

1. Write a JDBC program to make JDBC connections.

DATABASE:

```
CREATE TABLE s(id INTEGER,id1 INTEGER,id2 INTEGER);
SELECT * FROM s;
```

	id integer	id1 integer	id2 integer

JAVA:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class jdbc {
    public static void main(String args[]){
        Connection c=null;
        Statement stmt=null;
        try{
            Class.forName("org.postgresql.Driver");
c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgre
s","cse");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");
            stmt=c.createStatement();
            String sql="Insert into s values(1,2,3);";
            stmt.executeUpdate(sql);
            stmt.close();
            c.commit();
            c.close();
        }
        catch(Exception e){
            System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
            System.exit(0);
        }
        System.out.println("Record executed successfully");
    }
}
```

OUTPUT IN JAVA:

```
Opened database successfully
Record executed successfully
```

OUTPUT IN DATABASE:

	id integer	id1 integer	id2 integer
1	1	2	3

2. Write a JDBC program to implement ETL for names.

DATABASE:

```
create table firstname(id integer primary key,name varchar(50));
insert into firstname values(1,'Kanakalatha'),(2,'Tejaswini'),(3,'Vishruta'),
(4,'Akshara'),(5,'Likitha'),(6,'Nikhila'),(7,'Sindhuja'),(8,'Sravya'),
(9,'Neeharika'),(10,'Yashasvi');
select * from firstname;
```

	id integer	name character varying(50)
1	1	Kanakalatha
2	2	Tejaswini
3	3	Vishruta
4	4	Akshara
5	5	Likitha
6	6	Nikhila
7	7	Sindhuja
8	8	Sravya
9	9	Neeharika
10	10	Yashasvi

```
create table lastname(id integer primary key,name varchar(50));
insert into lastname values (1,'Vemuru'),(2,'Pemmaraju'),(3,'Dochibotla'),
(4,'Kappaganti'),(5,'Tallada'),(6,'Surineni'),(7,'Alugadda'),(8,'Bhallamudi'),
(9,'Bukhya'),(10,'Nancherla');
select * from lastname;
```

	id integer	name character varying(50)
1	1	Vemuru
2	2	Pemmaraju
3	3	Dochibotla
4	4	Kappaganti
5	5	Tallada
6	6	Surineni
7	7	Alugadda
8	8	Bhallamudi
9	9	Bukhya
10	10	Nancherla

```
create table fullname(id integer primary key,name varchar(50));
select * from fullname;
```

	id integer	name character varying(50)

JAVA:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class etl {
    public static void main(String args[]){
        Connection c=null;
        Statement stmt=null;
        Statement stmt1=null;
        Statement stmt2=null;
        String a;
        String b;
        String d;

        try{
            Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/names","postgres"
,"cse");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");
            stmt=c.createStatement();
            String sql="select * from firstname";
            ResultSet rs;
            rs=stmt.executeQuery(sql);
            while(rs.next()){
                stmt1=c.createStatement();
                stmt2=c.createStatement();
                ResultSet rs1;
                int i=rs.getInt("id");
                String sql1="select * from lastname where id="+Integer.toString(i)+" ";
                rs1=stmt1.executeQuery(sql1);
                rs1.next();
                a=rs.getString("name");
                b=rs1.getString("name");
                d=a+" "+b;
                String sql2="insert into fullname values("+Integer.toString(i)+",'"+d+"' )";
                stmt2.executeUpdate(sql2);
                rs1.close();
                stmt1.close();
            }
            stmt.close();
            stmt1.close();
            stmt2.close();
            c.commit();
            c.close();
        }
        catch(Exception e){
            System.err.println(e.getClass().getName()+": "+e.getMessage()+"!");
            System.exit(0);
        }
        System.out.println("Record executed successfully");
    }
}

```

OUTPUT IN JAVA:

Opened database successfully
Record executed successfully

OUTPUT IN DATABASE:

```
select * from fullname;
```

	id integer	name character varying(50)
1	1	Kanakalatha Vemuru
2	2	Tejaswini Pemmaraju
3	3	Vishruta Dochibotla
4	4	Akshara Kappaganti
5	5	Likitha Tallada
6	6	Nikhila Surineni
7	7	Sindhuja Alugadda
8	8	Sravva Bhallamudi
9	9	Neeharika Bukhya
10	10	Yashasvi Nancherla

3. Write a JDBC program to implement ETL for bmi.

DATABASE:

```
CREATE TABLE weight(id int, weight int);
```

```
INSERT INTO weight VALUES (1, 70), (2, 75), (3, 55), (4, 30), (5, 60), (6, 50),  
(7, 83), (8, 55), (9, 100), (10, 46);
```

```
SELECT * FROM weight;
```

	id integer	weight real
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

```
CREATE TABLE height(id int, height int);
```

```
INSERT INTO height VALUES (1, 150), (2, 157), (3, 155), (4, 130), (5, 160), (6,  
150), (7, 183), (8, 155), (9, 150), (10, 146);
```

```
SELECT * FROM height;
```

	id integer	height integer
1	1	150
2	2	157
3	3	155
4	4	130
5	5	160
6	6	150
7	7	183
8	8	155
9	9	150
10	10	146

JAVA:

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;
```

```
public class bmi {  
    public static void main(String args[]){  
        Connection c=null;  
        Statement stmt=null;  
        Statement stmt1=null;  
        Statement stmt2=null;  
        int a;  
        int b;  
        float d;  
  
        try{  
            Class.forName("org.postgresql.Driver");  
  
c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgre  
s","cse");  
            c.setAutoCommit(false);  
            System.out.println("Opened database successfully");  
            stmt=c.createStatement();  
            String sql="select * from weight";  
            ResultSet rs;  
            rs=stmt.executeQuery(sql);  
            stmt2=c.createStatement();  
            String sql2="create table bmi(id int, bmi float);";  
            stmt2.executeUpdate(sql2);  
            while(rs.next()){  
                stmt1=c.createStatement();  
                stmt2=c.createStatement();  
                ResultSet rs1;  
                int i=rs.getInt("id");  
                String sql1="select * from height where id="+Integer.toString(i)+";";
```

```
rs1=stmt1.executeQuery(sql1);
rs1.next();
a=rs.getInt("weight");
b=rs1.getInt("height");
d=(float)(a*10000)/(b*b);
sql2="insert into bmi values("+Integer.toString(i)+","+d+");";
stmt2.executeUpdate(sql2);
rs1.close();
stmt1.close();
}
stmt.close();
stmt1.close();
stmt2.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}
```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from bmi;

	id integer	bmi double precision
1	1	31.11111
2	2	30.427198
3	3	22.89282
4	4	17.75148
5	5	23.4375
6	6	22.222221
7	7	24.784258
8	8	22.89282
9	9	44.444443
10	10	21.580034

4. Write a JDBC program to do data cleaning.

DATABASE:

```
CREATE TABLE student(id int, weight real, height real);
```

```
INSERT INTO student(id, weight) VALUES (1, 70), (3, 55), (5, 60), (7, 83), (9, 100);
```

```
INSERT INTO student(id, height) VALUES (2, 157), (4, 130), (6, 150), (8, 155), (10, 146);
```

```
INSERT INTO student(id) VALUES (11);
```

```
SELECT * FROM student;
```

	id integer	weight real	height real
1	1	70	
2	2		157
3	3	55	
4	4		130
5	5	60	
6	6		150
7	7	83	
8	8		155
9	9	100	
10	10		146
11	11		

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.lang.Math;
```

```
public class bmi {
```

```
    public static void main(String args[]){
```

```
        Connection c=null;
```

```
        Statement stmt=null;
```

```
        Statement stmt2=null;
```

```
        Double h_mean, w_mean;

        int a,b,d;

try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select avg(weight) from student;";

ResultSet rs;

ResultSet rs1;

stmt2=c.createStatement();

rs1=stmt.executeQuery(sql);

rs1.next();

w_mean = rs1.getDouble(1);

sql = "select avg(height) from student;";

rs1=stmt.executeQuery(sql);

rs1.next();

h_mean = rs1.getDouble(1);

sql = "select * from student;";

rs=stmt.executeQuery(sql);

while(rs.next()){
a=rs.getInt("weight");

b=rs.getInt("id");

d=rs.getInt("height");

if(a==0)

{

        String sql2= "update student set weight="+w_mean+"where id="+b+";";
```

```
        stmt2.executeUpdate(sql2);
    }
    if(d==0)
    {
        String sql2= "update student set height="+h_mean+"where id="+b+"";
        stmt2.executeUpdate(sql2);
    }
}
stmt.close();
stmt2.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}
```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from student;

	id integer	weight real	height real
1	1	70	147.6
2	3	55	147.6
3	5	60	147.6
4	7	83	147.6
5	9	100	147.6
6	2	73.6	157
7	4	73.6	130
8	6	73.6	150
9	8	73.6	155
10	10	73.6	146
11	11	73.6	147.6

5. Write a JDBC program to implement data integration.

DATABASE:

```
create table student(id integer primary key,name varchar(50), gender
varchar(10));
```

```
insert into student values(1,'a', 'male'),(2,'e', 'female'),(3,'s', 'male'),
(4,'k', 'female'),(5,'i', 'male'),(6,'h', 'female');
```

```
select * from student;
```

	id integer	name character varying(50)	gender character varying(10)
1	1	a	male
2	2	e	female
3	3	s	male
4	4	k	female
5	5	i	male
6	6	h	female

```
create table hostlers(id integer primary key,name varchar(50), gender int);
```

```
insert into hostlers values(1,'ab', 0),(2,'y', 1),(3,'z', 1),(4,'q', 1),
(5,'p',0),
(6,'b', 0);
```

```
select * from hostlers;
```

	id integer	name character varying(50)	gender integer
1	1	ab	0
2	2	y	1
3	3	z	1
4	4	q	1
5	5	p	0
6	6	b	0

JAVA:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.lang.Math;

public class bmi {
    public static void main(String args[]){
        Connection c=null;
        Statement stmt=null;
        Statement stmt2=null;
```

```

        Double h_mean, w_mean;
        int a,b,d;
        String name,gen;

try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);
System.out.println("Opened database successfully");
stmt=c.createStatement();
String sql="create table names(id int, name varchar(50), gender
varchar(10));";
stmt.executeUpdate(sql);
sql="select * from student;";
ResultSet rs1;
stmt2=c.createStatement();
rs1=stmt.executeQuery(sql);

while(rs1.next())
{
name = rs1.getString("name");
gen = rs1.getString("gender");
if(gen.equals("female")){
String sql2 = "insert into names values      (" +rs1.getInt("id")
+", '"+name+"', '"+'f')";
stmt2.executeUpdate(sql2);
}
else{
String sql2 = "insert into names values (" +rs1.getInt("id")
+", '"+name+"', '"+'m')";
stmt2.executeUpdate(sql2);
}
}
sql="select * from hostlers;";
stmt2=c.createStatement();
rs1=stmt.executeQuery(sql);
while(rs1.next())
{
name = rs1.getString("name");
int gen1 = rs1.getInt("gender");
if(gen1==0){
String sql2 = "insert into names values (" +rs1.getInt("id")
+", '"+name+"', '"+'f')";
stmt2.executeUpdate(sql2);
}
else{
String sql2 = "insert into names values (" +rs1.getInt("id")
+", '"+name+"', '"+'m')";
stmt2.executeUpdate(sql2);
}
}
}
stmt.close();

```

```

stmt2.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from names;

	id integer	name character varying(50)	gender character varying(10)
1	1	a	m
2	2	e	f
3	3	s	m
4	4	k	f
5	5	i	m
6	6	h	f
7	1	ab	f
8	2	y	m
9	3	z	m
10	4	q	m
11	5	p	f
12	6	b	f

6. Write a JDBC program to implement min max normalization.

DATABASE:

```
CREATE TABLE weight(id int, weight int);
```

```
INSERT INTO weight VALUES (1, 70), (2, 75), (3, 55), (4, 30), (5, 60), (6, 50),  
(7, 83), (8, 55), (9, 100), (10, 46);
```

```
SELECT * FROM weight;
```

	id integer	weight real
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

JAVA:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class bmi {

    public static void main(String args[]){

        Connection c=null;

        Statement stmt=null;

        Statement stmt2=null;

        int min;

        int max,a,b;

        float d;
```



```
try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select * from weight";

ResultSet rs;

ResultSet rs2;

stmt2=c.createStatement();

String sql2="select min(weight) from weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

min=rs2.getInt(1);

sql2="select max(weight) from weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

max=rs2.getInt(1);

rs2=stmt2.executeQuery(sql2);

sql2="ALTER TABLE weight ADD min_max_norm real;";

stmt2.executeUpdate(sql2);

rs=stmt.executeQuery(sql);

while(rs.next()){

    stmt2=c.createStatement();

    a=rs.getInt("weight");

    b=rs.getInt("id");

    d=(float)(a-min)/(max-min);

    System.out.println(d);
```

```

        sql2="update weight set min_max_norm = "+d+" where id = "+b+"";
        stmt2.executeUpdate(sql2);

    }

    stmt.close();
    stmt2.close();
    c.commit();
    c.close();
}

catch(Exception e){
    System.err.println(e.getClass().getName()+": "+e.getMessage()+"!");
    System.exit(0);
}

System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from weight;

	id integer	weight integer	min_max_norm real
1	1	70	0.571429
2	2	75	0.642857
3	3	55	0.357143
4	4	30	0
5	5	60	0.428571
6	6	50	0.285714
7	7	83	0.757143
8	8	55	0.357143
9	9	100	1
10	10	46	0.228571

7. Write a JDBC program to implement z-score normalization.

DATABASE:

```
CREATE TABLE weight(id int, weight int);
```

```
INSERT INTO weight VALUES (1, 70), (2, 75), (3, 55), (4, 30), (5, 60), (6, 50),  
(7, 83), (8, 55), (9, 100), (10, 46);
```

```
SELECT * FROM weight;
```

	id integer	weight real
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.lang.Math;
```

