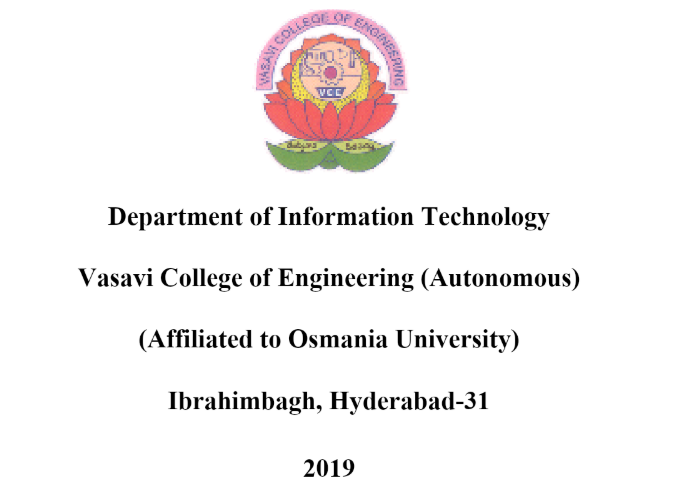
**JAVA AWT BASED- SQL CONNECTIVITY USING JDBC**

A REPORT BASED ON THE PROJECT:

IPL DATABASE

BY:

N.SANDALI <1602-18-737-103>



**BONAFIDE CERTIFICATE**

This is to certify that the project report titled “**IPL database**” project work of Miss N.Sandali bearing Roll no:1602-18-737-103 who carried out this project under my supervision in the IV semester for the academic year 2019-2020.

*Signature Signature*

*External examine* *Internal examine*

**ABSTRACT**

To create an IPL database, we will require a total of 6 table. One- for the team details, two- for the player details, three- for the sponsors, four- for the match venue details, five- for the city it represents and six- for the owner who owns the team. The basic attributes would be the name and ID of any entity, besides this descriptive attributes can also be present. Entity names can have a domain type varchar2 where the ID can be number. The relationships between various entity sets helps in creating an IPL database. Few of these entities have primary keys which turn out to be the foreign keys in the rest.

**INTRODUCTION**

REQUIREMENTS FOR IPL DATABASE:

* Functional Requirements:

1. User-Interface: The system shall provide an easy-to-use user-interface. Any detail about the matches or players can be fetched without any trouble.
2. Detailed data: All the data that is available in the database can be accessed by the users. A detailed information is provided according to the user’s need.
3. Match Status: The statistics of the matches is recorded in the database which makes it easy for the users to plan for the next match strategies.
4. Player intake: A detailed elaboration of the players and captains of the particular teams are recorded in the database.
5. Mobility: Any changes in the database regarding the information can be made without any major changes occurring in the database.

* Security Requirements:

1. Player Authenticity: Ensures that the player has his own unique identity and permits only if that identity is provided to access sensitive information.
2. System Integrity: Ensure that the system cannot be re-configured during operation.
3. Data Integrity: Ensure that each detail is recorded as intended and cannot be tampered with in any manner, once recorded.
4. Secrecy / Privacy: No one should be able to determine any team’s strategy.
5. Reliability: IPL database should work robustly, without loss of any details, even in the face of numerous failures. The database shall be developed in a manner that ensures there is no malicious code or bugs.

* Through the project:

The main goal to be achieved through this project was to provide an

opportunity to display the details of various players, teams, owners, sponsors, matches, and cities taking part in the IPL. The project also ensure that the information that are recorded are pretty much confidential and are provided only if user has access to this database.

SQL particular player, team, owner, sponsor, match, city 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 players, teams, owners, sponsors, matches and cities 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

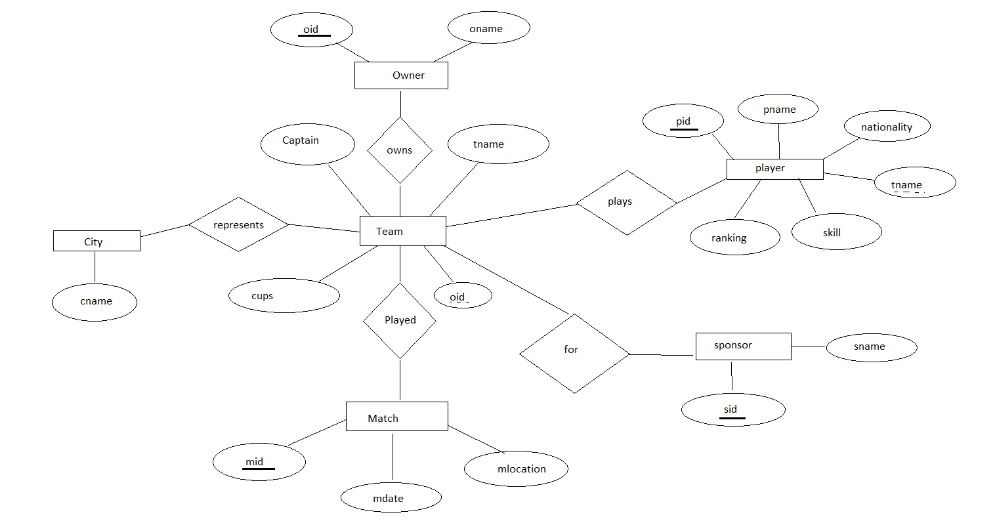
front end application is designed through Eclipse, runs and has the

capacity to connect with the database which has data inserted using

SQL.

