**JAVA AWT BASED- Online MOOC’s year wise student database management system - 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

**S.Hemanth Kumar <1602-18-737-072>**

****

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**Ibrahimbagh, Hyderabad-31**

**2020**

**BONAFIDE CERTIFICATE**

Certified that this project report titled”Online MOOC’s year wise student database management system” is bonafide work of Mr S.Hemanth Kumar, who carried out the mini project work under my supervision.

Certified further that, to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion or any other candidate.

Signature of the Examiner

**B.LEELAVATHY**

Lecturer

Department of Information Technology.

**ABSTRACT**

Online MOOC’s year wise Student Management System is a database management system which is helpful for students as well as the Mooc’s providers. In the current system all the activities are done manually. It is very time consuming and costly. Our online Mooc’s Student Management System deals with the various activities related to the students and mooc’s provider. In the database can register as a user and user has of two types, student and administrator. Administrator has the power to add new user and can edit and delete a user. A student can register as user and can add edit and delete his profile. The administrator can add edit and delete marks for the student. All the users can see the marks.

**INTRODUCTION**

* **REQUIREMENTS FOR ONLINE MOOC’S YEAR WISE DATABASE MANAGEMENT SYSTEM:**

List of tables :

* Online MOOC’s provider
* Courses
* Student
* Enrolls
* Assignments
* Results

List of attributes with their domain types:

|  |  |  |
| --- | --- | --- |
| ENTITY | ATTRIBUTES | DOMAIN |
| Online Mooc’s provider | 1. P\_id 2. P\_name 3. Type 4. Headquartes 5. Found | Number(5)  Varchar2(20)  Varchar2(20)  Varchar2(20)  Number(5) |
| Courses | 1. C\_id 2. C\_name 3. Duration 4. Min\_grade 5. Price 6. Status | Number(5)  Varchar2(20)  Varchar2(20)  Char(5)  Number(5)  Varchar(10) |
| Student | 1. S\_id 2. First\_name 3. Last\_name 4. User\_name 5. Password | Number(5)  Varchar(10)  Varchar(10)  Varchar(10)  Varchar(10) |
| Enrolls | 1. S\_id 2. C\_id 3. Year | Number(5)  Number(5)  Number(5) |
| Assignments | 1. A\_id 2. C\_id 3. Deadline 4. Score | Number(5)  Number(5)  Varchar(10)  Number(5) |
| Results | 1. S\_id 2. C\_id 3. A\_id 4. Score 5. Grade | Number(5)  Number(5)  Number(5)  Number(5)  Char(5) |

* Online mooc’s provider offers as many courses to the students who want to persue the course so, it is a one to many mapping. As it is not necessary that one provider should offer only one course.
* Student enrols into courses, it is many to many mapping as any number of students can enrol into any number of courses.
* Student submits assignment, it is one to one as one student should submit one assignment as per the provider instructions.
* Student gets results if he/submit the assignments and attend the exam, it is one to one mapping as one student get only one result.
* **SPECIFIC GOAL OF THE PROJECT:**

The main goal to be achieved through this project was to provide a facility to the Online MOOC’s providers to display the details of various courses, students who enroll into those courses, assignments they do, and the results they get based on their assignment submissions online.

The project also ensure that the details of the students are confidential and are stored in the database.

SQL particular Online MOOC’s provider, courses, student, assignments and results can be executed.

* **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.

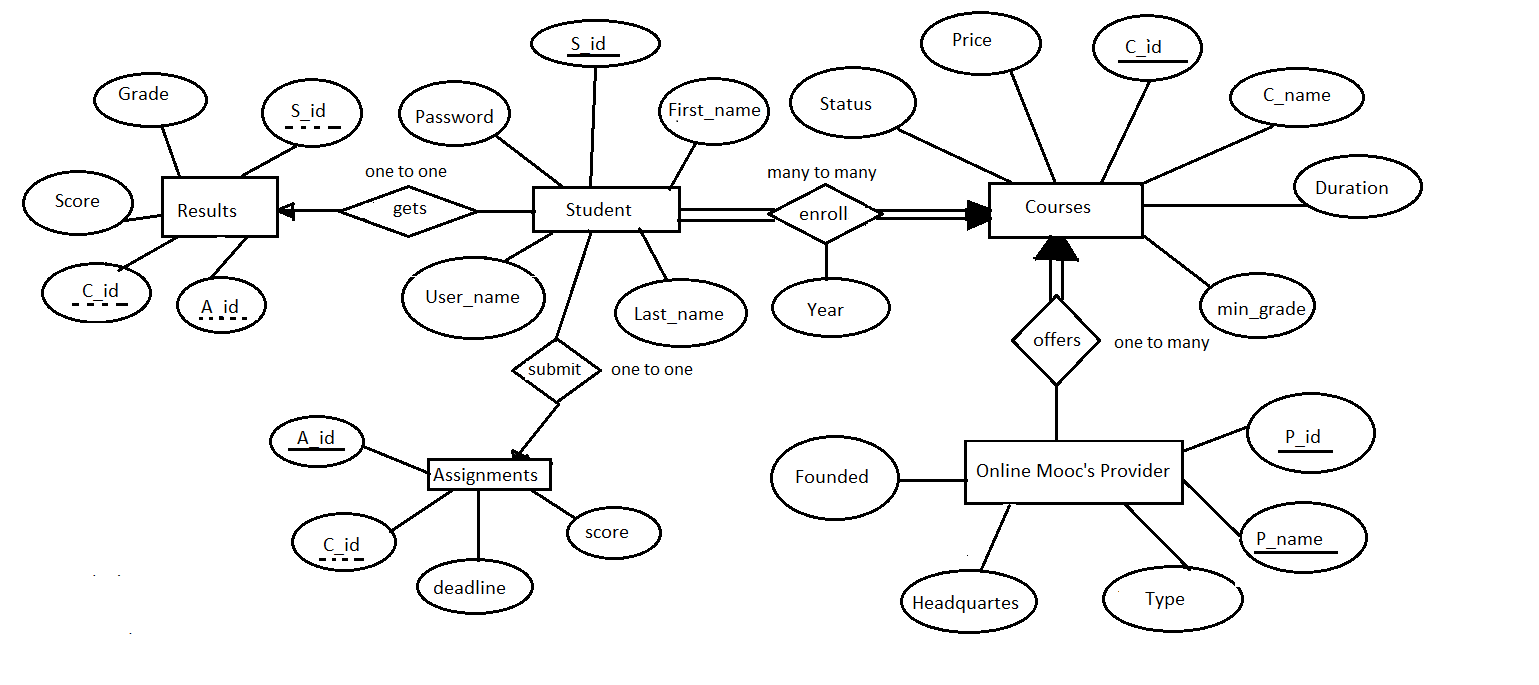
The interface of SQL Plus is used for creating the database. DDL and DML commands are implemented for operations being executed. The details of various Online MOOC’s provider, courses, student, assignments, and results are stored in the form of tables in the database.

