, int n) {

int max = d[0];

for(int i=1; i<n; i++) {

if(max < d[i]) {

max = d[i];

}

}

int count[max+1];

for(int i=0; i<max+1; i++) {

count[i] = 0;

}

for(int i=0; i<n; i++) {

count[d[i]]++;

}

int mode = 0;

int temp = count[0];

for (int i = 1; i < max+1; i++) {

if (count[i] > temp) {

temp = count[i];

mode = i;

}

}

return mode;

}

float dataSetStandardDevaition(vector<int> d, int n) {

float sum = 0, mean, sd = 0;

int i;

for(i = 0; i < n; ++i) {

sum += d[i];

}

mean = sum / n;

for(i = 0; i < n; ++i) {

sd += pow(d[i] - mean, 2);

}

return sqrt(sd / n);

}

int main() {

int n, b;

cout<<"How many data you want to input: "<<endl;

cin>>n;

cout<<"How many bins you want: "<<endl;

cin>>b;

int nFinal = n;

int paddCount = 0;

if(n%b !=0) {

for(int i=n+1; ; i++) {

paddCount++;

if(i%b == 0) {

nFinal = i;

break;

}

}

}

cout<<"Total elem: "<<nFinal<<endl;

cout<<"padd: "<<paddCount<<endl;

cout<<"Enter the Data: "<<endl;

// vector<float> data;

vector<int> data;

for(int i=0; i<n; i++) {

int temp;

cin>>temp;

data.push\_back(temp);

}

for(int i=n; i<nFinal; i++) {

data.push\_back(0);

}

sort(data.begin(), data.end());

for(int i=0; i<nFinal; i++) {

cout<<data[i]<<" ";

}

map<int, vector<int>> binsData{};

int binCount = 1;

int binFull = 0;

int oneBinSize = nFinal/b;

for(int i=0; i<nFinal; i++) {

vector<int> temp;

if(binFull <= oneBinSize) {

binsData[binCount].push\_back(data[i]);

binFull++;

}

if(binFull == oneBinSize) {

binCount++;

binFull = 0;

cout<<endl;

}

}

cout<<"Original data without data preproccesing: "<<endl;

for (auto p : binsData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

map<int, vector<float>> binsMeanData{};

for(auto& p : binsData) {

float mean = dataSetMean(p.second, p.second.size());

// cout<<mean<<endl;

for(int i=0; i<oneBinSize; i++) {

binsMeanData[p.first].push\_back(mean);

}

}

cout<<endl;

cout<<"Replacing all bins with mean of the data: "<<endl;

for (auto p : binsMeanData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

map<int, vector<float>> binsMedianData{};

for(auto& p : binsData) {

float median = dataSetMedian(p.second, p.second.size());

// cout<<median<<endl;

for(int i=0; i<oneBinSize; i++) {

binsMedianData[p.first].push\_back(median);

}

}

cout<<endl;

cout<<"Replacing all bins with median of the data: "<<endl;

for (auto p : binsMedianData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

map<int, vector<int>> binsModeData{};

for(auto& p : binsData) {

int mode = dataSetMode(p.second, p.second.size());

// cout<<mode<<endl;

for(int i=0; i<oneBinSize; i++) {

binsModeData[p.first].push\_back(mode);

}

}

cout<<endl;

cout<<"Replacing all bins with mode of the data: "<<endl;

for (auto p : binsModeData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

map<int, vector<float>> binsSTDData{};

for(auto& p : binsData) {

float std = dataSetStandardDevaition(p.second, p.second.size());

for(int i=0; i<oneBinSize; i++) {

binsSTDData[p.first].push\_back(std);

}

}

cout<<endl;

cout<<"Replacing all bins with Standard deviation of the data: "<<endl;

for (auto p : binsSTDData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

map<int, vector<float>> binsBoundariesData{};

for(auto& p : binsData) {

binsBoundariesData[p.first].push\_back(p.second[0]);

for(int i=1; i<oneBinSize; i++) {

float tempStart, tempEnd;

tempStart = abs(p.second[i] - p.second[0]);

tempEnd = abs(p.second[i] - p.second[oneBinSize-1]);

if(tempStart <= tempEnd) {

binsBoundariesData[p.first].push\_back(p.second[0]);

} else {

binsBoundariesData[p.first].push\_back(p.second[oneBinSize-1]);

}

}

}

cout<<endl;

cout<<"Replacing all bins with Boundaries of the data: "<<endl;

for (auto p : binsBoundariesData) {

cout<<p.first<<": ";

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

cout<<endl;

}

cout<<endl<<"Data after Smoothing by Mean: "<<endl;

for (auto p : binsMeanData) {

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

}

cout<<endl<<"Data after Smoothing by Median: "<<endl;

for (auto p : binsMedianData) {

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

}

cout<<endl<<"Data after Smoothing by Mode: "<<endl;

for (auto p : binsModeData) {

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

}

cout<<endl<<"Data after Smoothing by Standard Devaition: "<<endl;

for (auto p : binsSTDData) {

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

}

cout<<endl<<"Data after Smoothing by Boundaries: "<<endl;

for (auto p : binsBoundariesData) {

for(int i=0; i<oneBinSize; i++) {

cout<<p.second[i]<<" ";

}

}

}

Output:

How many data you want to input:

8

How many bins you want:

3

Total elem: 9

padd: 1

Enter the Data:

23

89

94

45

11

15

33

78

0 11 15 23 33 45 78 89 94

Original data without data preproccesing:

1: 0 11 15

2: 23 33 45

3: 78 89 94

Replacing all bins with mean of the data:

1: 8.66667 8.66667 8.66667

2: 33.6667 33.6667 33.6667

3: 87 87 87

Replacing all bins with median of the data:

1: 11 11 11

2: 33 33 33

3: 89 89 89

Replacing all bins with mode of the data:

1: 0 0 0

2: 23 23 23

3: 78 78 78

Replacing all bins with Standard deviation of the data:

1: 6.3421 6.3421 6.3421

2: 8.99382 8.99382 8.99382

3: 6.68331 6.68331 6.68331

Replacing all bins with Boundaries of the data:

1: 0 15 15

2: 23 23 45

3: 78 94 94

Data after Smoothing by Mean:

8.66667 8.66667 8.66667 33.6667 33.6667 33.6667 87 87 87

Data after Smoothing by Median:

11 11 11 33 33 33 89 89 89

Data after Smoothing by Mode:

0 0 0 23 23 23 78 78 78

Data after Smoothing by Standard Devaition:

6.3421 6.3421 6.3421 8.99382 8.99382 8.99382 6.68331 6.68331 6.68331

Data after Smoothing by Boundaries:

0 15 15 23 23 45 78 94 94

**Practical 3**

**Aim:Write a program which implements Data Transformation by Min- max normalization.**

 package lab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

public class MinMaxNormalize {

public static double dataSetMean(double d[], int n) {

double sum = 0;

for(int i=0; i<n; i++) {

sum = sum + d[i];

}

return sum/n;

}

public static float dataSetStandardDevaition(double d[], int n) {

double sum = 0, mean, sd = 0;

int i;

for(i = 0; i < n; ++i) {

sum += d[i];

}

mean = sum / n;

for(i = 0; i < n; ++i) {

sd += Math.pow(d[i] - mean, 2);

}

return (float) Math.sqrt(sd / n);

}

public static void main(String[] args) {

try {

Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");

Connection con = DriverManager.getConnection("jdbc:ucanaccess://C://Users//User//Documents/emp.accdb");

Scanner sc = new Scanner(System.in);

PreparedStatement ps = con.prepareStatement("UPDATE Table1 SET Salary = ? Where ID = ?");

Statement s = con.createStatement();

while(true) {

System.out.println("1. Perform normalization");

System.out.println("2. Enter Data");

int choice = sc.nextInt();

if(choice == 2) {

System.out.println("Enter the data for which you want to perform normalization");

PreparedStatement psi = con.prepareStatement("insert into Table1(ID, Salary) values(?, ?)");

int check = 0;

while(!(check == 1))

{

System.out.println("Enter the ID of the person ");

int id = sc.nextInt();

System.out.println("Enter the salary of the person");

double salary = sc.nextInt();

psi.setInt(1, id);

psi.setDouble(2, salary);

psi.executeUpdate();

System.out.println("Type exit if u dont want to enter more data");

check = sc.nextInt();

}

psi.close();

}

else {

String query = "Select \* from Table1";

ResultSet rs = s.executeQuery(query);

double salary[] = new double[100];

int i=0;

while(rs.next()) {

salary[i] = rs.getDouble("Salary");

i++;

}

System.out.println("Original Salary of all the Employees fetch from the Database: ");

for(int j=0; j<i; j++) {

System.out.println(salary[j]);

}

System.out.println(" Min-Max normaliztion");

double max = salary[0], min = salary[0], new\_max = 1.0, new\_min = 0.0;

System.out.println("Max and Min values of the given range of salary: ");

for(int j=1; j<i; j++) {

if(salary[j] > max) {

max = salary[j];

}

if(salary[j] < min) {

min = salary[j];

}

}

System.out.println(max+" "+min);

System.out.println("Salary of all the Employess after Min-Max Normalization: ");

for(int j=0; j<i; j++) {

double temp;

temp = ((salary[j] - min)\*(new\_max - new\_min) + new\_min)/(max-min);

temp = Math.round(temp\*100)/100.0d;

salary[j] = temp;

System.out.println(salary[j]);

}

int out = 0;

out = sc.nextInt();

if(out == 1) {

break;

