JAVA AWT BASED-PLICKER CARD IMPLEMENTATION - SQL CONNECTIVITY USING JDBC

A

Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

V.RENU AAKANKSHA <1602-18-737-093>

Department of Information Technology

Vasavi College of Engineering (Autonomous)

Ibrahimbagh, Hyderabad-31

2020

BONAFIDE CERTIFICATE

This is to certify that the project report titled ”***Plicker Card Implementation Database Management System*** ” is the bonafide project work of ***Ms. Renu Aakanksha Veesam*** bearing hall ticket no. *1602-18-737-093* who carried out this project work under my supervision in the IV semester for the academic year 2019-2020

Signature

External Examiner

Signature

Internal Examiner

V.Renu Aakanksha
1602-18-737-093

AIM:

To create a Java GUI based registration form which takes the values like: Student name, Instructor name, Admin name, Student ID, Instructor ID, Admin ID, Student score, Student remarks, Subject taught by the Instructor, Instructor Experience, Student's class, class for which Instructor teaches, Question no: , Question, Marks for a particular question, Answer no: , four options for a given question are the details taken from the user

ABSTRACT

Plickers is a web tool that can be used for formative assessment even in a low technology classroom and hence found immensely beneficial to classrooms in rural south India where restrictions marginalise the 'Bring Your Own Device' concept. For its application, it requires only one computer, the Plickers mobile app installed to a smart phone and Plickers assessment cards. It has been proved to be an effective, timesaving, easy to use edu tech tool that can be used in Indian classrooms

V.Renu Aakanksha
1602-18-737-093

INTRODUCTION

REQUIREMENT ANALYSIS

We require a total of 5 tables in order to keep a track of that database. One to store the details of the student, another for the details of the admin , one for instructor details and other 2 to store the details of the question and its options with correct option. The basic attributes are id, name of any entity, besides this, descriptive attributes are also present. Entity name can have a data type of char for attributes like name, question options to choose correct answer, student remarks, subject taught by the instructor. Number for score , class, exp and vachar2 for id.

The relationship between various entity sets helps in retrieval of the information and feedback of the queries.

List of entities with their attributes and domain types

Student : id number(5) (primary key)

name char(20)

Score number(5)

Remarks char(20)

Class number(5)

Instructor : id number(5) (primary key)

Name char(20)

Subject char(20)

Experience number(5)

Class number(5)

Admin : id number(5) (primary key)

Name char(20)

Question : no. number(5) (primary key)

Question char(1000)

Marks awarded number(5)

Answer : id number(5) (primary key)

Opt1 char(20)

Opt2 char(20)

Opt3 char(20)

Opt4 char(20)

Crct_answer : id number (5) foreign key from answer

No. number(5) foreign key from question

Taught : id number(5) foreign key from studentd

Id number(5) foreign key from instructor

Subject char(20)

Manage : id number(5) foreign key from student

Id number(5) foreign key from admin

Given_ques: no. number(5) foreign key from question

Id number(5) foreign key from studentd

MAPPING CARDINALITIES AND PARTICIPATION CONSTRAINTS

Student details (many) given_ques question(many)-total participation

Student details(many) manage admin(one) – partial participation

Question(one) crct-ans answer(one)- total participation

Student details (many) taught Instructor(many)- partial participation

SPECIAL GOAL OF THE PROJECT

This project shows a different way of learning which makes studies interesting to both teachers and learners.

This project is made such that data is secure from all kinds of inputs and is safely stored in the database.

This project is user friendly .Even **a** non programmer can easily use the application.

➢ Architecture and technology used:

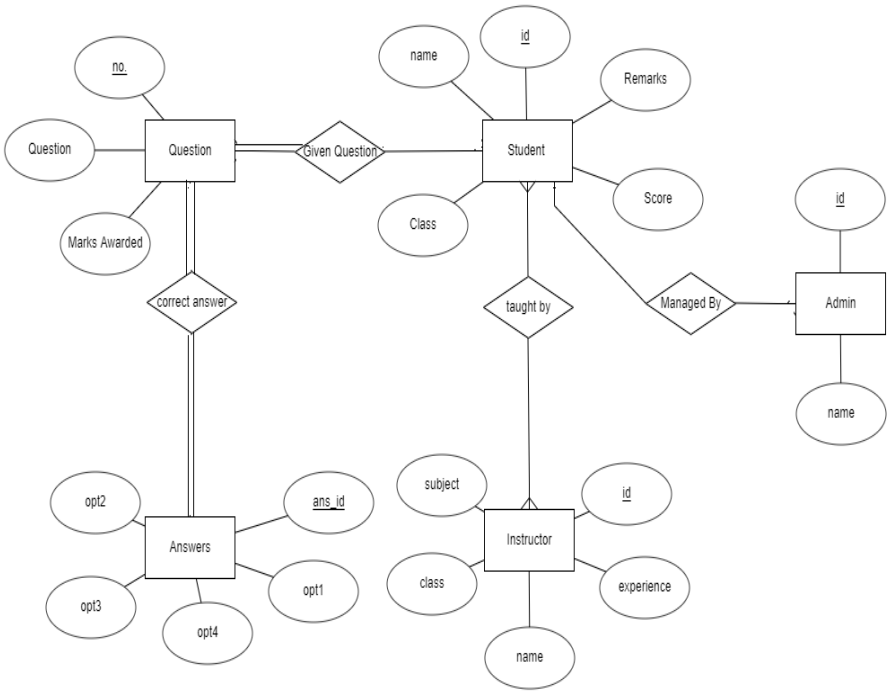
SQL Plus is the most basic Oracle Database utility with a basic command-line interface, commonly used by users, administrators and programmers.

The interface of SQL Plus is used for creating the database. DDL and DML commands are implemented for operations being executed. The details of various Online MOOC's provider, courses, student, assignments, and results are stored in the form of tables in the database.

Eclipse is an integrated development environment(IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Erlang, JavaScripts etc.

The front end application code is written in "**Java**" using Eclipse. The portal for front end application is designed through Eclipse, runs and has the capacity to connect with the database which has data inserted using SQL.

ENTITY RELATIONSHIP DIAGRAM



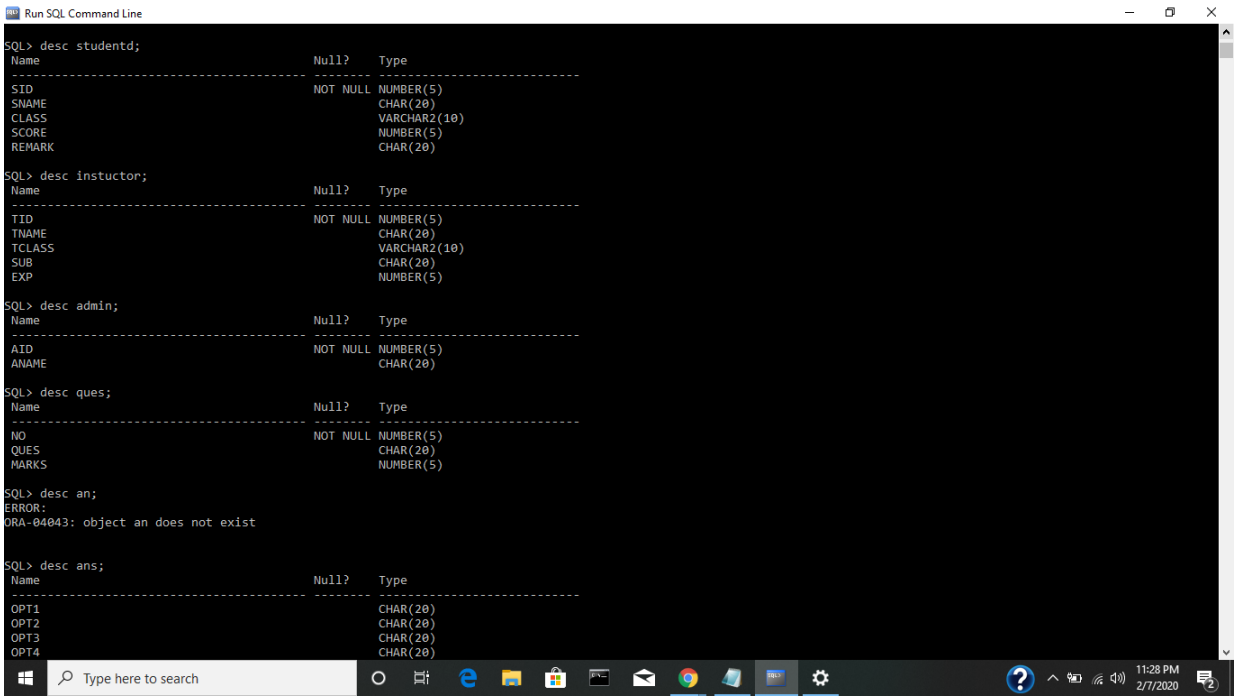
V.Renu Aakanksha
1602-18-737-093

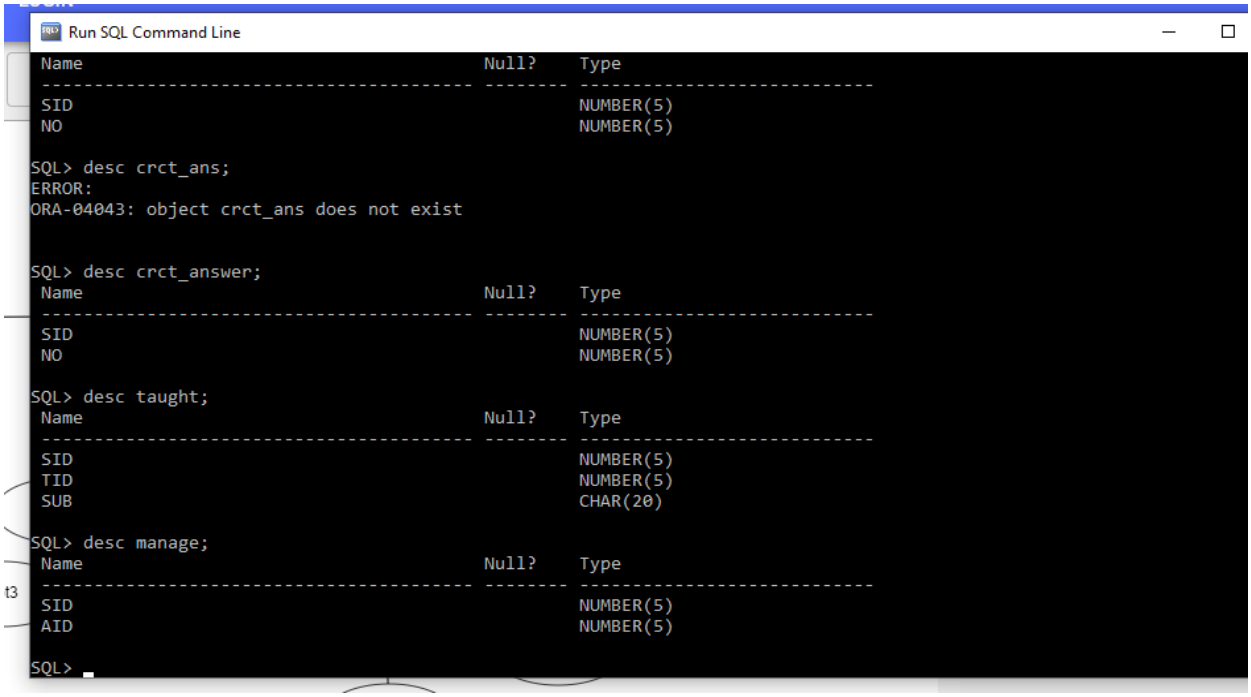
DDL COMMANDS:

```
SQL*Plus: Release 11.2.0.2.0 Production on Fri Feb 7 23:02:21 2020

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

SQL> conn system;
Enter password:
Connected.
SQL> create table studentD(sid number(5),sname char(20),class varchar2(10),score number(5),remark char(20));

Table created.

SQL> create table instuctor(tid number(5),tname char(20),tclass varchar(10),sub char(20),exp number(5));

Table created.

SQL> create table admin(aid number(5),aname char(20));

Table created.

SQL> create table ques(no number(5),ques char(20),marks number(5));

Table created.

SQL> create table ans(opt1 char(20),opt2 char(20),opt3 char(20),opt4 char(20));

Table created.

SQL>
```