**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.

* **DESIGN:**

i)ER DIAGRAM:

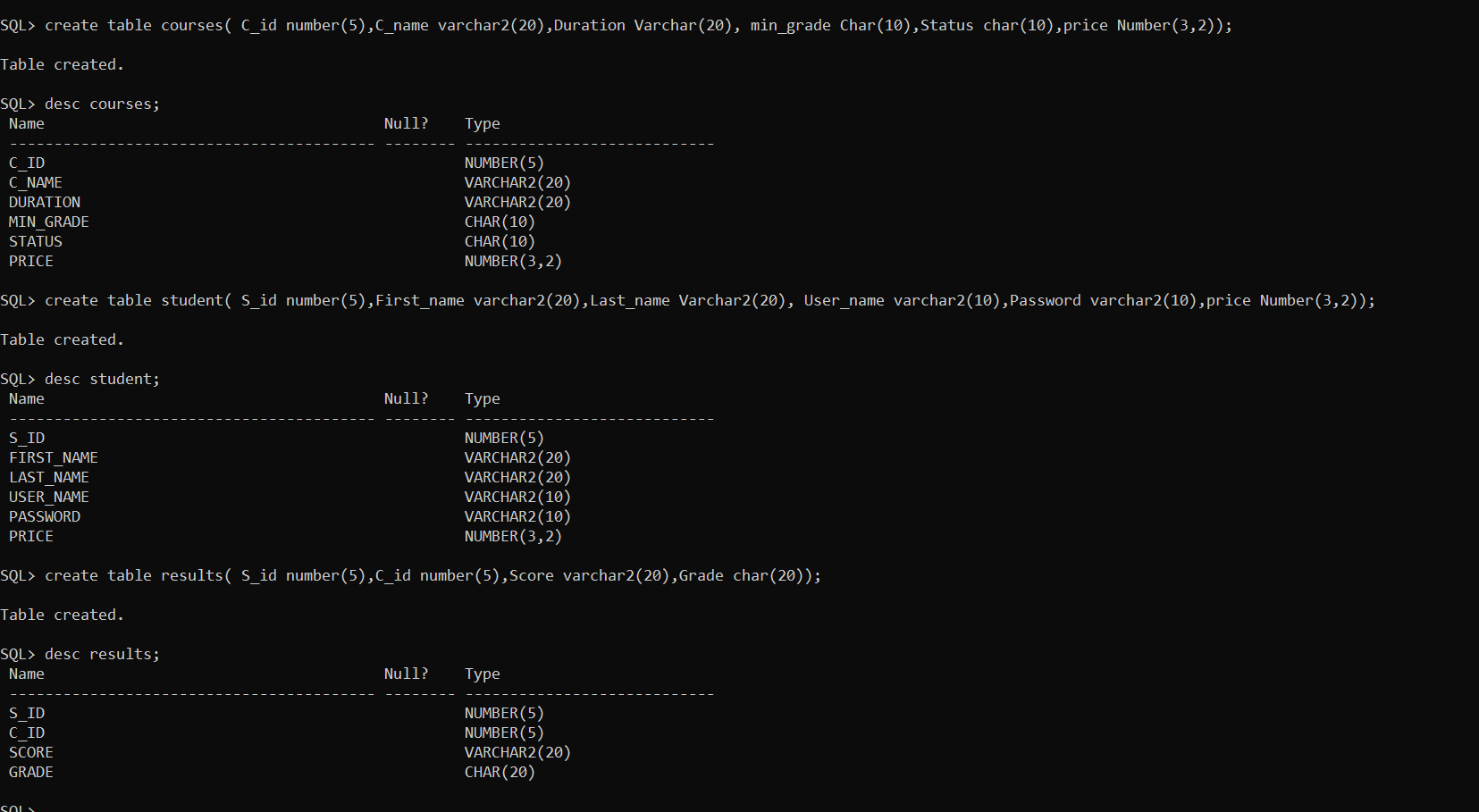


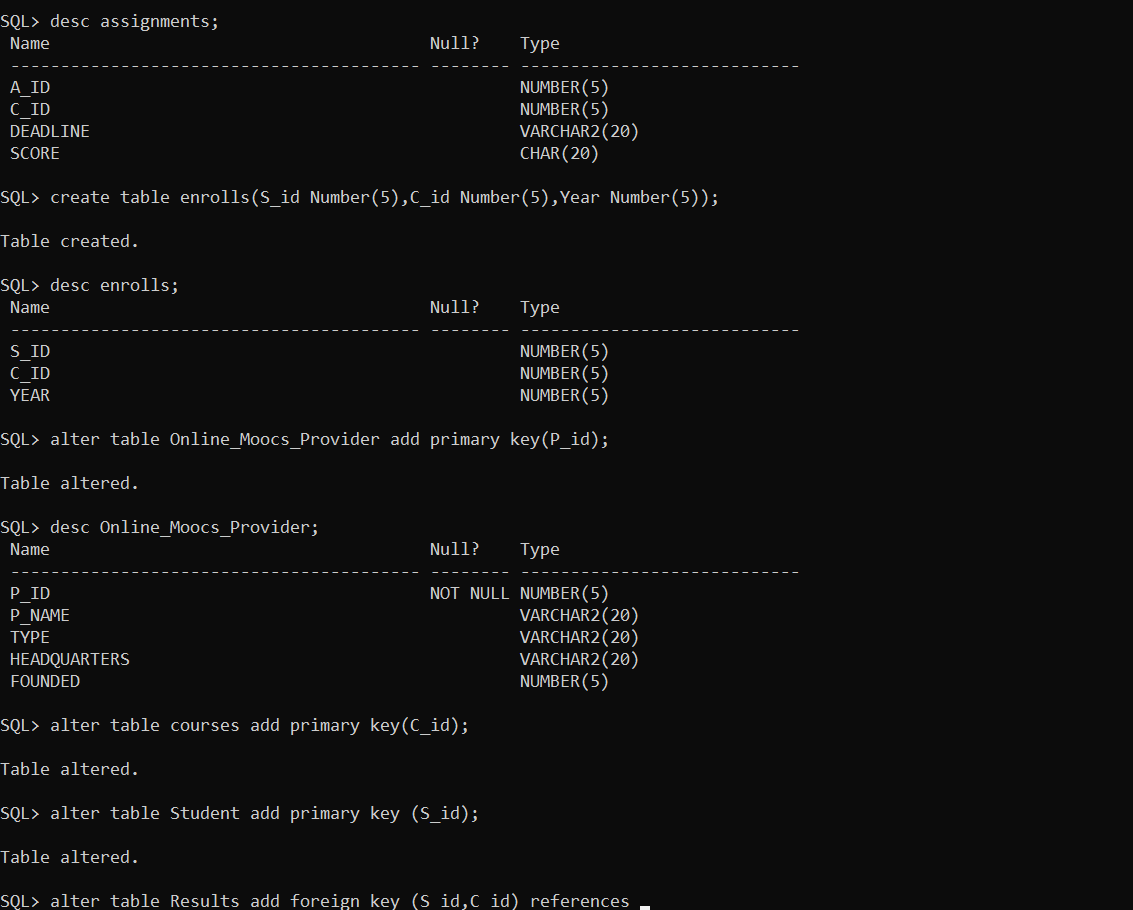
**MAPPING CARDINALITIES AND PARTICIPATION CONSTRAINTS:**

* Online mooc’s provider offers as many courses to the students who want to persue the course so, it is a one to many mapping. As it is not necessary that one provider should offer only one course.
* Student enrols into courses, it is many to many mapping as any number of students can enrol into any number of courses.
* Student submits assignment, it is one to one as one student should submit one assignment as per the provider instructions.
* Student gets results if he/submit the assignments and attend the exam, it is one to one mapping as one student get only one result.

