PROGRAM 1.

Develop a database application to demonstrate storing and retrieving of BLOB and CLOB object.

Theory:-

CLOB stands for Character Large Object. It is a datatype used to store and retrieve large amount of text data in character format. CLOB datatypes are created by using CREATE commands. It can store Single byte and multiple byte character data. It supports both fixed width and variable width character set.

A CLOB contains a logical pointer to a CLOB, not the CLOB itself. CLOB columns are referred as LONG VARCHAR.

BLOB stands for Binary Large object or Basic Large Object. It is a datatype used to store unstructured binary large objects. It is an array of bytes(byte[]) stored in the database. It is not case sensitive Blob's are used to hold multimedia objects.
BLOB's fields are normally used to store graphics, audio, video, still images and so on.

Insertb.java

```
Packagevemana;
importjava.io.File;
importjava.io.FileInputStream;
importjava.io.IOException;
importjava.io.PrintWriter;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.PreparedStatement;
importjava.sql.SQLException;
importjava.util.logging.Level;
importjava.util.logging.Logger;
```

```
importjavax.servlet.ServletException;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

public class insertb extends HttpServlet {
protected void doGet (HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException {
        FileInputStream is = null;
        response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
         // Connect to Oracle
        out.println("<html>");
        out.println("<body>");
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
        , "hr", "hr");
        PreparedStatementps = con.prepareStatement("insert into pic values(?,?,?)
  ");
        ps.setInt(1,3);
        ps.setString(2,"Java Logo");

        File fBlob = new File
        ("C:\\Users\\insbins\\Documents\\NetBeansProjects\\BLOBCLOBTrial\\sr
        c\\java\\vemana\\Jellyfish.jpg");
        is = new FileInputStream ( fBlob );
        ps.setBinaryStream (3, is, (int) fBlob.length() );
        ps.executeUpdate();
        if (is != null)
        is.close();
        out.println("Image sucessfully inserted into database");
        out.println("</body>");
```

```
                out.println("</html>");
                        }
                catch(Exception e)
                {
                        System.out.println(e);
                 }
                finally {
                        out.close();
                         }
        }
}
```

GetBLOB.java

```
packagevemana;
importjava.io.IOException;
importjava.io.InputStream;
importjava.io.OutputStream;
importjava.sql.Blob;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.PreparedStatement;
importjava.sql.ResultSet;
importjavax.servlet.ServletException;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

public class GetBLOB extends HttpServlet
{
        protected void processRequest (HttpServletRequest request, HttpServletResponse
        response)
        throwsServletException, IOException
        {
```

```
        try
        {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connectioncon=DriverManager.getConnection("jdbc:oracle:thin:@localho
        st:1521:xe", "hr", "hr");
                PreparedStatementps = con.prepareStatement("select image from pic
        where id = ?");
                String id = request.getParameter("id");
                ps.setString(1,id);
                ResultSetrs = ps.executeQuery();
                rs.next();
                Blob b = rs.getBlob("image");
                response.setContentType("image/jpeg");
                response.setContentLength( (int) b.length());
                InputStream is = b.getBinaryStream();
                OutputStreamos = response.getOutputStream();
                bytebuf[] = new byte[(int) b.length()];
                is.read(buf);
                os.write(buf);
                os.close();
        }
        catch(Exception ex)
        {
                System.out.println(ex.getMessage());
        }
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throwsServletException, IOException
{
        processRequest(request, response);
}
protected void doPost(HttpServletRequest request, HttpServletResponseresponse)
throwsServletException, IOException
{
```

```
        processRequest(request, response);
 }
```

Bdisplay.java

```
packagevemana;
importjava.io.IOException;
importjava.io.PrintWriter;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.PreparedStatement;
importjava.sql.ResultSet;
importjavax.servlet.ServletException;
importjavax.servlet.http.HttpServlet;
importjavax.servlet.http.HttpServletRequest;
importjavax.servlet.http.HttpServletResponse;

public class bdisplay extends HttpServlet
{
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throwsServletException, IOException
        {
                String id;
                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out = response.getWriter();
                try
                {
                out.println("<html>");
                out.println("<body>");
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con =
                DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
                "hr", "hr");
                PreparedStatementps = con.prepareStatement("select * from pic");
                ResultSetrs = ps.executeQuery();
                out.println("<h1>Photos</h1>");
```

```
                while ( rs.next())
                {
                        id=rs.getString("id");
                        out.println("<h4>" +id+ "</h4>");
                        out.println("<img width='160' height='160'
                        src=GetBLOB?id="+id+"></img><p/>");
                }
                out.println("</body>");
                out.println("</html>");
                con.close();
        }
           catch(Exception e)
           {
                System.out.println(e);
           }
           finally
           {
                out.close();
           }
    }
}
```

Steps for storing and retrieving of BLOB and CLOB object:

Step 1: Connect to Oracle JDBC driver.