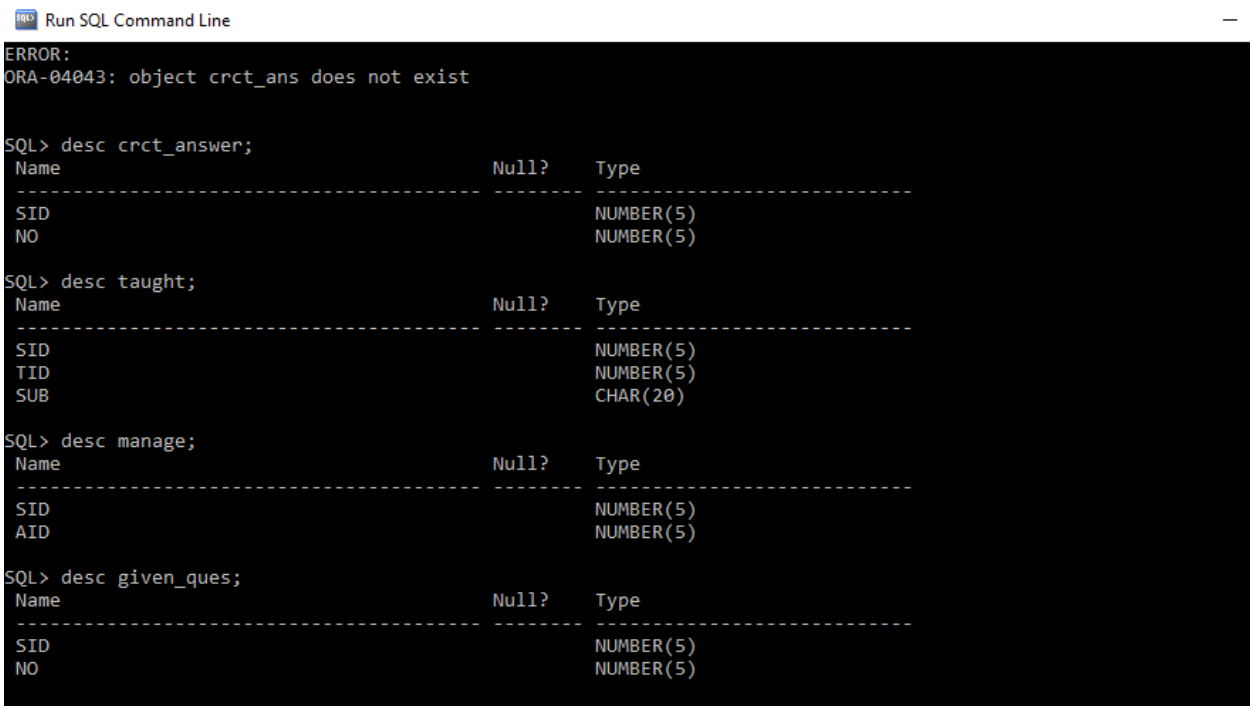
V.Renu Aakanksha
1602-18-737-093

PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT



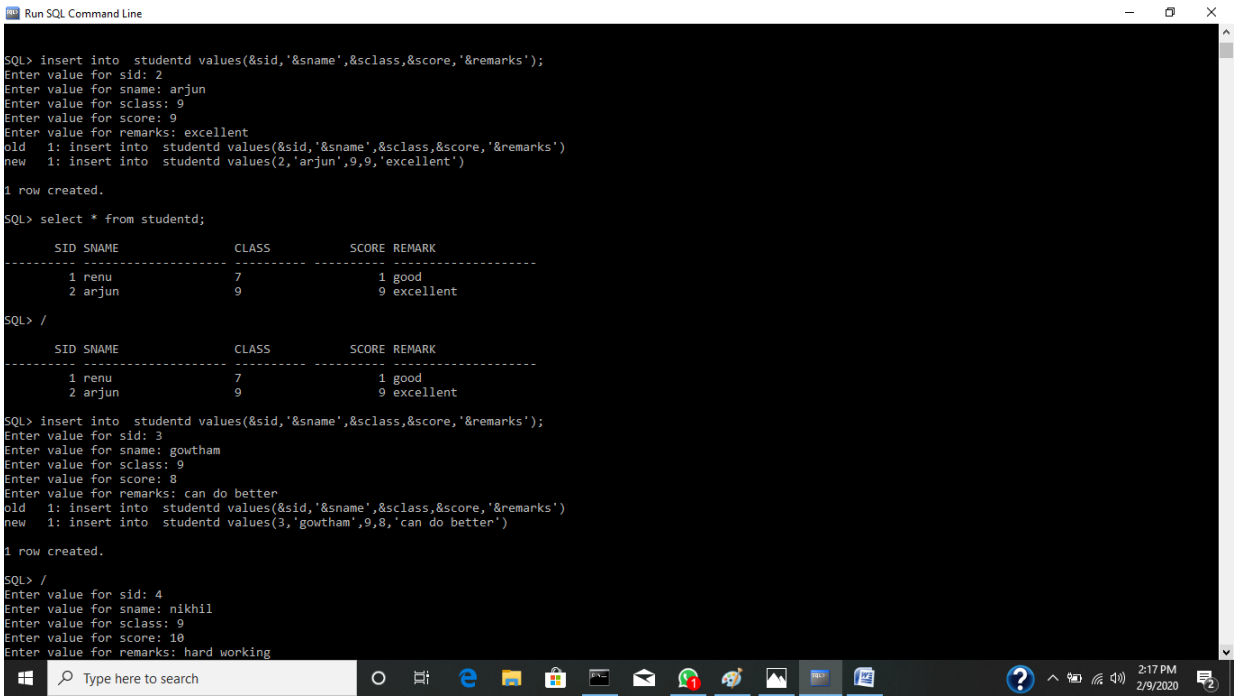


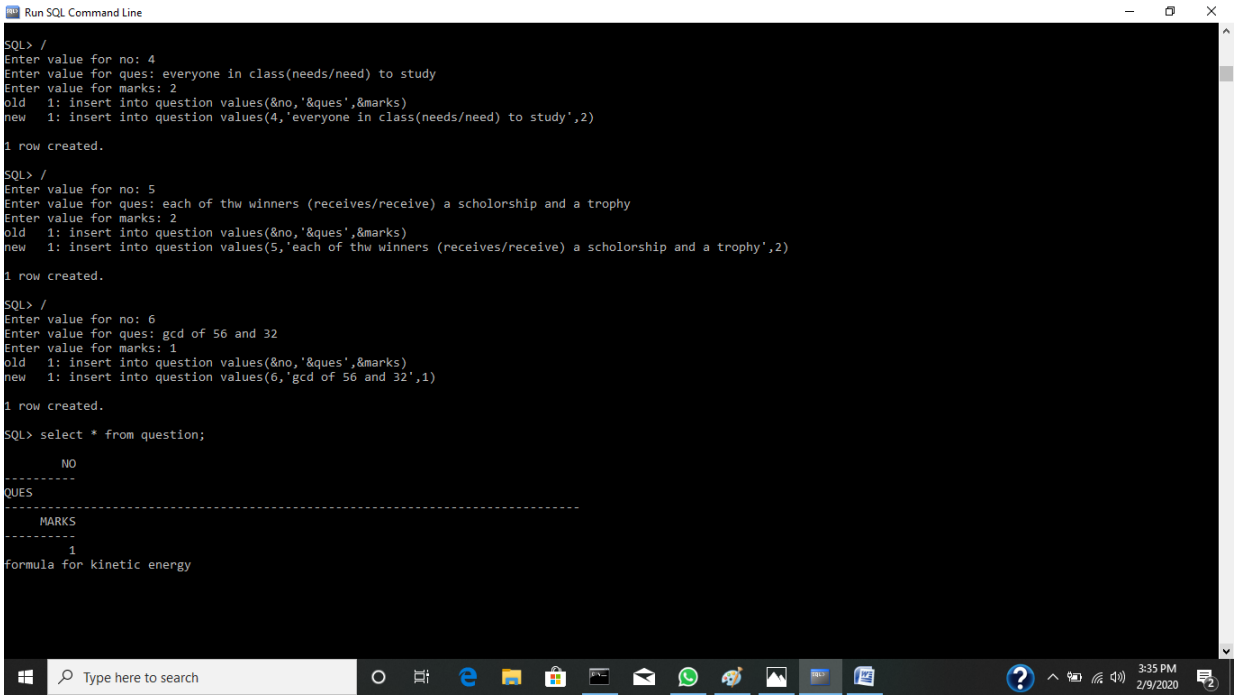
V.Renu Aakanksha
1602-18-737-093



DML COMMANDS

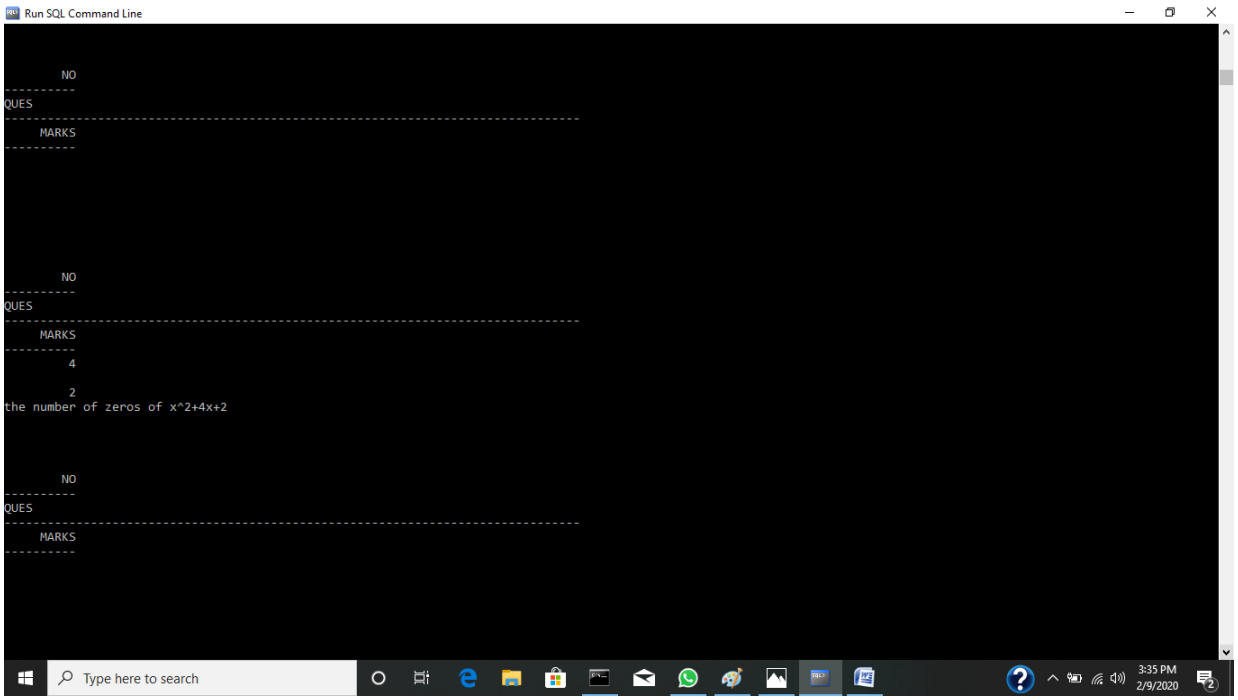
PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT

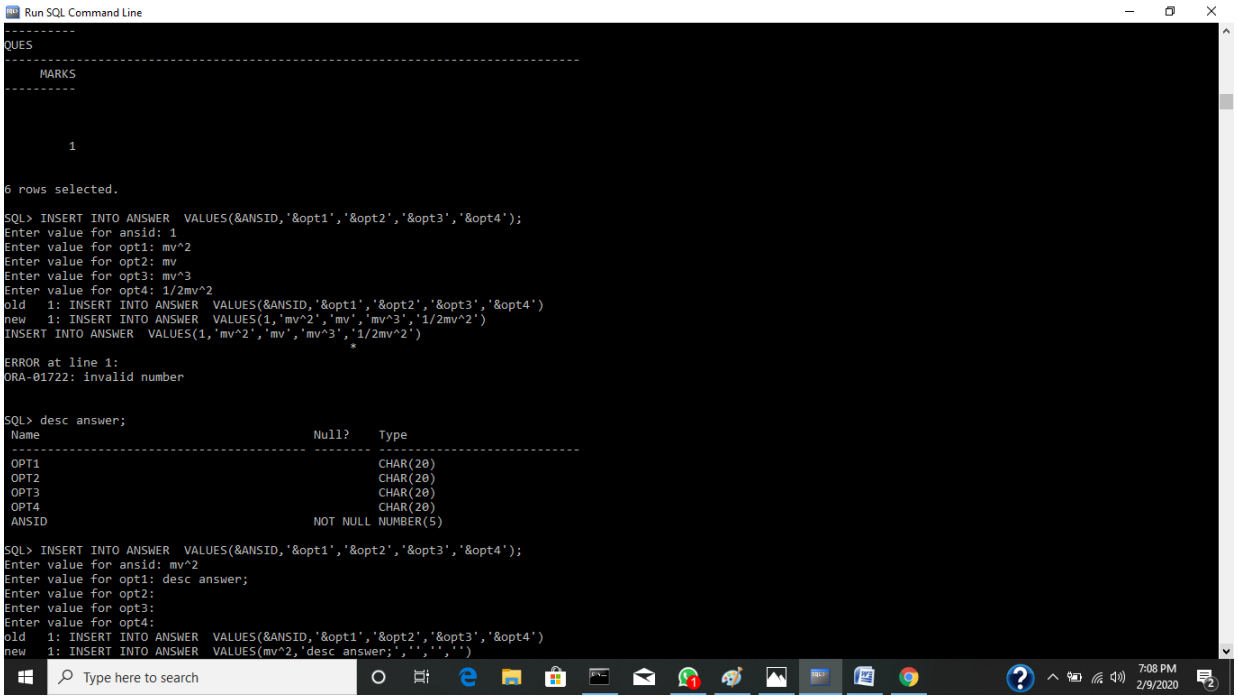




V.Renu Aakanksha
1602-18-737-093

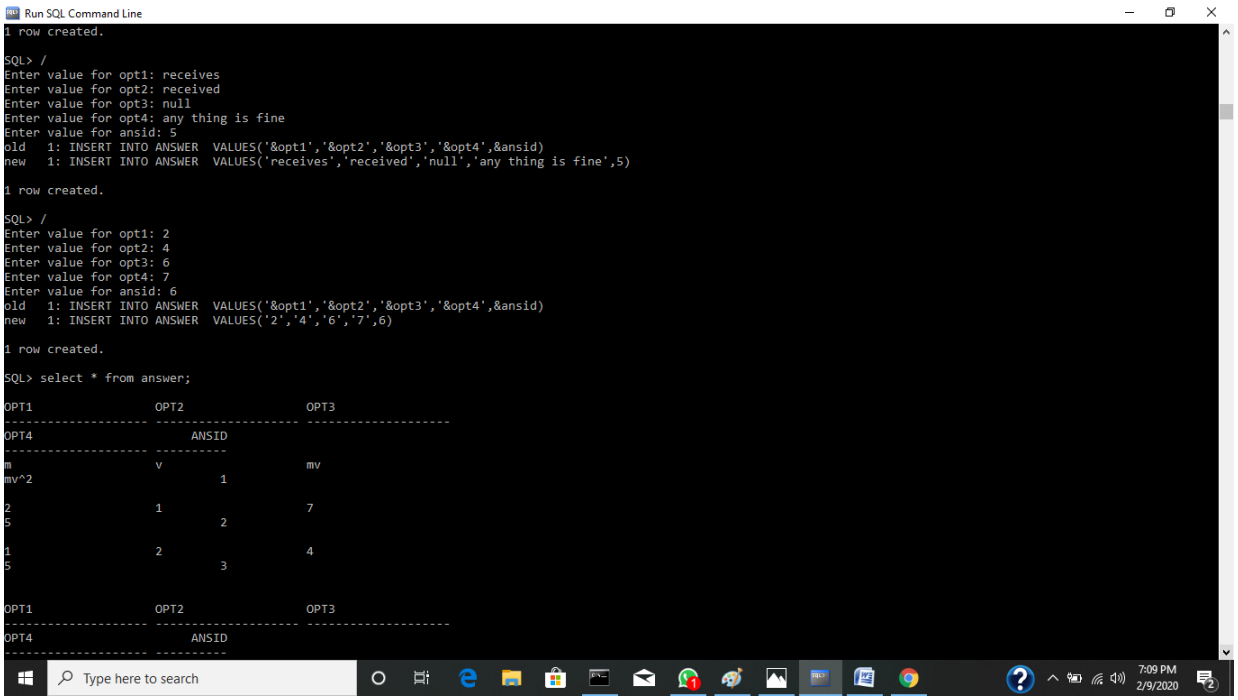
PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT

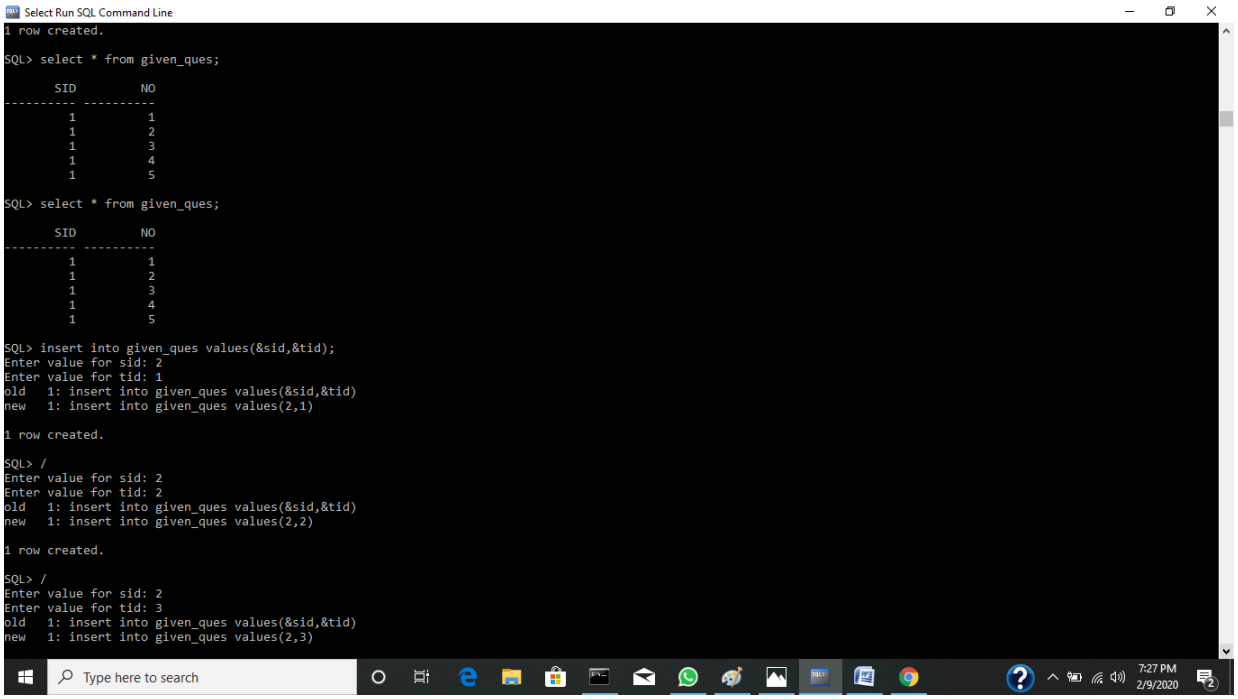




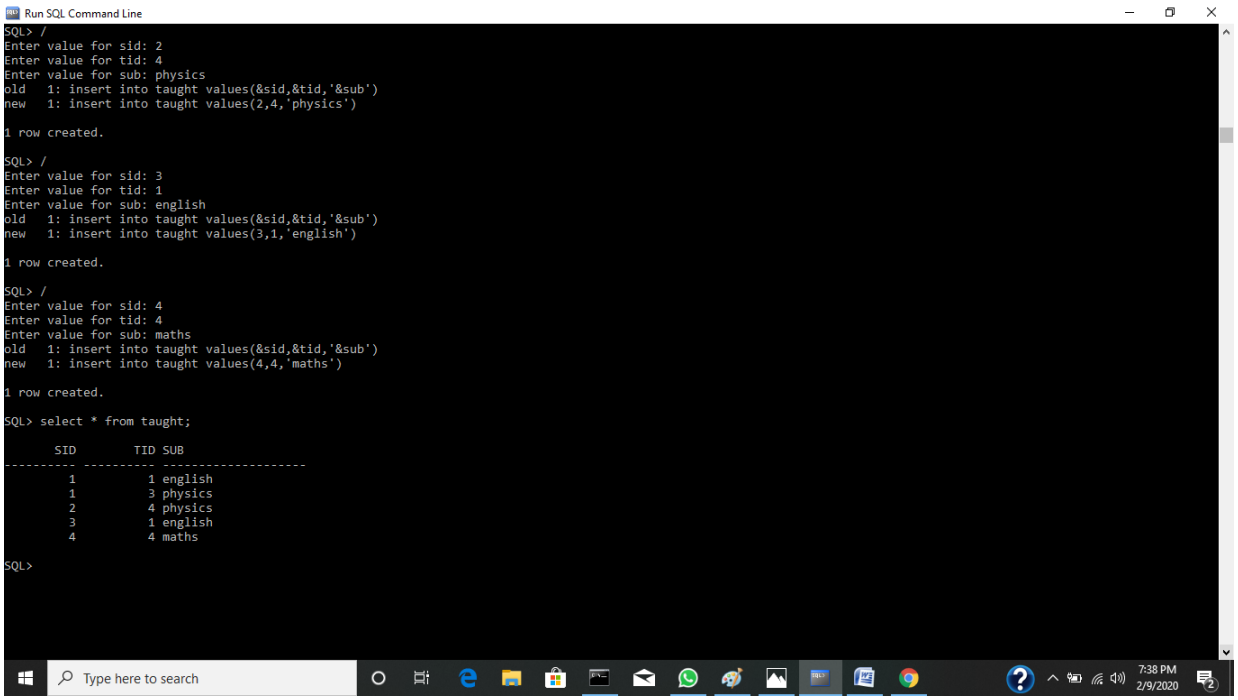
V.Renu Aakanksha
1602-18-737-093

PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT





V.Renu Aakanksha
1602-18-737-093



IMPLEMENTATION

FRONT END PROGRAMS:

INSERT STUDENT DETAILS:

```
import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class InsertStudent extends Panel

{

        Button insertStudentButton;

        TextField sidText, snameText, classText, scoreText,remarkText;
```

```
        TextArea errorText;
        Connection connection;
        Statement statement;
        public InsertStudent()
        {
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                }
                catch (Exception e)
                {
                        System.err.println("Unable to find and load driver");
                        System.exit(1);
                }
                connectToDB();
        }
        public void connectToDB()
    {
        try
                {
                   connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
                   statement = connection.createStatement();
                }
                catch (SQLException connectException)
                {
```

```
                    System.out.println(connectException.getMessage());
                    System.out.println(connectException.getSQLState());
                    System.out.println(connectException.getErrorCode());
                    System.exit(1);
                }
    }
        public void buildGUI()
        {
                //Handle Insert Account Button
                insertStudentButton = new Button("Insert Student");
                insertStudentButton.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                try
                                {
                                    Statement statement = connection.createStatement();
                                    //String query = "INSERT INTO sailors (SID,SNAME, RATING, AGE)
VALUES (2,'Divya',7,20)";
                                    String query= "INSERT INTO studentd VALUES(" + sidText.getText() +
", " + "'" + snameText.getText() + "'," + classText.getText() + "," + scoreText.getText() +",'" +
remarkText.getText() + "'"+ ")";
                                    int i = statement.executeUpdate(query);
                                    errorText.append("\nInserted " + i + " rows successfully");
                                }
                                catch (SQLException insertException)
```

```
                    {
                        displaySQLErrors(insertException);
                    }
                }
            });
            sidText = new TextField(15);
            snameText = new TextField(15);
            classText = new TextField(15);
            scoreText = new TextField(15);
            remarkText = new TextField(15);

            errorText = new TextArea(10, 4);
            errorText.setEditable(false);

            Panel first = new Panel();
            first.setLayout(new GridLayout(5, 1));
            first.add(new Label("Student ID:"));
            first.add(sidText);
            first.add(new Label("Name:"));
            first.add(snameText);
            first.add(new Label("Class:"));
            first.add(classText);
            first.add(new Label("Score:"));
            first.add(scoreText);
            first.add(new Label("Remark:"));
```

```
                first.add(remarkText);
                first.setBounds(125,90,200,100);
                Panel second = new Panel(new GridLayout(4, 1));
                second.add(insertStudentButton);
           second.setBounds(125,220,150,100);
                Panel third = new Panel();
                third.add(errorText);
                third.setBounds(125,320,300,200);


                setLayout(null);


                add(first);
                add(second);
                add(third);


                setSize(500, 600);
                setVisible(true);
        }
        private void displaySQLErrors(SQLException e)
        {
                errorText.append("\nSQLException: " + e.getMessage() + "\n");
                errorText.append("SQLState:          " + e.getSQLState() + "\n");
                errorText.append("VendorError:    " + e.getErrorCode() + "\n");
        }
        public static void main(String[] args)
```

```
	{
		InsertStudent std = new InsertStudent();
		std.buildGUI();
	}
}
```