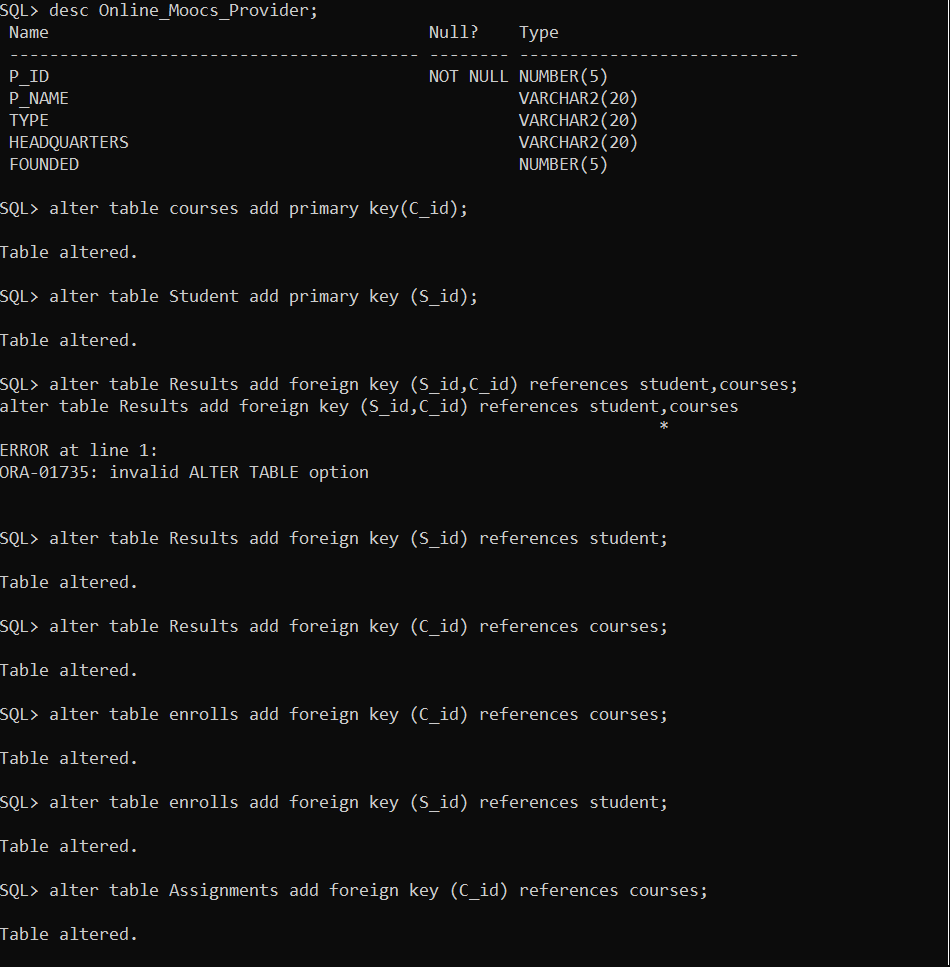
**DDL Commands:**

Creating all the required tables.



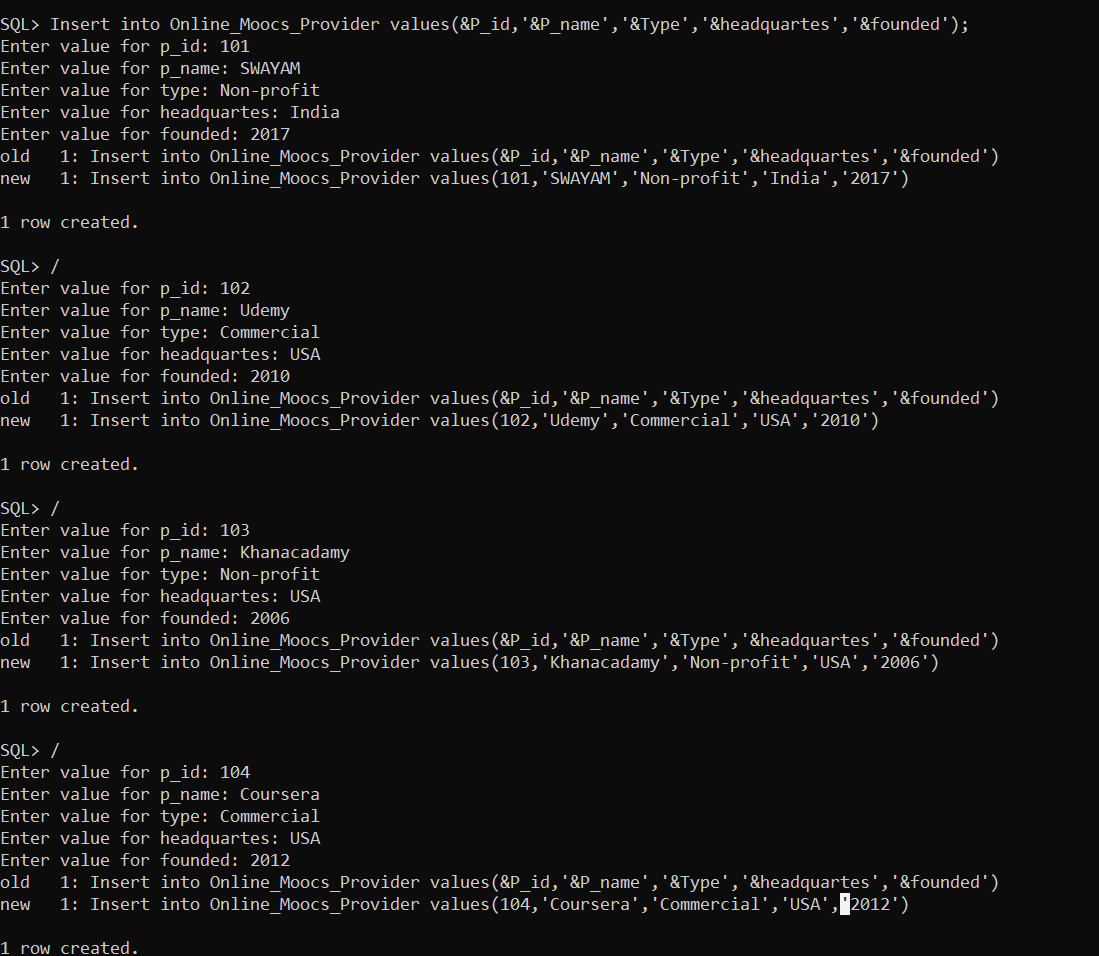


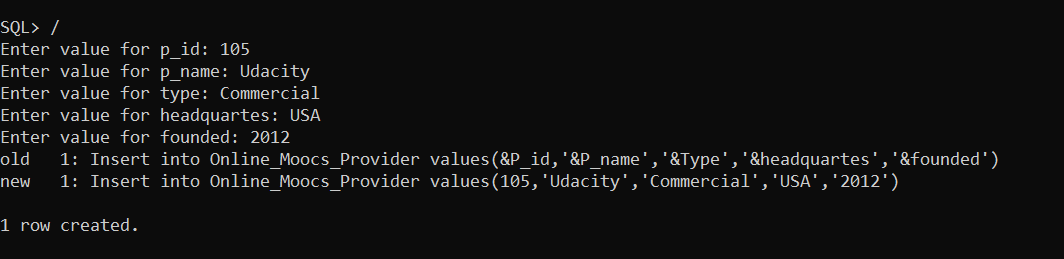
**Enforcing constraints to primary, forein key constraints:**