Step 2: Creating a BLOB.

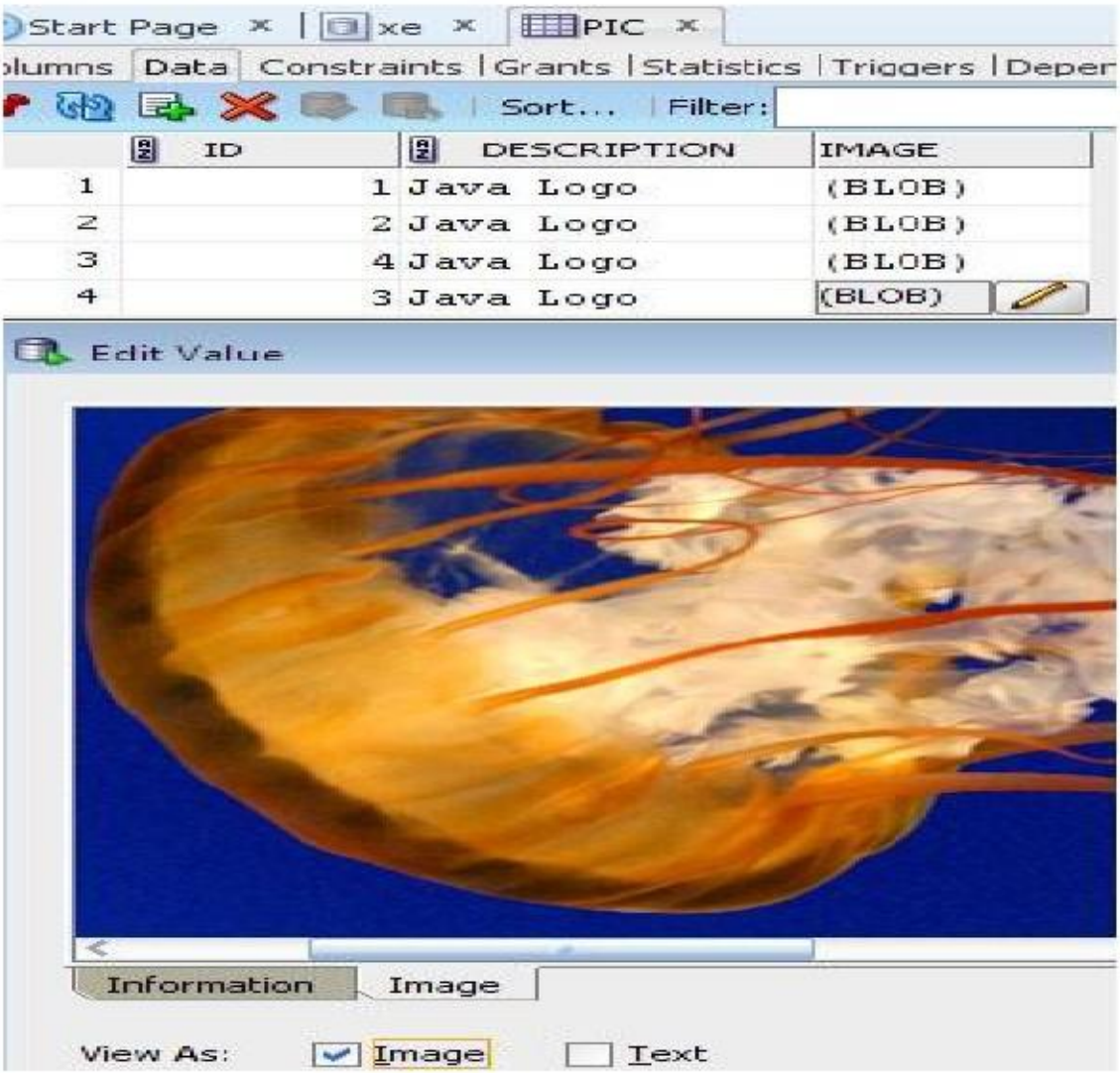
Step 3: Inserting image into a BLOB.

Step 4: Querying a database for a BLOB.

Step 5: Executing a binary data.

Step 6: Storing and Retrieving of BLOB and CLOB object was displayed successfully.