}

// Updating Normalized Salary of all the Employees in the Database

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

ps.setDouble(1, salary[j]);

ps.setInt(2, j+1);

ps.executeUpdate();

}

}

}

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

Output:

1. Perform normalization

2. Enter Data

2

Enter the data for which you want to perform normalization

Enter the ID of the person

12

Enter the salary of the person

2340

Type exit if u dont want to enter more data

1

1. Perform normalization

2. Enter Data

1

Original Salary of all the Employees fetch from the Database:

2000.0

1400.0

4500.0

6000.0

12000.0

11000.0

7800.0

9400.0

6700.0

12.0

12000.0

13000.0

12111.0

11111.0

12222.0

33333.0

5345.0

2500.0

3400.0

2340.0

Min-Max normaliztion

Max and Min values of the given range of salary:

33333.0 12.0

Salary of all the Employess after Min-Max Normalization:

0.06

0.04

0.13

0.18

0.36

0.33

0.23

0.28

0.2

0.0

0.36

0.39

0.36

0.33

0.37

1.0

0.16

0.07

0.1

0.07

**Practical 4**

**Aim: Write a program which implements Data Transformation by Z- Score normalization.**

package lab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

public class ZScoreNormalize {

public static double dataSetMean(double d[], int n) {

double sum = 0;

for(int i=0; i<n; i++) {

sum = sum + d[i];

}

return sum/n;

}

public static float dataSetStandardDevaition(double d[], int n) {

double sum = 0, mean, sd = 0;

int i;

for(i = 0; i < n; ++i) {

sum += d[i];

}

mean = sum / n;

for(i = 0; i < n; ++i) {

sd += Math.pow(d[i] - mean, 2);

}

return (float) Math.sqrt(sd / n);

}

public static void main(String[] args) {

try {

Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");

Connection con = DriverManager.getConnection("jdbc:ucanaccess://C://Users//User//Documents/emp.accdb");

Scanner sc = new Scanner(System.in);

PreparedStatement ps = con.prepareStatement("UPDATE Table1 SET Salary = ? Where ID = ?");

Statement s = con.createStatement();

while(true) {

System.out.println("1. Perform normalization");

System.out.println("2. Enter Data");

int choice = sc.nextInt();

if(choice == 2) {

System.out.println("Enter the data for which you want to perform normalization");

PreparedStatement psi = con.prepareStatement("insert into Table1(ID, Salary) values(?, ?)");

int check = 0;

while(!(check == 1))

{

System.out.println("Enter the ID of the person ");

int id = sc.nextInt();

System.out.println("Enter the salary of the person");

double salary = sc.nextInt();

psi.setInt(1, id);

psi.setDouble(2, salary);

psi.executeUpdate();

System.out.println("Type exit if u dont want to enter more data");

check = sc.nextInt();

}

psi.close();

}

else {

String query = "Select \* from Table1";

ResultSet rs = s.executeQuery(query);

double salary[] = new double[100];

int i=0;

while(rs.next()) {

salary[i] = rs.getDouble("Salary");

i++;

}

System.out.println("Original Salary of all the Employees fetch from the Database: ");

for(int j=0; j<i; j++) {

System.out.println(salary[j]);

}

System.out.println(" Z-score normaliztion");

double mean = dataSetMean(salary, 10);

double sd = dataSetStandardDevaition(salary, 10);

System.out.println("Salary of all the Employess after Z-score Normalization: ");

for(int j=0; j<i; j++) {

double temp;

temp = (salary[j] - mean)/sd;

temp = Math.round(temp\*100)/100.0d;

salary[j] = temp;

System.out.println(salary[j]);

}

int out = 0;

out = sc.nextInt();

if(out == 1) {

break;

}

// Updating Normalized Salary of all the Employees in the Database

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

ps.setDouble(1, salary[j]);

ps.setInt(2, j+1);

ps.executeUpdate();

}

}

}

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

Output:

1. Perform normalization

2. Enter Data

2

Enter the data for which you want to perform normalization

Enter the ID of the person

1

Enter the salary of the person

5466

Type exit if u dont want to enter more data

1

1. Perform normalization

2. Enter Data

2

Enter the data for which you want to perform normalization

Enter the ID of the person

4

Enter the salary of the person

2111

Type exit if u dont want to enter more data

1

1. Perform normalization

2. Enter Data

1

Original Salary of all the Employees fetch from the Database:

2000.0

1400.0

4500.0

6000.0

12000.0

11000.0

7800.0

9400.0

6700.0

12.0

12000.0

13000.0

12111.0

11111.0

12222.0

33333.0

5345.0

2500.0

3400.0

2340.0

5466.0

2111.0

Z-score normaliztion

Salary of all the Employess after Z-score Normalization:

-1.05

-1.2

-0.41

-0.02

1.52

1.26

0.44

0.85

0.16

-1.56

1.52

1.78

1.55

1.29

1.58

7.0

-0.19

-0.92

-0.69

-0.96

-0.16

-1.02

**Practical 5**

**Aim: To perform multi-dimensional data model using SQL queries. Star schemas.**

**Time Table:**

CREATE TABLE TimeTable

(

time\_key int primary key,

time\_day int,

time\_month int,

time\_quater int,

time\_year varchar(50)

);

**Insert query of Time Table:**

INSERT INTO `starschema`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('1', '10', '1', '1', '2022');

INSERT INTO `starschema`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('2', '15', '6', '2', '2022');

INSERT INTO `starschema`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('3', '18', '1', '1', '2022');

INSERT INTO `starschema`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('4', '22', '5', '2', '2022');

Time Table:

time\_key time\_day time\_month time\_quater time\_year

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10 | 1 | 1 | 2022 |
| 2 | 15 | 6 | 2 | 2022 |
| 3 | 18 | 1 | 1 | 2022 |
| 4 | 22 | 5 | 2 | 2022 |
|  |  |  |  |  |

**Item Table:**

CREATE TABLE ItemTable

(

item\_key int primary key,

item\_name varchar(50) ,

item\_brand varchar(50) ,

item\_type varchar(50) ,

item\_supplier\_type varchar(50)

);

**Insert query of Item Table:**

INSERT INTO `starschema`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `item\_price`) VALUES ('1', 'realme 10', 'realme', 'Phone', '15000');

INSERT INTO `starschema`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `item\_price`) VALUES ('2', 'iphone 10', 'apple', 'Phone', '25000');

INSERT INTO `starschema`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `item\_price`) VALUES ('3', 'microsoft surface', 'microsoft', 'Laptop', '100000');

INSERT INTO `starschema`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `item\_price`) VALUES ('4', 'redmi 12', 'mi', 'Phone', '12000');

**Item Table:**

Item\_key item\_name item\_brand item\_type item\_price

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | realme 10 | realme | Phone | 15000 |
| 2 | iphone 10 | apple | Phone | 25000 |
| 3 | microsoft surface | microsoft | Laptop | 100000 |
| 4 | redmi 12 | mi | Phone | 12000 |
|  |  |  |  |  |

**Branch Table:**

CREATE TABLE BranchTable

(

branch\_key int primary key,

branch\_name varchar(50) ,

branch\_type varchar(50)

);

**Insert query for Branch Table:**

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('1', 'branch\_1', 'type\_1');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('2', 'branch\_2', 'type\_2');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('3', 'branch\_3', 'type\_3');

**Branch Table:**

branch\_key branch\_name branch\_type

1 branch\_1 type\_1

2 branch\_2 type\_2

3 branch\_3 type\_3

**Location Table:**

CREATE TABLE LocationTable

(

location\_key int primary key,

location\_street varchar(50) ,

location\_city varchar(50) ,

location\_state varchar(50) ,

location\_country varchar(50)

);

**Insert query for Location Table:**

INSERT INTO `starschema`.`locationtable` (`location\_key`, `location\_city`, `location\_state`, `location\_country`) VALUES ('1', 'anand', 'gujarat', 'india');

INSERT INTO `starschema`.`locationtable` (`location\_key`, `location\_city`, `location\_state`, `location\_country`) VALUES ('2', 'baroda', 'gujarat', 'india');

INSERT INTO `starschema`.`locationtable` (`location\_key`, `location\_city`, `location\_state`, `location\_country`) VALUES ('3', 'atlanta', 'georgia', 'usa');

INSERT INTO `starschema`.`locationtable` (`location\_key`, `location\_city`, `location\_state`, `location\_country`) VALUES ('5', 'amd', 'gujarat', 'india');

**Location Table:**

location\_key location\_name location\_state location\_country

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | anand | gujarat | india |
| 2 | baroda | gujarat | India |
| 3 | atlanta | georgia | usa |
| 4 | amd | gujarat | india |
|  |  |  |  |
|  |  |  |  |

**SalesFact Table:**

CREATE TABLE SalesTable

(

time\_key int,

item\_key int,

branch\_key int,

location\_key int,

units\_sold int,

dollars\_sold int,

FOREIGN KEY (time\_key) REFERENCES TimeTable(time\_key),

FOREIGN KEY (item\_key) REFERENCES ItemTable(item\_key),

FOREIGN KEY (branch\_key) REFERENCES BranchTable(branch\_key),

FOREIGN KEY (location\_key) REFERENCES LocationTable(branch\_key),

);

**Insert query for Salesfact Table:**

INSERT INTO `starschema`.`salesfact` (`number\_of\_units\_sold`, `price`, `item\_key`, `time\_key`, `location\_key`, `branch\_key`, `id`) VALUES ('200', '15000', '1', '1', '1', '1', '1');