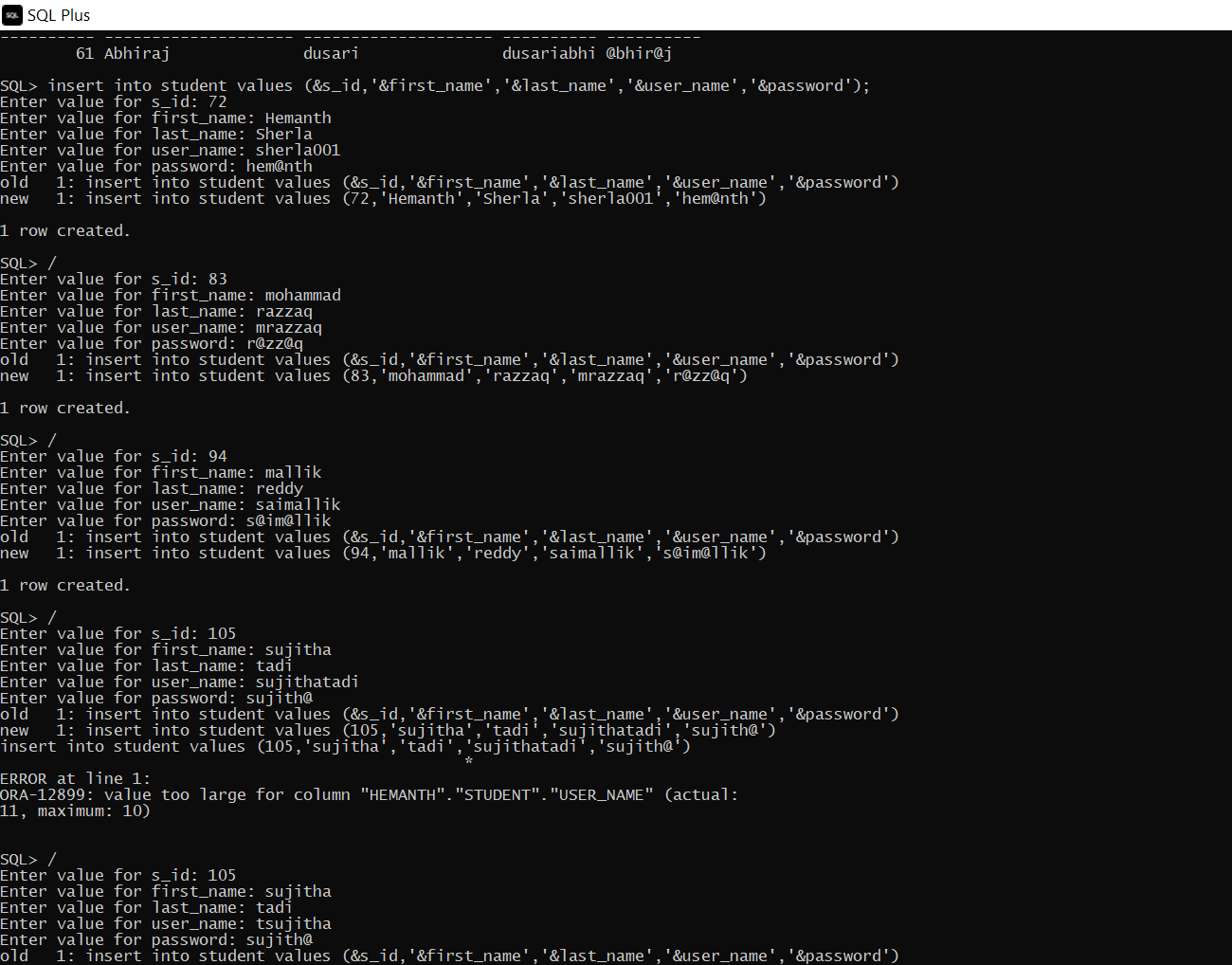
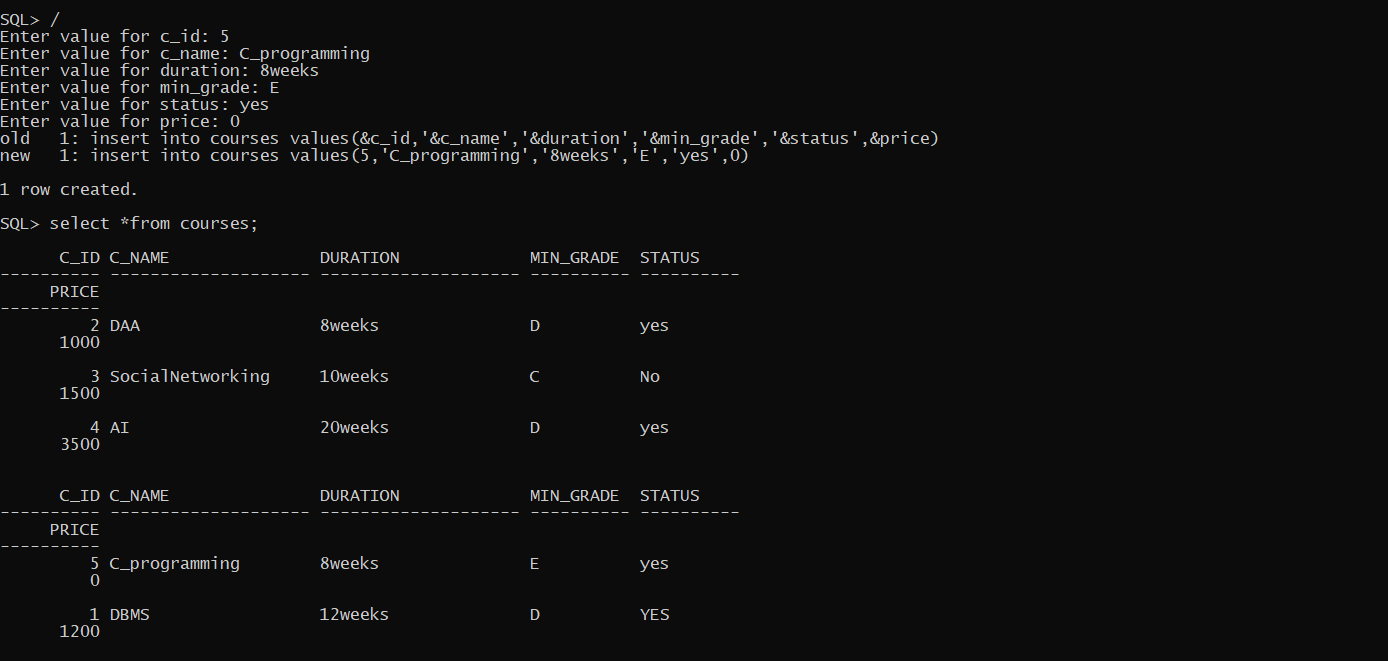