OUTPUT:



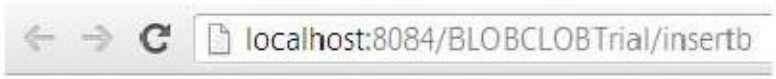
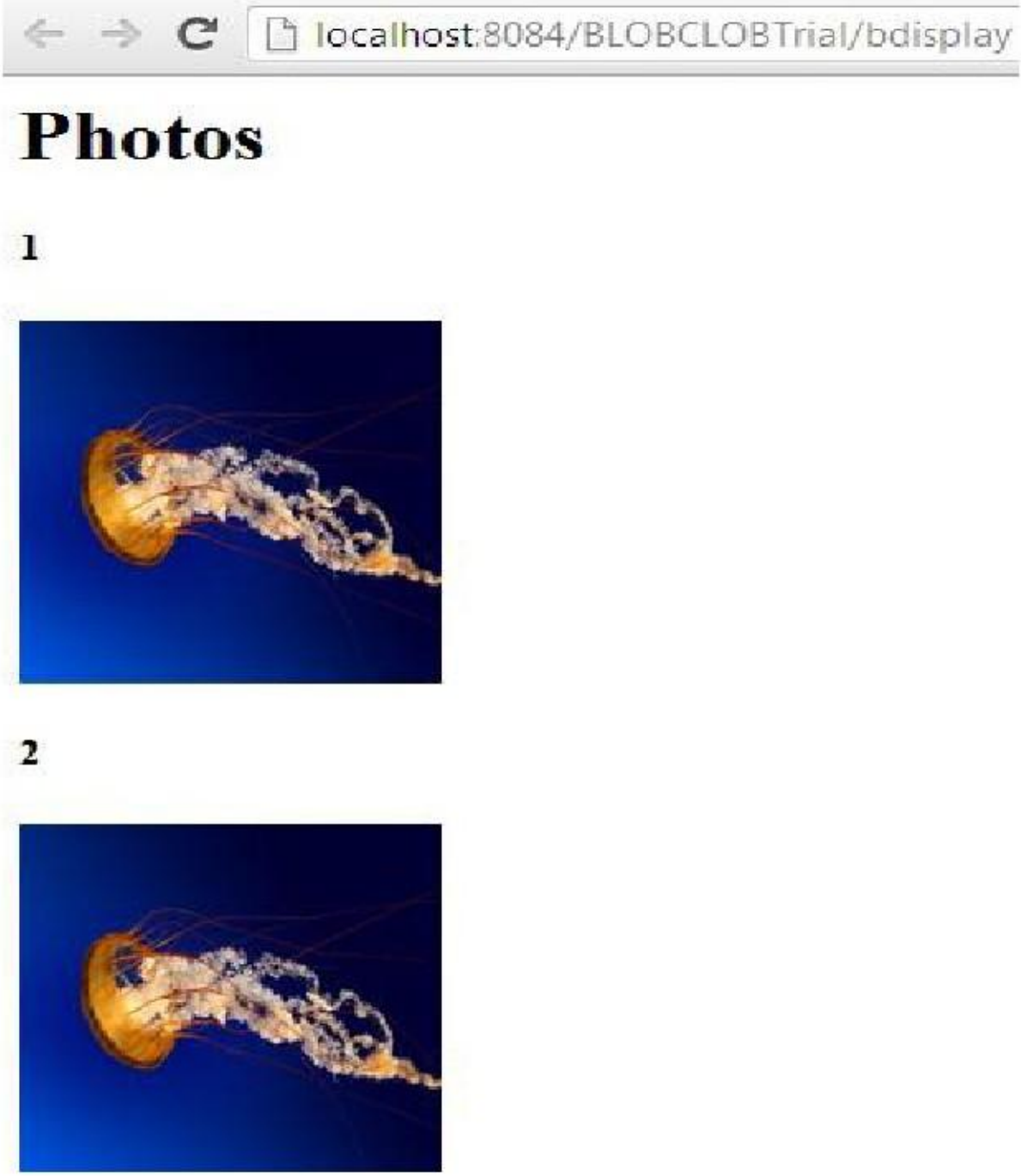


Image sucessfully inserted into database



CONCLUSION: - An application for storing and retrieving of BLOB and CLOB object wasimplemented successfully.

PROGRAM 2.

Develop a database application to demonstrate the representation of multivalued attributes, and use of nested tables to represent complex objects. Write suitable queries to demonstrate it.

Theory:-

Nested table is an Oracle data type used to support columns containing multivalued attributes, in this case, columns that can hold an entire sub-table.

Example:

Create a table with NESTED TABLE column:

SQL>CREATE TYPE my_tab_t AS TABLE OF VARCHAR2(30);

Output:

Type Created.

SQL >CREATE TABLE nested_table (id NUMBER, col1 my_tab_v)

NESTED TABLE col1 STORE AS col1_tab;

Output:

Table created.