INSERT INTO `starschema`.`salesfact` (`number\_of\_units\_sold`, `price`, `item\_key`, `time\_key`, `location\_key`, `branch\_key`, `id`) VALUES (50, '25000', '2', '1', '2', '2', '2');

**Salesfact Table:**

number\_of\_units\_sold price item\_key time\_key location\_key branch\_key id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 200 | 15000 | 1 | 1 | 1 | 1 | 1 |
| 50 | 25000 | 2 | 1 | 2 | 2 | 2 |
|  |  |  |  |  |  |  |

**Multidimensional Query for querying star schema:**

1).

select s.time\_key, s.item\_key, s.branch\_key, s.location\_key, sum(s.number\_of\_units\_sold \* s.price), sum(s.number\_of\_units\_sold)

from timetable t, itemtable i, branchtable b, locationtable l, salesfact s

where s.time\_key = t.time\_key and s.item\_key = i.item\_key and s.branch\_key = b.branch\_key and s.location\_key = l.location\_key

GROUP BY s.time\_key, s.item\_key , s.branch\_key, s.location\_key;

time\_key item\_key branch\_key location\_key sum(s.number\_of\_units\_sold \* s.price)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 3000000 |  |
| 1 | 2 | 2 | 2 | 1250000 |  |
|  |  |  |  |  |  |

sum(s.number\_of\_units\_sold)

|  |
| --- |
| 200 |
| 50 |
|  |

2).

select i.item\_name, t.time\_quater, b.branch\_name, l.location\_city, sum(s.number\_of\_units\_sold \* s.price), sum(s.number\_of\_units\_sold)

from timetable t, itemtable i, branchtable b, locationtable l, salesfact s

where s.time\_key = t.time\_key and s.item\_key = i.item\_key and s.branch\_key = b.branch\_key and s.location\_key = l.location\_key

GROUP BY s.time\_key, s.item\_key , s.branch\_key, s.location\_key;

Item\_name time\_quater branch\_name location\_city sum(s.number\_of\_units\_sold \* s.price)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| realme 10 | 1 | branch\_1 | anand | 3000000 |
| iphone 10 | 1 | branch\_2 | baroda | 1250000 |

sum(s.number\_of\_units\_sold)

|  |
| --- |
| 200 |
| 50 |

**Practical 6**

**Aim. To perform multi-dimensional data model using SQL queries. Snowflake and fact constellation schemas.**

**Snowflake Schema:**

**Time Table:**

CREATE TABLE TimeTable

(

time\_key int primary key,

time\_day int,

time\_month int,

time\_quater int,

time\_year varchar(50)

);

**Insert query of Time Table:**

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('1', '10', '1', '1', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('2', '15', '6', '2', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('3', '18', '1', '1', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('4', '22', '5', '2', '2022');Time Table:

time\_key time\_day time\_month time\_quater time\_year

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10 | 1 | 1 | 2022 |
| 2 | 15 | 6 | 2 | 2022 |
| 3 | 18 | 1 | 1 | 2022 |
| 4 | 22 | 5 | 2 | 2022 |

**Item Table & Supplier Table:**

CREATE TABLE ItemTable

(

item\_key int primary key,

item\_name varchar(50) ,

item\_brand varchar(50) ,

item\_type varchar(50) ,

supplier\_key int,

FOREIGN KEY (supplier\_key) REFERENCES SupplierTable(supplier\_key)

);

CREATE TABLE SupplierTable

(

supplier\_key int primary key,

supplier\_type varchar(50)

);

**Insert query of Item Table:**

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('1', 'realme 10', 'realme', 'Phone', 1);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('2', 'iphone 10', 'apple', 'Phone', 2);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('3', 'microsoft surface', 'microsoft', 'Laptop', 1);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('4', 'redmi 12', 'mi', 'Phone', 2);

**Insert query of Supplier Table:**

INSERT INTO `snowflake`.`suppliertable` (`supplier\_key`, `supplier\_type`) VALUES ('1', 'type\_1');

INSERT INTO `snowflake`.`suppliertable` (`supplier\_key`, `supplier\_type`) VALUES ('2', 'type\_2');

**Supplier Table:**

supplier\_key supplier\_name

|  |  |
| --- | --- |
| 1 | type\_1 |
| 2 | type\_2 |
|  |  |

**Item Table:**

Item\_key item\_name item\_brand item\_type supplier\_key

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | realme 10 | realme | Phone | 1 |
| 2 | iphone 10 | apple | Phone | 2 |
| 3 | microsoft surface | microsoft | Laptop | 1 |
| 4 | redmi 12 | mi | Phone | 2 |
|  |  |  |  |  |

**Branch Table:**

CREATE TABLE BranchTable

(

branch\_key int primary key,

branch\_name varchar(50) ,

branch\_type varchar(50)

);

**Insert query for Branch Table:**

INSERT INTO ` snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('1', 'branch\_1', 'type\_1');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('2', 'branch\_2', 'type\_2');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('3', 'branch\_3', 'type\_3');

**Branch Table:**

branch\_key branch\_name branch\_type

1 branch\_1 type\_1

2 branch\_2 type\_2

3 branch\_3 type\_3

**Location Table & City Table:**

CREATE TABLE LocationTable

(

location\_key int primary key,

location\_street varchar(50) ,

city\_key int,

FOREIGN KEY (city\_key) REFERENCES CityTable(city\_key)

);

CREATE TABLE CityTable

(

city\_key int primary key,

city\_name varchar(50),

city\_state varchar(50),

city\_country varchar(50)

);

**Insert query for Location Table:**

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('1', 'street-1', '1');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('2', 'street-2', '2');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('3', 'street-3', '3');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('4', 'street-4', '4');

**Location Table:**

location\_key location\_street city\_key

1 street-1 1

2 street-2 2

3 street-3 3

4 street-4 4

**Insert query for City Table:**

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('1', 'anand', 'gujarat', 'india');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('2', 'baroda', 'gujarat', 'india');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('3', 'atlanta', 'georgia', 'usa');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('4', 'amd', 'gujarat', 'india');

**City Table:**

city\_key city\_name city\_state city\_country

1 anand gujarat india

2 baroda gujarat india

3 atlanta georgia usa

4 amd gujarat india

**SalesFact Table:**

CREATE TABLE SalesTable

(

time\_key int,

item\_key int,

branch\_key int,

location\_key int,

units\_sold int,

dollars\_sold int,

FOREIGN KEY (time\_key) REFERENCES TimeTable(time\_key),

FOREIGN KEY (item\_key) REFERENCES ItemTable(item\_key),

FOREIGN KEY (branch\_key) REFERENCES BranchTable(branch\_key),

FOREIGN KEY (location\_key) REFERENCES LocationTable(branch\_key),

);

**Insert query for SalesFact Table:**

INSERT INTO `snowflake`.`salesfact` (`item\_key`, `time\_key`, `branch\_key`, `location\_key`, `dollars\_sold`, `units\_sold`, `id`) VALUES ('3', '3', '2', '3', '20000', '110', '1');

INSERT INTO `snowflake`.`salesfact` (`item\_key`, `time\_key`, `branch\_key`, `location\_key`, `dollars\_sold`, `units\_sold`, `id`) VALUES ('4', '2', '3', '4', '10000', '230', '2');

**SalesTable of SnowFlake:**

Item\_key time\_key branch\_key location\_key dollars\_sold units\_sold id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 3 | 2 | 3 | 20000 | 110 | 1 |
| 4 | 2 | 3 | 4 | 10000 | 230 | 2 |
|  |  |  |  |  |  |  |

**Multidimensional Query for querying snowflake schema:**

1).

select i.item\_name, t.time\_quater, b.branch\_name, l.location\_street, sum(s.units\_sold \* s.dollars\_sold), sum(s.units\_sold)

from timetable t, itemtable i, branchtable b, locationtable l, salesfact s

where s.time\_key = t.time\_key and s.item\_key = i.item\_key and s.branch\_key = b.branch\_key and s.location\_key = l.location\_key

GROUP BY s.time\_key, s.item\_key , s.branch\_key, s.location\_key;

Item\_name time\_quater branch\_name location\_street

|  |  |  |  |
| --- | --- | --- | --- |
| redmi 12 | 2 | branch\_3 | Street-4 |
| Microsoft surface | 1 | branch\_2 | Street-3 |

sum(s.number\_of\_units\_sold \* s.price) sum(s.number\_of\_units\_sold)

|  |
| --- |
| 2300000 |
| 2200000 |
|  |

230

110

**Fact constellation schemas:**

**Time Table:**

CREATE TABLE TimeTable

(

time\_key int primary key,

time\_day int,

time\_month int,

time\_quater int,

time\_year varchar(50)

);

**Insert query of Time Table:**

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('1', '10', '1', '1', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('2', '15', '6', '2', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('3', '18', '1', '1', '2022');

INSERT INTO `snowflake`.`timetable` (`time\_key`, `time\_day`, `time\_month`, `time\_quater`, `time\_year`) VALUES ('4', '22', '5', '2', '2022');Time Table:

time\_key time\_day time\_month time\_quater time\_year

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10 | 1 | 1 | 2022 |
| 2 | 15 | 6 | 2 | 2022 |
| 3 | 18 | 1 | 1 | 2022 |
| 4 | 22 | 5 | 2 | 2022 |

**Item Table & Supplier Table:**

CREATE TABLE ItemTable

(

item\_key int primary key,

item\_name varchar(50) ,

item\_brand varchar(50) ,

item\_type varchar(50) ,

supplier\_key int,

FOREIGN KEY (supplier\_key) REFERENCES SupplierTable(supplier\_key)