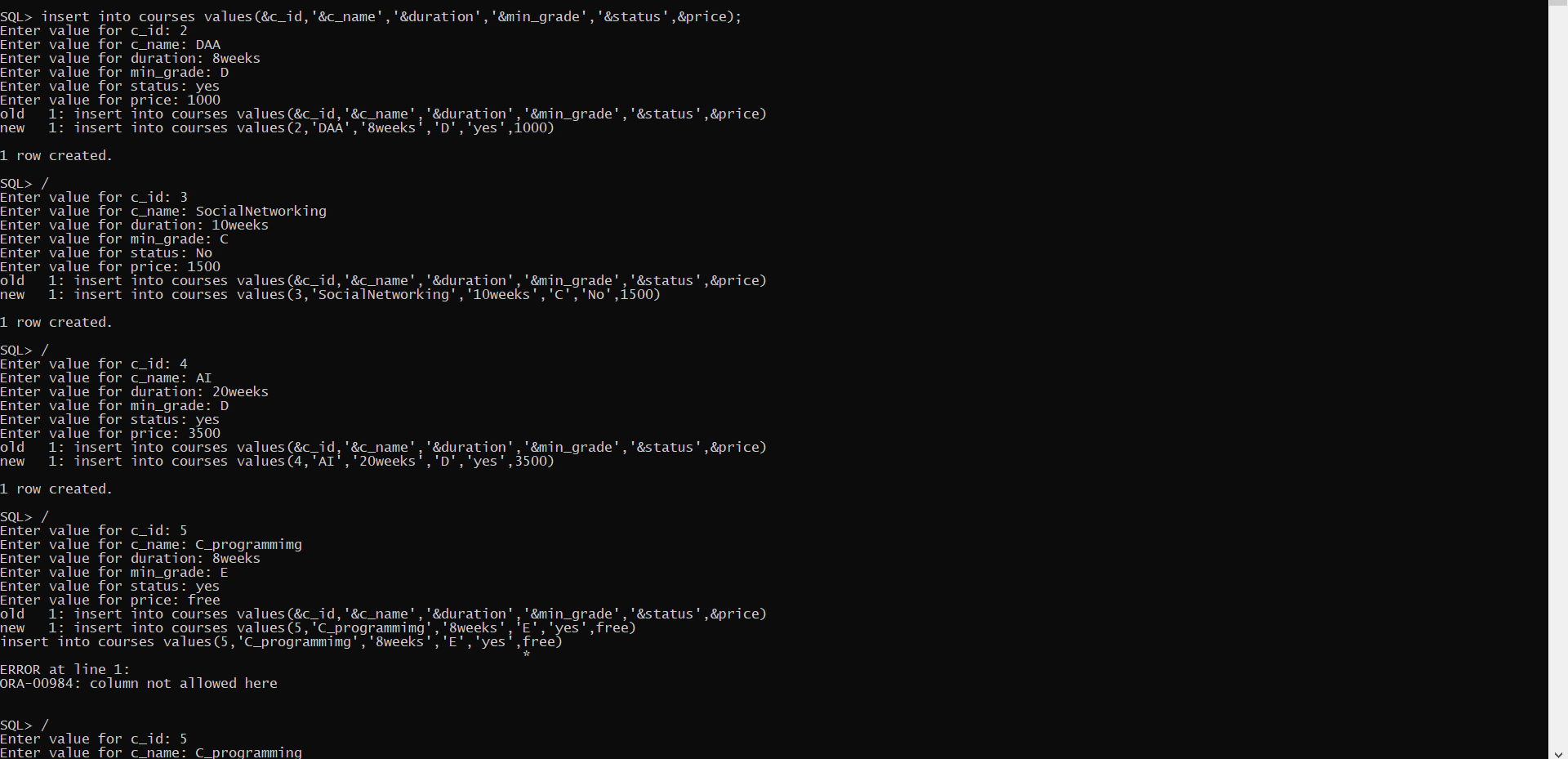
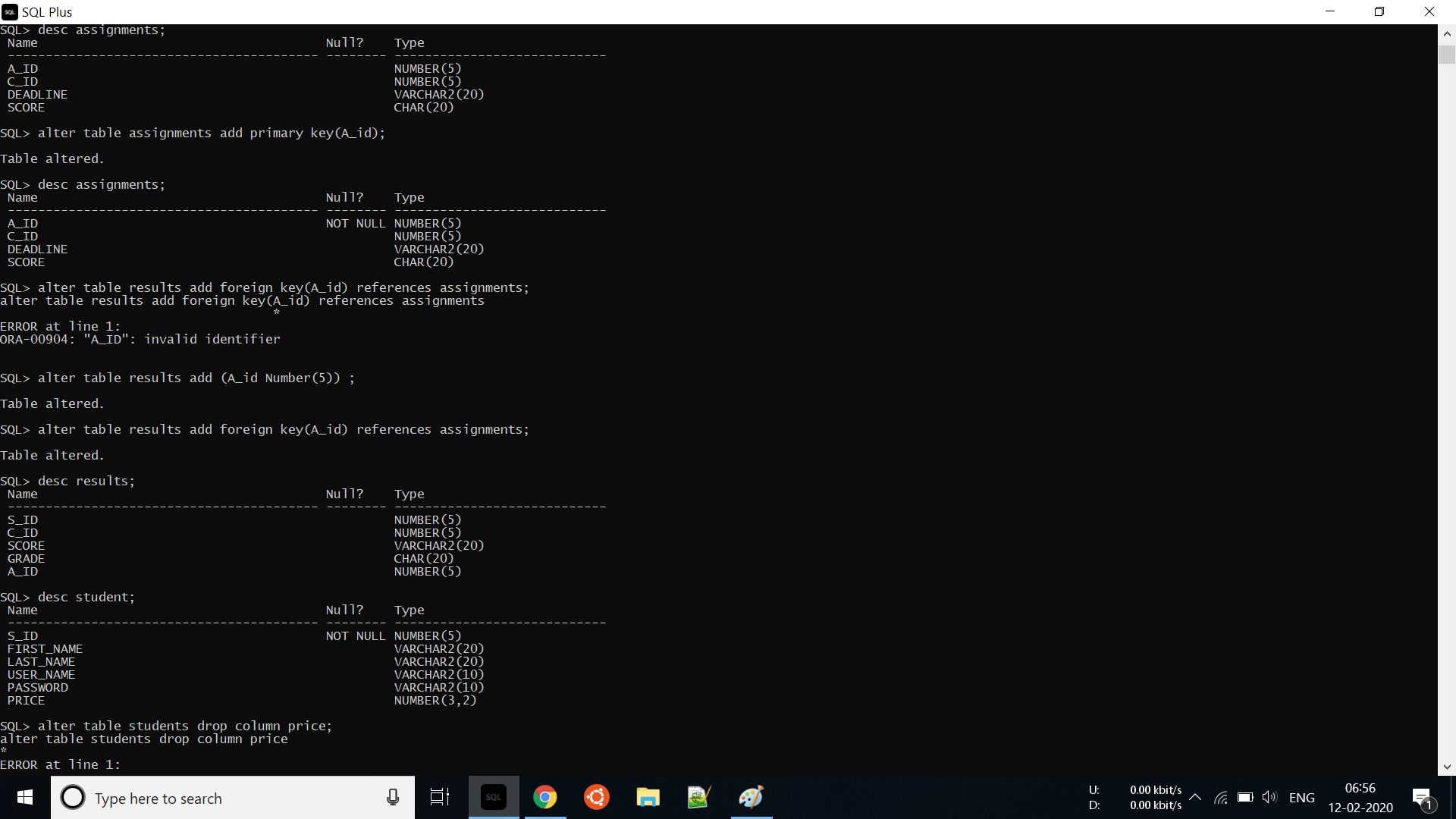
****