```
public class bmi {
```

```
    public static void main(String args[]){
```

```
        Connection c=null;
```

```
        Statement stmt=null;
```

```
        Statement stmt2=null;
```

```
        float mean;
```

```
        int a,b;
```

```
double d,std;

try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select * from weight";

ResultSet rs;

ResultSet rs2;

stmt2=c.createStatement();

String sql2="select sum(weight) from weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

mean=rs2.getInt(1);

sql2="select count(weight) from weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

a=rs2.getInt(1);

mean = mean/a;

sql2="select sum(pow(weight-"+mean+", 2)) FROM weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

std=rs2.getFloat(1);

rs2=stmt2.executeQuery(sql2);

sql2="ALTER TABLE weight ADD z_score real;";

stmt2.executeUpdate(sql2);
```

```
std = Math.sqrt(std/a);
rs=stmt.executeQuery(sql);
while(rs.next()){
    stmt2=c.createStatement();
    a=rs.getInt("weight");
    b=rs.getInt("id");
    d=(float)(a-mean)/(std);
    sql2="update weight set z_score = "+d+" where id = "+b+"";
    stmt2.executeUpdate(sql2);
}
stmt.close();
stmt2.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+": "+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}
```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from weight;

	id integer	weight integer	z_score real
1	1	70	0.398217
2	2	75	0.660202
3	3	55	-0.387738
4	4	30	-1.69766
5	5	60	-0.125753
6	6	50	-0.649723
7	7	83	1.07938
8	8	55	-0.387738
9	9	100	1.97013
10	10	46	-0.85931

8. Write a JDBC program to implement decimal scaling.

DATABASE:

```
CREATE TABLE weight(id int, weight int);
```

```
INSERT INTO weight VALUES (1, 70), (2, 75), (3, 55), (4, 30), (5, 60), (6, 50),  
(7, 83), (8, 55), (9, 100), (10, 46);
```

```
SELECT * FROM weight;
```

	id integer	weight real
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.lang.Math;
```

```
public class bmi {
```

```
    public static void main(String args[]){
```

```
        Connection c=null;
```

```
        Statement stmt=null;
```

```
        Statement stmt2=null;
```

```
        int max;
```

```
        int a,b;
```

```
        int val;

        double d;

try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select * from weight";

ResultSet rs;

ResultSet rs2;

stmt2=c.createStatement();

String sql2="select max(weight) from weight;";

rs2=stmt2.executeQuery(sql2);

rs2.next();

max=rs2.getInt(1);

val = (max+"").length();

sql2="ALTER TABLE weight ADD d_scaling real;";

stmt2.executeUpdate(sql2);

rs=stmt.executeQuery(sql);

while(rs.next()){

    stmt2=c.createStatement();

    a=rs.getInt("weight");

    b=rs.getInt("id");

    d=(double)a/Math.pow(10,val);

    sql2="update weight set d_scaling = "+d+" where id = "+b+"";

    stmt2.executeUpdate(sql2);
```



```

    }

    stmt.close();

    stmt2.close();

    c.commit();

    c.close();

    }

    catch(Exception e){

        System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");

        System.exit(0);

    }

    System.out.println("Record executed successfully");

}

}

```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from weight;

	id integer	weight integer	z_score real	d_scaling real
1	1	70	0.07	
2	2	75	0.075	
3	3	55	0.055	
4	4	30	0.03	
5	5	60	0.06	
6	6	50	0.05	
7	7	83	0.083	
8	8	55	0.055	
9	9	100	0.1	
10	10	46	0.046	

9. Write a JDBC program to implement horizontal partitioning.

DATABASE:

```
CREATE TABLE weight(id int, weight int);
```

```
INSERT INTO weight VALUES (1, 70), (2, 75), (3, 55), (4, 30), (5, 60), (6, 50),  
(7, 83), (8, 55), (9, 100), (10, 46);
```

```
SELECT * FROM weight;
```

	id integer	weight real
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.lang.Math;
```

```
public class bmi {
```

```
    public static void main(String args[]){
```

```
        Connection c=null;
```

```
        Statement stmt=null;
```

```
        Statement stmt2=null;
```

```
        int a,b;
```

```
try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select * from weight";

ResultSet rs;

ResultSet rs2;

stmt2=c.createStatement();

String sql2="create table under_weight(id int, weight int);";

stmt2.executeUpdate(sql2);

sql2="create table over_weight(id int, weight int);";

stmt2.executeUpdate(sql2);

rs=stmt.executeQuery(sql);

while(rs.next()){

    a=rs.getInt("weight");

    b=rs.getInt("id");

    if(a<=60){

        sql2="insert into under_weight values("+b+", "+a+");";

        stmt2.executeUpdate(sql2);

    }

    else{

        sql2="insert into over_weight values("+b+", "+a+");";

        stmt2.executeUpdate(sql2);

    }

}

stmt.close();
```

```

    stmt2.close();
    c.commit();
    c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from under_weight;

	id integer	weight integer
1	3	55
2	4	30
3	5	60
4	6	50
5	8	55
6	10	46

Select * from over_weight;

	id integer	weight integer
1	1	70
2	2	75
3	7	83
4	9	100

10. Write a JDBC program to implement vertical partitioning.

DATABASE:

```
CREATE TABLE student(id int, weight int, height int);
```

```
INSERT INTO student VALUES (1, 70, 150), (2, 75, 157), (3, 55, 155), (4, 30, 130), (5, 60, 160), (6, 50, 150), (7, 83, 183), (8, 55, 155), (9, 100, 150), (10, 46, 146);
```

```
SELECT * FROM student;
```

	id integer	weight integer	height integer
1	1	70	150
2	2	75	157
3	3	55	155
4	4	30	130
5	5	60	160
6	6	50	150
7	7	83	183
8	8	55	155
9	9	100	150
10	10	46	146

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.lang.Math;
```

```
public class bmi {
```

```
    public static void main(String args[]){
```

```
        Connection c=null;
```

```
        Statement stmt=null;
```

```
        Statement stmt2=null;
```

```
        int a,b,d;

try{
Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","po
stgres","cse");

c.setAutoCommit(false);

System.out.println("Opened database successfully");

stmt=c.createStatement();

String sql="select * from student";

ResultSet rs;

ResultSet rs2;

stmt2=c.createStatement();

String sql2="create table weight(id int, weight int);";

stmt2.executeUpdate(sql2);

sql2="create table height(id int, height int);";

stmt2.executeUpdate(sql2);

rs=stmt.executeQuery(sql);

while(rs.next()){

    a=rs.getInt("weight");

    b=rs.getInt("id");

    d=rs.getInt("height");

    sql2="insert into weight values("+b+", "+a+")";

    stmt2.executeUpdate(sql2);

    sql2="insert into height values("+b+", "+d+")";

    stmt2.executeUpdate(sql2);

}

stmt.close();

stmt2.close();
```

```

        c.commit();
        c.close();
    }
    catch(Exception e){
        System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
        System.exit(0);
    }
    System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened database successfully

Record executed successfully

OUTPUT IN DATABASE:

Select * from weight;

	id integer	weight integer
1	1	70
2	2	75
3	3	55
4	4	30
5	5	60
6	6	50
7	7	83
8	8	55
9	9	100
10	10	46

Select * from height;

	id integer	height integer
1	1	150
2	2	157
3	3	155
4	4	130
5	5	160
6	6	150
7	7	183
8	8	155
9	9	150
10	10	146

11. Write a JDBC program to implement apriori algorithm.

DATABASE:

```
CREATE TABLE transactions(id int, item1 int, item2 int, item3 int, item4 int,
item5 int, item6 int);
```

```
INSERT INTO transactions VALUES (1,1,0,1,0,1,1), (2,1,0,0,0,1,0),
(3,1,1,0,1,0,1), (4,0,1,1,0,0,1), (5,1,0,0,1,1,1), (6,1,0,0,1,0,1),
(7,1,1,1,1,1,1);
```

```
SELECT * FROM transactions;
```

	id integer	item1 integer	item2 integer	item3 integer	item4 integer	item5 integer	item6 integer
1	1	1	0	1	0	1	1
2	2	1	0	0	0	1	0
3	3	1	1	0	1	0	1
4	4	0	1	1	0	0	1
5	5	1	0	0	1	1	1
6	6	1	0	0	1	0	1
7	7	1	1	1	1	1	1

JAVA:

```
package javaapplication3;

import java.sql.*;
import java.util.*;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Vector;
```



```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class NewClass6 {
    static Vector<String> transactions=new Vector();
    static Vector<String> rec=new Vector();
    static int[] bitSet;
    static Vector<String> vec=new Vector();
    static Vector<String> rec1=new Vector();
    static int len;
    static void createRec(){
        int l=len;
        String str="";
        for(int i=1;i<=len;i++){
            str="";
            for(int j=0;j<len;j++){
                if((i-1)==j)
                    str+='1';
                else
                    str+=".";
            }
            rec.add(str);
        }
    }
    static String add(String a,String b){
        String y="";
        if(a.length()!=b.length())
            return y;
        for(int i=0;i<a.length();i++){
```

```
    if(a.charAt(i)=='.' && b.charAt(i)=='.')
        y+='.';
    else
        y+='1';
    }
    return y;
}

static int common_elements(String a,String b){
    int count=0;
    if(a.length()!=b.length())
        return count;
    for(int i=0;i<a.length();i++){
        if(a.charAt(i)=='1' && b.charAt(i)=='1')
            count++;
    }
    return count;
}

static void join(int len){
    if(len==NewClass6.len)
        return;
    int n=rec.size();
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            int c=common_elements(rec.get(i),rec.get(j));
            if(c==(len-1)){
                String y;
                y=add(rec.get(i),rec.get(j));
                int t=0;
                for(t=0;t<vec.size();t++){
```

```

        if(vec.get(t).equals(y))
            break;
    }
    if(t==vec.size())
        vec.add(y);
    }
}
}
}
}

```

```

static void GenSubSet(int reqLen, int start, int[] check,int currLen,int len)
{
    if(start<len){
        if(currLen>reqLen)
            return;
        if (currLen == reqLen) {
            String str="";
            for (int i = 0; i < len; i++) {
                if (check[i] != 0)
                    str+='1';
                else
                    str+='.';
            }
            vec.add(str);
            return;
        }
        else if ((currLen+1) == reqLen&&(start+1)==len&&bitSet[start]!=0){
            String str="";
            for (int i = 0; i < len; i++) {
                if (check[i] != 0||i==start) {