);

CREATE TABLE SupplierTable

(

supplier\_key int primary key,

supplier\_type varchar(50)

);

**Insert query of Item Table:**

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('1', 'realme 10', 'realme', 'Phone', 1);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('2', 'iphone 10', 'apple', 'Phone', 2);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('3', 'microsoft surface', 'microsoft', 'Laptop', 1);

INSERT INTO `snowflake`.`itemtable` (`item\_key`, `item\_name`, `item\_brand`, `item\_type`, `supplier\_key`) VALUES ('4', 'redmi 12', 'mi', 'Phone', 2);

**Insert query of Supplier Table:**

INSERT INTO `snowflake`.`suppliertable` (`supplier\_key`, `supplier\_type`) VALUES ('1', 'type\_1');

INSERT INTO `snowflake`.`suppliertable` (`supplier\_key`, `supplier\_type`) VALUES ('2', 'type\_2');

**Supplier Table:**

supplier\_key supplier\_name

|  |  |
| --- | --- |
| 1 | type\_1 |
| 2 | type\_2 |
|  |  |

**Item Table:**

Item\_key item\_name item\_brand item\_type supplier\_key

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | realme 10 | realme | Phone | 1 |
| 2 | iphone 10 | apple | Phone | 2 |
| 3 | microsoft surface | microsoft | Laptop | 1 |
| 4 | redmi 12 | mi | Phone | 2 |
|  |  |  |  |  |

**Branch Table:**

CREATE TABLE BranchTable

(

branch\_key int primary key,

branch\_name varchar(50) ,

branch\_type varchar(50)

);

**Insert query for Branch Table:**

INSERT INTO ` snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('1', 'branch\_1', 'type\_1');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('2', 'branch\_2', 'type\_2');

INSERT INTO `snowflake`.`branchtable` (`branch\_key`, `branch\_name`, `branch\_type`) VALUES ('3', 'branch\_3', 'type\_3');

**Branch Table:**

branch\_key branch\_name branch\_type

1 branch\_1 type\_1

2 branch\_2 type\_2

3 branch\_3 type\_3

**Location Table & City Table:**

CREATE TABLE LocationTable

(

location\_key int primary key,

location\_street varchar(50) ,

city\_key int,

FOREIGN KEY (city\_key) REFERENCES CityTable(city\_key)

);

CREATE TABLE CityTable

(

city\_key int primary key,

city\_name varchar(50),

city\_state varchar(50),

city\_country varchar(50)

);

**Insert query for Location Table:**

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('1', 'street-1', '1');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('2', 'street-2', '2');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('3', 'street-3', '3');

INSERT INTO `snowflake`.`locationtable` (`location\_key`, `location\_street`, `city\_key`) VALUES ('4', 'street-4', '4');

**Location Table:**

location\_key location\_street city\_key

1 street-1 1

2 street-2 2

3 street-3 3

4 street-4 4

**Insert query for City Table:**

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('1', 'anand', 'gujarat', 'india');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('2', 'baroda', 'gujarat', 'india');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('3', 'atlanta', 'georgia', 'usa');

INSERT INTO `snowflake`.`citytable` (`city\_key`, `city\_name`, `city\_state`, `city\_country`) VALUES ('4', 'amd', 'gujarat', 'india');

**City Table:**

city\_key city\_name city\_state city\_country

1 anand gujarat india

2 baroda gujarat india

3 atlanta georgia usa

4 amd gujarat india

**Shipper Table:**

CREATE TABLE ShipperTable

(

shipper\_key int primary key,

shipper\_name varchar(50),

shipper\_type varchar(50),

location\_key int,

FOREIGN KEY (location\_key) REFERENCES LocationTable(location\_key)

);

**Insert query of Shipper Table:**

INSERT INTO `snowflake`.`new\_table` (`shipper\_key`, `shipper\_name`, `location\_key`, `shipper\_type`) VALUES ('1', 'e-kart', '1', 'type-1');

INSERT INTO `snowflake`.`new\_table` (`shipper\_key`, `shipper\_name`, `location\_key`, `shipper\_type`) VALUES ('2', 'ecom', '2', 'type-2');

**Shipper Table:**

Shipper\_key shipper\_name location\_key shipper\_type

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | e-kart | 1 | type-1 |
| 2 | ecom | 2 | type-2 |
|  |  |  |  |

**SalesFact Table:**

CREATE TABLE SalesTable

(

time\_key int,

item\_key int,

branch\_key int,

location\_key int,

units\_sold int,

dollars\_sold int,

FOREIGN KEY (time\_key) REFERENCES TimeTable(time\_key),

FOREIGN KEY (item\_key) REFERENCES ItemTable(item\_key),

FOREIGN KEY (branch\_key) REFERENCES BranchTable(branch\_key),

FOREIGN KEY (location\_key) REFERENCES LocationTable(branch\_key),

);

**Insert query for SalesFact Table:**

INSERT INTO `snowflake`.`salesfact` (`item\_key`, `time\_key`, `branch\_key`, `location\_key`, `dollars\_sold`, `units\_sold`, `id`) VALUES ('3', '3', '2', '3', '20000', '110', '1');

INSERT INTO `snowflake`.`salesfact` (`item\_key`, `time\_key`, `branch\_key`, `location\_key`, `dollars\_sold`, `units\_sold`, `id`) VALUES ('4', '2', '3', '4', '10000', '230', '2');

**SalesTable of Fact Constellation:**

Item\_key time\_key branch\_key location\_key dollars\_sold units\_sold id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 3 | 2 | 3 | 20000 | 110 | 1 |
| 4 | 2 | 3 | 4 | 10000 | 230 | 2 |
|  |  |  |  |  |  |  |

**ShippingFact Table:**

CREATE TABLE ShippingTable

(

time\_key int,

item\_key int,

shipper\_key int,

from\_location int,

to\_location int,

units\_shipped int,

dollars\_cost int,

FOREIGN KEY (time\_key) REFERENCES TimeTable(time\_key),

FOREIGN KEY (item\_key) REFERENCES ItemTable(item\_key),

FOREIGN KEY (shipper\_key) REFERENCES ShipperTable(shipper\_key),

FOREIGN KEY (from\_location) REFERENCES LocationTable(location\_key),

FOREIGN KEY (to\_location) REFERENCES LocationTable(location\_key)

);

**Insert query for ShippingFact Table:**

INSERT INTO `snowflake`.`shippingtable` (`item\_key`, `time\_key`, `shipper\_key`, `from\_location`, `to\_location`, `dollars\_cost`, `units\_shipped`, `id`) VALUES ('3', '3', '1', '3', '4', '2000', '100', '1');

INSERT INTO `snowflake`.`shippingtable` (`item\_key`, `time\_key`, `shipper\_key`, `from\_location`, `to\_location`, `dollars\_cost`, `units\_shipped`, `id`) VALUES ('4', '4', '2', '4', '1', '1500', '45', '2');

**ShippingFactTable of Fact Constellation:**

Item\_key time\_key shipping\_key from\_location to\_location dollars\_cost

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 3 | 1 | 3 | 4 | 2000 |  |  |
| 4 | 4 | 2 | 4 | 1 | 1500 |  |  |
|  |  |  |  |  |  |  |  |

units\_shipped id

|  |  |
| --- | --- |
| 100 | 1 |
| 45 | 2 |

**Multidimensional Query for querying SalesFact schema:**

1).

select i.item\_name, t.time\_quater, b.branch\_name, l.location\_street, sum(s.units\_sold \* s.dollars\_sold), sum(s.units\_sold)

from timetable t, itemtable i, branchtable b, locationtable l, salesfact s

where s.time\_key = t.time\_key and s.item\_key = i.item\_key and s.branch\_key = b.branch\_key and s.location\_key = l.location\_key

GROUP BY s.time\_key, s.item\_key , s.branch\_key, s.location\_key;

Item\_name time\_quater branch\_name location\_street

|  |  |  |  |
| --- | --- | --- | --- |
| redmi 12 | 2 | branch\_3 | Street-4 |
| Microsoft surface | 1 | branch\_2 | Street-3 |

sum(s.number\_of\_units\_sold \* s.price) sum(s.number\_of\_units\_sold)

|  |
| --- |
| 2300000 |
| 2200000 |
|  |

230

110

**Multidimensional Query for querying ShippingFact schema:**

select i.item\_name, t.time\_quater,sum(s.units\_shipped \* s.dollars\_cost), sum(s.units\_shipped)

from timetable t, itemtable i, locationtable l, shippingtable s

where s.time\_key = t.time\_key and s.item\_key = i.item\_key

GROUP BY s.time\_key, s.item\_key , s.from\_location;

Item\_name time\_quater sum(s.units\_shipped\*s.dollars\_cost) sum(s.units\_shipped)

|  |  |  |  |
| --- | --- | --- | --- |
| microsoft surface | 1 | 800000 | 400 |
| redmi 12 | 2 | 270000 | 180 |

**Practical 7**

**Aim: Write a program to find out support and confidence for given dataset**

#include <bits/stdc++.h>

using namespace std;

int findInTransactions(set<set<int>> trns, set<int> tt)

{

int count = 0;

for (auto e = trns.begin(); e != trns.end(); e++)

{

set<int> s = \*e;

if (includes(s.begin(), s.end(), tt.begin(), tt.end()))

count++;

}

return count;

}

void reduse(set<set<int>> trns, set<set<int>> &kk, int support, int len)

{

for (auto i = kk.begin(); i != kk.end(); i++)