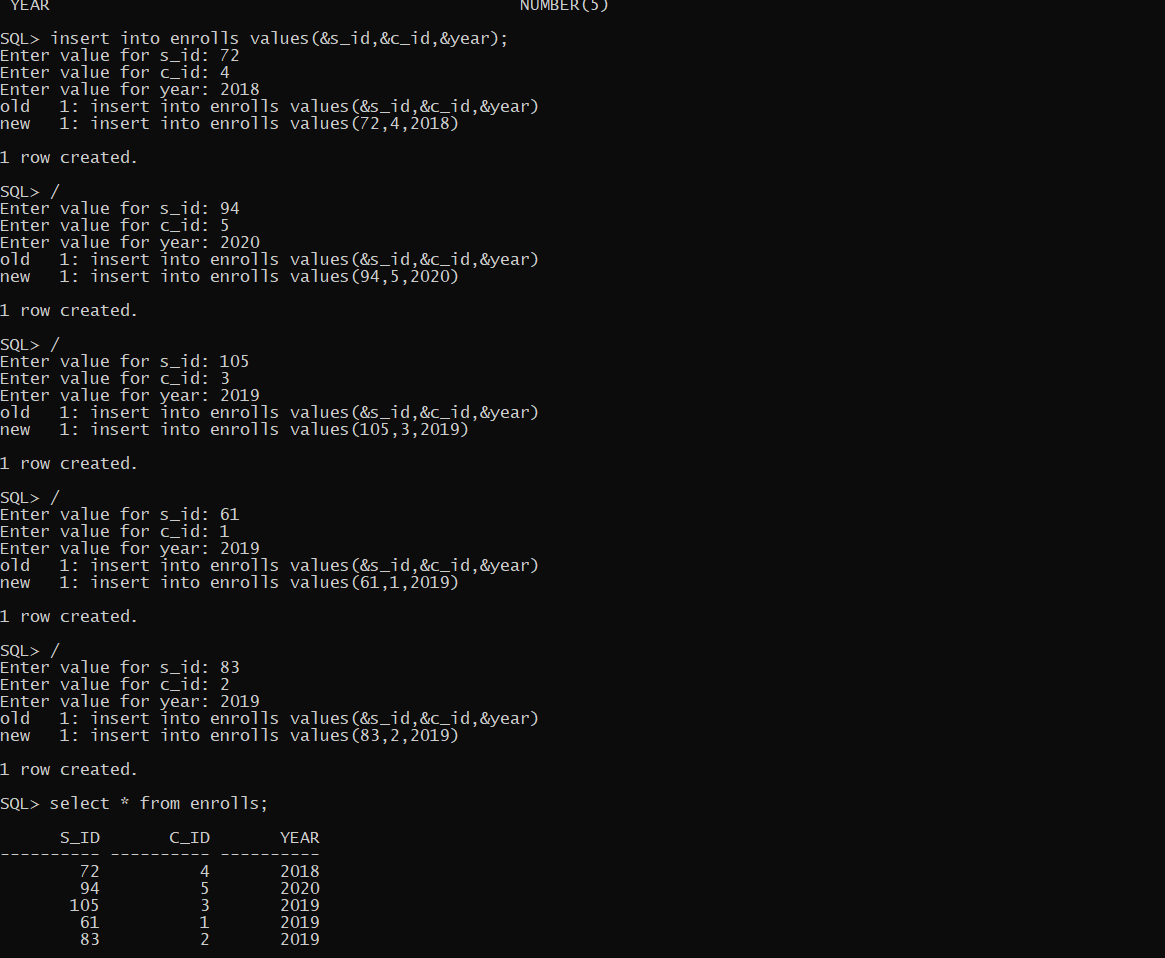
**DML commands:**

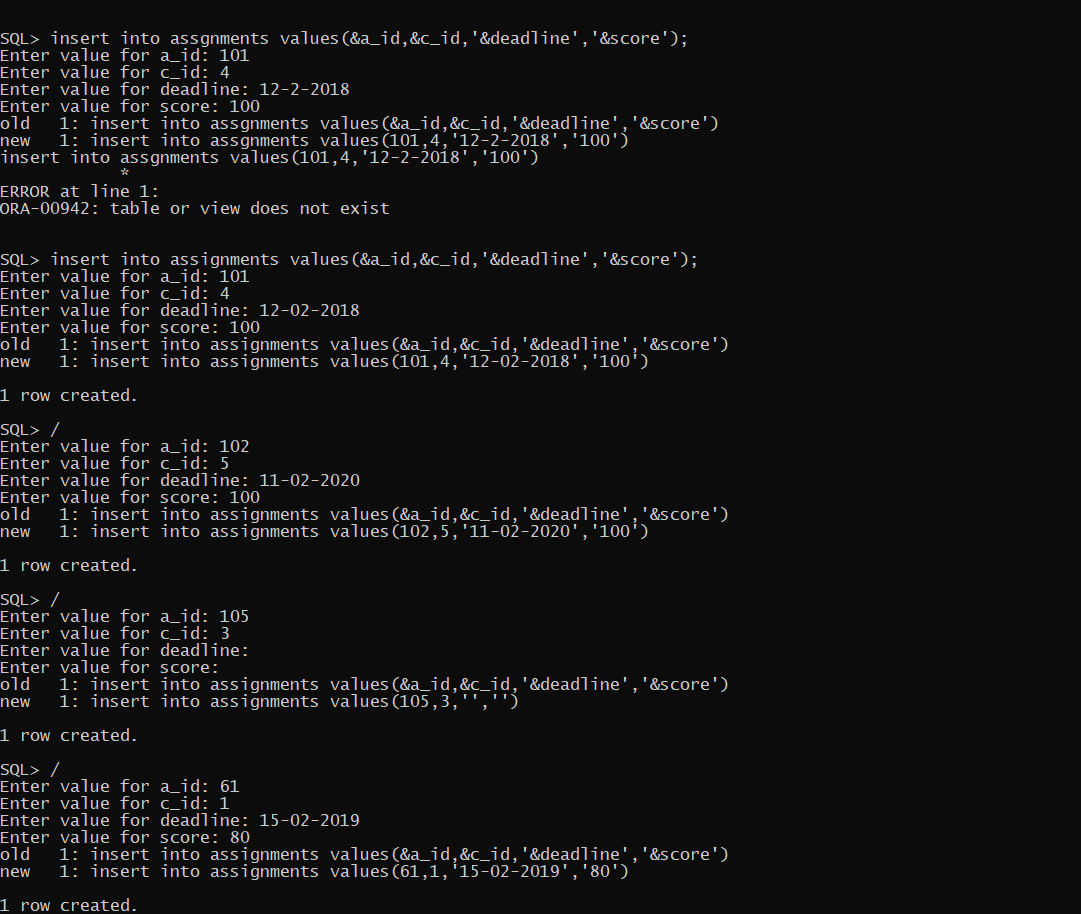
Inserting values into the tables.