```

```
        str+="1";
    }
    else
        str+=".";
    }
    vec.add(str);
    return;
}
if (start == len)
    return;
if(bitSet[start]==1){
    check[start] = 1;
    GenSubSet(reqLen, start + 1, check, currLen + 1, len);
}
check[start] = 0;
GenSubSet(reqLen, start + 1, check ,currLen, len);}
}

public static void main(String[] args){
    Connection c=null;
    Statement stmt=null;
    try{
        Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgre
s","cse");

        c.setAutoCommit(false);
        System.out.println("Opened Database succesfully");
        stmt=c.createStatement();
        String sql="SELECT * FROM transactions";
        ResultSet rs;
```

```
rs=stmt.executeQuery(sql);
ResultSetMetaData rsmd = rs.getMetaData();
int columnCount= rsmd.getColumnCount();
sql="SELECT count(*) FROM transactions";
rs=stmt.executeQuery(sql);
sql="SELECT * FROM transactions";
rs=stmt.executeQuery(sql);
len = columnCount-1;
while(rs.next()){
    String str="";
    rs.getInt(1);
    for(int y=1;y<columnCount;y++){
        int d=rs.getInt(y+1);
        if(d==1)
            str+='1';
        else
            str+='0';
    }
    transactions.add(str);
}
int start=0;
createRec();
int count2=1;
while(!rec.isEmpty()){
    Iterator value = rec.iterator();
    rec1.clear();
    while(value.hasNext())
        rec1.add((String)value.next());
    vec.clear();
```

```
join(count2);
count2++;
if(vec.isEmpty())
    break;
value = vec.iterator();
rec.clear();
while (value.hasNext()) {
    String y=(String)(value.next());
    Pattern pattern = Pattern.compile(y);
    int count1=0;
    for(int i=0;i<transactions.size();i++){
        Matcher matcher = pattern.matcher(transactions.get(i));
        if(matcher.find())
            count1++;
    }
    if(count1>=2)
        rec.add(y);
    }
}

Iterator value4 = rec1.iterator();
System.out.print("The frequent item sets generated are ");
while(value4.hasNext())
    System.out.println(value4.next());
System.out.println();
Iterator value1 = rec1.iterator();
int[] check=new int[len];
Dictionary geek = new Hashtable();
bitSet = new int[len];
while (value1.hasNext()) {
```

```
int total=0;
int reqLen=1;
String y=(String)(value1.next());
while(reqLen<(y.length()-1)){
    for(int u=0;u<y.length();u++){
        if(y.charAt(u)=='1')
            bitSet[u]=1;
        else
            bitSet[u]=0;
        check[u]=0;
    }
    vec.clear();
    GenSubSet(reqLen,0,check,0,len);
    reqLen++;
    Iterator value2 = vec.iterator();
    while(value2.hasNext()){
        String y1=(String)value2.next();
        Pattern pattern = Pattern.compile(y1);
        int count1=0;
        for(int i=0;i<transactions.size();i++){
            Matcher matcher = pattern.matcher(transactions.get(i));
            if(matcher.find())
                count1++;
        }
        geek.put(y1, count1);
        if(y1.equals(y))
            total = count1;
    }
}
```

```

Enumeration k = geek.keys();
for (Enumeration i = geek.elements(); i.hasMoreElements();) {
    System.out.print("Support for : [");
    y = (String)k.nextElement();
    for(int t=0;t<y.length();t++){
        if(y.charAt(t)=='1')
            System.out.print(t+",");
    }
    System.out.println("] is "+((total*100)/(int)i.nextElement())+"%");

}
System.out.println();
}
stmt.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+" : "+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened Database succesfully

The frequent item sets generated are 11.1.1

1.1.11

1..111

Support for : [0,1,3,5,] is 100%

Support for : [1,3,5,] is 100%

Support for : [0,1,3,] is 100%

Support for : [1,3,] is 100%

Support for : [0,3,5,] is 50%

Support for : [0,1,5,] is 100%

Support for : [3,5,] is 50%

Support for : [0,3,] is 50%

Support for : [1,5,] is 66%

Support for : [3,] is 50%

Support for : [0,1,] is 100%

Support for : [1,] is 66%

Support for : [0,5,] is 40%

Support for : [5,] is 33%

Support for : [0,] is 33%

Support for : [0,2,4,5,] is 100%

Support for : [1,5,] is 66%

Support for : [0,2,] is 100%

Support for : [0,1,3,5,] is 100%

Support for : [0,2,4,] is 100%

Support for : [1,] is 66%

Support for : [0,1,3,] is 100%

Support for : [5,] is 33%

Support for : [4,5,] is 66%

Support for : [0,3,5,] is 50%

Support for : [4,] is 50%

Support for : [0,3,] is 50%

Support for : [0,1,5,] is 100%

Support for : [0,1,] is 100%

Support for : [0,5,] is 40%

Support for : [0,4,5,] is 66%

Support for : [2,5,] is 66%

Support for : [0,] is 33%

Support for : [2,4,5,] is 100%

Support for : [0,4,] is 50%

Support for : [1,3,5,] is 100%

Support for : [2,] is 66%

Support for : [2,4,] is 100%

Support for : [1,3,] is 100%

Support for : [3,5,] is 50%

Support for : [3,] is 50%

Support for : [0,2,5,] is 100%

Support for : [0,2,4,5,] is 100%

Support for : [1,5,] is 66%

Support for : [0,2,] is 100%

Support for : [0,1,3,5,] is 100%

Support for : [0,2,4,] is 100%

Support for : [1,] is 66%

Support for : [0,1,3,] is 100%

Support for : [5,] is 33%

Support for : [4,5,] is 66%

Support for : [0,3,5,] is 50%

Support for : [0,3,4,5,] is 100%

Support for : [4,] is 50%

Support for : [0,3,] is 50%

Support for : [0,3,4,] is 100%

Support for : [0,1,5,] is 100%

Support for : [0,1,] is 100%

Support for : [0,5,] is 40%

Support for : [0,4,5,] is 66%

Support for : [2,5,] is 66%

Support for : [0,] is 33%

Support for : [2,4,5,] is 100%

Support for : [0,4,] is 50%

Support for : [1,3,5,] is 100%

Support for : [2,] is 66%

Support for : [2,4,] is 100%

Support for : [1,3,] is 100%

Support for : [3,5,] is 50%

Support for : [3,4,5,] is 100%

Support for : [3,] is 50%

Support for : [3,4,] is 100%

Support for : [0,2,5,] is 100%

Record executed successfully

12. Write a JDBC program to implement k-means algorithm.

DATABASE:

```
CREATE TABLE data(id int, x int, y int);
```

```
INSERT INTO data VALUES (1, 1, 1), (2, 3, 4), (3, 7, 8), (4, 5, 8), (5, 9, 9),  
(6, 8, 0), (7, 0, 4);
```

```
SELECT * FROM data;
```

	id integer	x integer	y integer
1	1	1	1
2	2	3	4
3	3	7	8
4	4	5	8
5	5	9	9
6	6	8	0
7	7	0	4

JAVA:

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import java.lang.Math;  
import java.sql.ResultSetMetaData;  
import java.sql.SQLException;  
import java.util.Random;
```

```
import java.util.Scanner;

public class bmi {
    public static void main(String args[]) throws SQLException{
        Connection c=null;
        Statement stmt=null;
        Statement stmt2=null;
        int a,b,d,k;
        ResultSet rs1,rs;

        try{
            Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgre
s","cse");

            c.setAutoCommit(false);
            System.out.println("Opened database successfully");
            stmt=c.createStatement();
            stmt2 = c.createStatement();
            String sql = "ALTER TABLE data ADD class int";
            stmt.executeUpdate(sql);
            System.out.print("Enter the number of clusters:");
            Scanner myObj = new Scanner(System.in);
            String u = myObj.nextLine();
            k = Integer.parseInt(u);
            double[][] means=new double[k][2];
            sql="SELECT count(*) FROM data;";
            rs=stmt.executeQuery(sql);
            rs.next();
            int n=rs.getInt(1);
```

```
Random rand = new Random();
for(int i=0;i<k;i++){
    int v;
    do{
        v = rand.nextInt(n+1);
    }while(v==0);
    String sql2 = "SELECT * FROM data WHERE id="+v+"";
    rs=stmt2.executeQuery(sql2);
    rs.next();
    a = rs.getInt("id");
    b = rs.getInt("x");
    d = rs.getInt("y");
    means[i][0]=b;
    means[i][1]=d;
}
int count=-1;
while(count!=0){
    count=0;
    sql="SELECT * FROM data";
    stmt2=c.createStatement();
    rs1=stmt.executeQuery(sql);
    while(rs1.next())
    {
        a = rs1.getInt("id");
        b = rs1.getInt("x");
        d = rs1.getInt("y");
        int Class = rs1.getInt("class");
        double min=-1;
```

```

        int mini = -1;

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

            if(min==-1||min>(Math.sqrt(Math.pow((means[i][0]-
b),2)+Math.pow((means[i][1]-d),2)))){

                min = (Math.sqrt(Math.pow((means[i][0]-
b),2)+Math.pow((means[i][1]-d),2)));

                mini = i;

            }

        }

        if(Class!=(mini+1)){

            String sql2 = "update data set class = "+(mini+1)+"where id =
"+a+"";

            stmt2.executeUpdate(sql2);

            count++;}

    }

    if(count==0)

        break;

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

        String sql2 = "select avg(x),avg(y) from data where class = "+i+"";

        rs=stmt2.executeQuery(sql2);

        rs.next();

        means[i][0]=rs.getDouble(1);

        means[i][1]=rs.getDouble(2);

    }

}

System.out.print("The means are: ");

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

    System.out.println("(" +means[i][0]+", "+means[i][1]+")");

}

stmt.close();

```

```

    stmt2.close();

    c.commit();

    c.close();

}

catch(Exception e){

    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");

    System.exit(0);

}

System.out.println("Record executed successfully");

}

}

```

OUTPUT IN JAVA:

Opened database successfully

Enter the number of clusters:2

The means are: (0.0, 0.0)

(4.5, 0.5)

Record executed successfully

OUTPUT IN DATABASE:

```
select * from data;
```

	id integer	x integer	y integer	class integer
1	1	1	1	2
2	2	3	4	2
3	3	7	8	1
4	4	5	8	1
5	5	9	9	1
6	6	8	0	1
7	7	0	4	2

13. Write a JDBC program to implement k-medoids algorithm.

DATABASE:

```
CREATE TABLE data(id int, x int, y int);
```

```
INSERT INTO data VALUES (1, 2, 6), (2, 3, 4), (3, 3, 8), (4, 4, 7), (5, 6, 2),  
(6, 6, 4), (7, 7, 3), (8, 7, 4), (9, 8, 5), (10, 7, 6);
```

```
SELECT * FROM data;
```

	id integer	x integer	y integer
1	1	2	6
2	2	3	4
3	3	3	8
4	4	4	7
5	5	6	2
6	6	6	4
7	7	7	3
8	8	7	4
9	9	8	5
10	10	7	6

JAVA:

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import java.lang.Math;  
import java.sql.ResultSetMetaData;
```

```
import java.sql.SQLException;
import java.util.Random;
import java.util.Scanner;

public class bmi {
    public static void main(String args[]) throws SQLException{
        Connection c=null;
        Statement stmt=null;
        Statement stmt2=null;
        int a,b,d,k;
        ResultSet rs1,rs;

        try{
            Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgres",
"s","cse");

            c.setAutoCommit(false);
            System.out.println("Opened database successfully");
            stmt=c.createStatement();
            stmt2 = c.createStatement();
            String sql = "ALTER TABLE data ADD class int";
            stmt.executeUpdate(sql);
            System.out.print("Enter the number of clusters:");
            Scanner myObj = new Scanner(System.in);
            String u = myObj.nextLine();
            k = Integer.parseInt(u);
            double[][] medoids=new double[k][2];
            double[][] medoids1=new double[k][2];
            sql="SELECT count(*) FROM data;";
```

```
rs=stmt.executeQuery(sql);
rs.next();
int n=rs.getInt(1);

Random rand = new Random();
for(int i=0;i<k;i++){
    int v;
    do{
        v = rand.nextInt(n+1);
    }while(v==0);
    String sql2 = "SELECT * FROM data WHERE id="+v+"";
    rs=stmt2.executeQuery(sql2);
    rs.next();
    a = rs.getInt("id");
    b = rs.getInt("x");
    d = rs.getInt("y");
    medoids[i][0]=b;
    medoids[i][1]=d;
}

double cost=0;
double cost1=0;
cost=0;
sql="SELECT * FROM data";
stmt2=c.createStatement();
rs1=stmt.executeQuery(sql);
while(rs1.next())
{
    a = rs1.getInt("id");
    b = rs1.getInt("x");
```

```

        d = rs1.getInt("y");
        int Class = rs1.getInt("class");
        double min=-1;
        int mini = -1;
        for(int i=0;i<k;i++){
            if(min==-1||min>Math.abs(medoids[i][0]-b)+Math.abs(medoids[i][1]-
d)){
                min = Math.abs(medoids[i][0]-b)+Math.abs(medoids[i][1]-d);
                mini = i;
            }
        }
        if(Class!=(mini+1)){
            String sql2 = "update data set class = "+(mini+1)+"where id =
"+a+"";
            stmt2.executeUpdate(sql2);}
        else{
            mini=Class-1;}
        cost+= Math.abs(medoids[mini][0]-b)+Math.abs(medoids[mini][1]-d);

    }
    while((cost-cost1)>0||(cost1==0)){
        if(cost1!=0)
            cost = cost1;
        int v;
        int k1;
        do{
            v = rand.nextInt(n+1);
            k1 = rand.nextInt(k+1);
        }while(v==0||k1==0);
        String sql2 = "SELECT * FROM data WHERE id="+v+"";

```

```

rs=stmt2.executeQuery(sql2);
rs.next();
a = rs.getInt("id");
b = rs.getInt("x");
d = rs.getInt("y");
for(int i=0;i<k;i++){
    if(i==k1){
        medoids1[i][0]=b;
        medoids1[i][1]=d;}
    else{
        medoids1[i][0]=medoids[i][0];
        medoids1[i][1]=medoids[i][1];
    }
}
sql="SELECT * FROM data";
stmt2=c.createStatement();
rs1=stmt.executeQuery(sql);
while(rs1.next())
{
    a = rs1.getInt("id");
    b = rs1.getInt("x");
    d = rs1.getInt("y");
    int Class = rs1.getInt("class");
    double min=-1;
    int mini = -1;
    for(int i=0;i<k;i++){
        if(min==-1||min>Math.abs(medoids1[i][0]-b)+Math.abs(medoids1[i]
[1]-d)){
            min = Math.abs(medoids1[i][0]-b)+Math.abs(medoids1[i][1]-d);

```

```

        mini = i;
    }
}
if(Class!=(mini+1)){
    sql2 = "update data set class = "+(mini+1)+"where id = "+a+"";
    stmt2.executeUpdate(sql2);}
else{
    mini=Class-1;
}
cost1+= Math.abs(medoids1[mini][0]-b)+Math.abs(medoids1[mini][1]-d);
}
}
System.out.print("The medoids are: ");
for(int i=0;i<k;i++){
    System.out.println("(" +medoids1[i][0]+", "+medoids1[i][1]+")");
}
stmt.close();
stmt2.close();
c.commit();
c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+": "+e.getMessage()+"!");
    System.exit(0);
}
System.out.println("Record executed successfully");
}
}

```

OUTPUT IN JAVA:

Opened database successfully

Enter the number of clusters:2

The medoids are: (6.0, 2.0)

(4.0, 7.0)

Record executed successfully

OUTPUT IN DATABASE:

```
select * from data;
```

	id integer	x integer	y integer	class integer
1	1	2	6	2
2	2	3	4	2
3	3	3	8	2
4	4	4	7	2
5	5	6	2	1
6	6	6	4	1
7	7	7	3	1
8	8	7	4	1
9	9	8	5	1
10	10	7	6	2

14. Write a JDBC program to implement k-nn algorithm.

DATABASE:

```
CREATE TABLE data(id int, x int, y int, class int);
```

```
INSERT INTO data VALUES (1, 1, 1, 0), (2, 3, 4, 1), (3, 7, 8, 0), (4, 5, 8, 1),  
(5, 9, 9, 0), (6, 8, 0, 1), (7, 0, 4, 0);
```

```
SELECT * FROM data;
```

	id integer	x integer	y integer	class integer
1	1	1	1	0
2	2	3	4	1
3	3	7	8	0
4	4	5	8	1
5	5	9	9	0
6	6	8	0	1
7	7	0	4	0

JAVA:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```



```
import java.sql.Statement;
import java.lang.Math;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Random;
import java.util.Scanner;

public class bmi {
    public static void sortByColumn(double arr[][], int col)
    {
        // Using built-in sort function Arrays.sort
        Arrays.sort(arr, new Comparator<double[]>() {