{

set<int> temp = \*i;

if (temp.size() == len)

{

int sp = findInTransactions(trns, temp);

if (sp < support)

{

kk.erase(i);

}

}

}

}

void printSet(set<set<int>> kk, set<set<int>> trns)

{

for (auto i = kk.begin(); i != kk.end(); i++)

{

set<int> temp1 = \*i;

for (auto j = temp1.begin(); j != temp1.end(); j++)

{

cout << \*j << " ";

}

cout << "Counts : " << findInTransactions(trns, temp1) << endl;

}

}

void findPermutations(set<set<int>> &s, int k)

{

set<set<int>> kk;

for (auto i = s.begin(); i != s.end(); i++)

{

set<int> temp = \*i;

if (temp.size() == k - 1)

{

kk.insert(temp);

}

}

if (kk.size() == 0)

return;

for (auto i = kk.begin(); i != kk.end(); i++)

{

for (auto j = kk.begin(); j != kk.end(); j++)

{

set<int> temp1 = \*i;

set<int> temp2 = \*j;

set<int> hh;

merge(temp1.begin(), temp1.end(), temp2.begin(), temp2.end(), inserter(hh, hh.begin()));

if (hh.size() == k)

s.insert(hh);

}

}

}

int main()

{

int n;

cout << "Enter n : ";

cin >> n;

vector<vector<int>> t;

set<set<int>> ff;

set<set<int>> trans;

map<set<int>, int> mp;

for (int i = 0; i < n; i++)

{

int k;

cout << "Enter k : ";

cin >> k;

cout << "Enter k elements : ";

vector<int> vv;

set<int> vvv;

for (int j = 0; j < k; j++)

{

int l;

cin >> l;

vvv.insert(l);

set<int> c;

c.insert(l);

ff.insert(c);

mp[c]++;

}

trans.insert(vvv);

t.push\_back(vv);

}

int ll = ff.size();

int cl = 0;

int spll = 2;

int support;

cout << "Enter mininmum support : ";

cin >> support;

while (ll != cl)

{

cl = ll;

findPermutations(ff, spll);

reduse(trans, ff, spll, support);

spll++;

ll = ff.size();

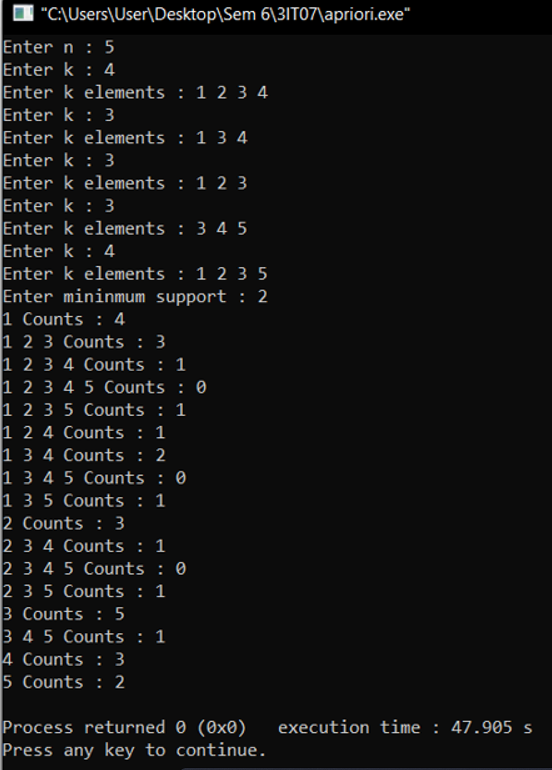
}

printSet(ff, trans);

return 0;

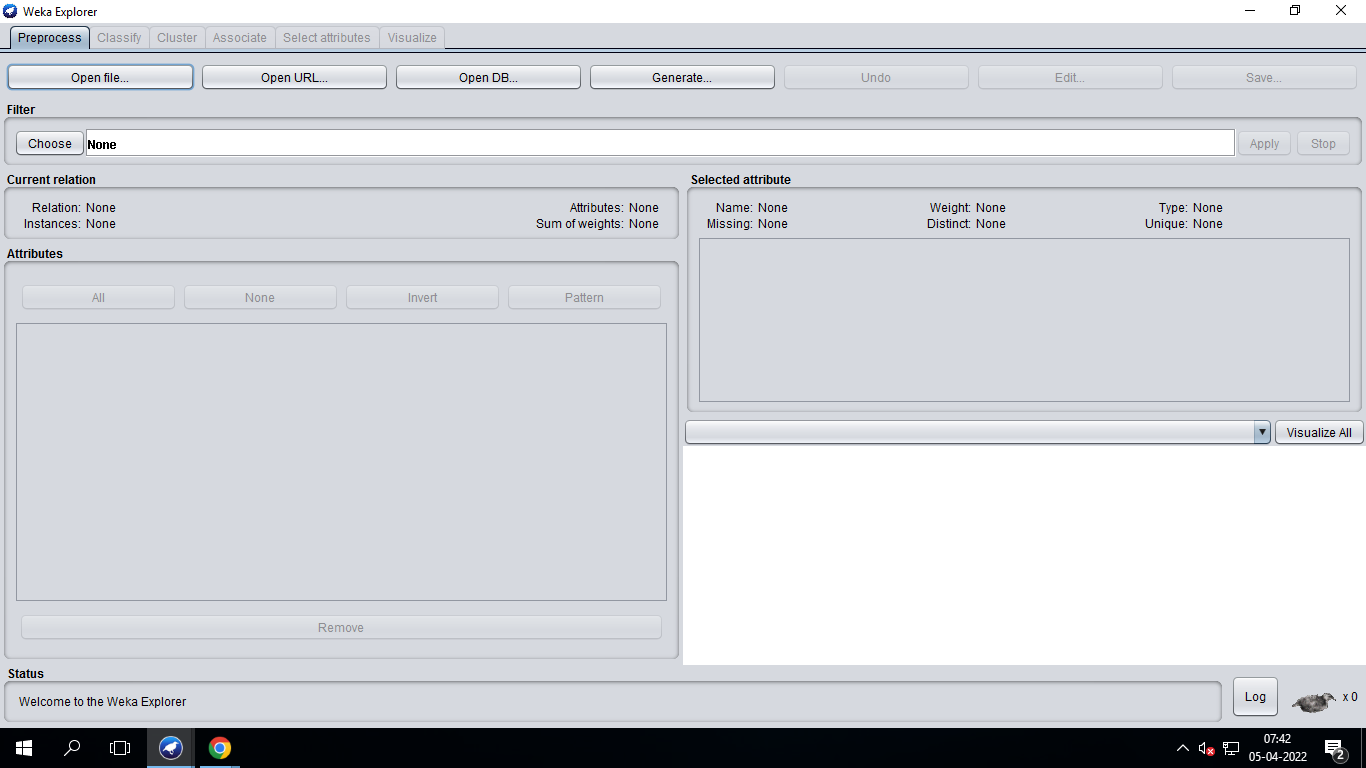
}

Output:

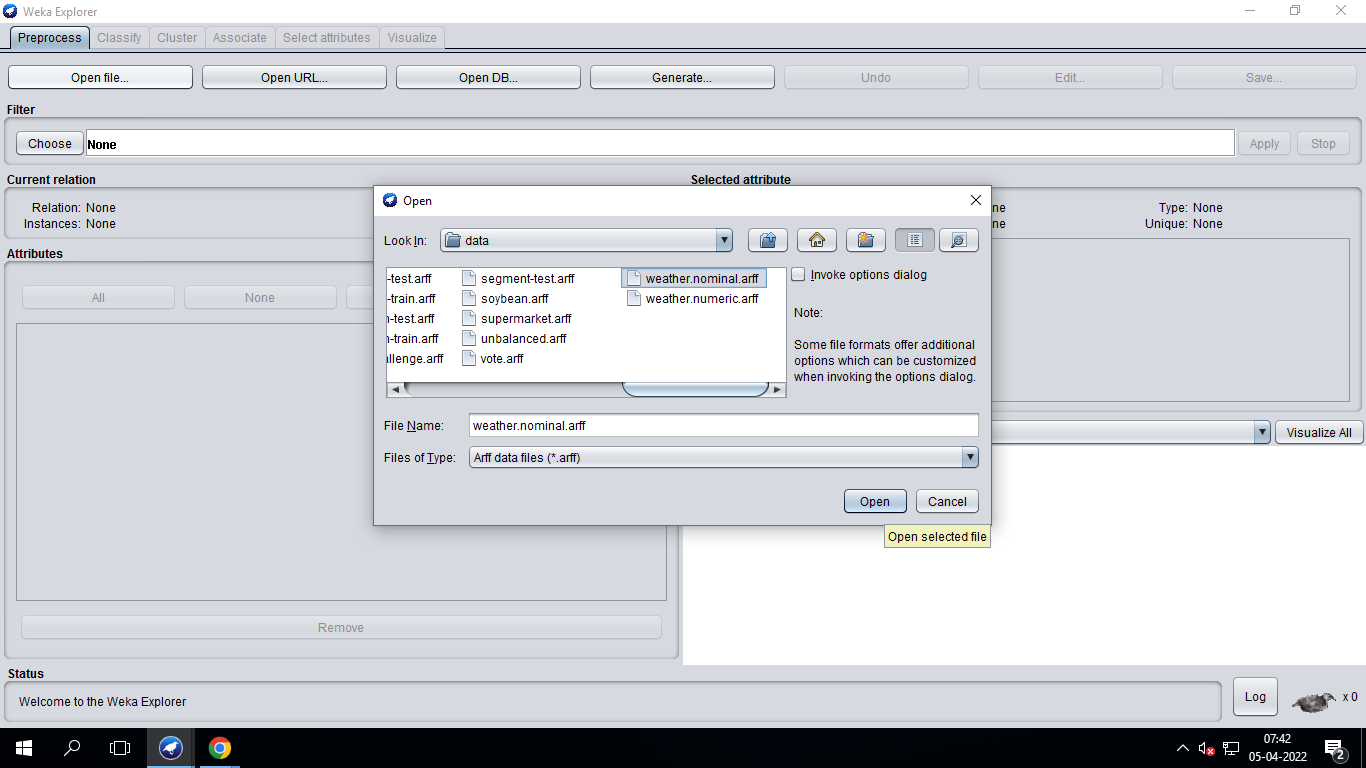


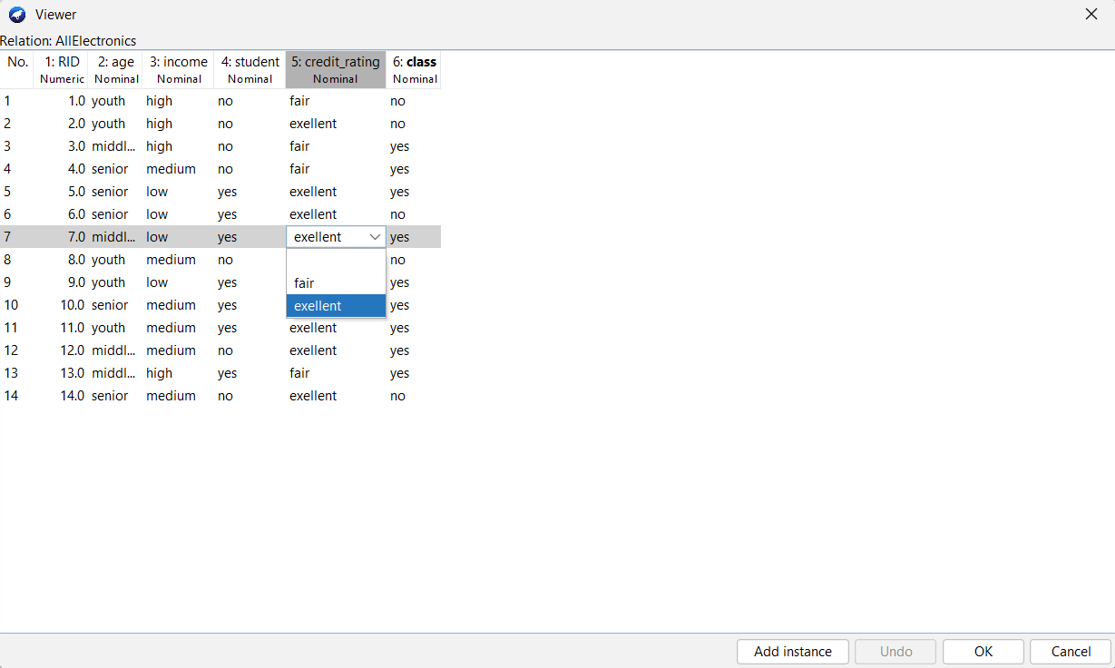
**Practical 8**

**Aim: Introducing WEKA for Data Mining. Perform steps to add and edit in weka.**

****

* Open Weka Tool.Above given picture is home screen of weka tool,where different functionality given to perform data minig on data set.

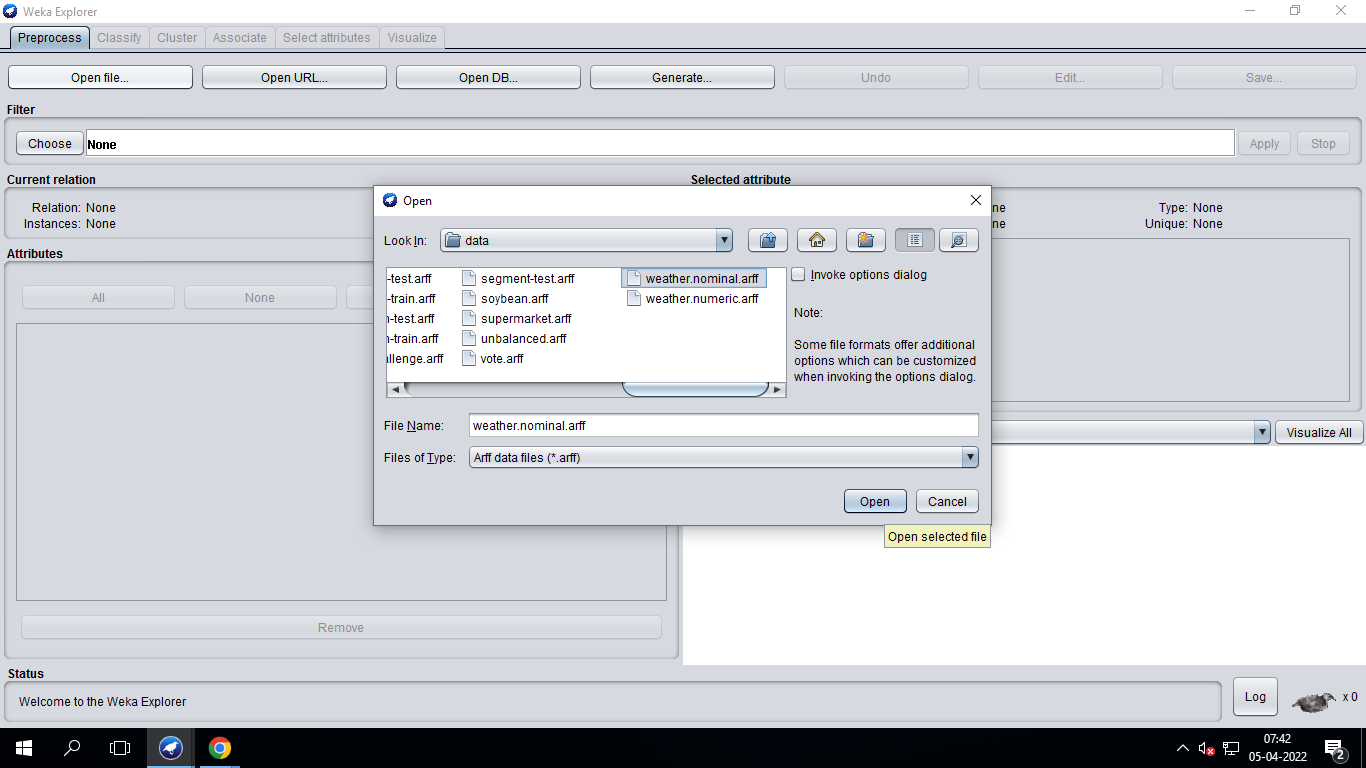
****

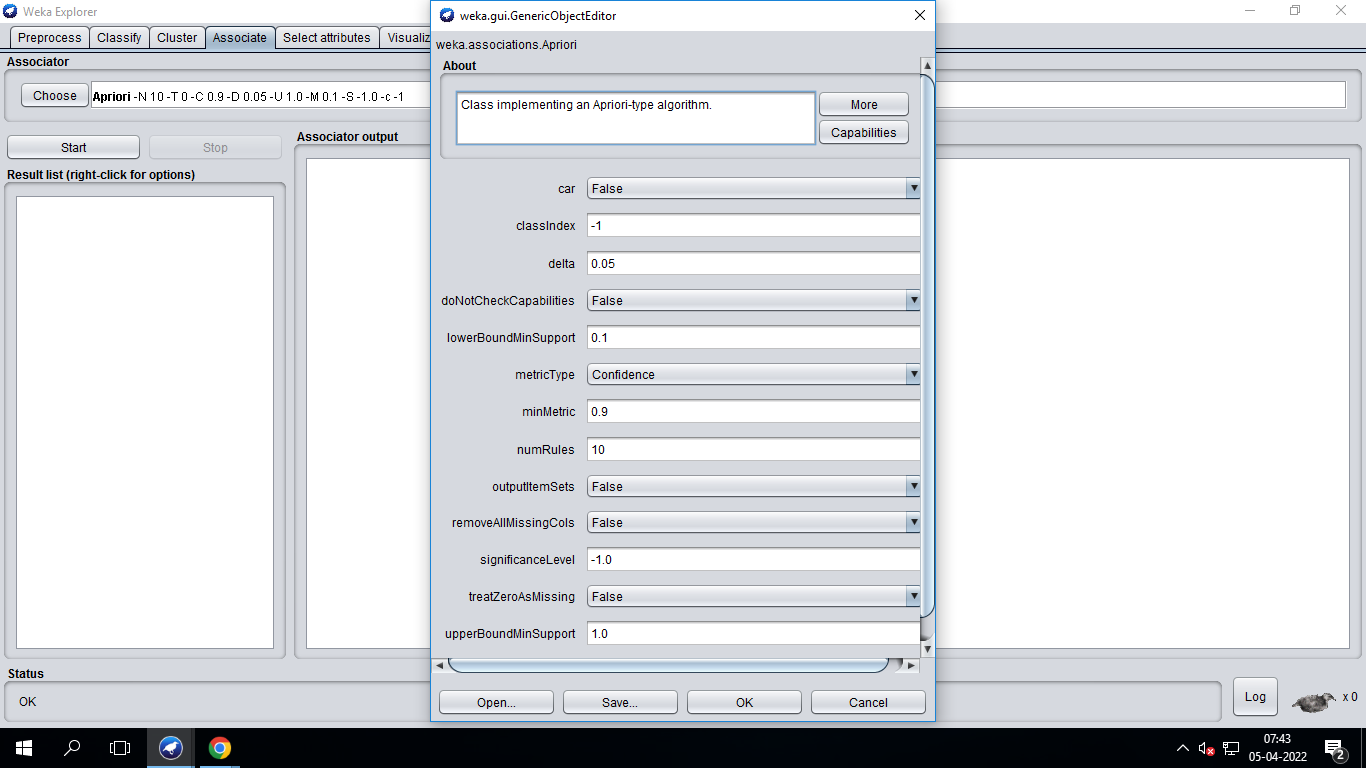
* To open data set in weka tool click on Open File button and then above given dialog box will open from that you choose data set.
* Weka provide some in build data set.You can choose your data set also but it must be in .arff formate.
* To edit data set, choose edit button from home page. It will allow you to edit data in data set.
* To Edit dataset in WEKA click on Edit button.
* Now you will see a popup in which you can view/edit any value of any attribute
* 