* Design:

ER diagram



* DATABASE DESIGN:

CONTENT:

● Abstract

● ER Diagram

● Logical database design – DDL commands

● Enforcing primary and foreign keys.

● DML operation and outputs.

**ABSTRACT**

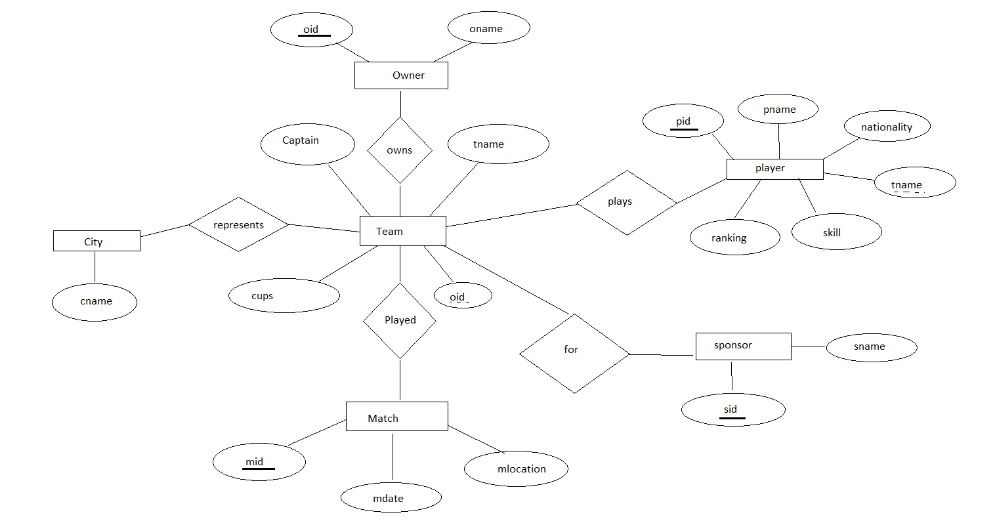
To create an IPL database, we will require a total of 6 table. One- for the team details, two- for the player details, three- for the sponsors, four- for the match venue details, five- for the city it represents and six- for the owner who owns the team. The basic attributes would be the name and ID of any entity, besides this descriptive attributes can also be present. Entity names can have a domain type varchar2 where the ID can be number. The relationships between various entity sets helps in creating an IPL database. Few of these entities have primary keys which turn out to be the foreign keys in the rest. The six tables are:

* Player
* Team
* Owner
* Sponsor
* Match
* City

LIST OF REQUIREMENTS:

|  |  |
| --- | --- |
| List of Players. |  |
| Details of players, teams and owners. |  |
| Details of sponsors. |  |
| Location of the match being played. |  |

**ER Diagram:**

****

**CONSTRAINTS APPLIED:**

The database has two constraints that are applied- the primary key constraint and foreign key constraint. The primary keys are:

1)oid in the owner table.

2)pid in the player table.

3)mid in the match table.

4)sid in the sponsor table.

The attributes oid and tname act as foreign keys in the team table and player table respectively.

**DDL Commands:**

**For creating player table:**

Query: create table player( pid number(5) primary key, pname varchar2(20), ranking number(5), nationality varchar2(20), skill varchar2(20), tname varchar2(20));

**For creating Team table:**

Query: create table team( captain varchar2(20), tname varchar2(20), cups number(5));

**For creating Owner table:**

Query: create table owner( oid number(5) primary key, oname varchar2(20));

**For creating Sponsor table:**

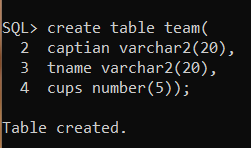
Query: create table sponsor( sid number(5) primary key, sname varchar2(20));