Inserting data into table:

SQL >INSERT INTO nested_table VALUES (1, my_tab_v('A'));

1 row created.

SQL >INSERT INTO nested_table VALUES (2, my_tab_v('B', 'C'));

1 row created.

SQL >INSERT INTO nested_table VALUES (3, my_tab_v('D', 'E', 'F'));

1 row created.

SQL >COMMIT;

Commit complete.

Selecting from nested table:

SQL >SELECT * FROM nested_table;

Output:

ID	COL1
1	MY_TAB_V('A')
2	MY_TAB_V('B', 'C')
3	MY_TAB_V('D', 'E', 'F')

SQL >SELECT id, COLUMN_VALUE FROM nested_table t1, TABLE(t1.col1) t2;

Output:

ID	ID COLUMN_VALUE
1	A
2	B
2	C
3	D
3	E
3	F

6 rows selected.

Conclusion: - Application to demonstrate the representation of multivalued attributes, and use of nested tables to represent complex objects was successfully executed with the help of above queries.

PROGRAM 3.

Design and develop a suitable student database application .one of the attributes to be maintained is the attendance of a student in each subject for which he/she has enrolled. Using TRIGGERS, write active rules to do the following

a) Whenever the attendance is updated, check if the attendance is less than 85%, if so notify the head of the department concerned.

b) Whenever the marks in an internal assessment test are entered, check if the marks is less than 40%, if, so notify the head of the department concerned.

create table stud_rec(regno varchar(10) primary key,name varchar2(10) not null,major varchar2(10),mark number(4),attendence number(4));

Table created.

SQL> desc stud_rec;

Name	Null?	Type
REGNO	NOT NULL	VARCHAR2(10)
NAME	NOT NULL	VARCHAR2(10)
MARKS		VARCHAR2(10)
MAJOR		NUMBER(4)
ATTENDENCE		NUMBER(4)

SQL> insert into stud_rec values('&regno','&name','&major',&mark,&atttendance);

Enter value for regno: 001

Enter value for name: pinky

Enter value for major: cse

Enter value for mark: 85

Enter value for atttendance: 70

old 1: insert into stud_rec values('&regno','&name','&major',&mark, &atttendance)

new 1: insert into stud_rec values('001','pinky','cse',85,70)

1 row created.

SQL> /

Enter value for regno: 002

Enter value for name: john

Enter value for major: cse

Enter value for mark: 30

Enter value for atttendance: 70

old 1: insert into stud_rec values('&regno','&name','&major',&mark,&atttendance)

new 1: insert into stud_rec values('002','john','cse',30,70)

1 row created.

SQL> /

Enter value for regno: 003

Enter value for name: arun

Enter value for major: cse

Enter value for mark: 90

Enter value for atttendance: 95

old 1: insert into stud_rec values('&regno','&name','&major',&mark,&atttendance)

new 1: insert into stud_rec values('003','arun','cse',90,95)

1 row created.

SQL> select * from stud_rec;

REGNO	NAME	MARKS	MAJOR	ATTENDENCE
001	pinky	cse	85	70
002	john	cse	30	70
003	arun	cse	90	95

```
create TRIGGER trig_attee_mark after update on stud_rec
for each row
begin
if ( :new.attendence < 85 ) then dbms_output.put_line (:new.name || ' bearing REG.NO. '||
:new.regno || ' has attendence less than 85 ' );
end if;
if ( :new.mark < 40 ) then dbms_output.put_line (:new.name || ' bearing REG.NO. '||
:new.regno || ' got mark less than 40 ' );
end if;
end;
```

Trigger created.

SQL> select * from stud_rec;

REGNO	NAME	MARKS	MAJOR	ATTENDENCE
001	pinky	cse	85	70

002	john	cse	30	70
003	arun	cse	90	95

SQL> update stud_rec set attendence=25 where regno='001';

1 row updated.

SQL> select * from stud_rec;

REGNO	NAME	MARKS	MAJOR	ATTENDENCE
001	pinky	cse	85	25
002	john	cse	30	70
003	arun	cse	90	95

PROGRAM 4.

Design, develop and execute a program in a language of your choice to implement any one algorithm for mining association rules. Run the program against any large database available in the public domain and discuss the results.

Theory:-

Apriori is a classic algorithm for frequent itemset mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

Steps in execution:

STEP 1: Create two notepad files and name them config.txt and transa.txt respectively.

STEP 2: In config.txt file, insert 3 lines of input

line 1 - number of items per transaction

line 2 - number of transactions

line 3 –minsup

STEP 3: In transa.txt file, which is the transaction file, where the input is separated by space.