MODIFY STUDENT DETAILS:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateStudent extends Panel
{
	Button updateStudentButton;
	List studentIDList;
	TextField sidText, snameText, scoreText, remarkText, classText;
	TextArea errorText;
	Connection connection;
	Statement statement;
	ResultSet rs;

	public UpdateStudent()
	{
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
                }
                catch (Exception e)
                {
                        System.err.println("Unable to find and load driver");
                        System.exit(1);
                }
                connectToDB();
        }
        public void connectToDB()
    {   try     {
                  connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
                  statement = connection.createStatement();
                }
                catch (SQLException connectException)
                {
                                System.out.println(connectException.getMessage());
                        System.out.println(connectException.getSQLState());
                  System.out.println(connectException.getErrorCode());
                  System.exit(1);
                }
    }
        private void loadStudent()
        {
        try
                {
                  rs = statement.executeQuery("SELECT * FROM studentd");
```

```
                while (rs.next())
                {
                        studentIDList.add(rs.getString("SID"));
                }
            }
            catch (SQLException e)
            {
                displaySQLErrors(e);
            }
        }
        public void buildGUI()
        {
            studentIDList = new List(6);
            loadStudent();
            add(studentIDList);
            //When a list item is selected populate the text fields
            studentIDList.addItemListener(new ItemListener()
            {
                    public void itemStateChanged(ItemEvent e)
                    {
                            try
                            {
                                    rs = statement.executeQuery("SELECT * FROM studentd");
                                    while (rs.next())
                                    {
                                            if
(rs.getString("SID").equals(studentIDList.getSelectedItem()))
```

```
                                        break;
                                }
                                if (!rs.isAfterLast())
                                {
                                        sidText.setText(rs.getString("SID"));
                                        snameText.setText(rs.getString("SNAME"));
                                        classText.setText(rs.getString("CLASS"));
                                        scoreText.setText(rs.getString("SCORE"));
                                        remarkText.setText(rs.getString("REMARK"));
                                }
                        }
                        catch (SQLException selectException)
                        {
                                displaySQLErrors(selectException);
                        }
                }
        });
        //Handle Update Sailor Button
        updateStudentButton = new Button("Update Student");
        updateStudentButton.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                                Statement statement = connection.createStatement();
```

```
                    int i = statement.executeUpdate("UPDATE studentd "
                    + "SET sname='" + snameText.getText()   + "', "
                    + "class=" + classText.getText() +   ", "
                    + "remark='" + remarkText.getText() +   "', "
                    + "score=" + scoreText.getText() +" WHERE sid = "
                    + studentIDList.getSelectedItem());
                    errorText.append("\nUpdated " + i + " rows successfully");
                    studentIDList.removeAll();
                    loadStudent();
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
                }
            }
        });

        sidText = new TextField(15);
        sidText.setEditable(false);
        snameText = new TextField(15);
        classText = new TextField(15);
        scoreText = new TextField(15);
        remarkText = new TextField(15);
        errorText = new TextArea(10, 40);
        errorText.setEditable(false);
        Panel first = new Panel();
```

```
            first.setLayout(new GridLayout(5, 1));
            first.add(new Label("Student ID:"));
            first.add(sidText);
            first.add(new Label("Name :"));
            first.add(snameText);
            first.add(new Label("class :"));
            first.add(classText);
            first.add(new Label("Score :"));
            first.add(scoreText);
            first.add(new Label("Remark :"));
            first.add(remarkText);
            Panel second = new Panel(new GridLayout(4, 1));
            second.add(updateStudentButton);
            Panel third = new Panel();
            third.add(errorText);
            add(first);
            add(second);
            add(third);
            setSize(500, 600);
            setLayout(new FlowLayout());
            setVisible(true);
        }
        private void displaySQLErrors(SQLException e)
        {
            errorText.append("\nSQLException: " + e.getMessage() + "\n");
            errorText.append("SQLState:      " + e.getSQLState() + "\n");
```

```
                errorText.append("VendorError:    " + e.getErrorCode() + "\n");
        }


        public static void main(String[] args)
        {
                UpdateStudent ups = new UpdateStudent();
                ups.buildGUI();
        }
}
```

DELETE STUDENT DETAILS :

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteStudent extends Panel
{
        Button deleteStudentButton;
        List studentIDList;
        TextField sidText, snameText, scoreText, remarkText,classText;
        TextArea errorText;
        Connection connection;
        Statement statement;
        ResultSet rs;
        public DeleteStudent()
        {
                try
                {
```

```
                Class.forName("oracle.jdbc.driver.OracleDriver");
            }
            catch (Exception e)
            {
                System.err.println("Unable to find and load driver");
                System.exit(1);
            }
            connectToDB();
        }
        public void connectToDB()
    {
            try
            {
              connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
              statement = connection.createStatement();
            }
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
    }


        private void loadStudent()
```

```
        {
                try
                {
                  rs = statement.executeQuery("SELECT * FROM studentd");
                  while (rs.next())
                  {
                        studentIDList.add(rs.getString("SID"));
                  }
                }
                catch (SQLException e)
                {
                  displaySQLErrors(e);
                }
        }
        public void buildGUI()
        {
             studentIDList = new List(10);
                loadStudent();
                add(studentIDList);


                //When a list item is selected populate the text fields
                studentIDList.addItemListener(new ItemListener()
                {
                        public void itemStateChanged(ItemEvent e)
                        {
                                try
```

```
                                {
                                        rs = statement.executeQuery("SELECT * FROM studentd");
                                        while (rs.next())
                                        {
                                                if
(rs.getString("SID").equals(studentIDList.getSelectedItem()))
                                                break;
                                        }
                                        if (!rs.isAfterLast())
                                        {
                                                sidText.setText(rs.getString("SID"));
                                                snameText.setText(rs.getString("SNAME"));
                                                scoreText.setText(rs.getString("SCORE"));
                                                remarkText.setText(rs.getString("REMARK"));
                                                classText.setText(rs.getString("CLASS"));
                                        }
                                }
                                catch (SQLException selectException)
                                {
                                        displaySQLErrors(selectException);
                                }
                        }
                });



                //Handle Delete Sailor Button
                deleteStudentButton = new Button("Delete Student");
```

```
deleteStudentButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("DELETE FROM studentd
WHERE SID = "
                    + studentIDList.getSelectedItem());
            errorText.append("\nDeleted " + i + " rows successfully");
            sidText.setText(null);
            snameText.setText(null);
            scoreText.setText(null);
            remarkText.setText(null);
            studentIDList.removeAll();
            loadStudent();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});
sidText = new TextField(15);
snameText = new TextField(15);
scoreText = new TextField(15);
```

```
remarkText = new TextField(15);

classText = new TextField(15);


errorText = new TextArea(10, 40);

errorText.setEditable(false);


Panel first = new Panel();

first.setLayout(new GridLayout(5, 1));

first.add(new Label("Student ID:"));

first.add(sidText);

first.add(new Label("Name:"));

first.add(snameText);

first.add(new Label("Score:"));

first.add(scoreText);

first.add(new Label("Remarks:"));

first.add(remarkText);

first.add(new Label("Class:"));

first.add(classText);


Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteStudentButton);


Panel third = new Panel();

third.add(errorText);


add(first);
```

```
            add(second);
            add(third);

            setSize(450, 600);
            setLayout(new FlowLayout());
            setVisible(true);

        }

        private void displaySQLErrors(SQLException e)
        {
            errorText.append("\nSQLException: " + e.getMessage() + "\n");
            errorText.append("SQLState:      " + e.getSQLState() + "\n");
            errorText.append("VendorError:   " + e.getErrorCode() + "\n");
        }
        public static void main(String[] args)
        {
            DeleteStudent dels = new DeleteStudent();
            dels.buildGUI();
        }
}
```

MAIN PROGRAM:

import java.awt.*;