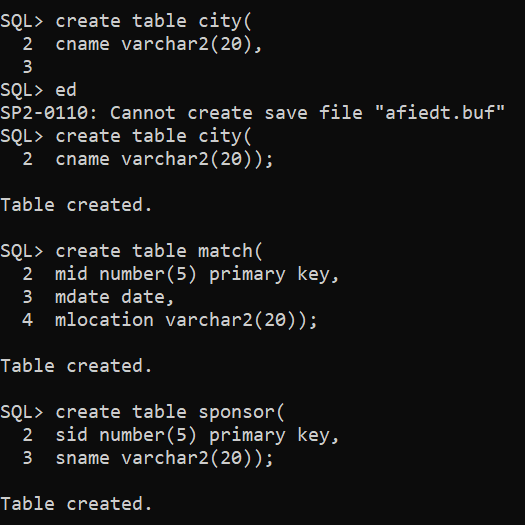
**For creating Match table:**

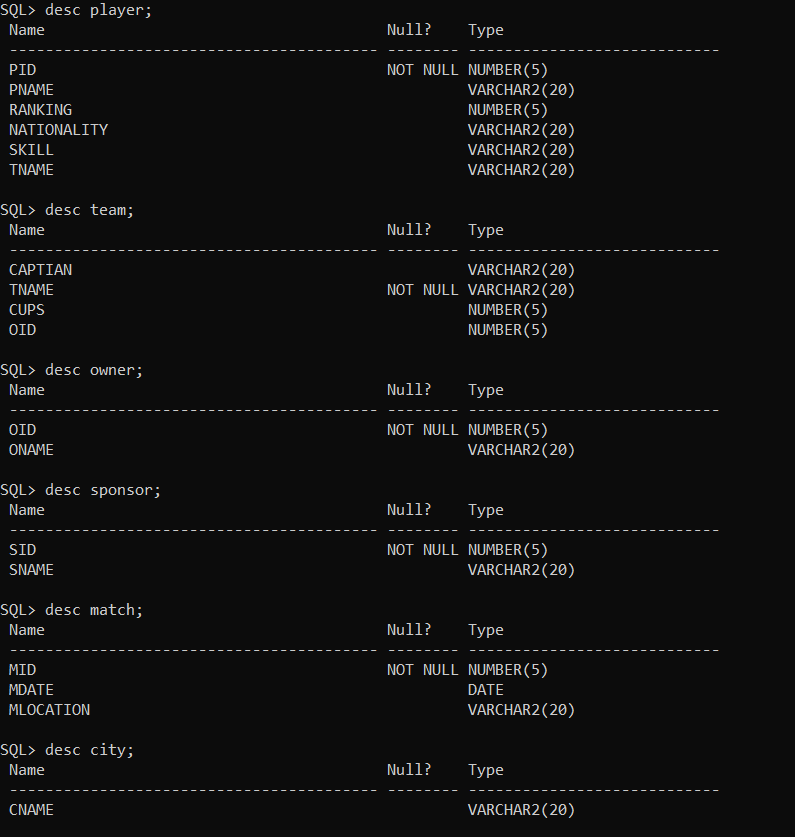
Query: create table match( mid number(5) primary key, mdate date, mlocation varchar2(20));

**For creating Team table:**

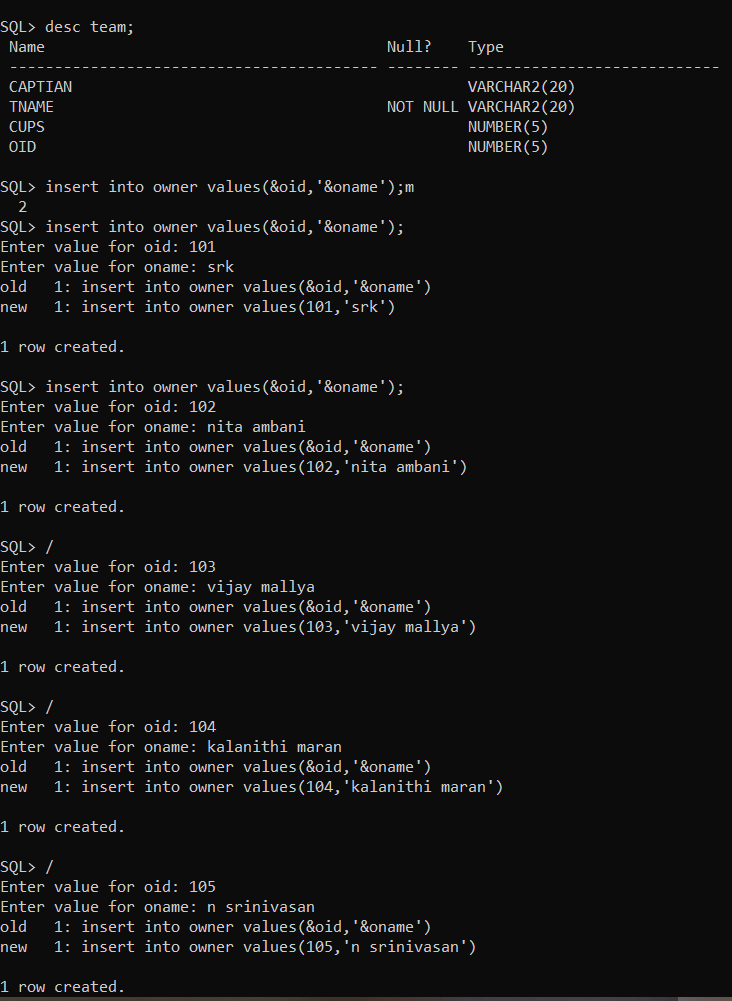
Query: create table city( cname varchar2(20));