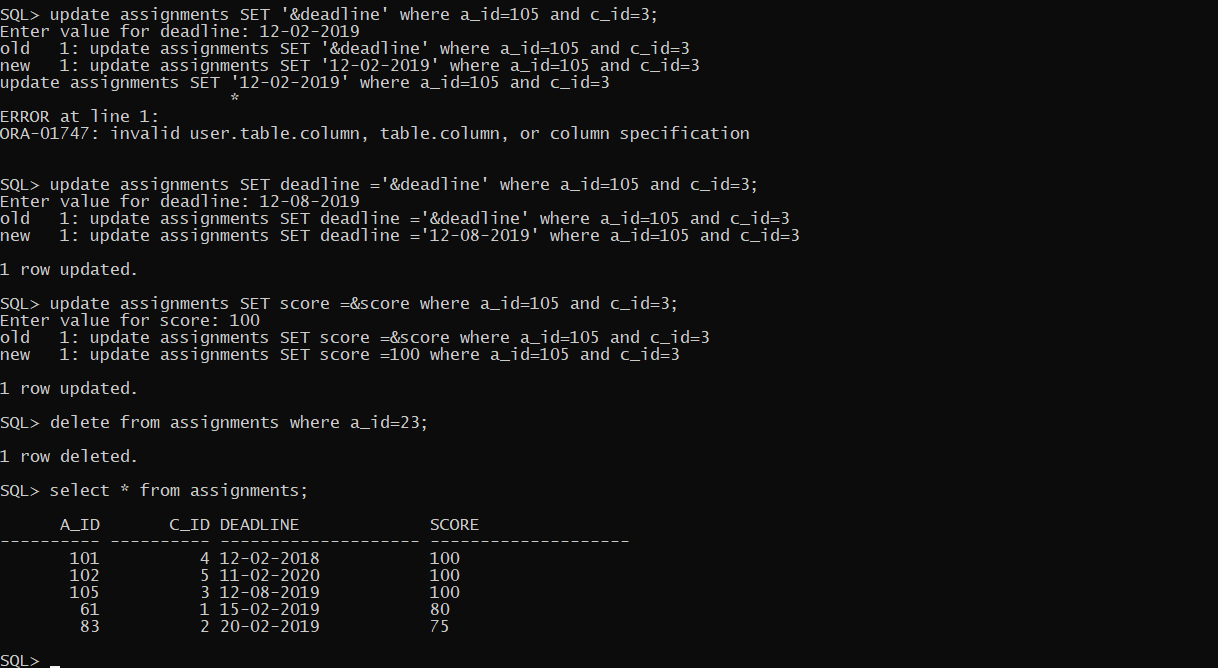


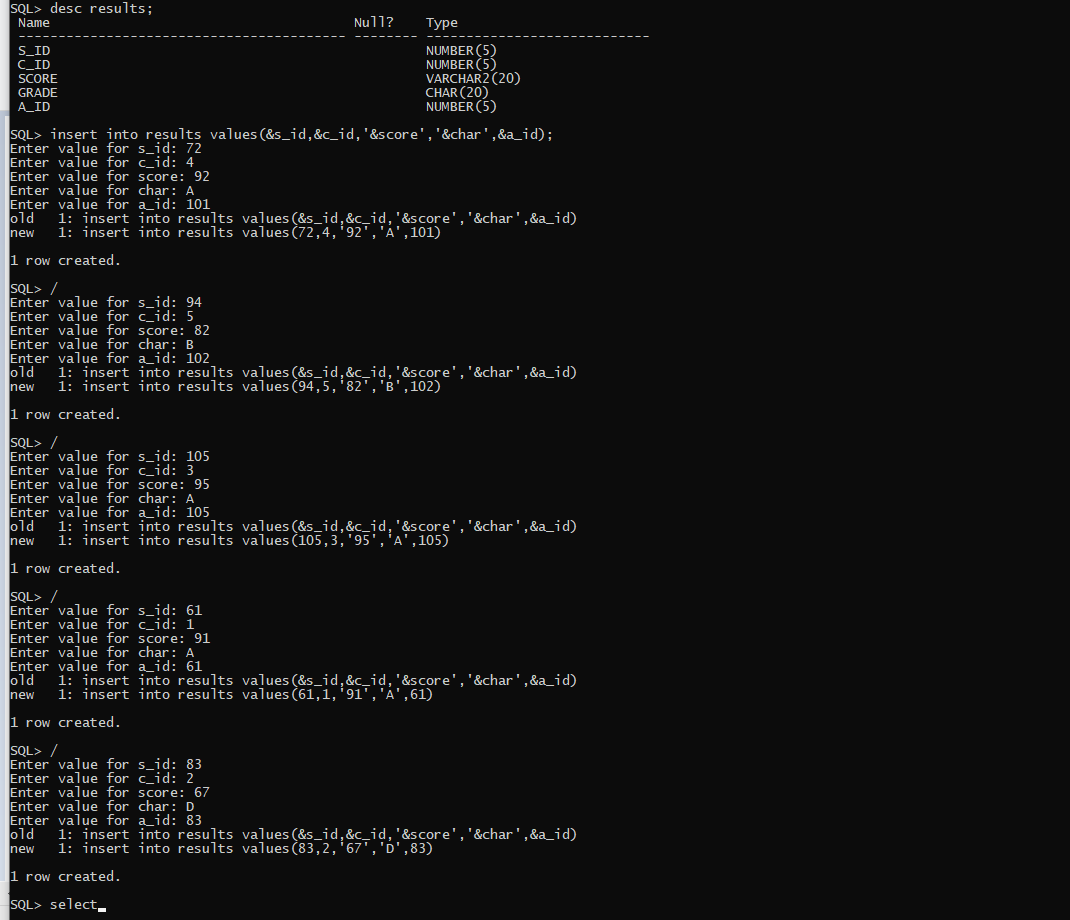


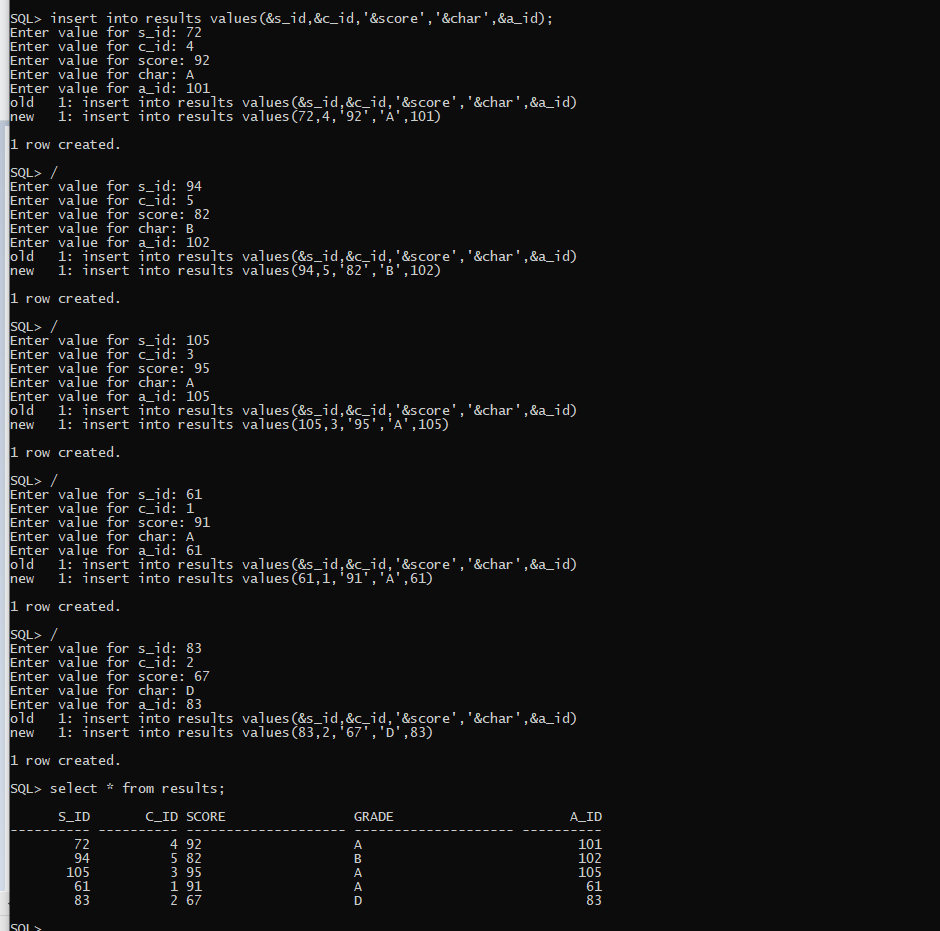












**Implementation**

* Front end programs:

1)Insert a Student:

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class InsertStudent extends Panel

{

Button insertStudentButton;

TextField sidText, fnameText, lnameText,unameText, passwordText;

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: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);

}

}

public void buildGUI()

{

//Handle Insert Account Button

insertStudentButton = new Button("Submit");

insertStudentButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

String query= "INSERT INTO student VALUES(" + sidText.getText() + ", " + "'" + fnameText.getText() + "'," + "'" + lnameText.getText() + "',"+"'"+unameText.getText()+"'," +"'"+passwordText.getText()+"'"+")";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

fnameText = new TextField(15);

sidText = new TextField(15);

lnameText = new TextField(15);

unameText = new TextField(15);

passwordText = new TextField(15);

errorText = new TextArea(10,40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5,2));

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

first.add(sidText);

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

first.add(fnameText);

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

first.add(lnameText);

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

first.add(unameText);

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

first.add(passwordText);

first.setBounds(125,90,300,150);

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

second.add(insertStudentButton);

second.setBounds(195,290,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(80,410,430,300);

setLayout(null);

add(first);

add(second);

add(third);

setSize(400,180);

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 ins = new InsertStudent();

ins.buildGUI();

}

}

2)Delete a Student:

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class DeleteStudent extends Panel{

Button deleteStudentButton;

List studentIDList;

TextField sidText, fnameText, lnameText, unameText,passwordText;

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: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);

}

}

private void loadStudent()

{

try

{

rs = statement.executeQuery("SELECT \* FROM STUDENT");

while (rs.next())

{

studentIDList.add(rs.getString("S\_ID"));

}

}

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 STUDENT");

while (rs.next())

{

if (rs.getString("S\_ID").equals(studentIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

sidText.setText(rs.getString("S\_ID"));

fnameText.setText(rs.getString("First\_NAME"));

lnameText.setText(rs.getString("Last\_Name"));

unameText.setText(rs.getString("User\_Name"));

passwordText.setText(rs.getString("Password"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteStudentButton = new Button("Delete");

deleteStudentButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM student WHERE S\_ID = "

+ studentIDList.getSelectedItem());

errorText.append("\nDeleted " + i + " rows successfully");

sidText.setText(null);

fnameText.setText(null);

lnameText.setText(null);

unameText.setText(null);

passwordText.setText(null);

studentIDList.removeAll();

loadStudent();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

sidText = new TextField(15);

fnameText = new TextField(15);

lnameText = new TextField(15);

unameText = new TextField(15);

passwordText= new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(6, 1));

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

first.add(sidText);

sidText.setEditable(false);

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

first.add(fnameText);

fnameText.setEditable(false);

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

first.add(lnameText);

lnameText.setEditable(false);

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

first.add(unameText);

unameText.setEditable(false);

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

first.add(passwordText);

passwordText.setEditable(false);

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();

}

}

3) Update a Student:

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class UpdateStudent extends Panel

{

Button updateStudentButton;

List studentIDList;

TextField sidText,fnameText,lnameText,unameText,passwordText;

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: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);

}

}

private void loadStudent()

{

try

{

rs = statement.executeQuery("SELECT S\_ID FROM STUDENT");

while (rs.next())

{

studentIDList.add(rs.getString("S\_ID"));

}

}

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 Student");

while (rs.next())

{

if (rs.getString("S\_ID").equals(studentIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

sidText.setText(rs.getString("S\_ID"));

fnameText.setText(rs.getString("First\_NAME"));

lnameText.setText(rs.getString("Last\_name"));

unameText.setText(rs.getString("USER\_NAME"));

passwordText.setText(rs.getString("PASSWORD"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Sailor Button

updateStudentButton = new Button("Modify");

updateStudentButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE STUDENT SET First\_name='" + fnameText.getText() + "', Last\_name='" + lnameText.getText() + "', "