            @Override
            public int compare(double[] o1, double[] o2) {
                if (o1[col] > o2[col])
                    return 1;
                else
                    return -1;
            }
        }); // End of function call sort().
    }

    public static void main(String args[]) throws SQLException{
        Connection c=null;
        Statement stmt=null;
        Statement stmt2=null;
```

```
        int a,b,d,x,y,k;

        ResultSet rs1,rs;

try{
    Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/student","postgres","cse");

    c.setAutoCommit(false);

    System.out.println("Opened database successfully");

    stmt=c.createStatement();

        stmt2 = c.createStatement();

        double[][] dist=new double[7][2];

        System.out.println("Enter the x co-ordinate");

        Scanner myObj = new Scanner(System.in);

        String u = myObj.nextLine();

        x = Integer.parseInt(u);

        System.out.println("Enter the y co-ordinate");

        u = myObj.nextLine();

        y = Integer.parseInt(u);

        String sql="SELECT * FROM data";

        stmt2=c.createStatement();

        rs1=stmt.executeQuery(sql);

        int i=0;

        while(rs1.next())

        {

            a = rs1.getInt("id");

            b = rs1.getInt("x");

            d = rs1.getInt("y");

            int Class = rs1.getInt("class");
```

```
        dist[i][0] = (Math.sqrt(Math.pow((x-b),2)+Math.pow((y-d),2)));
        dist[i][1] = Class;
        i++;
    }
    System.out.print("The means are: ");
    sortByColumn(dist, 0);
    System.out.println("Enter the number of k");
    u = myObj.nextLine();
    k = Integer.parseInt(u);
    int c0=0,c1=0;
    for(int j=0;j<k;j++){
        if(dist[j][1]==0)
            c0++;
        else
            c1++;
    }
    if(c0>c1)
        System.out.println("The point belongs to class 0");
    else
        System.out.println("The point belongs to class 1");
    stmt.close();
    stmt2.close();
    c.commit();
    c.close();
}
catch(Exception e){
    System.err.println(e.getClass().getName()+":"+e.getMessage()+"!");
    System.exit(0);
}
```

```

        System.out.println("Record executed successfully");
    }
}

```

OUTPUT IN JAVA:

```

Opened database successfully
Enter the x co-ordinate
5
Enter the y co-ordinate
5
The means are: Enter the number of k
3
The point belongs to class 1
Record executed successfully

```

15. Write a JDBC program to implement baiyes classifier.

DATABASE:

JAVA:

```

package pr4;

import java.sql.*;
import java.util.Scanner;
public class knearest {

    Connection conn;

    knearest() throws ClassNotFoundException, SQLException
    {
        Class.forName("org.postgresql.Driver");
        conn =
            DriverManager.getConnection("jdbc:postgresql://localhost:5432/postgres","pos
            tgres","cse");
    }

    public void algo() throws SQLException
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("enter the colour");
        String colour=sc.next();
        System.out.println("enter the income");
        int income=sc.nextInt();
    }
}

```

```

System.out.println("enter the age");
String age=sc.next();
System.out.println("enter the value of k");
int k=sc.nextInt();
Statement s1=conn.createStatement();
Statement s2=conn.createStatement();
ResultSet rs1= s1.executeQuery("SELECT MAX(income) FROM table9" );
rs1.next();
int max=rs1.getInt(1);
ResultSet rs2= s1.executeQuery("SELECT MIN(income) FROM table9" );
rs2.next();
int min=rs2.getInt(1);
income=(income-min)/(max-min);
ResultSet rs3= s1.executeQuery("SELECT * FROM table9" );
double f = 0;
double f1[]=new double[10];
String s[]=new String[10];
int i = 0;
int c ,c3;
while(rs3.next())
{
    f=0;
    c=0;
    c3=0;
    if(colour.equals(rs3.getString(2)))
        c=1;
    if(age.equals(rs3.getString(4)))
        c3=1;
    if(c==0)
        f=f+1;
    if(c3==0)
        f=f+1;
    float incomel=rs3.getInt(3);
    incomel=(incomel-min)/(max-min);
    float f2=income-incomel;
    f2=f2*f2;
    f=f+f2;
    f1[i]=Math.sqrt(f);
    s[i]=rs3.getString(5);
    i++;
}
int p,j;
double temp;
String temp2;
for(i=0;i<f1.length;i++){
    for(j=i+1;j<f1.length;j++){
        if(f1[i]>f1[j]){
            temp=f1[i];
            temp2=s[i];
            f1[i]=f1[j];
            s[i]=s[j];
            f1[j]=temp;
            s[j]=temp2;
        }
    }
}

```

```

    }
    int c1 = 0, c2 = 0;
    for( j=0; j<10; j++){
        //System.out.println(f1[j]);
    }

    if(k==1)
        System.out.println("the class label is "+s[0]);
    else
    {
        for(i=0; i<k; i++)
        {
            if(s[i].equals("yes"))
                c1++;

            if(s[i].equals("no"))
                c2++;

        }
        if(c1>c2)
        {
            System.out.println("the class label is  yes");
        }
        else{
            System.out.println("the class label is no"); }

    }
}

public static void main(String args[]) throws SQLException,
ClassNotFoundException
{
    knearest kn=new knearest();
    kn.algo();
}
}

```

OUTPUT IN JAVA:

```

enter the colour
red
enter the income
12000
enter the age
youth
enter the value of k

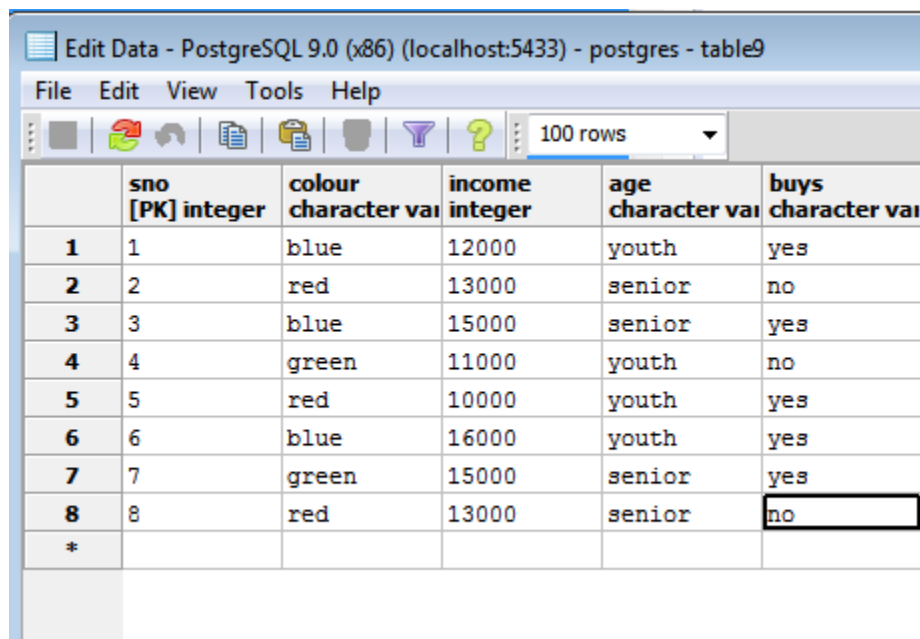
2
16000 10000
-0.33333334
-0.5
-0.8333333
-0.16666667

```

0.0
 -1.0
 -0.8333333
 -0.5
 1.0540925573162605
 1.118033988749895
 1.6414762922301045
 1.0137937560704264
 0.0
 1.4142135623730951
 1.6414762922301045
 1.118033988749895
 0.0
 0.0
 the class label is no

OUTPUT IN DATABASE:

select * from table9;



The screenshot shows the 'Edit Data' window for PostgreSQL 9.0 (x86) connected to localhost:5433, displaying data for 'table9'. The window has a menu bar (File, Edit, View, Tools, Help) and a toolbar with icons for various database operations. A dropdown menu indicates '100 rows' are displayed. The data is presented in a table with 6 columns: 'sno' (integer, primary key), 'colour' (character variable), 'income' (integer), 'age' (character variable), and 'buys' (character variable). The table contains 8 rows of data, with the last row (sno=8) having 'no' selected in the 'buys' column. A '*' row is visible at the bottom of the data grid.

	sno [PK] integer	colour character var	income integer	age character var	buys character var
1	1	blue	12000	youth	yes
2	2	red	13000	senior	no
3	3	blue	15000	senior	yes
4	4	green	11000	youth	no
5	5	red	10000	youth	yes
6	6	blue	16000	youth	yes
7	7	green	15000	senior	yes
8	8	red	13000	senior	no
*					

16. WEKA

- Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code.
- Weka contains tools for data pre-processing, classification, regression, clustering, association rules and visualization.
- It is also well-suited for developing new machine learning schemes.
- Weka can be found in weka website(Latest version 3.6):
<http://www.cs.waikato.ac.nz/ml/weka/>

Datasets in Weka

- Each entry in a dataset is an instance of the java class: `weka.core.Instance`.
- Each instance consists of a number of attributes.

Attributes

- Attributes are of 5 types:
 - 1.Nominal: one of a predefined list of values. Ex:red,green,blue
 - 2.Numeric: A real or integer number.
 - 3.String: Enclosed in "double quotes".
 - 4.Date
 - 5.Relational

ARFF Files

- The external representation of an Instances class consists of:
->A header:Describes the attribute types.

->Data Section: Comma separated list of data

- Example for creation of .arff file:

```
@relation AllElectronics
```

```
@attribute colour {red,blue,green}
```

```
@attribute income real
```

```
@attribute age {youth,senior}
```

```
@attribute buys {yes,no}
```

```
@data
```

```
blue,12000,youth,yes
```

```
red,13000,senior,no
```

```
blue,15000,senior,yes
```

```
green,11000,youth,no
```

```
red,10000,youth,yes
```

```
blue,16000,youth,yes
```

```
green,15000,senior,yes
```

Classifiers in Weka

- Any .arff file is split into train and test set
- For Ex: Soybean-train.arff and Soybean-test.arff



a) CLASSIFICATION

It is necessary to provide a clear classification of data mining systems which may help users to distinguish between such systems and to identify them.

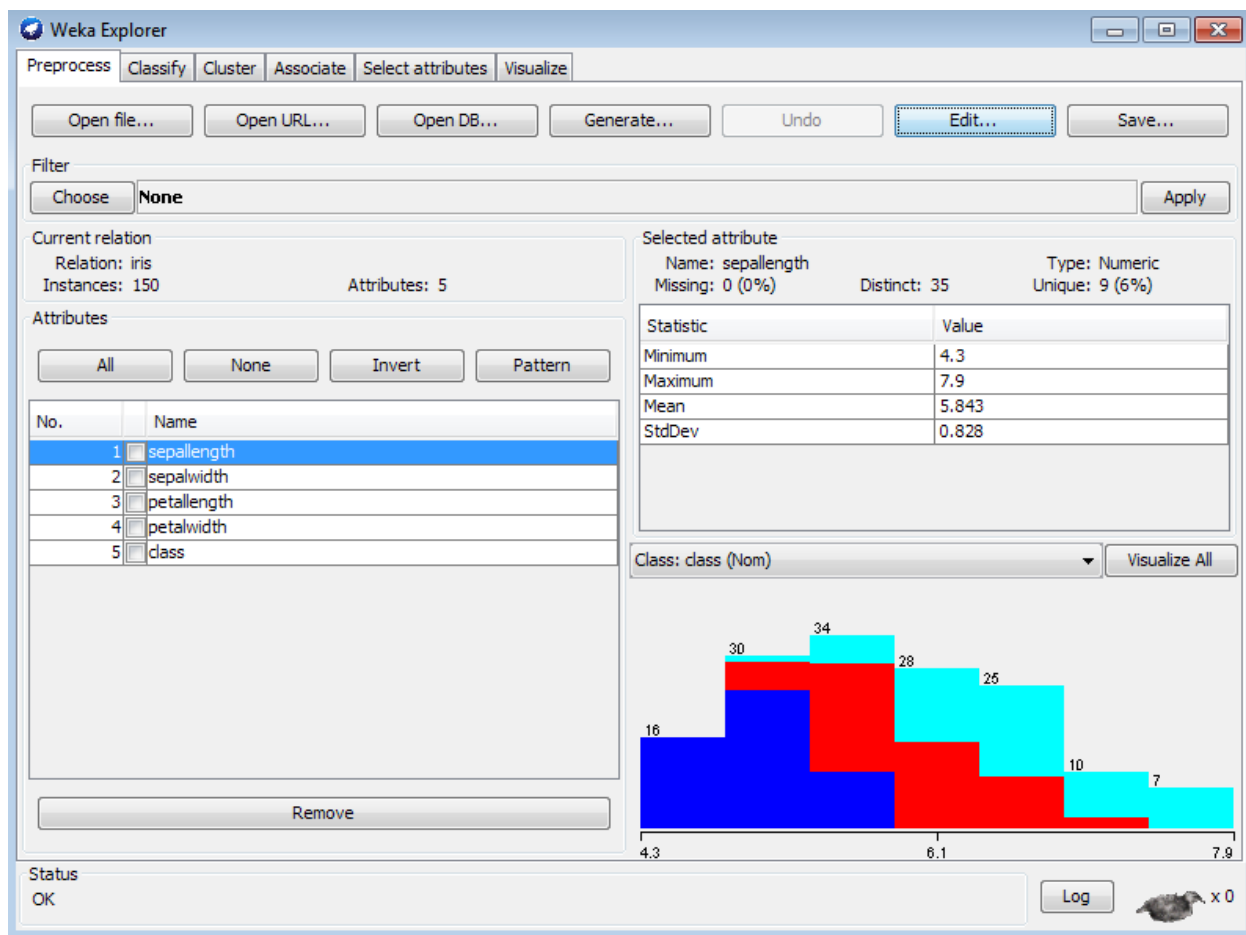
Data mining classification can be done in different ways:

- 1) Data mining can be classified according to the kinds of databases mined
- 2) Data mining can be Classified according to the kinds of knowledge mined which is done based on the mining functionalities like characterization, discrimination etc...
- 3) We can also classify the data mining systems according to the kinds of techniques utilized, applications adapted.

(i) ZeroR Algorithm

Open the WEKA GUI Chooser from start menuàall programs and click on the EXPLORER button.

Now click on the **Open File** button and choose the file named as “iris.arff”.



After loading the input choose the classify tab in the WEKA explorer window. Now select the “**use training set**” under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.

The screenshot shows the Weka Explorer window with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'ZeroR'. The 'Test options' section shows 'Cross-validation' selected with 'Folds' set to 10. The 'Result list' on the left shows two entries: '00:41:42 - rules.ZeroR' and '00:42:59 - rules.ZeroR'. The 'Classifier output' pane displays the following results:

Time taken to build model: 0 seconds

=== Evaluation on training set ===

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	50	33.3333 %
Incorrectly Classified Instances	100	66.6667 %
Kappa statistic	0	
Mean absolute error	0.4444	
Root mean squared error	0.4714	
Relative absolute error	100	%
Root relative squared error	100	%
Total Number of Instances	150	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	1	0.333	1	0.5	0.5	Iris-setosa	
0	0	0	0	0	0	Iris-versicolor	
0	0	0	0	0	0	Iris-virginica	
Weighted Avg.	0.333	0.333	0.111	0.333	0.167	0.5	

=== Confusion Matrix ===