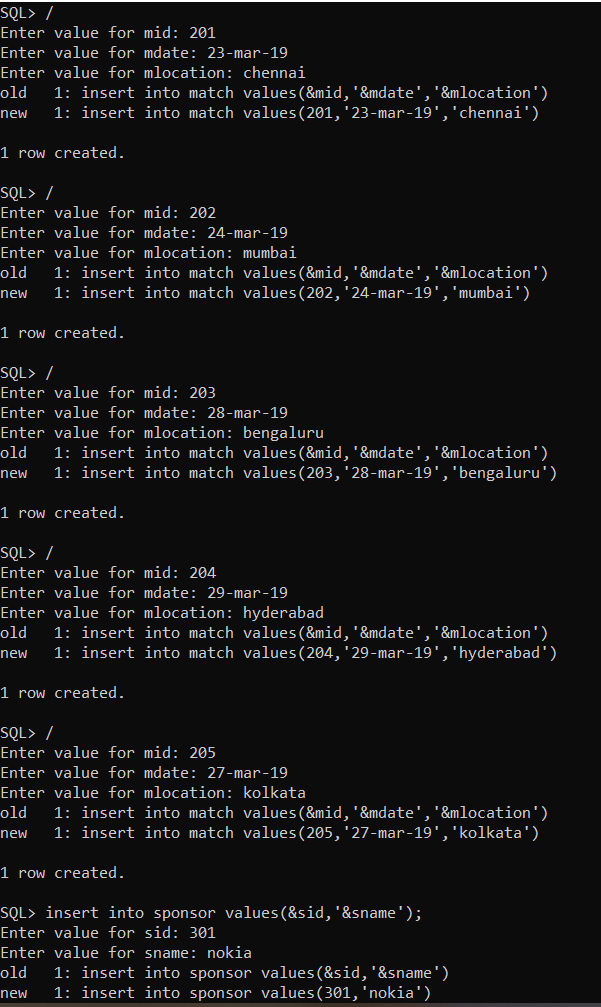


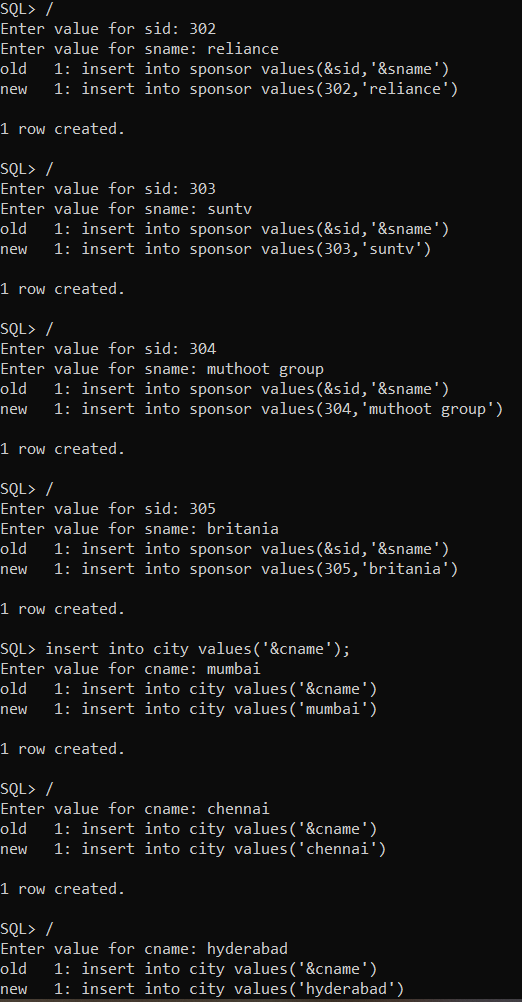


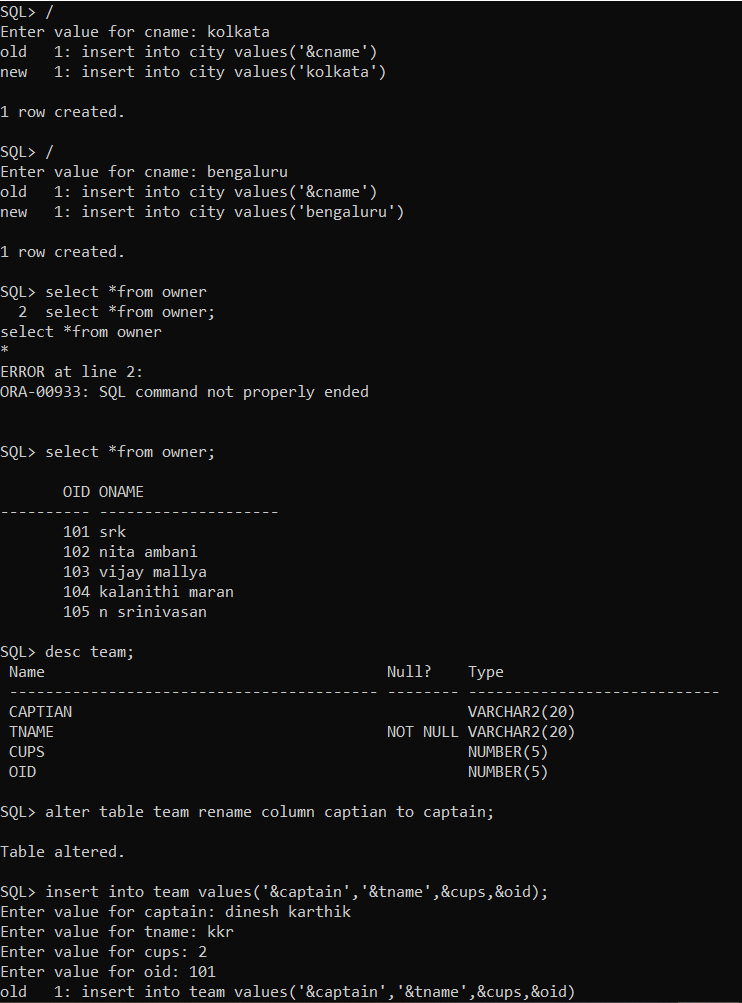