STEP 4: These two input files belongs to the class AprioriCalculation.Copy these two input files in the specified path

STEP 5: Run the program in netbeans.

```
package apriori;
import java.io.*;
import java.util.*;

public class Apriori {

    public static void main(String[] args) {
        AprioriCalculationap = new AprioriCalculation();

        ap.aprioriProcess();
    }
}
/******************************************************************
* Class Name : AprioriCalculation
* Purpose        : generate Aprioriitemsets
******************************************************************/
class AprioriCalculation
{
    Vector<String> candidates=new Vector<String>(); //the current candidates
    String configFile="config.txt"; //configuration file
    String transaFile="transa.txt"; //transaction file
    String outputFile="apriori-output.txt";//output file
    intnumItems; //number of items per transaction
    intnumTransactions; //number of transactions
    double minSup; //minimum support for a frequent itemset
    String oneVal[]; //array of value per column that will be treated as a '1'
    String itemSep = " "; //the separator value for items in the database
```

```
/*********************************************************************
 * Method Name : aprioriProcess
 * Purpose     : Generate the aprioriitemsets
 * Parameters : None
 * Return      : None
 *********************************************************************/
public void aprioriProcess()
{
    Date d; //date object for timing purposes
    long start, end; //start and end time
    intitemsetNumber=0; //the current itemset being looked at
    //get config
    getConfig();

    System.out.println("Apriori algorithm has started.\n");

    //start timer
    d = new Date();
    start = d.getTime();

    //while not complete
    do
    {
        //increase the itemset that is being looked at
        itemsetNumber++;

        //generate the candidates
        generateCandidates(itemsetNumber);

        //determine and display frequent itemsets
        calculateFrequentItemsets(itemsetNumber);
        if(candidates.size()!=0)
        {
            System.out.println("Frequent " + itemsetNumber + "-itemsets");
```

```
                System.out.println(candidates);
            }
        //if there are <=1 frequent items, then its the end. This prevents reading through the
database again. When there is only one frequent itemset.
        }while(candidates.size()>1);

        //end timer
        d = new Date();
        end = d.getTime();

        //display the execution time
        System.out.println("Execution time is: "+((double)(end-start)/1000) + " seconds.");
    }

    /**********************************************************************
     * Method Name : getInput
     * Purpose         : get user input from System.in
     * Parameters : None
     * Return          : String value of the users input
     **********************************************************************/
    public static String getInput()
    {
        String input="";
        //read from System.in
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        //try to get users input, if there is an error print the message
        try
        {
            input = reader.readLine();
        }
        catch (Exception e)
        {
            System.out.println(e);
```

```
        }
        return input;
    }

    /*********************************************************************
     * Method Name : getConfig
     * Purpose        : get the configuration information (config filename, transaction
filename)
     *: configFile and transaFile will be change appropriately
     * Parameters : None
     * Return         : None
     *********************************************************************/
    private void getConfig()
    {
        FileWriterfw;
        BufferedWriterfile_out;

        String input="";
        //ask if want to change the config
        System.out.println("Default Configuration: ");
        System.out.println("\tRegular transaction file with "' + itemSep + "' item
separator.");
        System.out.println("\tConfig File: " + configFile);
        System.out.println("\tTransa File: " + transaFile);
        System.out.println("\tOutput File: " + outputFile);
        System.out.println("\nPress 'C' to change the item separator, configuration file and
transaction files");
        System.out.print("or any other key to continue. ");
        input=getInput();

        if(input.compareToIgnoreCase("c")==0)
        {
            System.out.print("Enter new transaction filename (return for "'+transaFile+"'): ");
            input=getInput();
```

```
        if(input.compareToIgnoreCase("")!=0)
            transaFile=input;

        System.out.print("Enter new configuration filename (return for "'+configFile+'"):
");
        input=getInput();
        if(input.compareToIgnoreCase("")!=0)
            configFile=input;

        System.out.print("Enter new output filename (return for "'+outputFile+'"): ");
        input=getInput();
        if(input.compareToIgnoreCase("")!=0)
            outputFile=input;

        System.out.println("Filenames changed");

        System.out.print("Enter the separating character(s) for items (return for
"'+itemSep+'"): ");
        input=getInput();
        if(input.compareToIgnoreCase("")!=0)
            itemSep=input;
}
try
    {
        FileInputStreamfile_in = new FileInputStream(configFile);
        BufferedReaderdata_in = new BufferedReader(new InputStreamReader(file_in));
        //number of items
        numItems=Integer.valueOf(data_in.readLine()).intValue();

        //number of transactions
        numTransactions=Integer.valueOf(data_in.readLine()).intValue();

        //minsup
        minSup=(Double.valueOf(data_in.readLine()).doubleValue());
```

```
            //output config info to the user
            System.out.print("\nInput configuration: "+numItems+" items,
"+numTransactions+"                transactions, ");
            System.out.println("minsup = "+minSup+"%");
            System.out.println();
            minSup/=100.0;

           oneVal = new String[numItems];
           System.out.print("Enter 'y' to change the value each row recognizes as a '1':");
           if(getInput().compareToIgnoreCase("y")==0)
           {
              for(inti=0; i<oneVal.length; i++)
              {
                 System.out.print("Enter value for column #" + (i+1) + ": ");
                 oneVal[i] = getInput();
              }
           }

           else
              for(inti=0; i<oneVal.length; i++)
                 oneVal[i]="1";

           //create the output file
           fw= new FileWriter(outputFile);
           file_out = new BufferedWriter(fw);
           //put the number of transactions into the output file
           file_out.write(numTransactions + "\n");
           file_out.write(numItems + "\n******\n");
           file_out.close();
        }
        //if there is an error, print the message
        catch(IOException e)
        {
```

```
            System.out.println(e);
        }
    }

    /**********************************************************************
     * Method Name : generateCandidates
     * Purpose         : Generate all possible candidates for the n-thitemsets
     *                 : these candidates are stored in the candidates class vector
     * Parameters : n - integer value representing the current itemsets to be created
     * Return          : None
     **********************************************************************/
    private void generateCandidates(int n)
    {
        Vector<String>tempCandidates = new Vector<String>(); //temporary candidate
string vector
        String str1, str2; //strings that will be used for comparisons
        StringTokenizer st1, st2; //string tokenizers for the two itemsets being compared
            //if its the first set, candidates are just the numbers
        if(n==1)
        {
            for(inti=1; i<=numItems; i++)
            {
                tempCandidates.add(Integer.toString(i));
            }
        }
        else if(n==2) //second itemset is just all combinations of itemset 1
        {
            //add each itemset from the previous frequent itemsets together
            for(inti=0; i<candidates.size(); i++)
            {
                st1 = new StringTokenizer(candidates.get(i));
                str1 = st1.nextToken();
                for(int j=i+1; j<candidates.size(); j++)
                {
```