**Practical 9**

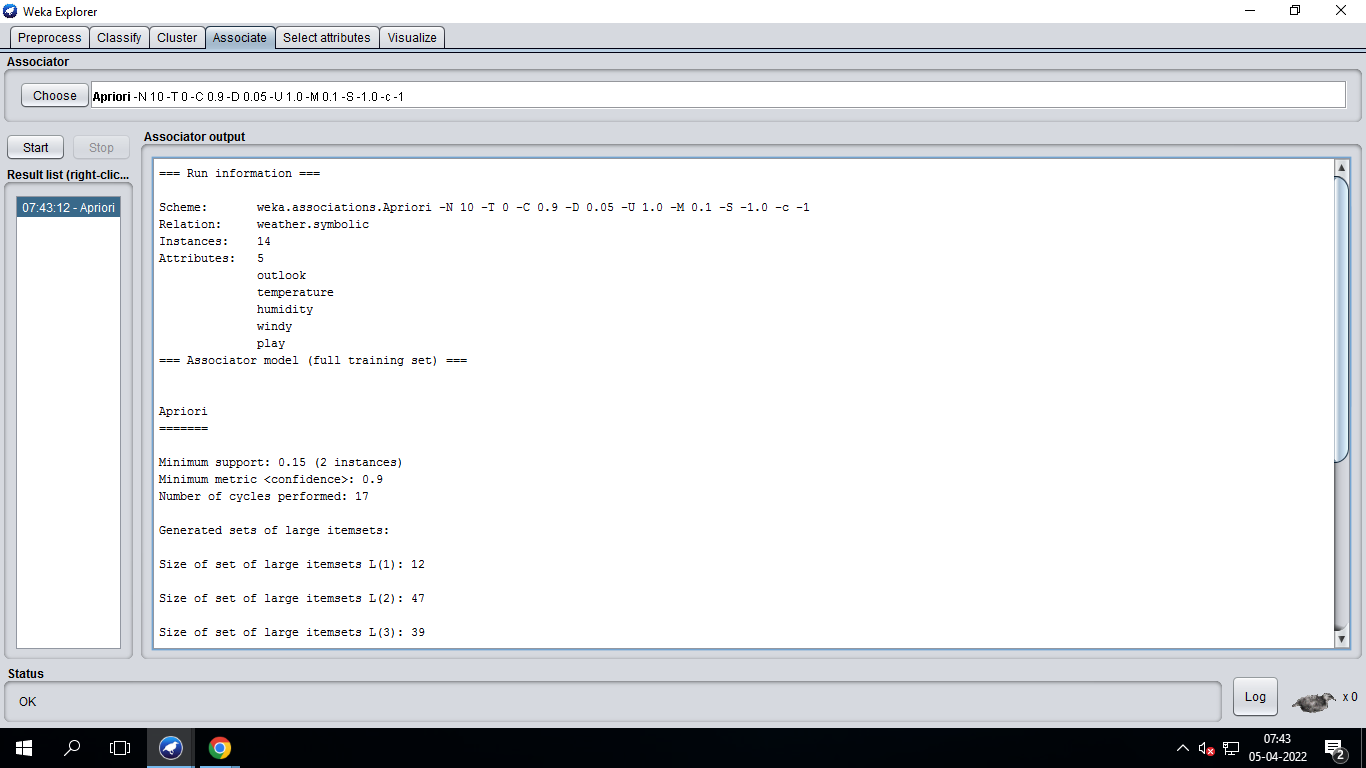
**Aim: To perform various algorithm of Association Mining using mining tool WEKA.Apriori Algorithm.**

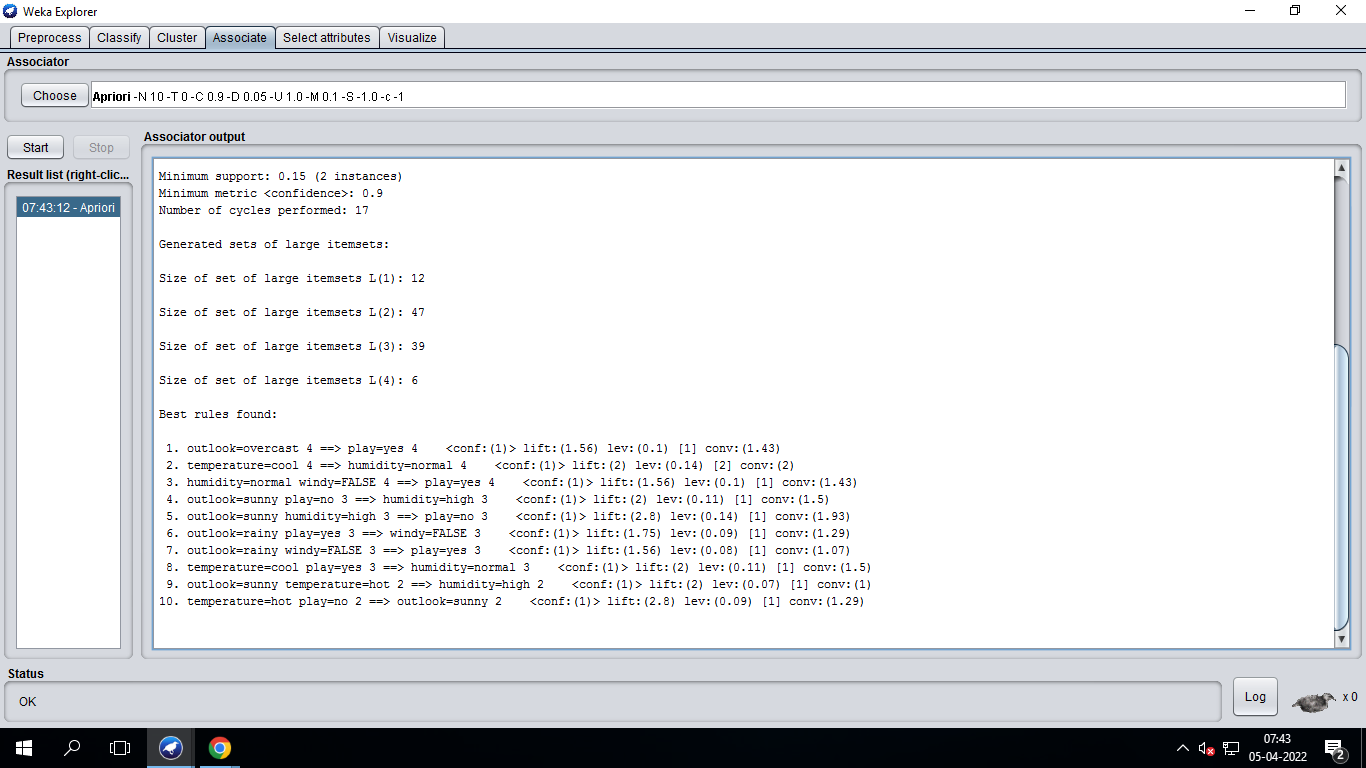
* First of all select the dataset you want to perform association mining. Here we will select weather nominal.arff dataset
* Generally file format for data set in WEKA tool is .arff and .csv.





* After choosing Data set, Associate Button is enable.click on that button.
* After that you will redirect to above given page, from that choose which algorithm you want to perform on data set.
* We select Apriori algorithm to perform data mining. Also you can set Support and Confidence.
* To perform mining task click on Start button.
* After clicking Start button mining result will show in Associator output screen.