```

a b c <-- Classified as
50 0 0 | a = Iris-setosa
50 0 0 | b = Iris-versicolor
50 0 0 | c = Iris-virginica

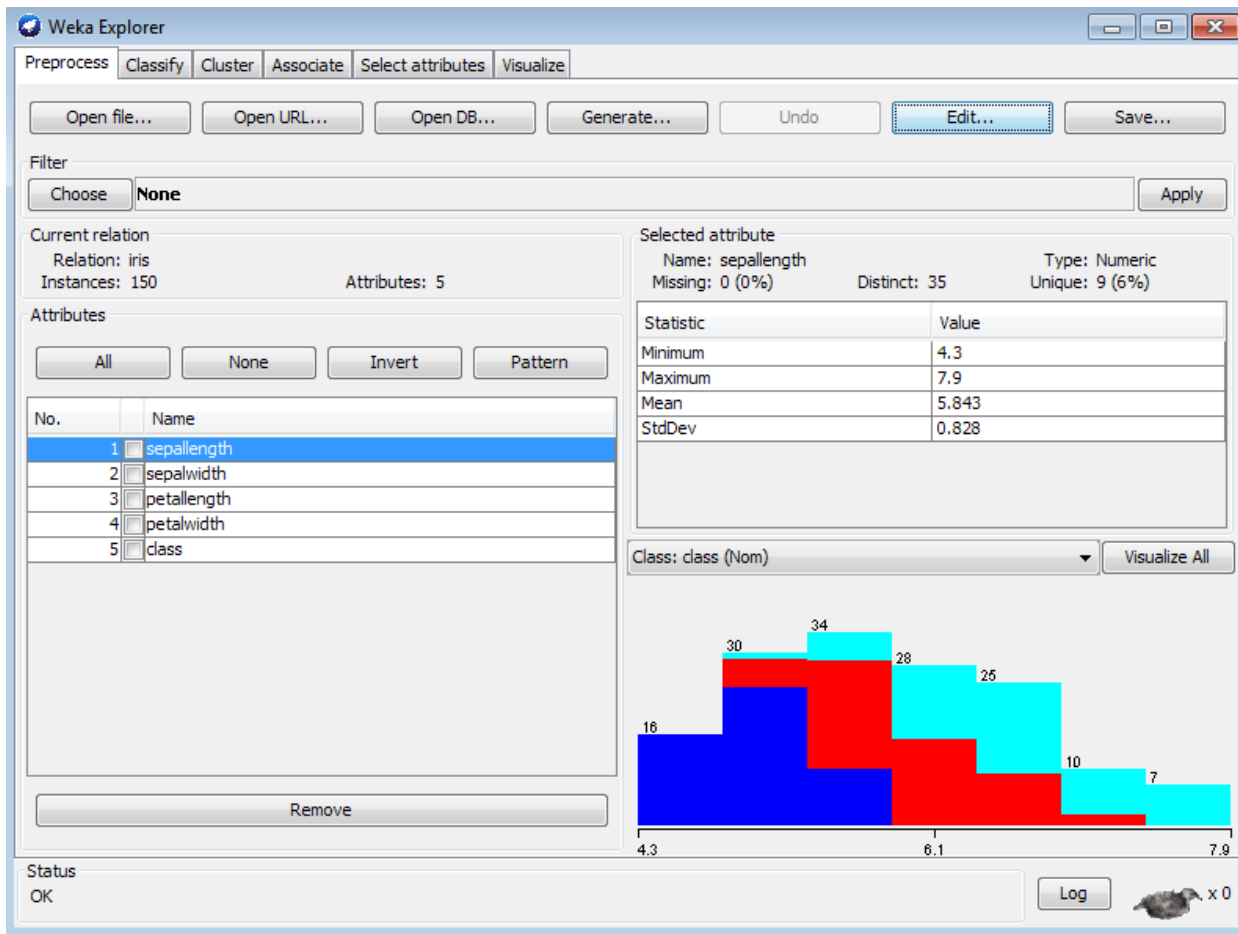
```

The status bar at the bottom shows 'Status OK' and a 'Log' button.

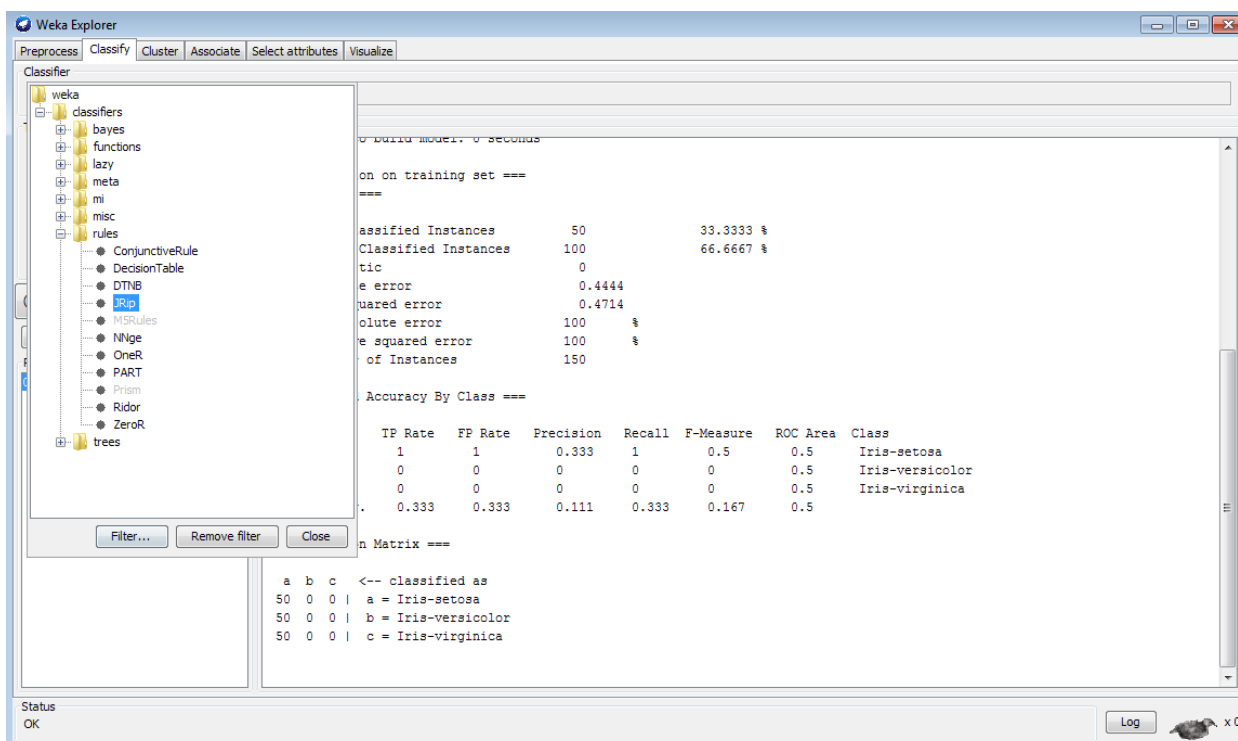
(ii) JRip Algorithm

Open the WEKA GUI Chooser from start menuàall programs and click on the EXPLORER button.

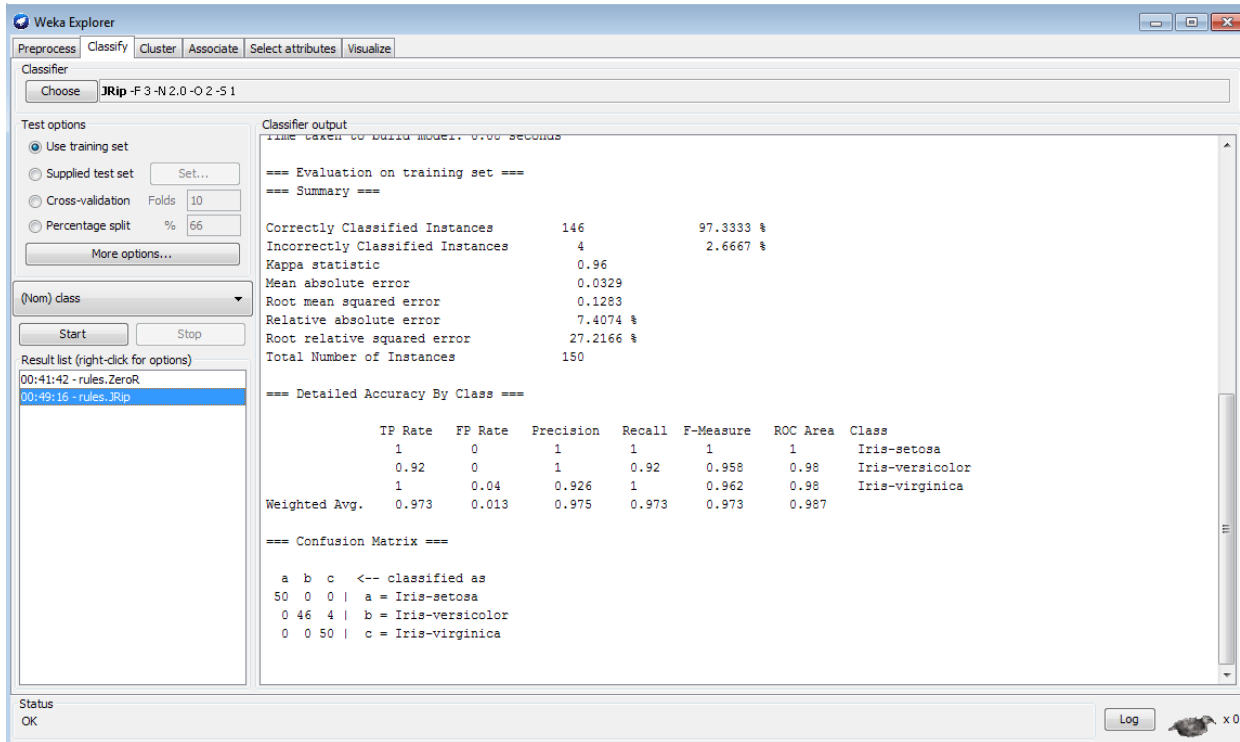
Now click on the **Open File** button and choose the file named as "iris.arff".



After loading the input choose the classify tab in the WEKA explorer window. Under the classify tab click on choose button and select the JRip.



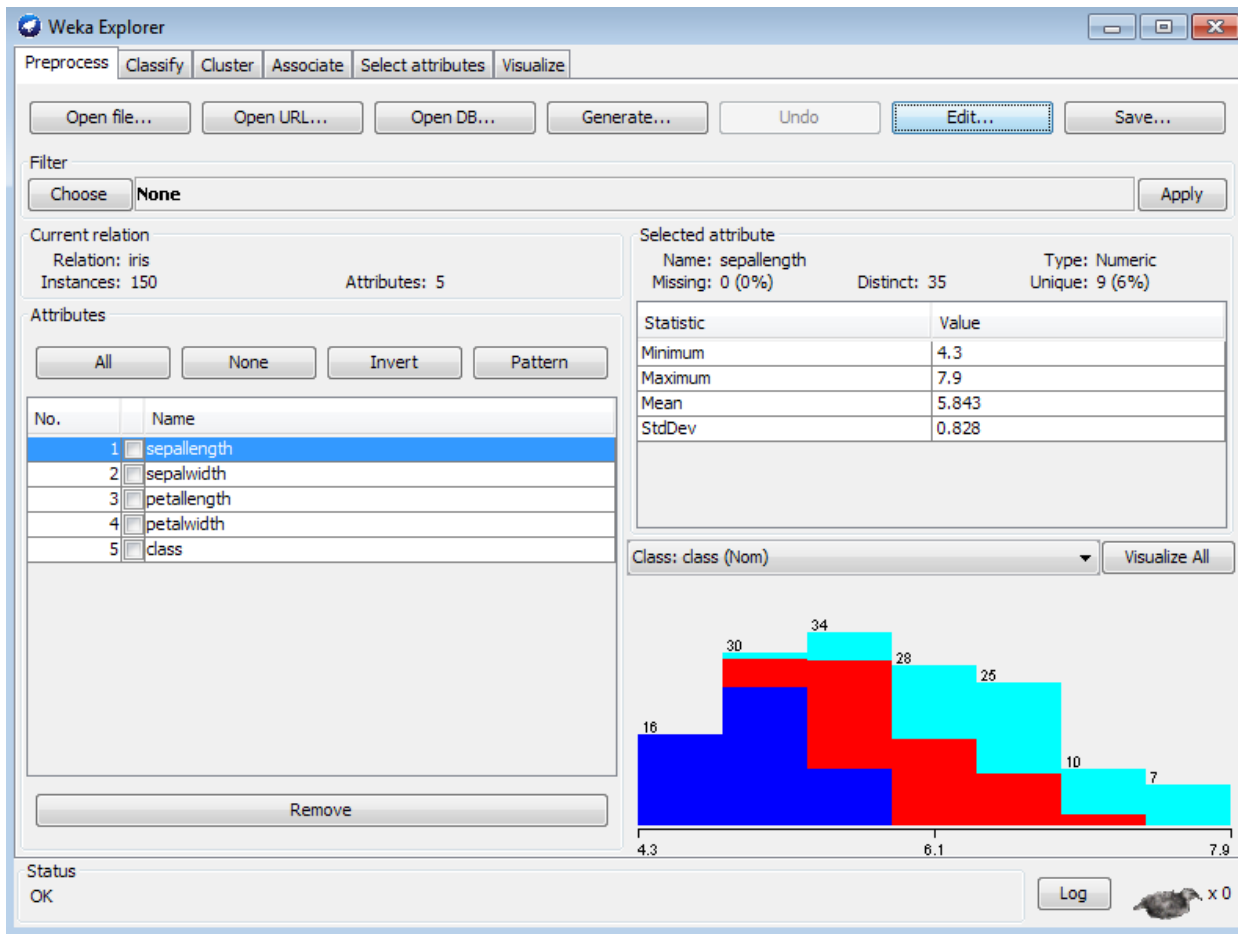
Now select the “**use training set**” under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.



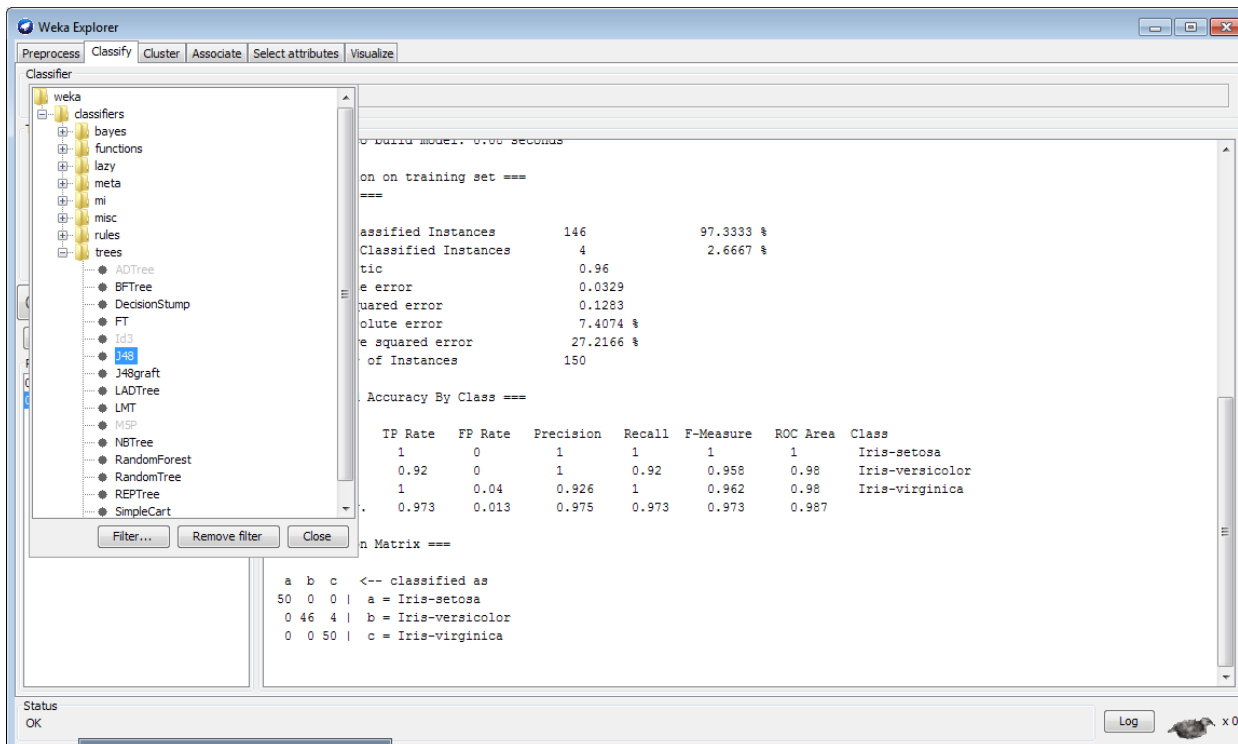
(iii) J48 Algorithm

Open the WEKA GUI Chooser from start menuàall programs and click on the EXPLORER button.

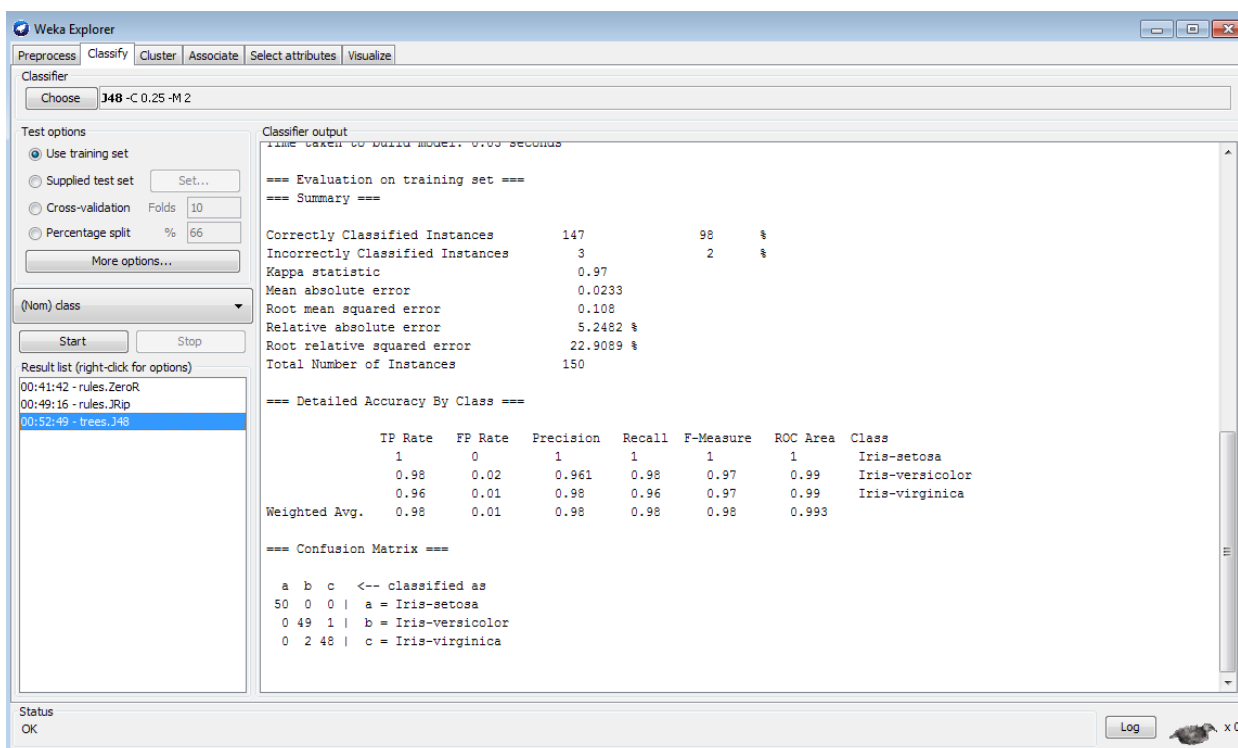
Now click on the **Open File** button and choose the file named as “iris.arff”.



After loading the input choose the classify tab in the WEKA explorer window. Under the classify tab click on choose button and select the J48.



Now select the “use training set “under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.



Under the **Result List** right click the item to get the options and select the option “visualize tree” option.

Weka Explorer

Classifier: J48-C 0.25-M 2

Test options: Use training set (selected), Supplied test set, Cross-validation (Folds: 10), Percentage split (%: 66)

Classifier output:

Time taken to build model: 0.05 seconds

=== Evaluation on training set ===