import java.awt.event.*;

```
class plickerCard extends Frame implements ActionListener
{
        String msg = "";
        Label ll;
        CardLayout cardLO;
        //Create Panels for each of the menu items, welcome screen panel and home screen panel with CardLayout
        InsertStudent is;
        UpdateStudent ups;
        DeleteStudent dels;
        InsertInstructor ii;
        UpdateInstructor ui;
        DeleteInstructor deli;
        taught tb;
        UpdateTaught upt;
        DeleteTaught delt;
        Manage_By mb;
        UpdateManage upm;
        DeleteManage delm;
            InsertAdmin ia;
            UpdateAdmin upa;
            DeleteAdmin dela;
            InsertQuestion iq;
            UpdateQuestion upq;
            DeleteQuestion delq;
            InsertAnswer ian;
            UpdateAnswer upan;
```

```
        DeleteAnswer delan;
        crct_answer ca;
        UpdateCrctAnswer upansid;
        DeleteCrctAns delca;
        given_ques gq;
        UpdateGiven upg;
        DeleteGivenQues delgq;
    Panel home,welcome;
    plickerCard()
    {
            cardLO = new CardLayout();
            //Create an empty home panel and set its layout to card layout
            home = new Panel();
            home.setLayout(cardLO);
            ll = new Label();
            ll.setAlignment(Label.CENTER);
            ll.setText("Welcome to PLICKER CARD");
            //Create welcome panel and add the label to it
            welcome = new Panel();
            welcome.add(ll);

            //create panels for each of our menu items and build them with
respective components
            is = new InsertStudent(); is.buildGUI();
            ups = new UpdateStudent(); ups.buildGUI();
            dels = new DeleteStudent(); dels.buildGUI();
            ui = new UpdateInstructor();        ui.buildGUI();
```

```
ii= new InsertInstructor();    ii.buildGUI();
deli= new DeleteInstructor(); deli.buildGUI();
tb= new taught();      tb.buildGUI();
upt=new UpdateTaught(); upt.buildGUI();
ia=new InsertAdmin(); ia.buildGUI();
upa=new UpdateAdmin(); upa.buildGUI();
dela=new DeleteAdmin();    dela.buildGUI();
iq=new InsertQuestion(); iq.buildGUI();
upq=new UpdateQuestion(); upq.buildGUI();
delq=new DeleteQuestion(); delq.buildGUI();
ian=new InsertAnswer(); ian.buildGUI();
upan=new UpdateAnswer(); upan.buildGUI();
delan=new DeleteAnswer(); delan.buildGUI();
ca=new crct_answer(); ca.buildGUI();
upansid=new UpdateCrctAnswer(); upansid.buildGUI();
delca=new DeleteCrctAns(); delca.buildGUI();
gq=new given_ques(); gq.buildGUI();
upg=new UpdateGiven(); upg.buildGUI();
delgq=new DeleteGivenQues(); delgq.buildGUI();
mb=new Manage_By(); mb.buildGUI();
upm=new UpdateManage(); upm.buildGUI();
delm=new DeleteManage(); delm.buildGUI();
 delt=new DeleteTaught(); delt.buildGUI();
//add all the panels to the home panel which has a cardlayout
home.add(welcome, "Welcome");
home.add(is, "InsertStudent");
```

```
home.add(ups, "UpdateStudent");
home.add(dels, "DeleteStudent");
home.add(ii,"InsertInstructor");
home.add(ui, "UpdateInstructor");
home.add(deli,"DeleteInstructor");
home.add(ia,"InsertAdmin");
home.add(upa,"UpdateAdmin");
home.add(dela,"DeleteAdmin");
home.add(iq,"InsertQuestion");
home.add(upq,"UpdateQuestion");
home.add(delq,"DeleteQuestion");
home.add(ian,"InsertAnswer");
home.add(upan,"UpdateAnswer");
home.add(delan,"DeleteAnswer");
home.add(tb, "taught");
home.add(upt,"UpdateTaught");
home.add(ca,"crct_answer");
home.add(upansid,"UpdateCrctAnswer");
home.add(gq,"given_ques");
home.add(upg,"UpdateGiven");
home.add(mb,"Manage_By");
home.add(upm,"UpdateManage");
home.add(delm,"DeleteManage");
home.add(delt,"DeleteTaught");
home.add(delca,"DeleteCrctAns");
home.add(delgq,"DeleteGivenQues");
```

```
// add home panel to main frame
add(home);


// create menu bar and add it to frame
MenuBar mbar = new MenuBar();
setMenuBar(mbar);
// create the menu items and add it to Menu
Menu Student = new Menu("Student Details");
MenuItem item1, item2, item3;
Student.add(item1 = new MenuItem("Submit Student Details"));
Student.add(item2 = new MenuItem("Modify Student details"));
Student.add(item3 = new MenuItem("Delete Student Details"));
mbar.add(Student);
Menu Instructor = new Menu("Instructor Details");
MenuItem item4, item5, item6;
Instructor.add(item4 = new MenuItem("Submit Instructor Details "));
Instructor.add(item5 = new MenuItem("Modify Instructor Details"));
Instructor.add(item6 = new MenuItem("Delete Instructor Details"));
mbar.add(Instructor);
Menu Admin= new Menu("Admin Details");
MenuItem item10, item11, item12;
Admin.add(item10=new MenuItem("Submit Admin Details"));
Admin.add(item11=new MenuItem("Modify Admin Details"));
Admin.add(item12=new MenuItem("Delete Admin Details"));
mbar.add(Admin);
Menu Question=new Menu("Question");
```

```
MenuItem item13,item14, item15;
Question.add(item13=new MenuItem("Insert Question"));
Question.add(item14=new MenuItem("Update Question"));
Question.add(item15=new MenuItem("Delete Question"));
mbar.add(Question);
Menu Answer = new Menu("Answer");
MenuItem item16, item17, item18;
Answer.add(item16= new MenuItem("Insert Answer"));
Answer.add(item17= new MenuItem("Update Answer"));
Answer.add(item18 = new MenuItem("Delete Answer"));
mbar.add(Answer);
Menu taught = new Menu("taught_By");
MenuItem item7, item8, item9;
taught.add(item7 = new MenuItem("Add Teacher Student pair"));
taught.add(item8 = new MenuItem("Update combinations"));
taught.add(item9 = new MenuItem("Delete a pair"));
mbar.add(taught);
Menu crct_ans = new Menu("crct_answer");
MenuItem item19, item20, item21;
crct_ans.add(item19 = new MenuItem("Add crct_answer"));
crct_ans .add(item20 = new MenuItem("Update crct_answer"));
crct_ans .add(item21 = new MenuItem("Delete crct_answer"));
mbar.add(crct_ans );
Menu given_ques = new Menu("given_question");
MenuItem item22, item23, item24;
given_ques.add(item22 = new MenuItem("Add given_ques"));
```

```
given_ques.add(item23 = new MenuItem("Update given_ques"));
given_ques.add(item24 = new MenuItem("Delete given_ques"));
mbar.add(given_ques );
Menu Manage_By = new Menu("Manage_By");
MenuItem item25, item26, item27;
Manage_By .add(item25 = new MenuItem("Add Admin Student
pair"));
Manage_By .add(item26 = new MenuItem("Update pair"));
Manage_By .add(item27 = new MenuItem("Delete pair"));
mbar.add(Manage_By );
// register listeners
item1.addActionListener(this);
item2.addActionListener(this);
item3.addActionListener(this);
item4.addActionListener(this);
item5.addActionListener(this);
item6.addActionListener(this);
item7.addActionListener(this);
item8.addActionListener(this);
item9.addActionListener(this);
item10.addActionListener(this);
item11.addActionListener(this);
item12.addActionListener(this);
item13.addActionListener(this);
item14.addActionListener(this);
item15.addActionListener(this);
item16.addActionListener(this);
```

```
item17.addActionListener(this);

item18.addActionListener(this);

item19.addActionListener(this);

item20.addActionListener(this);

item21.addActionListener(this);

item22.addActionListener(this);

item23.addActionListener(this);

item24.addActionListener(this);

item25.addActionListener(this);

item26.addActionListener(this);

item27.addActionListener(this);

 // Anonymous inner class which extends WindowAdaptor to handle the Window event: windowClosing

addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent we)

        {

                System.exit(0);

        }

});


//Frame properties

setTitle("PLICKER CARD");

Color clr = new Color(200, 100, 150);

setBackground(clr);

setFont(new Font("SansSerif", Font.BOLD, 14));

setSize(500, 600);

setVisible(true);
```

```
}

public void actionPerformed(ActionEvent ae)
{
    String arg = ae.getActionCommand();
    if(arg.equals("Submit Student Details"))
    {
        cardLO.show(home, "InsertStudent");
    }

    else if(arg.equals("Modify Student details"))
    {
        cardLO.show(home, "UpdateStudent");
    }

    else if(arg.equals("Delete Student Details"))
    {
        cardLO.show(home, "DeleteStudent");
    }
    else if(arg.equals("Submit Instructor Details"))
    {
        cardLO.show(home,"InsertInstructor");
    }

    else if(arg.equals("Modify Instructor Details"))
```

```
 {
        cardLO.show(home, "UpdateInstructor");
 }
else if(arg.equals("Delete Instructor Details"))
{
        cardLO.show(home,"DeleteInstructor");
}
else if(arg.equals("Submit Admin Details"))
{
        cardLO.show(home,"InsertAdmin");
}

else if(arg.equals("Update Admin Details"))
{
        cardLO.show(home,"UpdateAdmin");
}
else if(arg.equals("Delete Admin Details"))
{
        cardLO.show(home,"DeleteAdmin");
}
else if(arg.equals("Insert Question"))
{
        cardLO.show(home,"InsertQuestion");
}

else if(arg.equals("Update Question"))
```

```
{
        cardLO.show(home,"UpdateQuestion");
}
else if(arg.equals("Delete Question"))
{
        cardLO.show(home,"DeleteQuestion");
}
else if(arg.equals("Insert Answer"))
{
        cardLO.show(home,"InsertAnswer");
}
else if(arg.equals("Update Answer"))
{
        cardLO.show(home,"UpdateAnswer");
}
else if(arg.equals("Delete Answer"))
{
        cardLO.show(home,"DeleteAnswer");
}
else if(arg.equals("Add crct_answer"))
{
        cardLO.show(home,"crct_answer");
}
else if(arg.equals("Update crct_answer"))
{
        cardLO.show(home,"UpdateCrctAnswer");
```

```
}
else if(arg.equals("Delete crct_answer"))
{
        cardLO.show(home,"DeleteCrctAns");
}
else if(arg.equals("Add given_ques"))
{
        cardLO.show(home,"given_ques");
}
else if(arg.equals("Update given_ques"))
{
        cardLO.show(home,"UpdateGiven");
}
else if(arg.equals("Delete given_ques"))
{
        cardLO.show(home,"DeleteGivenQues");
}


else if(arg.equals("Update Answer"))
{
        cardLO.show(home,"UpdateAnswer");
}
else if(arg.equals("Delete Answer"))
{
        cardLO.show(home,"DeleteAnswer");
```

```
        }

        else if(arg.equals("Add Teacher Student pair"))
        {
                cardLO.show(home, "taught");
        }
        else if(arg.equals("View combinations"))
        {
                cardLO.show(home, "UpdateTaught");
        }
         else if(arg.equals("Delete a pair"))
        {
                cardLO.show(home, "DeleteTaught");
        }

        else if(arg.equals("Add Admin Student pair"))
        {
                cardLO.show(home, "Manage_By");
        }
        else if(arg.equals("Update pair"))
        {
                cardLO.show(home, "UpdateManage");
        }
        else if(arg.equals("Delete pair"))
        {
                cardLO.show(home, "DeleteManage");
```

```
            }
        }
        public static void main(String ... args)
        {
                new plickerCard();
        }
}
```

CONNECTIVITY WITH DATABASE:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

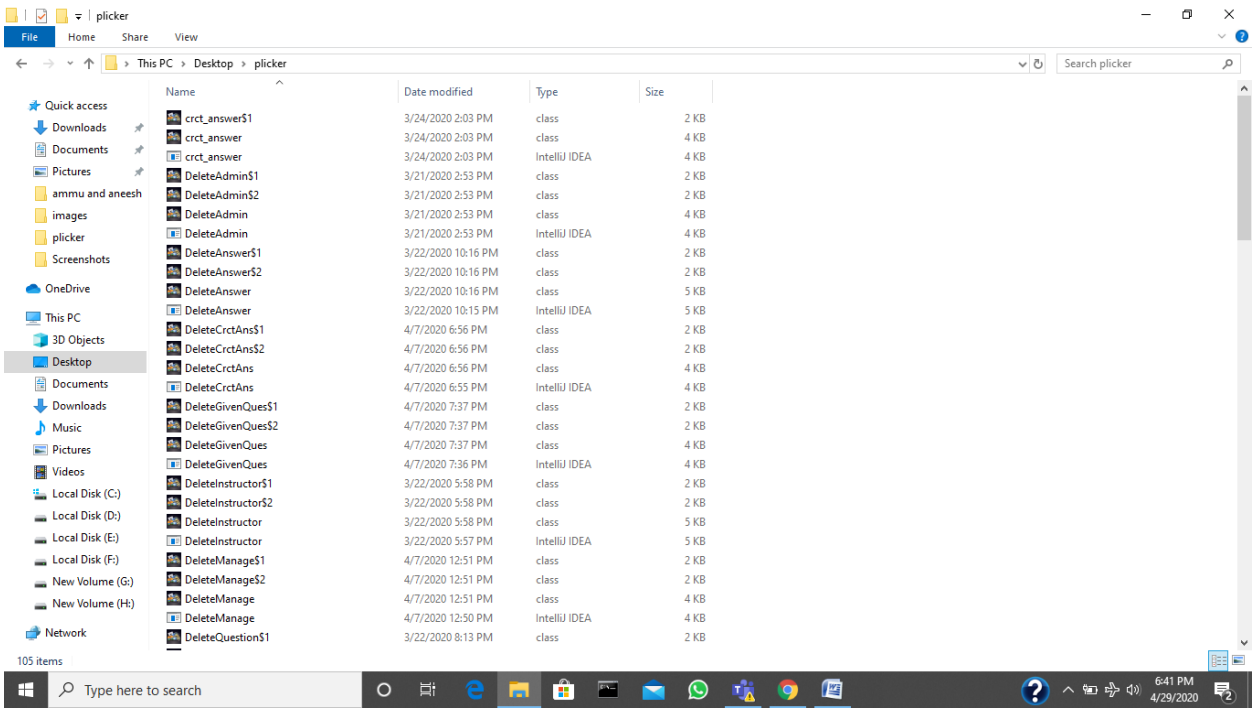
Block of code for JAVA- SQL connectivity with JDBC:

```
public void connectToDB()
    {
        try
        {

    connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:15
21:orcl","hemanth","oracle");
                statement=connection.createStatement();
        }
```

```
            catch(SQLException connectException)
            {
                System.out.println(connectException.getMessage());
                System.out.println(connectException.getSQLState());
                System.out.println(connectException.getErrorCode());
                System.exit(1);
            }
        }
```

FOLDER STRUCTURE:

This project is in the folder named plicker in which it has different cases for which data is being inserted or updated or deleted. All the source code for each program is named according to its functionality so that it would be easy to navigate.