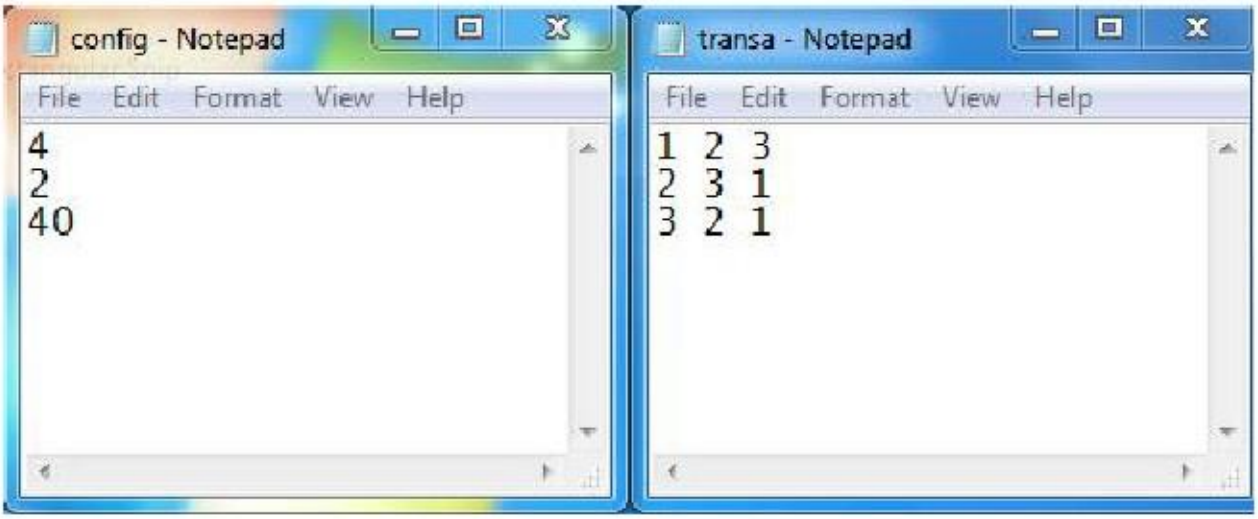
```
                st2 = new StringTokenizer(candidates.elementAt(j));
                str2 = st2.nextToken();
                tempCandidates.add(str1 + " " + str2);
            }
        }
    }
    else
    {
        //for each itemset
        for(inti=0; i<candidates.size(); i++)
        {
            //compare to the next itemset
            for(int j=i+1; j<candidates.size(); j++)
            {
                //create the strigns
                str1 = new String();
                str2 = new String();
                //create the tokenizers
                st1 = new StringTokenizer(candidates.get(i));
                st2 = new StringTokenizer(candidates.get(j));

                //make a string of the first n-2 tokens of the strings
                for(int s=0; s<n-2; s++)
                {
                    str1 = str1 + " " + st1.nextToken();
                    str2 = str2 + " " + st2.nextToken();
                }

                //if they have the same n-2 tokens, add them together
                if(str2.compareToIgnoreCase(str1)==0)
                    tempCandidates.add((str1 + " " + st1.nextToken() + " " +
st2.nextToken()).trim());
            }
        }
```

```
        }
        //clear the old candidates
        candidates.clear();
        //set the new ones
        candidates = new Vector<String>(tempCandidates);
        tempCandidates.clear();
    }

  /**********************************************************************
     * Method Name : calculateFrequentItemsets
     * Purpose         : Determine which candidates are frequent in the n-thitemsets
     * from all possible candidates
     * Parameters : n - iteger representing the current itemsets being evaluated
     **********************************************************************/
    private void calculateFrequentItemsets(int n)
    {
        Vector<String>frequentCandidates = new Vector<String>(); //the frequent
candidates for the current itemset
        FileInputStreamfile_in; //file input stream
        BufferedReaderdata_in; //data input stream
        FileWriterfw;
        BufferedWriterfile_out;
        StringTokenizerst, stFile; //tokenizer for candidate and transaction
        boolean match; //whether the transaction has all the items in an itemset
        boolean trans[] = new boolean[numItems]; //array to hold a transaction so that can be
checked
        int count[] = new int[candidates.size()]; //the number of successful matches

        try
        {
                //output file
                fw= new FileWriter(outputFile, true);
                file_out = new BufferedWriter(fw);
                //load the transaction file
```