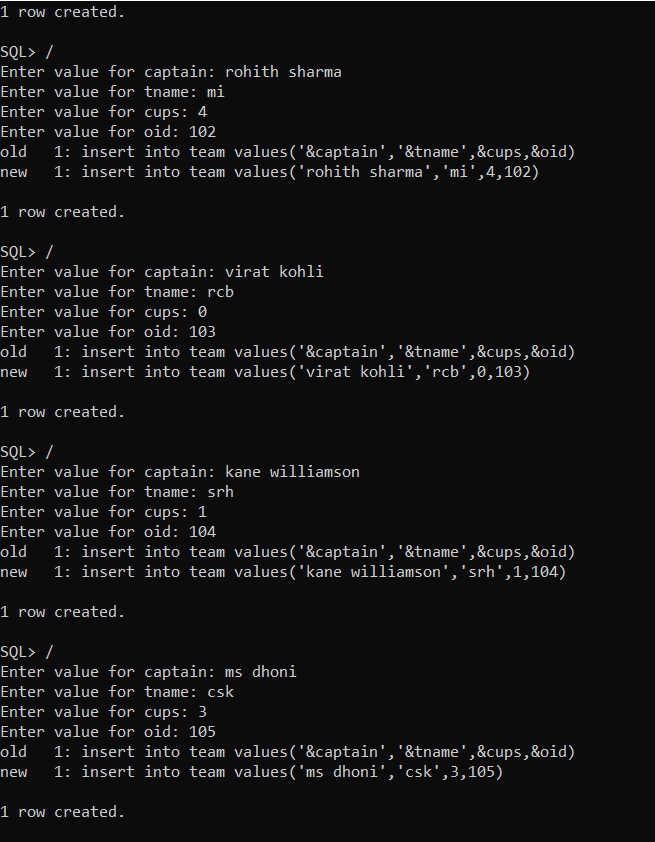
**DML Commands:**

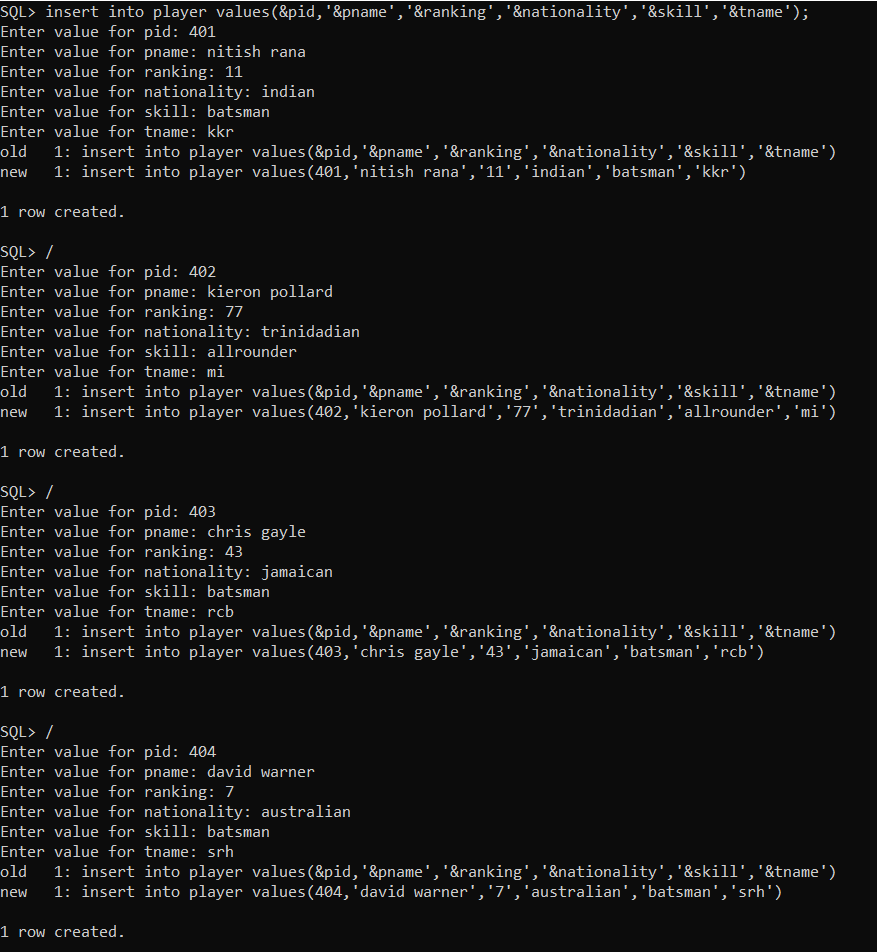


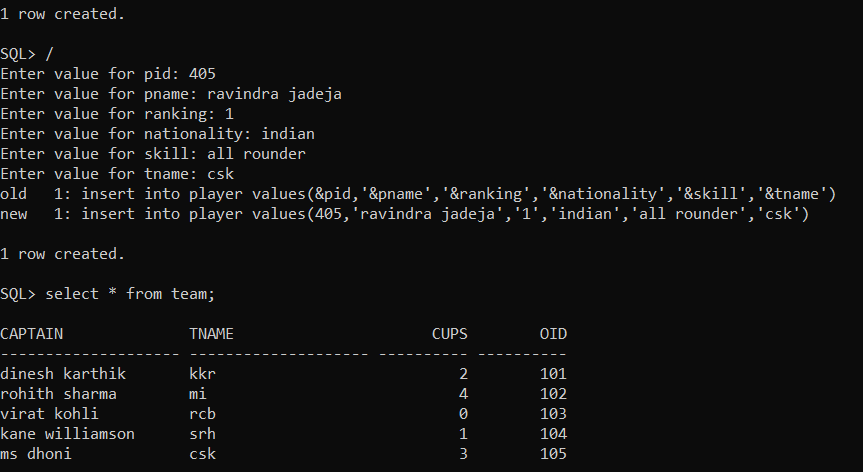