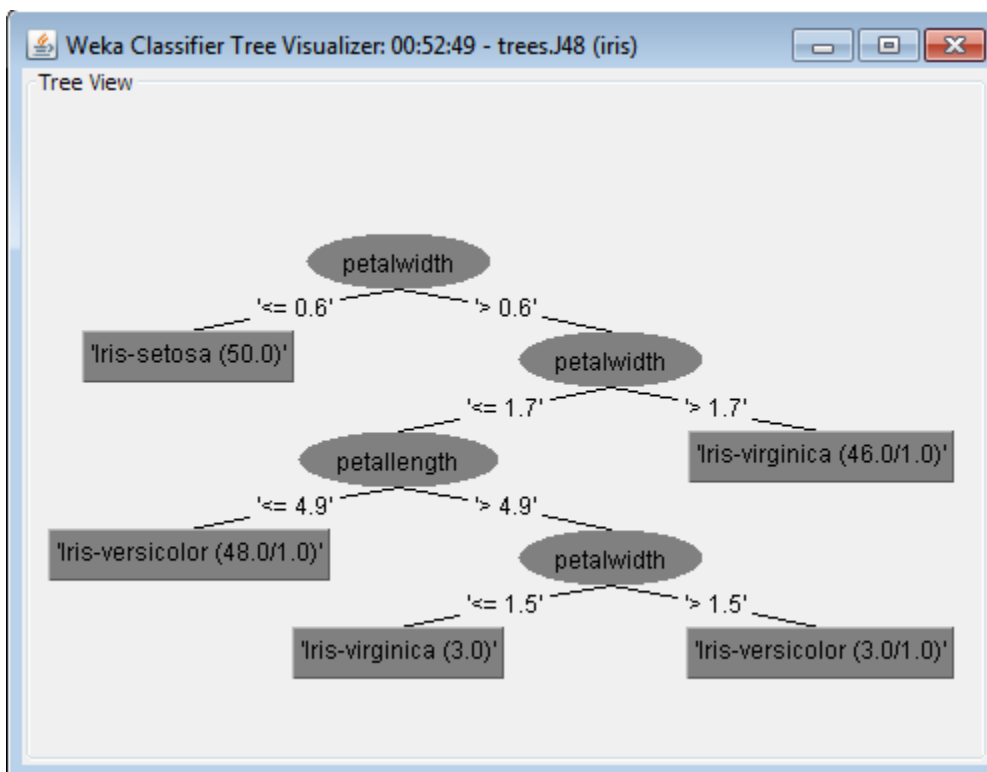
=== Summary ===

Correctly Classified Instances	147	98	%
Incorrectly Classified Instances	3	2	%
Kappa statistic	0.97		
Mean absolute error	0.0233		
Root mean squared error	0.108		
Relative absolute error	5.2482	%	
Root relative squared error	22.9089	%	
Total Number of Instances	150		

=== Detailed Accuracy By Class ===

	Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	Iris-setosa
	0.98	0.02	0.961	0.98	0.97	0.99	Iris-versicolor
	0.96	0.01	0.98	0.96	0.97	0.99	Iris-virginica
	0.98	0.01	0.98	0.98	0.98	0.993	

Context menu options: View in main window, View in separate window, Save result buffer, Delete result buffer, Load model, Save model, Re-evaluate model on current test set, Visualize classifier errors, Visualize tree (selected), Visualize margin curve, Visualize threshold curve, Cost/Benefit analysis, Visualize cost curve



b) ASSOCIATION

ASSOCIATION RULE: - It is a popular method for discovering interesting relations between variables in large databases.

It is used to identify the most frequent item sets or frequent patterns that occur frequently in a database.

To perform association rule mining we need two parameters like:

Support: - It is the total probability of the item sets that occurred in the transaction.

Confidence: - it is the ratio of probability of two item sets A,B and the probability of the item set A.

Association rule mining is used in different applications like basket analysis, catalog design, clustering, classification etc.

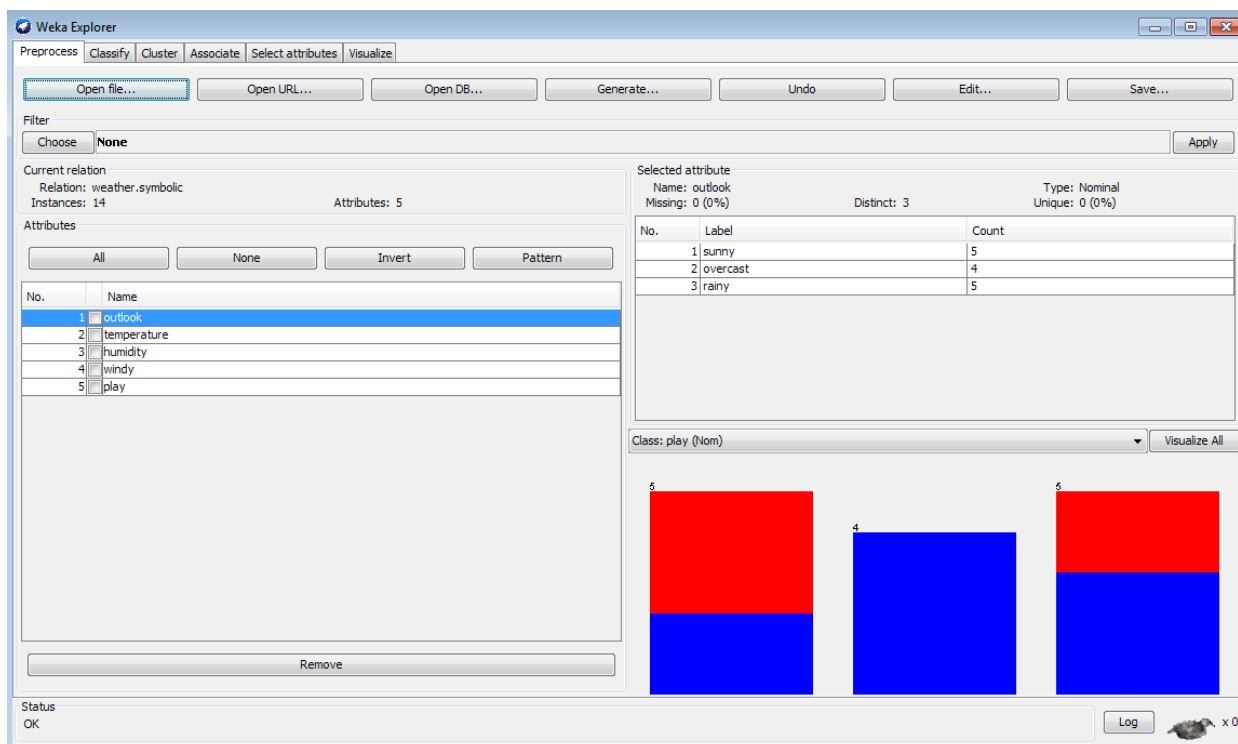
Apriori Algorithm: - To implement the association rule we use this algorithm.