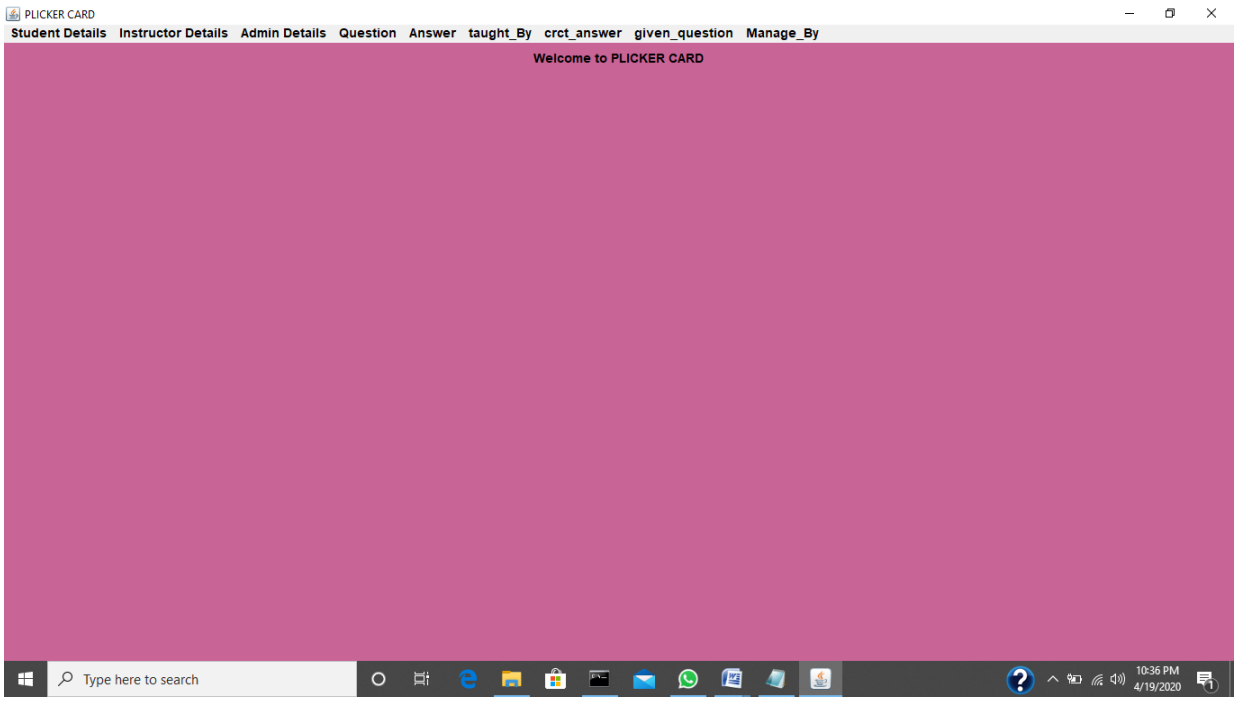


V.Renu Aakanksha
1602-18-737-093

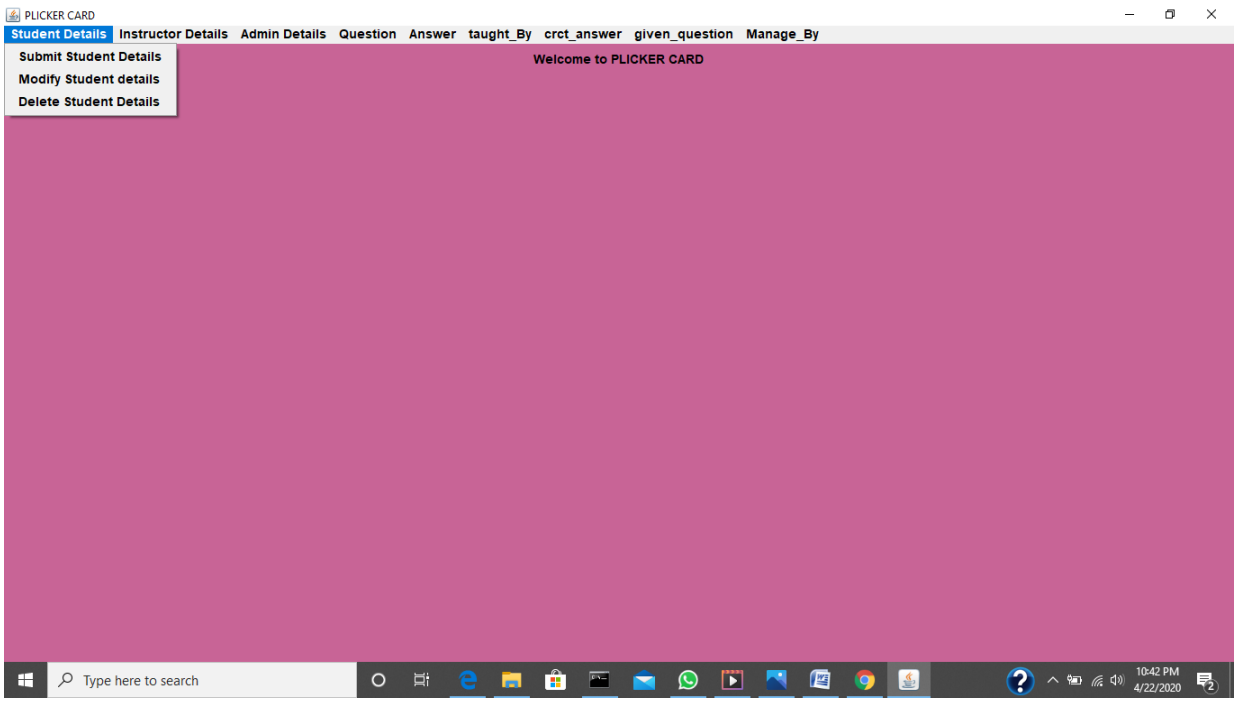
TESTING

The program runs for execution of three basic operations of insertion, update and delete on 5 different table. Along with this, it also has a output column which gives the information about how many rows have been edited. Errors, syntactical or exceptional will be shown if occurred.

HOME PAGE:



PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT

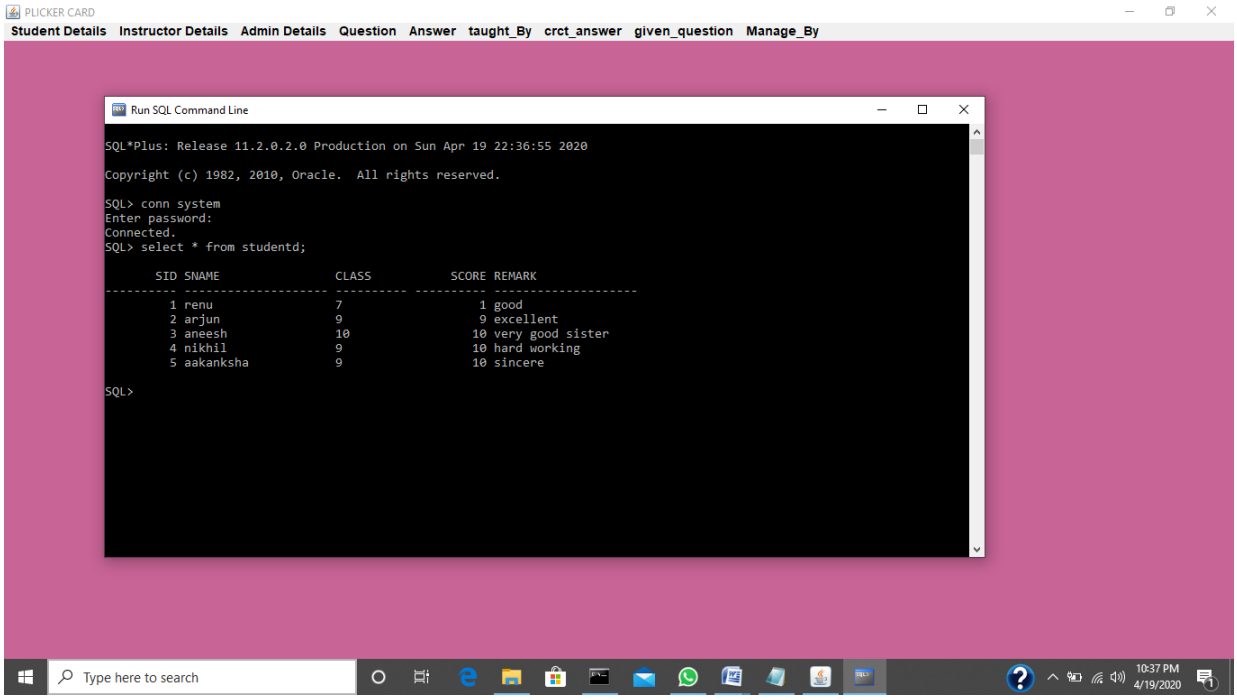


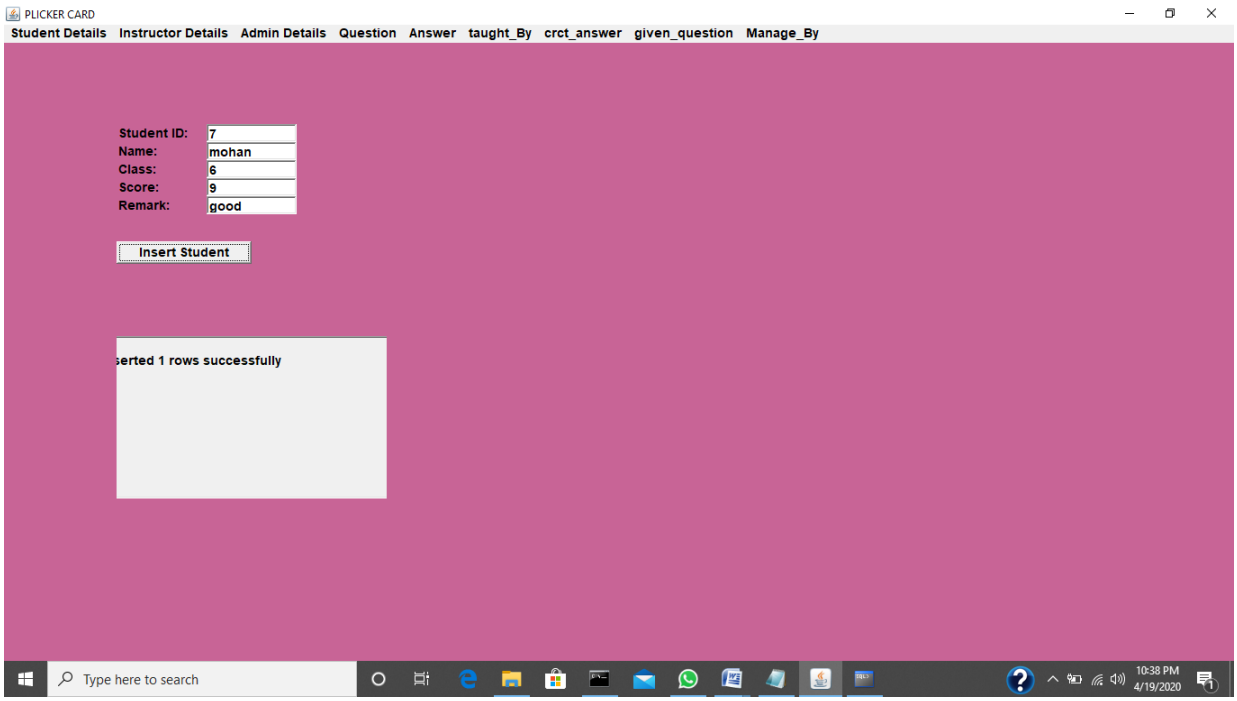
V.Renu Aakanksha
1602-18-737-093

PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT

SUMISSION OF STUDENT DETAILS:

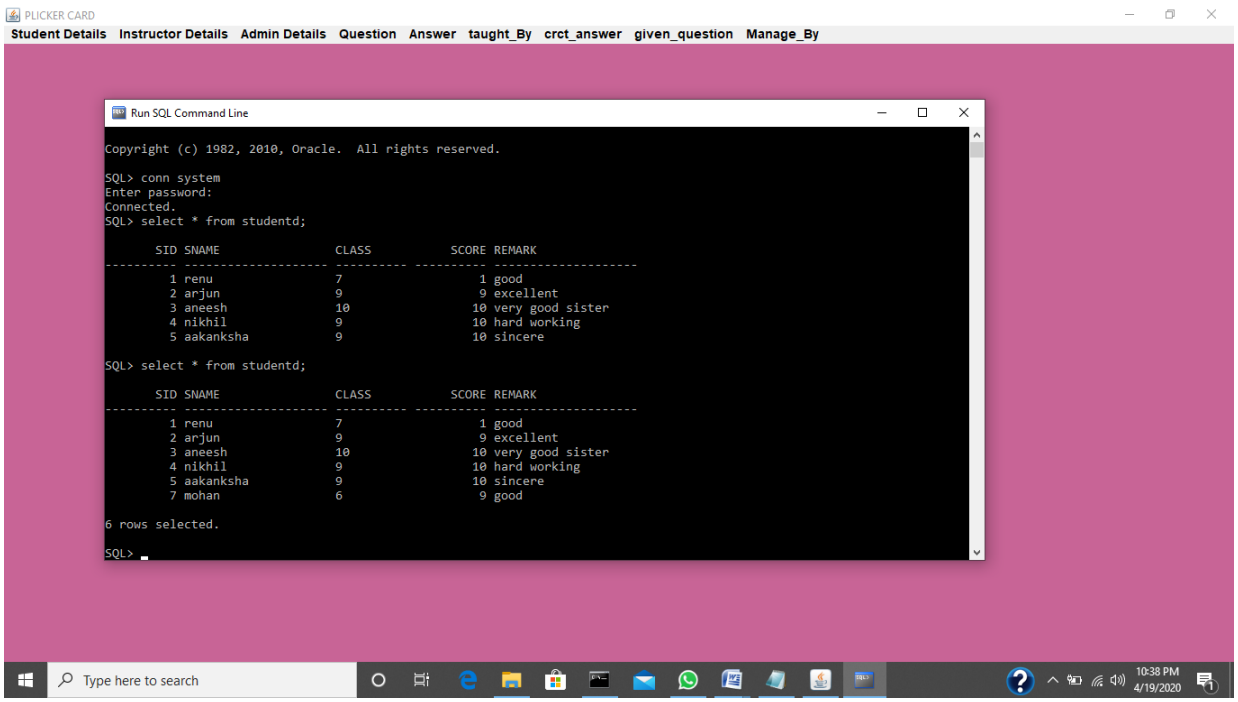
BEFORE





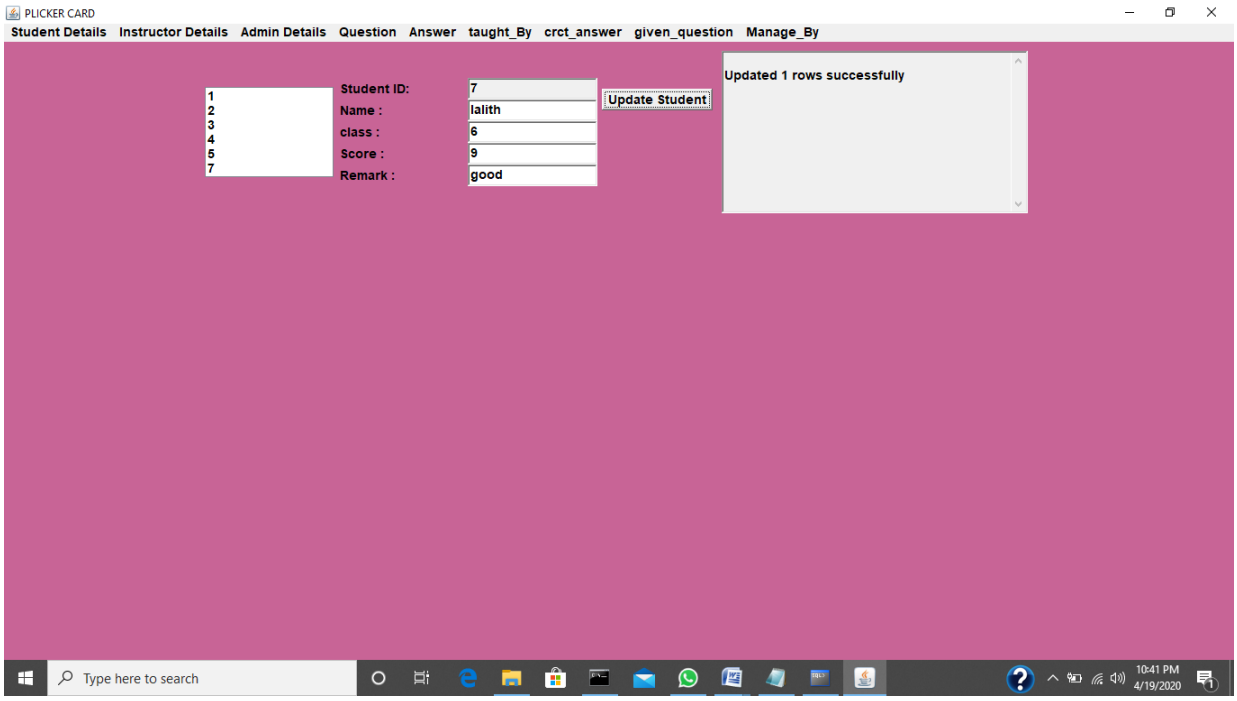
V.Renu Aakanksha
1602-18-737-093

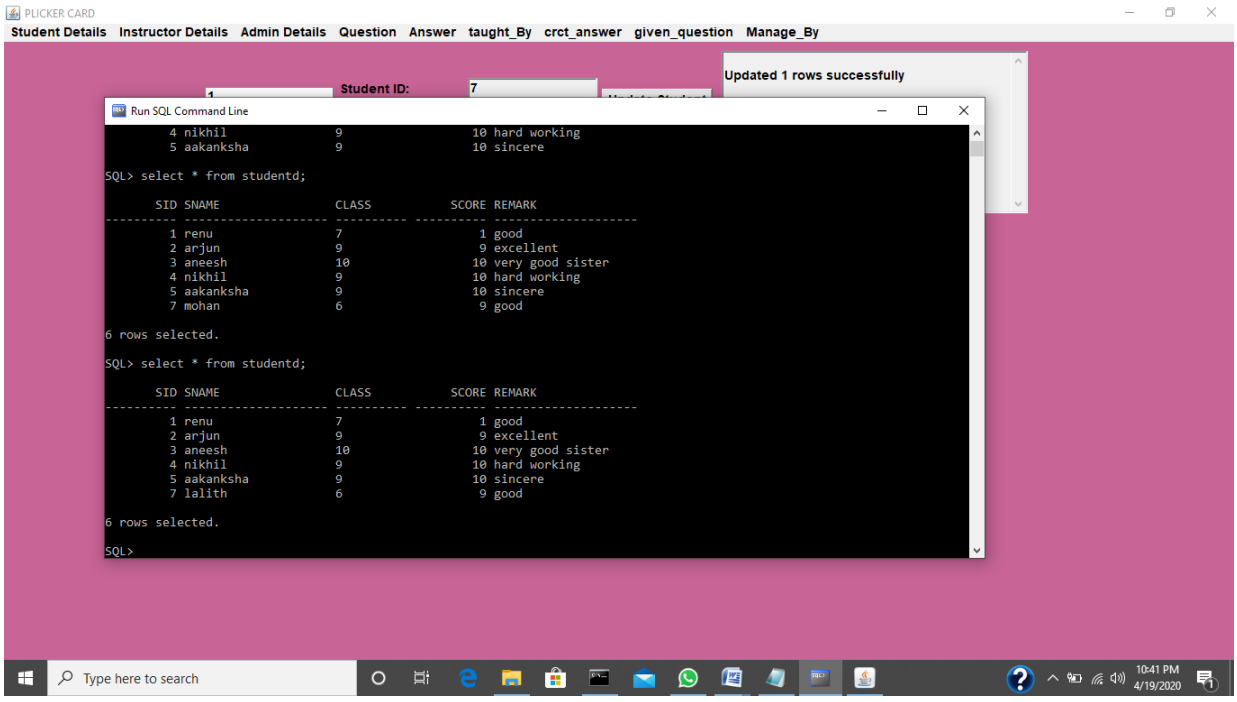
PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT



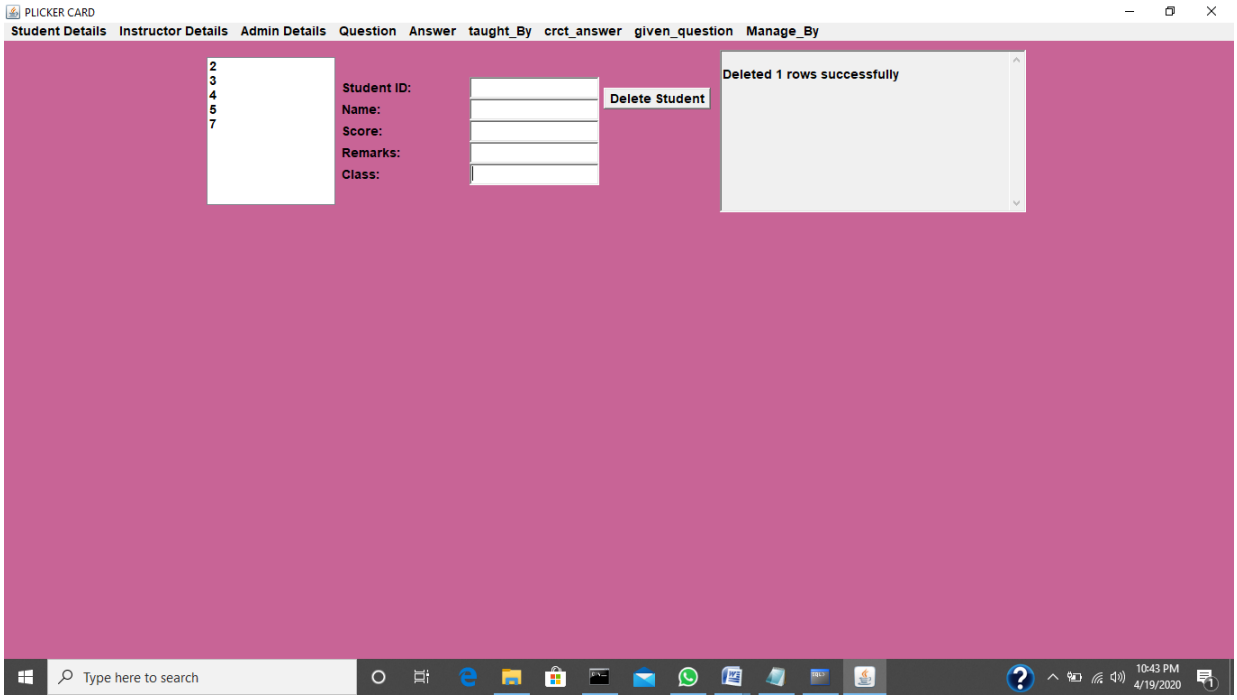
V.Renu Aakanksha
1602-18-737-093

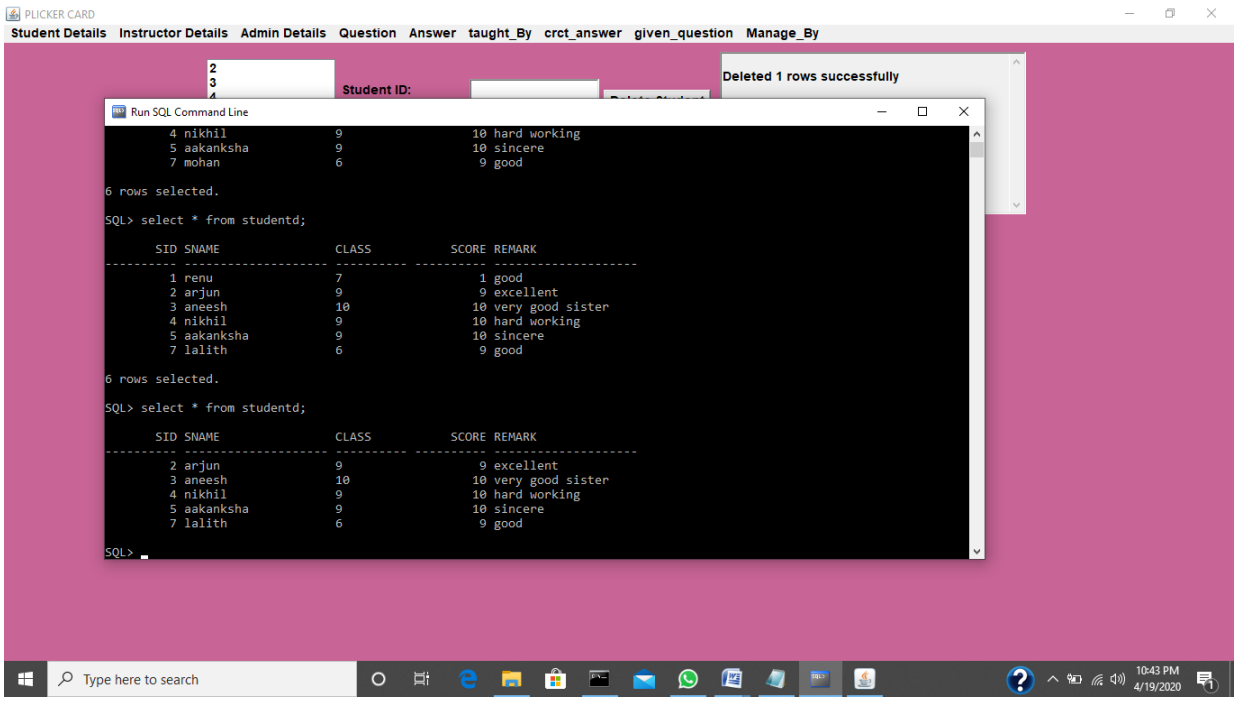
MODIFICATION OF STUDENT DETAILS:





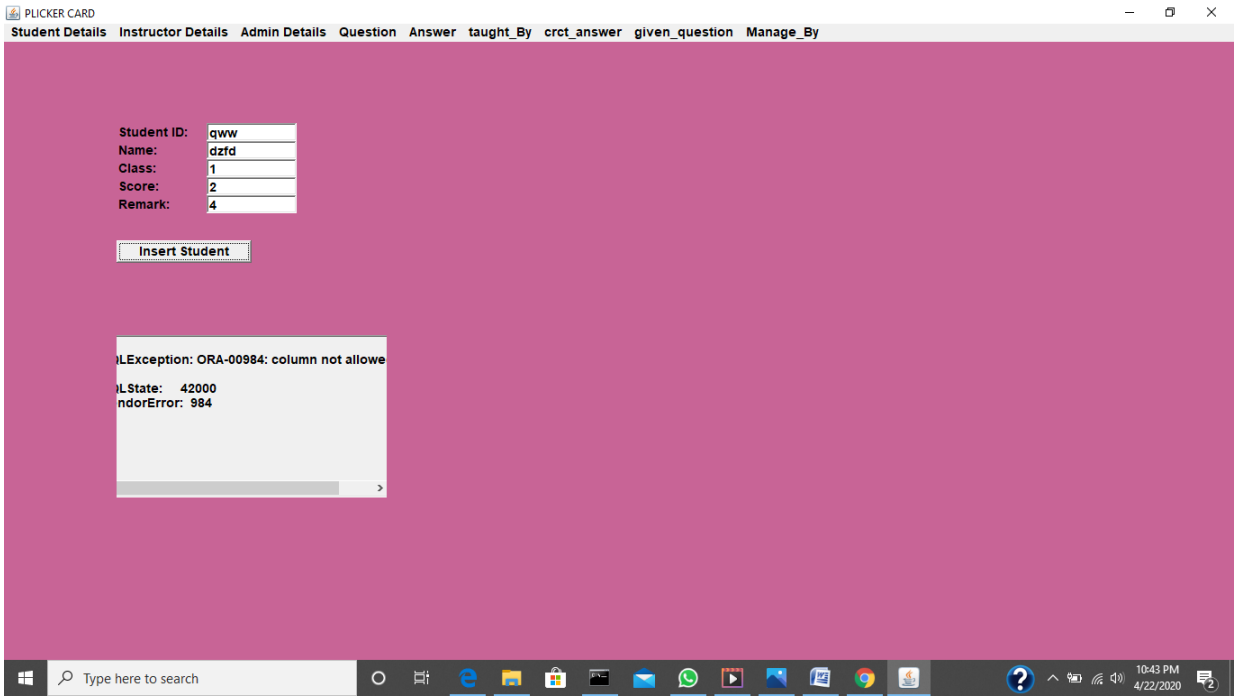
DELETION OF STUDENT DETAILS:





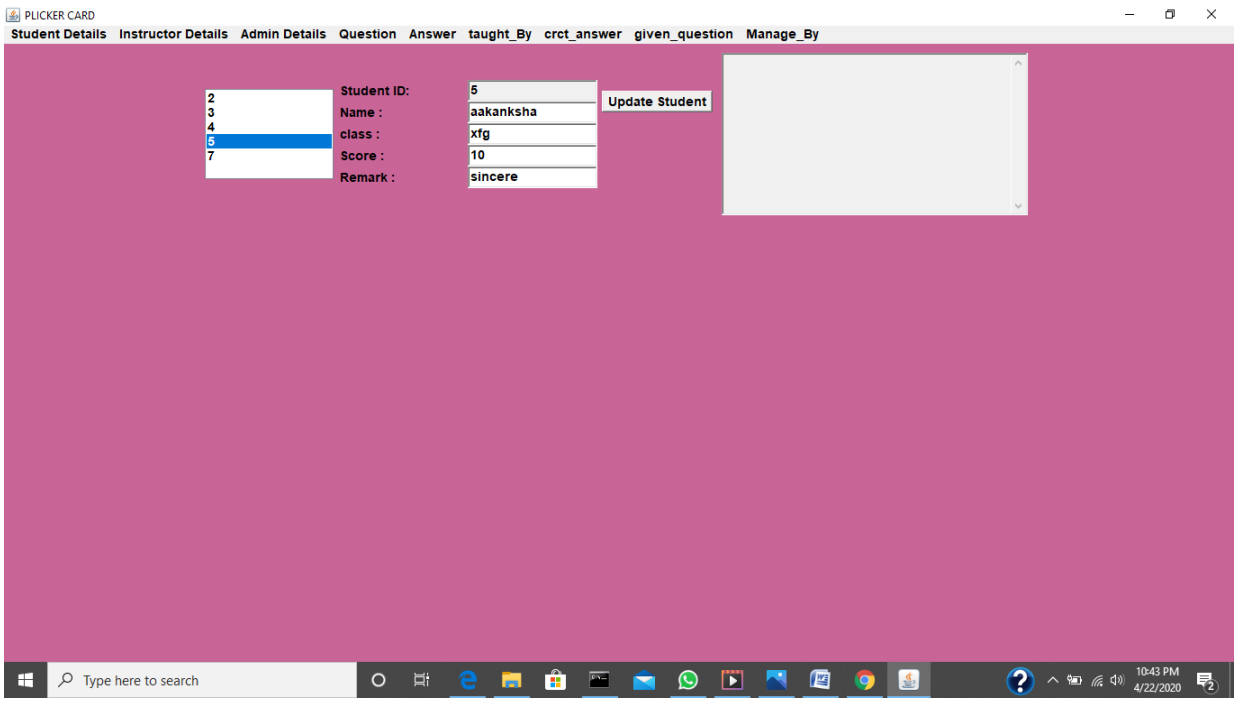
PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT

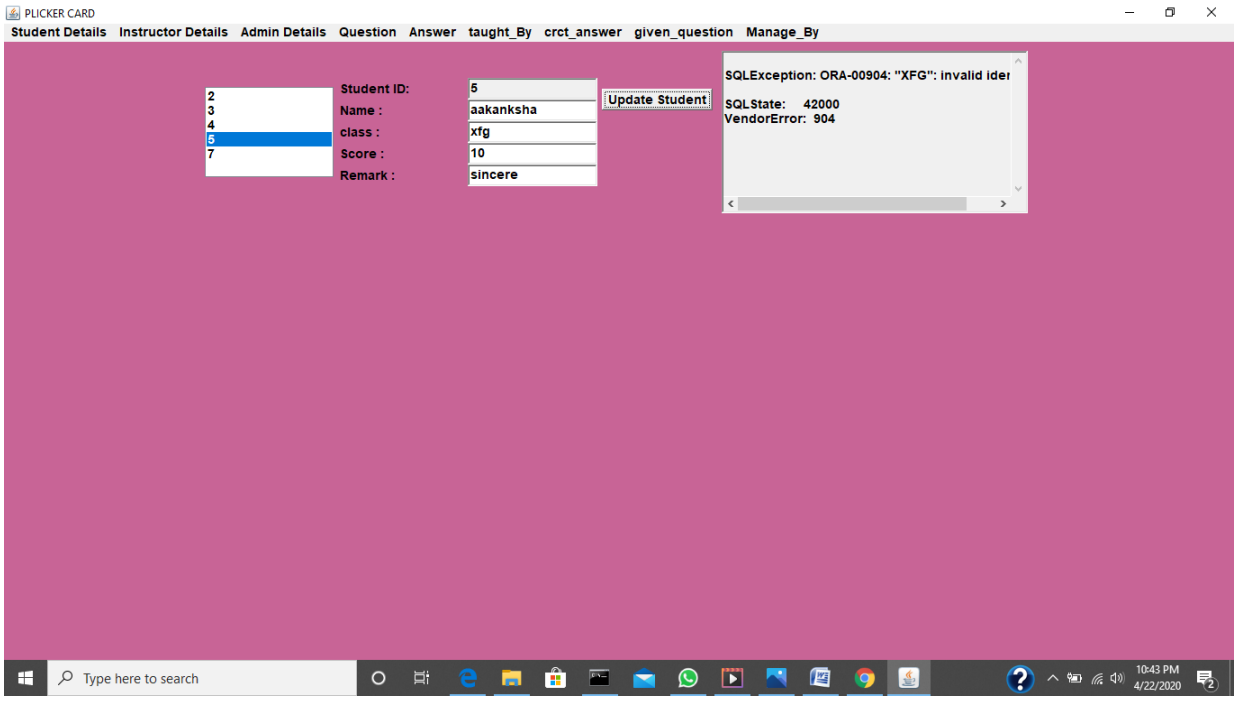
ERRORS:



V.Renu Aakanksha
1602-18-737-093

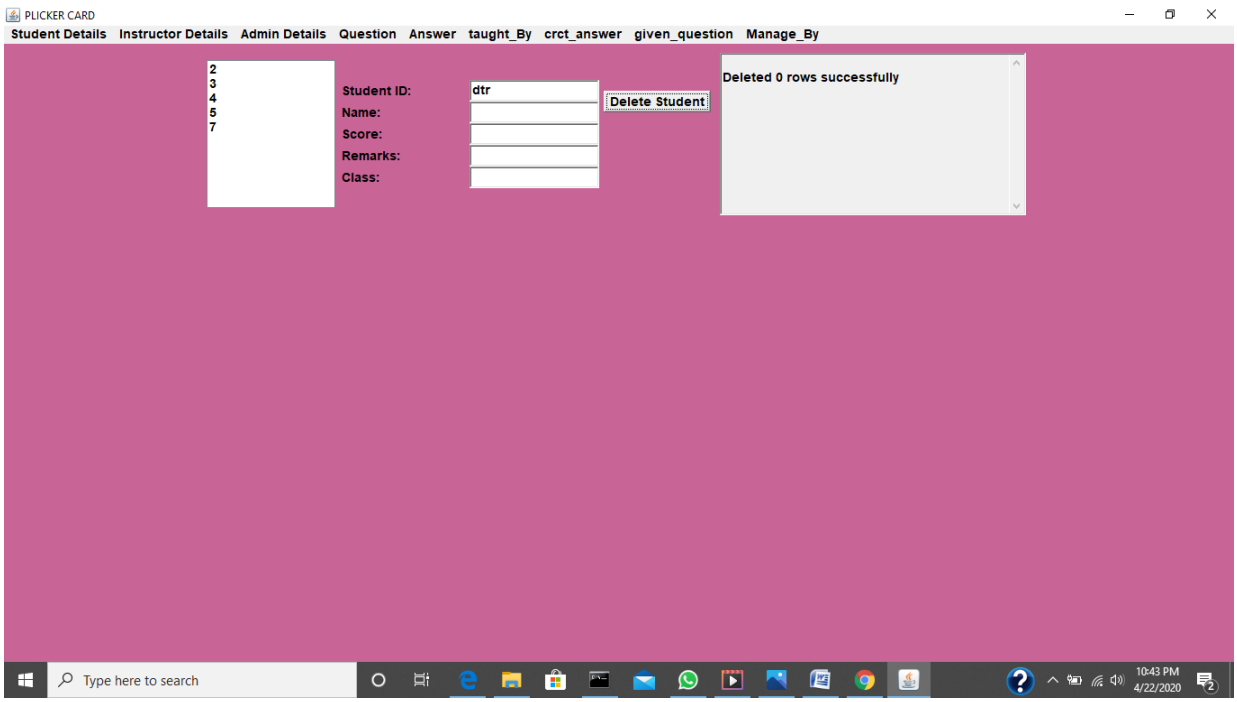
PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT





V.Renu Aakanksha
1602-18-737-093

PLICKER CARD IMPLEMENTATION
DBMS ASSIGNMENT



GITHUB LINK: https://github.com/r0614/pilcker-Card-Implementation

RESULTS

The DML commands, Insert, update and delete for one of the tables in given below:

For student table: (in java, as per the application)

FOR INSERT: "INSERT INTO studentd VALUES(" + sidText.getText() + ", " + "'" + snameText.getText() + "'," + classText.getText() + "," + scoreText.getText() +",'" + remarkText.getText() + "'"+ ")";

FOR UPDATE: UPDATE studentd "

V.Renu Aakanksha
1602-18-737-093

```
+ "SET sname='" + snameText.getText()    + "', "
+ "class=" + classText.getText() +    ", "
+ "remark='" + remarkText.getText() +    "', "
+ "score=" + scoreText.getText() +" WHERE sid = "
+ studentIDList.getSelectedItem());
```

FOR DELETE: "DELETE FROM studentd WHERE SID = "

```
+ studentIDList.getSelectedItem());
```

REFERENCES

1. http://sociallearningcommunity.com/10-of-the-best-mooc-providers/
2. https://en.wikipedia.org/wiki/List_of_MOOC_providers
3. GIT HUB LINK: https://github.com/r0614/pilcker-Card-Implementation
4. GIT HUB LINK OF ASSIGNMENT 1:
https://github.com/r0614/PLICKER-CARD-IMPLEMENTATION-UPDATED