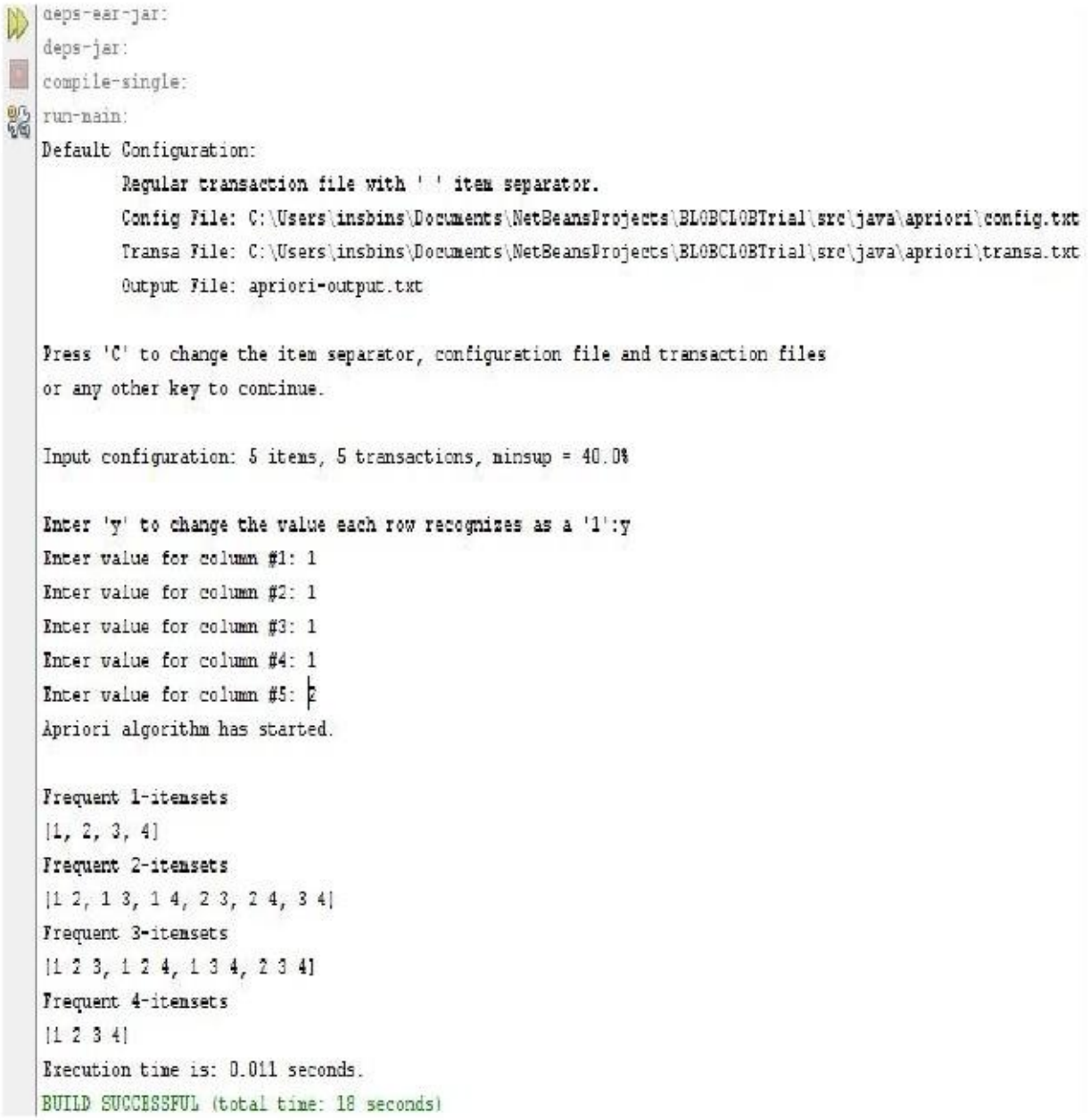
```
            file_in = new FileInputStream(transaFile);
            data_in = new BufferedReader(new InputStreamReader(file_in));

            //for each transaction
            for(inti=0; i<numTransactions; i++)
            {
                //System.out.println("Got here " + i + " times"); //useful to debug files that
you are unsure of the number of line
                stFile = new StringTokenizer(data_in.readLine(), itemSep); //read a line from
the file to the tokenizer
                //put the contents of that line into the transaction array
                for(int j=0; j<numItems; j++)
                {
                    trans[j]=(stFile.nextToken().compareToIgnoreCase(oneVal[j])==0); //if it
is not a 0, assign the value to true
                }

                //check each candidate
                for(int c=0; c<candidates.size(); c++)
                {
                    match = false; //reset match to false
                    //tokenize the candidate so that we know what items need to be present for
a match
                    st = new StringTokenizer(candidates.get(c));
                    //check each item in the itemset to see if it is present in the transaction
                    while(st.hasMoreTokens())
                    {
                        match = (trans[Integer.valueOf(st.nextToken())-1]);
                        if(!match) //if it is not present in the transaction stop checking
                            break;
                    }
                    if(match) //if at this point it is a match, increase the count
                        count[c]++;
                }
```

```
                }
                for(inti=0; i<candidates.size(); i++)
                {
                    // System.out.println("Candidate: " + candidates.get(c) + " with count: " +
count + " % is: " + (count/(double)numItems));
                    //if the count% is larger than the minSup%, add to the candidate to the
frequent candidates
                    if((count[i]/(double)numTransactions)>=minSup)
                    {
                        frequentCandidates.add(candidates.get(i));
//put the frequent itemset into the output file
                        file_out.write(candidates.get(i) + "," + count[i]/(double)numTransactions +
"\n");
                    }
                }
                file_out.write("-\n");
                file_out.close();
        }
        //if error at all in this process, catch it and print the error messate
        catch(IOException e)
        {
            System.out.println(e);
        }
        //clear old candidates
        candidates.clear();
        //new candidates are the old frequent candidates
        candidates = new Vector<String>(frequentCandidates);
        frequentCandidates.clear();
    }
}
```