Steps to follow while implementing the association rule are:

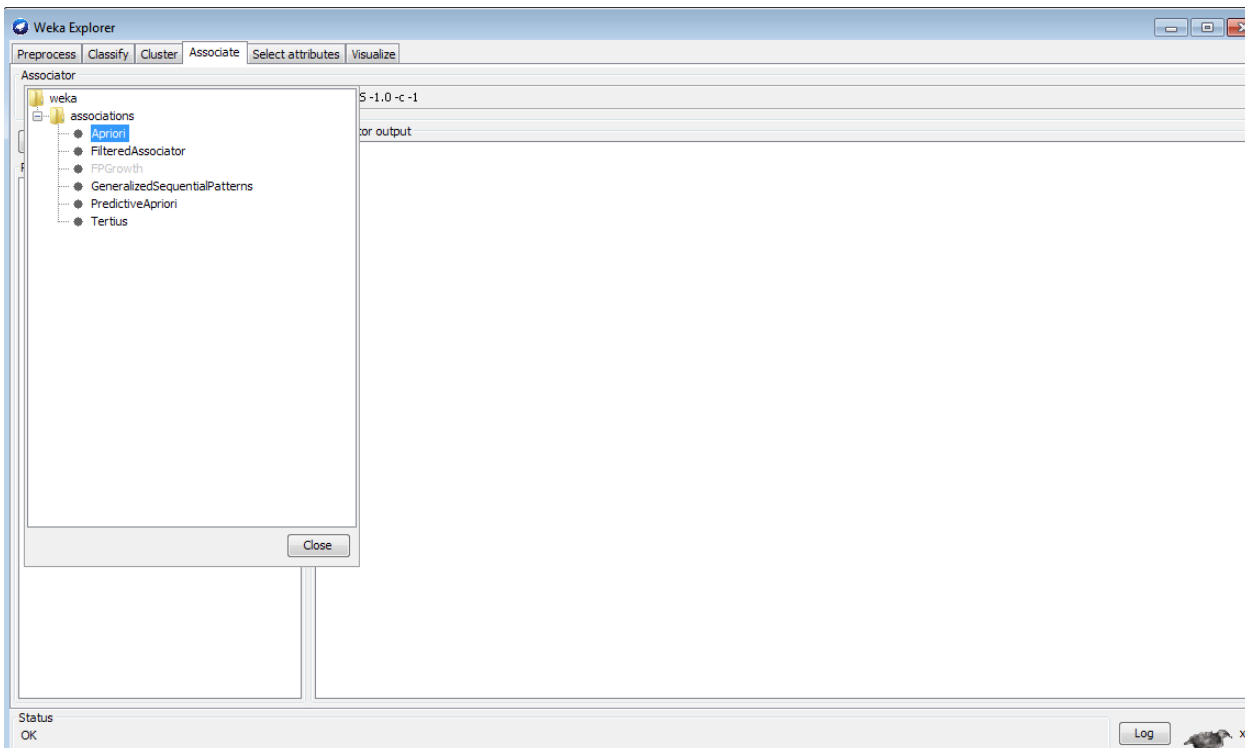
1. Find the frequent item sets where the items have the minimum support count.
2. Use the frequent item sets to generate association rules.
3. Generate the candidates by performing join operation.
4. Perform pruning operation i.e. if any one of the candidates that have a subset is not frequent in the set then delete it.
5. Also calculate support count for each candidate and delete those candidates whose minimum support count is not satisfied.

Open the WEKA GUI Chooser from start menuàall programs and click on the EXPLORER button.

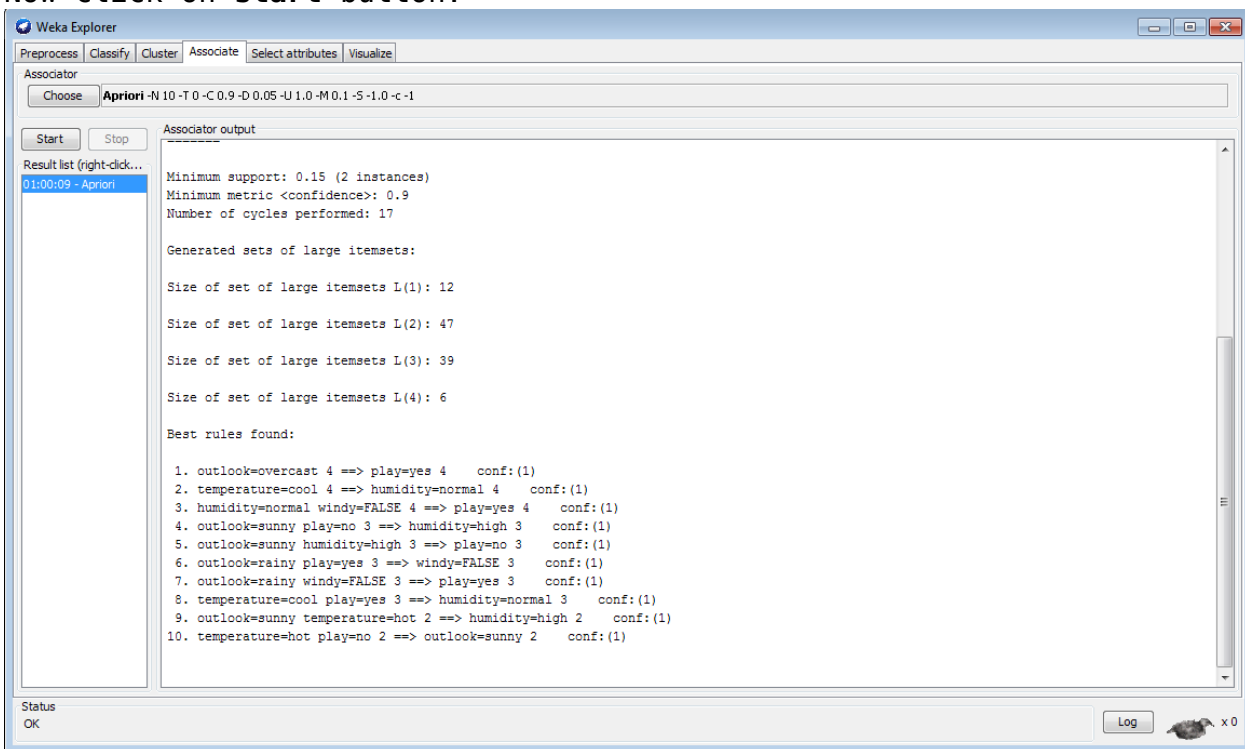
Now click on the **Open File** button and choose the file named as "weather.nominal.arff".



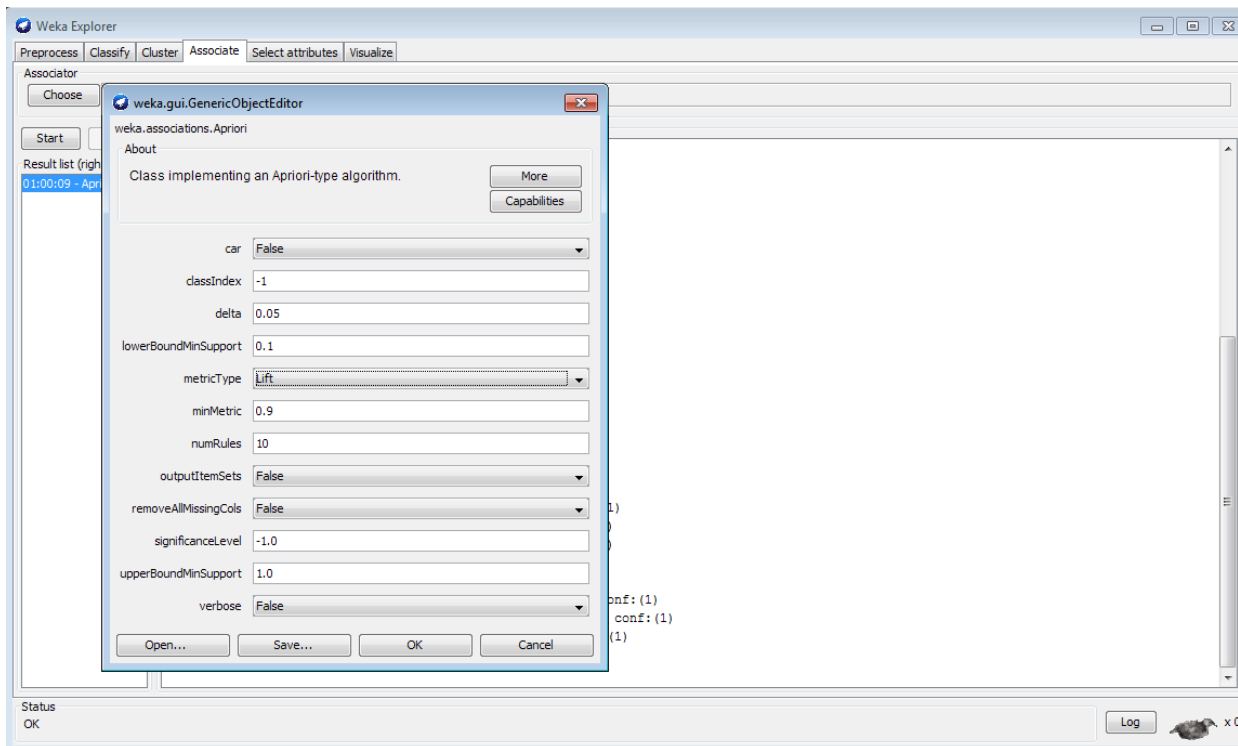
After loading the input file named association.csv as shown in figure 2, choose the associate tab in the WEKA explorer window. Under the associate tab click on choose button and select the **Apriori** Algorithm.



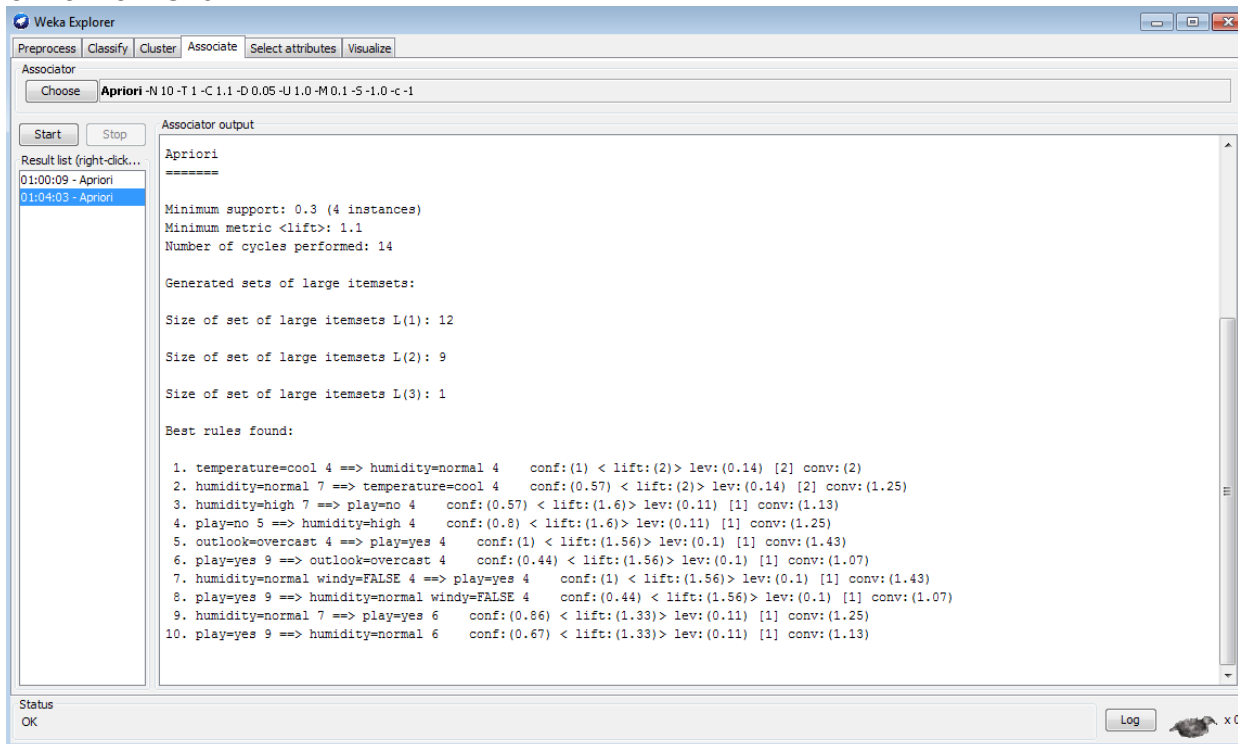
Now click on **start** button.



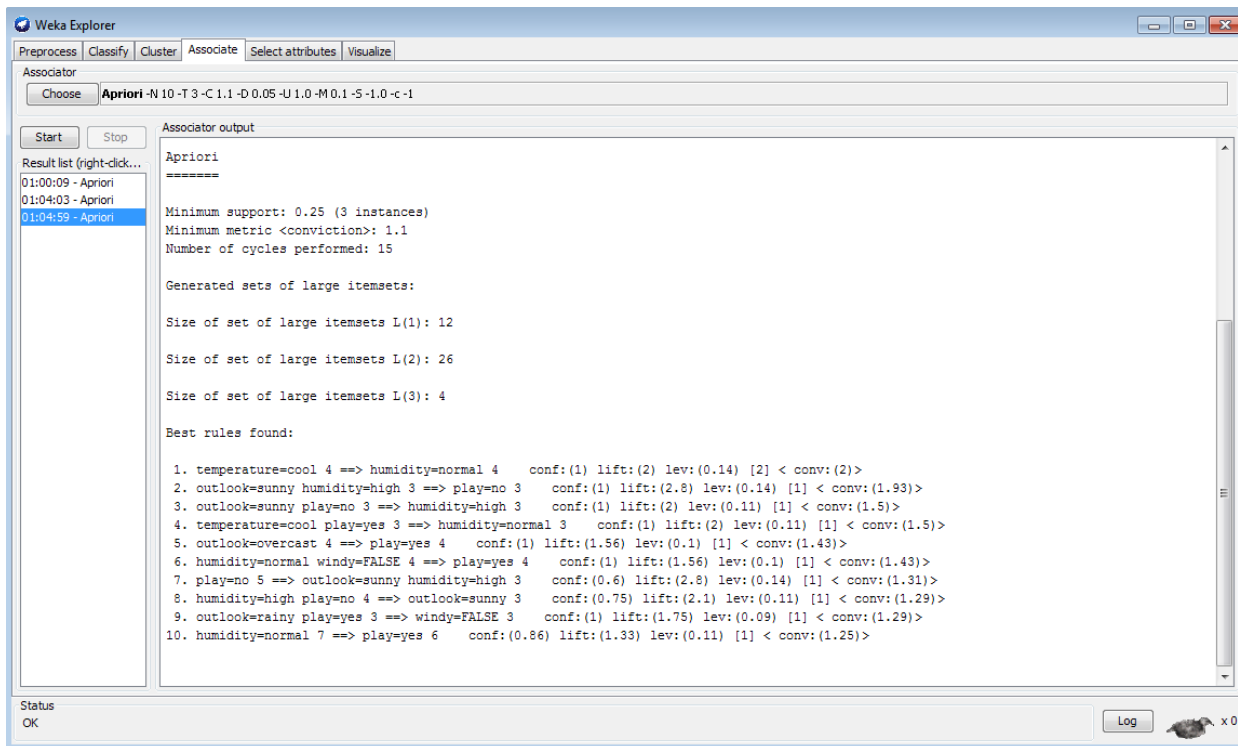
Now click on Apriori algorithm near choose button and change the metric from confidence to lift.



Click on start.



Change the metric to conviction and repeat.



c) CLUSTERING

Clustering is a task of assigning a set of objects into groups called as clusters. Clustering is also referred as cluster analysis where the objects in the same cluster are more similar to each other than to those objects in other clusters.