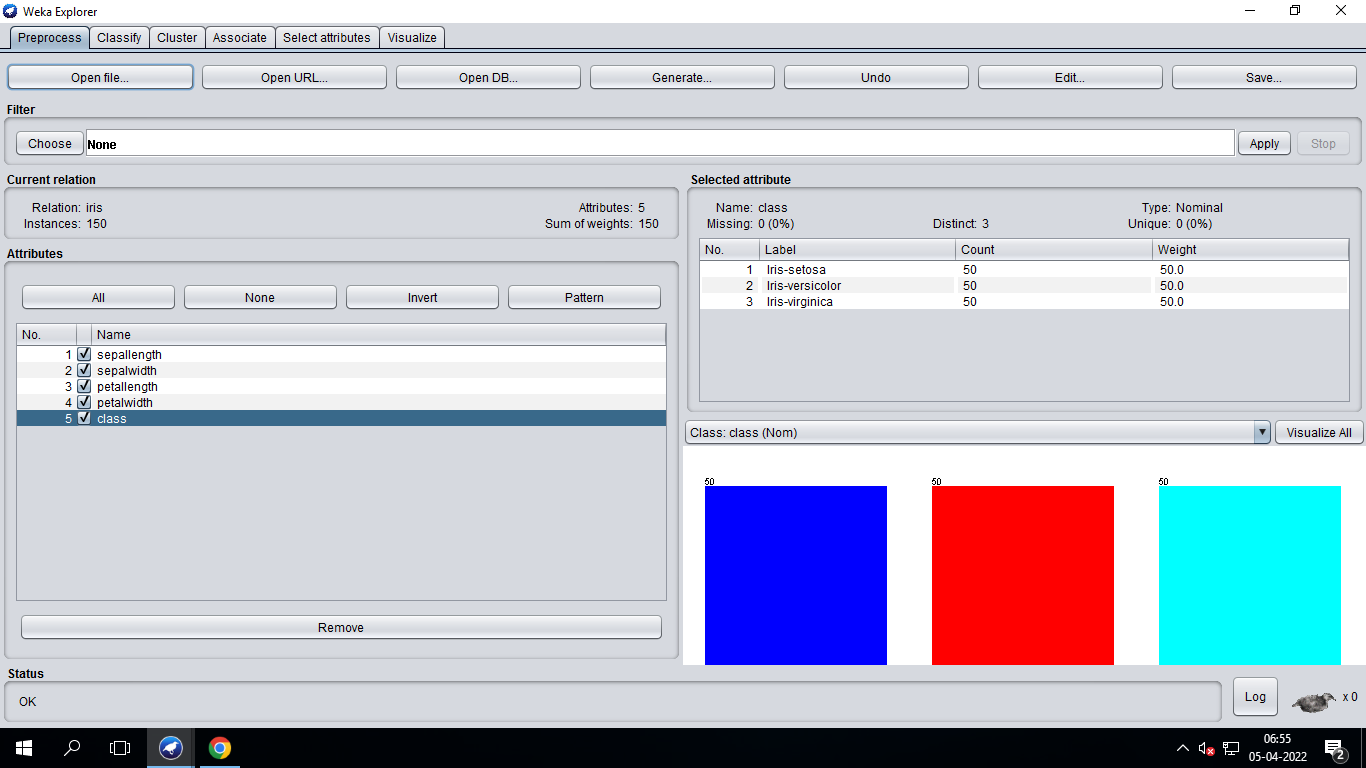




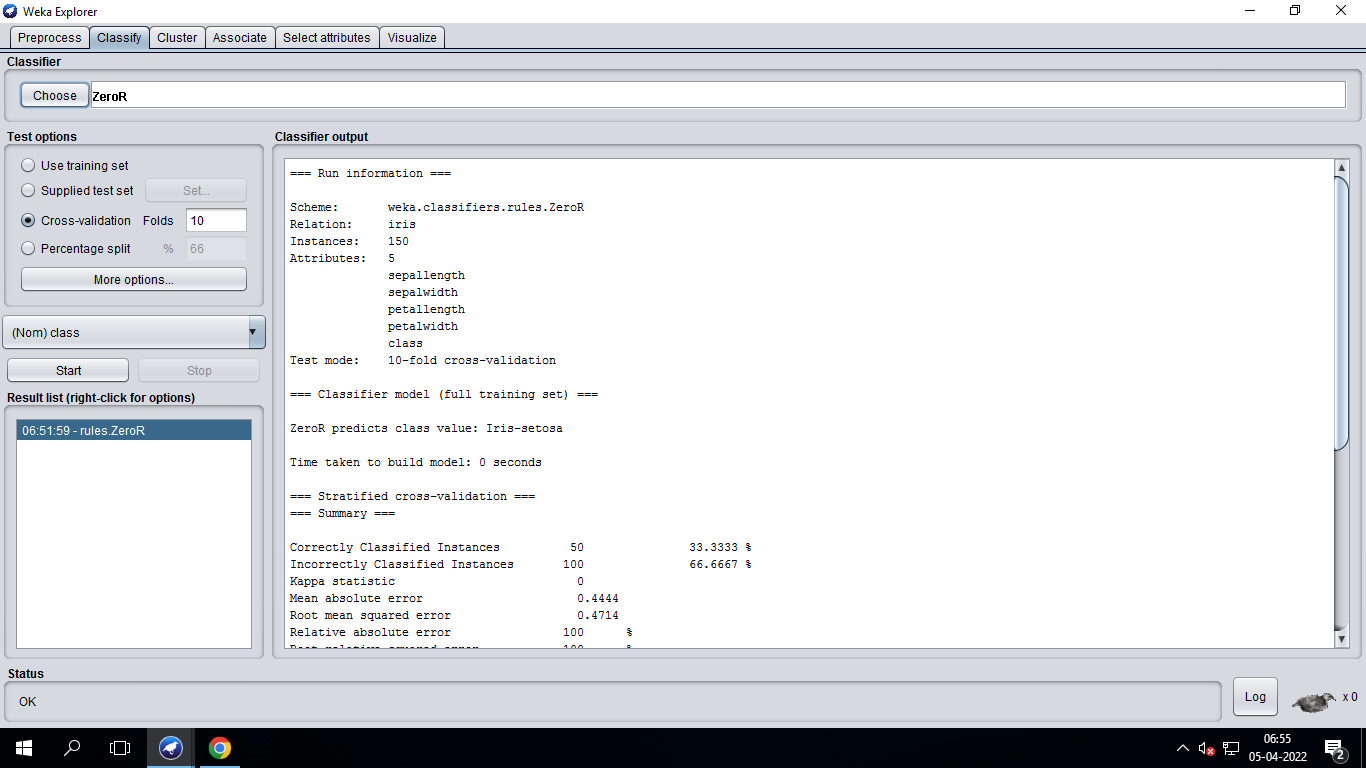
**Practical 10**

**Aim: To perform various algorithm of Classification Mining using mining tool WEKA.**

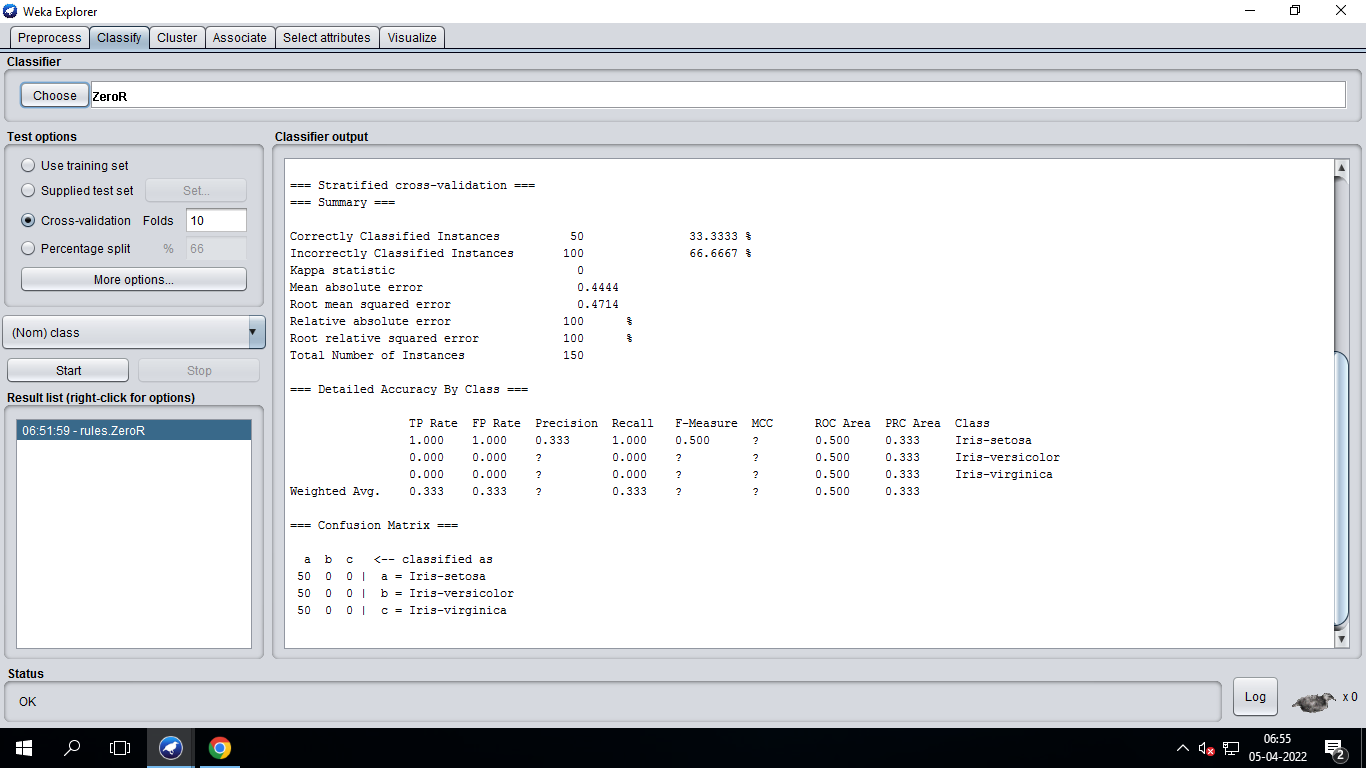
**Bayesian Classification.**



* To perform Classification select Classify button. You will redirect to above given page.

****

* Choose Algorithm from choose menu.
* Then click start button.
* After that you will see result as given above.

****

* You also can visualize your output by **visualize tree**