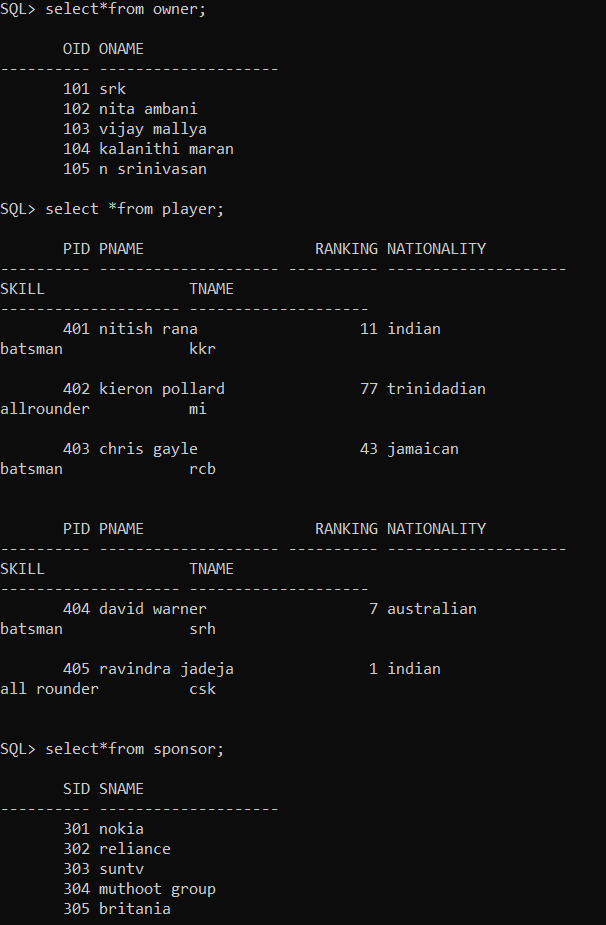


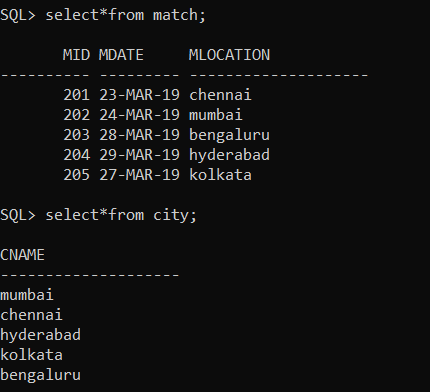












**Implementation**

* Front end programs:

1. Insert a Player:

import java.awt.Button;

import java.awt.Panel;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.\*;

import java.sql.\*;

public class insertPlayer extends Panel{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button insertPlayerButton;

TextField pid,pname,ranking,nationality,skill,tname;

TextArea errorText;

Connection connection;

Statement statement;

public insertPlayer()

{

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","sandali","sandali");

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()

{

insertPlayerButton = new Button("InsertPlayer");

insertPlayerButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

String query="INSERT INTO player VALUES(" + pid.getText() + ", " + "'" + pname.getText() + "',"+"" + ranking.getText() + ","+"" + nationality.getText() + ","+"" + skill.getText() + ","+"" + tname.getText() + ")";

int i=statement.executeUpdate(query);

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

}

catch(SQLException insertException)

{

displaySQLErrors(insertException);

}

}

}

);

pid=new TextField(20);

pname=new TextField(20);

ranking=new TextField(20);

nationality=new TextField(20);

skill=new TextField(20);

tname=new TextField(20);

errorText=new TextArea(10,80);

errorText.setEditable(false);

Panel first=new Panel();

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

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

first.add(pid);

first.add(new Label("PLAYER NAME:"));

first.add(pname);

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

first.add(ranking);

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

first.add(nationality);

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

first.add(skill);

first.add(new Label("TEAM NAME:"));

first.add(tname);

first.setBounds(125,90,200,100);

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

second.add(insertPlayerButton);

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("VoterError: "+e.getErrorCode()+"\n");

}

public static void main(String[] args)

{

insertPlayer player=new insertPlayer();

player.buildGUI();

}

}

1. Delete a Player:

import java.awt.Button;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.Label;

import java.awt.List;

import java.awt.Panel;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.\*;

import java.sql.\*;

public class deletePlayer extends Panel

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button deletePlayerButton;

List PlayerIDList;

TextField pid,pname,ranking,nationality,skill,tname;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public deletePlayer()

{

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","sandali","sandali");

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 loadProvider()

{

try

{

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

while (rs.next())

{

playerIDList.add(rs.getString("PID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

playerIDList = new List(10);

loadProvider();

add(playerIDList);

playerIDList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if(rs.getString("PID").equals(playerIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

pid.setText(rs.getString("PID"));

pname.setText(rs.getString("PNAME"));

ranking.setText(rs.getString("RANKING"));

nationality.setText(rs.getString("NATIONALITY"));

skill.setText(rs.getString("SKILL"));

tname.setText(rs.getString("TNAME"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deletePlayerButton = new Button("Delete Player");

deletePlayerButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM player WHERE PID = "+ playerIDList.getSelectedItem());

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

pid.setText(null);

pname.setText(null);

ranking.setText(null);

nationality.setText(null);

skill.setText(null);

tname.setText(null);

playerIDList.removeAll();

loadProvider();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

pid = new TextField(15);

pname = new TextField(15);

ranking = new TextField(15);

nationality = new TextField(15);

skill = new TextField(15);

tname = 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("Player ID:"));

first.add(pid);

pid.setEditable(false);

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

first.add(pname);

pname.setEditable(false);

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

first.add(ranking);

ranking.setEditable(false);

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

first.add(nationality);

nationality.setEditable(false);

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

first.add(skill);

skill.setEditable(false);

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

first.add(tname);

tname.setEditable(false);

Panel second = new Panel(new GridLayout(5, 1));

second.add(deletePlayerButton);

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)

{

deletePlayer del = new deletePlayer();

del.buildGUI();

}

}

1. Update a Player:

import java.awt.Button;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.Label;

import java.awt.List;

import java.awt.Panel;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.\*;

import java.sql.\*;

public class updatePLayer extends Panel

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button updatePlayerButton;

List PlayerIDList;

TextField pid,pname,ranking,nationality,skill,tname;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public updatePlayer()

{

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","sandali","sandali");

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 loadProvider()

{

try

{

rs = statement.executeQuery("SELECT PID FROM PLAYER");

while (rs.next())

{

PlayerIDList.add(rs.getString("PID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

PlayerIDList = new List(10);

loadProvider();

add(PlayerIDList);