Clustering is the main task of Explorative Data mining and is a common technique for statistical data analysis used in many fields like machine learning, pattern recognition, image analysis, bio informatics etc...

Cluster analysis is not an algorithm but is a general task to be solved.

Clustering is of different types like hierarchical clustering which creates a hierarchy of clusters, partial clustering, and spectral clustering.

SimpleK-Means: -

It is a method of cluster analysis called as partial cluster analysis or partial clustering.

K-Means clustering partition or divides n observations into K clusters.

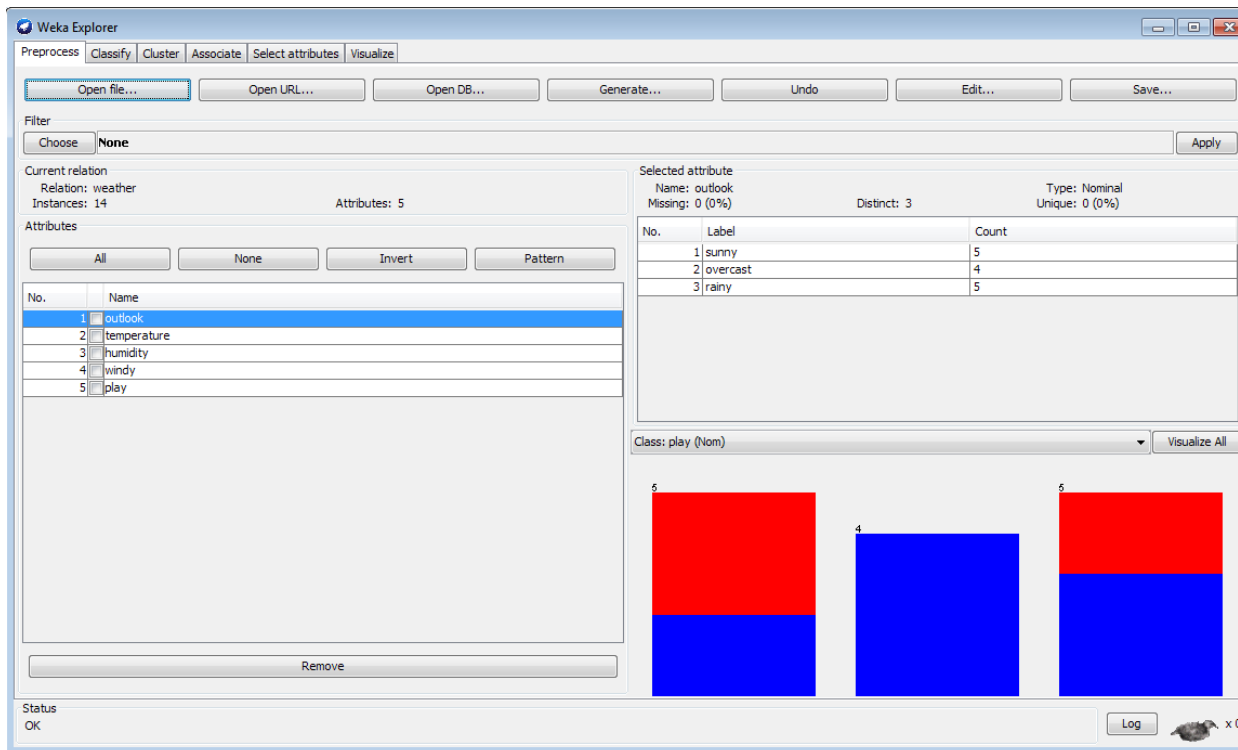
Each observation belongs to the cluster with the nearest mean.

K-means clustering is an algorithm to group the objects based on attributes/features into K number of groups where K is positive integer.

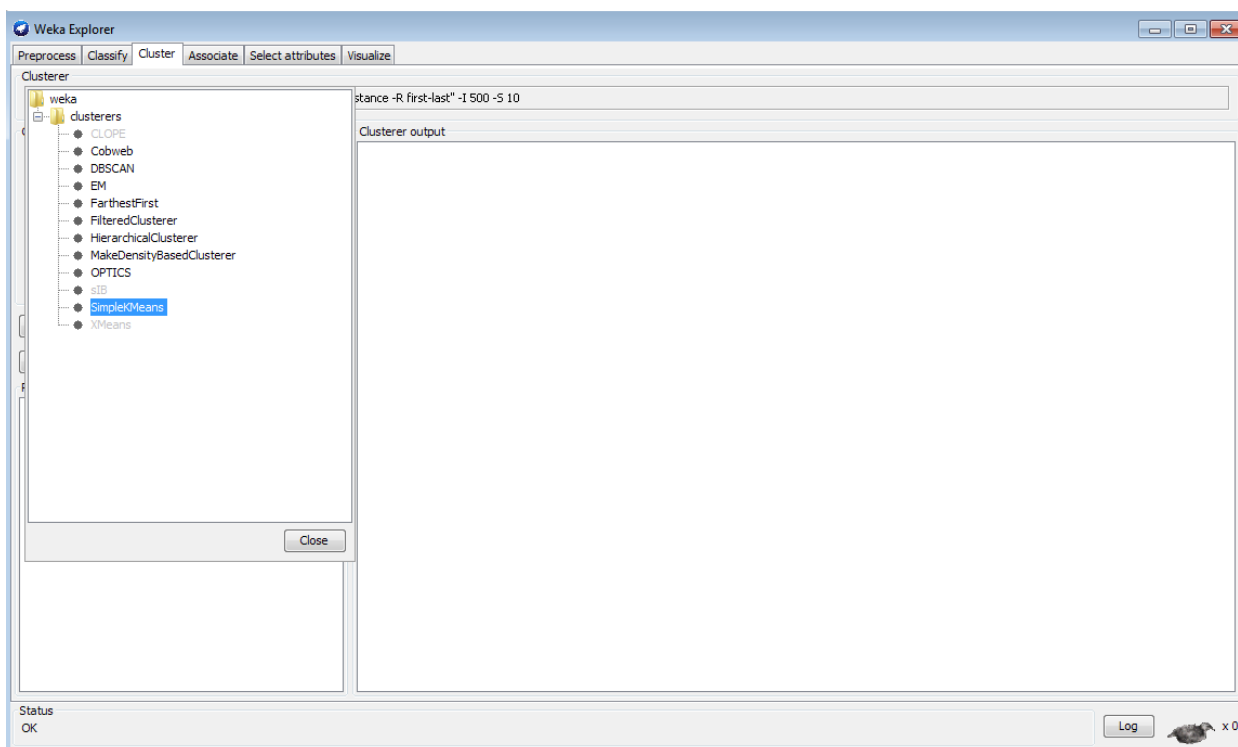
K-Means clustering is used in different types of applications like pattern recognition, artificial intelligent, image processing, etc.

Open the WEKA GUI Chooser from start menuàall programs and click on the EXPLORER button.

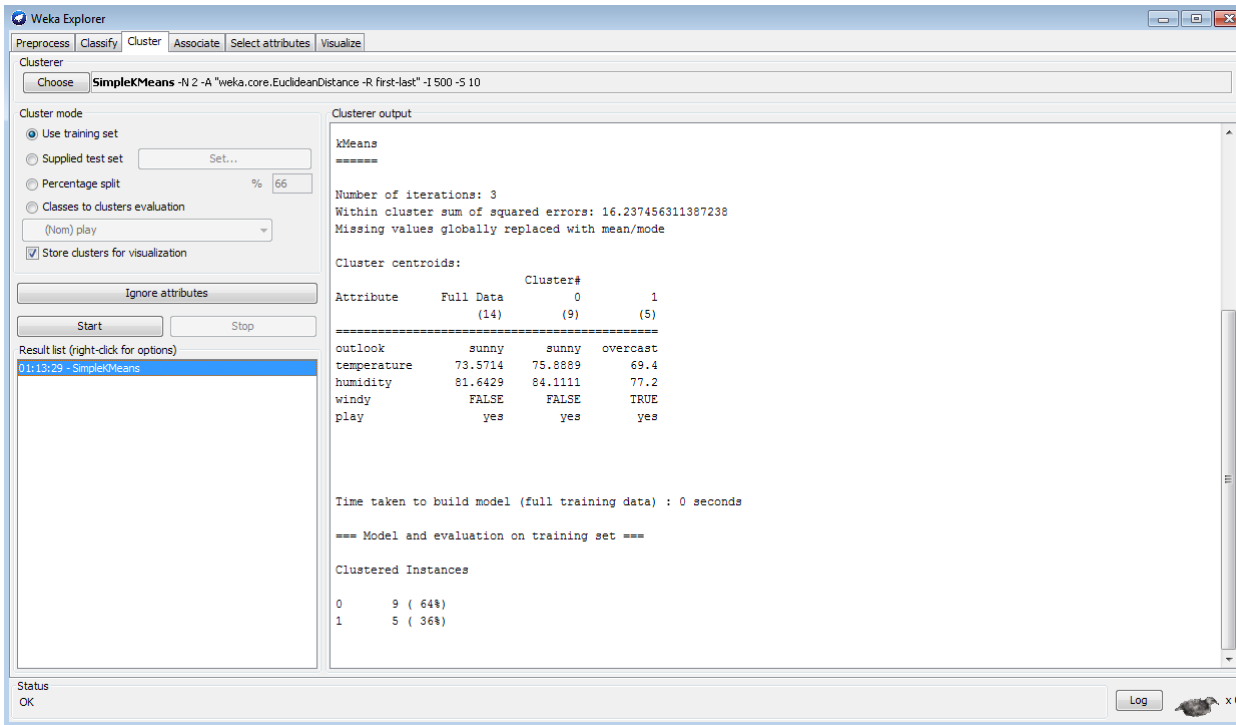
Now click on the **Open File** button and choose the file named as "weather.numeric.arff".



After loading the input file named cluster.csv as shown in figure 2, choose the cluster tab in the WEKA explorer window. Under the cluster tab click on choose button and select the **SimpleKMeans** under clusterers.



Now select the "use training set "under the **Test Options** located at the left of the WEKA explorer window and click on **start** button.



Under the **Result List** right click the item to get the options and select the option **"visualize cluster assignments"** option.