+ "User\_name ='"+unameText.getText()+"',"+"password ='"+passwordText.getText()+"'"+" WHERE S\_id ="

+ studentIDList.getSelectedItem());

errorText.append("\nUpdated " + i + " rows successfully");

studentIDList.removeAll();

loadStudent();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

sidText = new TextField(15);

sidText.setEditable(false);

fnameText = new TextField(15);

lnameText = new TextField(15);

unameText = new TextField(15);

passwordText=new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

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

first.add(sidText);

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

first.add(fnameText);

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

first.add(lnameText);

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

first.add(unameText);

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

first.add(passwordText);

Panel second = new Panel(new GridLayout(5, 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 us = new UpdateStudent();

us.buildGUI();

}

}

4)Main method

import java.awt.\*;

import java.awt.event.\*;

class OnlineMoocsProvider extends Frame implements ActionListener

{

String msg = "";

Label ll,l2;

CardLayout cardLO;

//Create Panels for each of the menu items, welcome screen panel and home screen panel with CardLayout

InsertProvider provide;

UpdateProvider upp;

DeleteProvider delp;

InsertCourses inc;

UpdateCourses upc;

DeleteCourses delc;

InsertStudent ins;

DeleteStudent dels;

UpdateStudent us;

Enroll mks;

UpdateEnroll upe;

DeleteEnroll dele;

InsertAssignments ina;

UpdateAssignments upa;

DeleteAssignment dela;

InsertResults inr;

DeleteResults delr;

UpdateResults upr;

Panel home,welcome;

OnlineMoocsProvider()

{

cardLO = new CardLayout();

//Create an empty home panel and set its layout to card layout

home = new Panel();

home.setLayout(cardLO);

ll = new Label();

l2 =new Label();

ll.setAlignment(Label.CENTER);

l2.setAlignment(Label.CENTER);

ll.setText("Welcome to Online MOOC's Provider");

l2.setText("All @rights are reserved");

//Create welcome panel and add the label to it

welcome = new Panel();

welcome.add(ll);

welcome.add(l2);

//create panels for each of our menu items and build them with respective components

provide = new InsertProvider(); provide.buildGUI();

upp = new UpdateProvider(); upp.buildGUI();

delp = new DeleteProvider(); delp.buildGUI();

inc = new InsertCourses();inc.buildGUI();

upc= new UpdateCourses();upc.buildGUI();

delc = new DeleteCourses();delc.buildGUI();

ins = new InsertStudent();ins.buildGUI();

dels = new DeleteStudent();dels.buildGUI();

us= new UpdateStudent();us.buildGUI();

mks= new Enroll(); mks.buildGUI();

upe= new UpdateEnroll();upe.buildGUI();

dele = new DeleteEnroll(); dele.buildGUI();

ina = new InsertAssignments();ina.buildGUI();

upa = new UpdateAssignments();upa.buildGUI();

dela = new DeleteAssignment();dela.buildGUI();

inr = new InsertResults();inr.buildGUI();

delr = new DeleteResults();delr.buildGUI();

upr = new UpdateResults();upr.buildGUI();

//add all the panels to the home panel which has a cardlayout

home.add(welcome, "Welcome");

home.add(provide, "InsertProvider");

home.add(upp, "UpdateProvider");

home.add(delp, "DeleteProvider");

home.add(inc,"InsertCourses");

home.add(upc,"UpdateCourses");

home.add(delc,"DeleteCourses");

home.add(ins,"InsertStudent");

home.add(dels,"DeleteStudent");

home.add(us,"UpdateStudent");

home.add(mks,"Enroll");

home.add(upe,"UpdateEnroll");

home.add(dele,"DeleteEnroll");

home.add(ina,"InsertAssignments");

home.add(upa,"UpdateAssignments");

home.add(dela,"DeleteAssignment");

home.add(inr,"InsertResults");

home.add(delr,"DeleteResults");

home.add(upr,"UpdateResults");

//home.add(upb, "UpdateBoat");

//home.add(mks, "MakeReserve");

// 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 provider = new Menu("OnlineMOOC'sProvider");

MenuItem item1, item2, item3;

provider.add(item1 = new MenuItem("Insert Provider"));

provider.add(item2 = new MenuItem("View Provider"));

provider.add(item3 = new MenuItem("Delete Provider"));

mbar.add(provider);

Menu courses = new Menu("Courses");

MenuItem item4, item5, item6;

courses.add(item4 = new MenuItem("Insert Courses"));

courses.add(item5 = new MenuItem("View Courses"));

courses.add(item6 = new MenuItem("Delete Courses"));

mbar.add(courses);

Menu student = new Menu("Student");

MenuItem item7, item8, item9;

student.add(item7 = new MenuItem("Insert Student"));

student.add(item8 = new MenuItem("View Student"));

student.add(item9 = new MenuItem("Delete Student"));

mbar.add(student);

Menu enroll= new Menu("Enroll");

MenuItem item10, item11, item12;

enroll.add(item10 = new MenuItem("Insert Enroll"));

enroll.add(item11= new MenuItem("View Enroll"));

enroll.add(item12 = new MenuItem("Delete Enroll"));

mbar.add(enroll);

Menu assignments= new Menu("Assignments");

MenuItem item13, item14, item15;

assignments.add(item13 = new MenuItem("Insert Assignments"));

assignments.add(item14= new MenuItem("View Assignments"));

assignments.add(item15 = new MenuItem("Delete Assignment"));

mbar.add(assignments);

Menu results= new Menu("Results");

MenuItem item16, item17, item18;

results.add(item16 = new MenuItem("Insert Results"));

results.add(item17= new MenuItem("View Results"));

results.add(item18 = new MenuItem("Delete Results"));

mbar.add(results);

// 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);

// 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("Online MOOC's Provider");

Color clr = new Color(50, 150, 100);

setBackground(clr);

setFont(new Font("Monaco", Font.BOLD, 20));

setSize(900, 1000);

setVisible(true);

}

public void actionPerformed(ActionEvent ae)

{

String arg = ae.getActionCommand();

if(arg.equals("Insert Provider"))

{

cardLO.show(home, "InsertProvider");

}

else if(arg.equals("View Provider"))

{

cardLO.show(home, "UpdateProvider");

}

else if(arg.equals("Delete Provider"))

{

cardLO.show(home, "DeleteProvider");

}

else if(arg.equals("Insert Courses"))

{

cardLO.show(home, "InsertCourses");

}

else if(arg.equals("Delete Courses"))

{

cardLO.show(home, "DeleteCourses");

}

else if(arg.equals("View Courses"))

{

cardLO.show(home, "UpdateCourses");

}

else if(arg.equals("Insert Student"))

{

cardLO.show(home, "InsertStudent");

}

else if(arg.equals("Delete Student"))

{

cardLO.show(home, "DeleteStudent");

}

else if(arg.equals("View Student"))

{

cardLO.show(home, "UpdateStudent");

}

else if(arg.equals("Insert Enroll"))

{

cardLO.show(home, "Enroll");

}

else if(arg.equals("View Enroll"))

{

cardLO.show(home, "UpdateEnroll");

}

else if(arg.equals("Delete Enroll"))

{

cardLO.show(home, "DeleteEnroll");

}

else if(arg.equals("Insert Assignments"))

{

cardLO.show(home, "InsertAssignments");

}

else if(arg.equals("View Assignments"))

{

cardLO.show(home, "UpdateAssignments");

}

else if(arg.equals("Insert Results"))

{

cardLO.show(home, "InsertResults");

}

else if(arg.equals("View Results"))

{

cardLO.show(home, "UpdateResults");

}

else if(arg.equals("Delete Results"))

{

cardLO.show(home, "DeleteResults");

}

}

public static void main(String ... args)

{

new OnlineMoocsProvider();

}

}

Connectivity with the 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:1521:orcl","sreya","sreya");

statement=connection.createStatement();

}

catch(SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

GITHUB LINK:

https://github.com/sherlahemanth001/dbms-assignment-2.git

**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.