Input:


config - Notepad
File Edit Format View Help
4
2
40
transa - Notepad
File Edit Format View Help
1 2 3
2 3 1
3 2 1

Output:-


deps-ear-jar:
deps-jar:
compile-single:
run-main:
Default Configuration:
Regular transaction file with ' ' item separator.
Config File: C:\Users\insbins\Documents\NetBeansProjects\BL0BCL0BTrial\src\java\apriori\config.txt
Transa File: C:\Users\insbins\Documents\NetBeansProjects\BL0BCL0BTrial\src\java\apriori\transa.txt
Output File: apriori-output.txt
Press 'C' to change the item separator, configuration file and transaction files
or any other key to continue.
Input configuration: 5 items, 5 transactions, minsup = 40.0%
Enter 'y' to change the value each row recognizes as a '1':y
Enter value for column #1: 1
Enter value for column #2: 1
Enter value for column #3: 1
Enter value for column #4: 1
Enter value for column #5: 2
Apriori algorithm has started.
Frequent 1-itemsets
[1, 2, 3, 4]
Frequent 2-itemsets
[1 2, 1 3, 1 4, 2 3, 2 4, 3 4]
Frequent 3-itemsets
[1 2 3, 1 2 4, 1 3 4, 2 3 4]
Frequent 4-itemsets
[1 2 3 4]
Execution time is: 0.011 seconds.
BUILD SUCCESSFUL (total time: 18 seconds)