PlayerIDList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("PID").equals(PlayerIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

pid.setText(rs.getString("PID"));

pname.setText(rs.getString("PNAME"));

ranking.setText(rs.getString("RANKING"));

nationality.setText(rs.getString("NATIONALITY"));

skill.setText(rs.getString("SKILL"));

tname.setText(rs.getString("TNAME"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

updatePlayerButton = new Button("Update Player");

updatePlayerButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE Player "

+ "SET pname='" + pname.getText() + "', "

+ "ranking=" + ranking.getText() + "', "

+ "nationality='" + nationality.getText() + "', "

+ "skill='" + skill.getText() + "', "

+ "tname='" + tname.getText() +

" WHERE pid = "+ playerIDList.getSelectedItem());

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

PlayerIDList.removeAll();

loadProvider();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

pid = new TextField(15);

pid.setEditable(false);

pname = new TextField(15);

ranking = new TextField(15);

nationality = new TextField(15);

skill = new TextField(15);

tname = 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("Player ID:"));

first.add(pid);

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

first.add(pname);

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

first.add(ranking);

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

first.add(nationality);

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

first.add(skill);

first.add(new Label("Team name:"));

first.add(tname);

Panel second = new Panel(new GridLayout(5, 1));

second.add(updatePlayerButton);

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)

{

updatePlayer ua = new updatePlayer();

ua.buildGUI();

}

}

1. Main method:

import java.awt.\*;

import java.awt.event.\*;

class Ipldatabase extends Frame implements ActionListener

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

String msg = "";

Label ll,l2;

CardLayout cardLO;

insertPlayer inp;

updatePlayer upp;

deletePlayer delp;

insertTeam int;

updateTeam upt;

deleteTeam delt;

insertOwner ino;

updateOwner upo;

deleteOwner delo;

insertCity inc;

updateCity upc;

deleteCity delc;

insertSponsor ins;

updateSponsor ups;

deleteSponsor dels;

insertMatch inm;

updateMatch upm;

deleteMatch delm;

Panel home,welcome;

Ipldatabase()

{

cardLO = new CardLayout();

home = new Panel();

home.setLayout(cardLO);

ll = new Label();

l2 =new Label();

ll.setAlignment(Label.CENTER);

l2.setAlignment(Label.CENTER);

ll.setText("Welcome to Indian Premier database");

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

welcome = new Panel();

welcome.add(ll);

welcome.add(l2);

inp = new insertPlayer();

inp.buildGUI();

upp = new updatePlayer();

upp.buildGUI();

delp = new deletePlayer();

delp.buildGUI();

int = new insertTeam();

int.buildGUI();

upt= new updateTeam();

upt.buildGUI();

delt = new deleteTeam();

delt.buildGUI();

ino = new insertOwner();

ino.buildGUI();

upo = new updateOwner();

upo.buildGUI();

delo = new deleteOwner();

delo.buildGUI();

inc = new insertCity();

inc.buildGUI();

upc = new updateCity();

upc.buildGUI();

delc = new deleteCity();

delc.buildGUI();

ins = new insertSponsor();

ins.buildGUI();

ups = new updateSponsor();

ups.buildGUI();

dels = new deleteSponsor();

dels.buildGUI();

inm =new insertMatch();

inm.buildGUI();

upm =new updateMatch();

upm.buildGUI();

delm=new deleteMatch();

delm.buildGUI();

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

home.add(welcome, "Welcome");

home.add(inp, "inserPlayer");

home.add(upp, "updatePlayer");

home.add(delp, "deletePlayer");

home.add(int, "insertTeam");

home.add(upt, "updateTeam");

home.add(delt, "deleteTeam");

home.add(ino, "insertOwner");

home.add(upo, "updateOwner");

home.add(delo, "deleteOwner");

home.add(inc, "insertCity");

home.add(upc, "updateCity");

home.add(delc, "deleteCity");

home.add(ins, "insertSponsor");

home.add(ups, "updateSponsor");

home.add(dels, "deleteSponsor");

home.add(inm, "insertMatch");

home.add(upm, "updateMatch");

home.add(delm, "deleteMatch");

// 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 Player = new Menu("Player");

MenuItem item1, item2, item3;

player.add(item1 = new MenuItem("Insert Player"));

player.add(item2 = new MenuItem("View Player"));

player.add(item3 = new MenuItem("Delete Player"));

mbar.add(player);

Menu Owner = new Menu("Owner");

MenuItem item4, item5, item6;

Owner.add(item4 = new MenuItem("Insert Owner"));

Owner.add(item5 = new MenuItem("View Owner"));

Owner.add(item6 = new MenuItem("Delete Owner"));

mbar.add(owner);

Menu Team = new Menu("Team");

MenuItem item7, item8, item9;

Team.add(item7 = new MenuItem("Insert Team"));

Team.add(item8 = new MenuItem("Update Team"));

Team.add(item9 = new MenuItem("Delete Team"));

mbar.add(team);

Menu Sponsor = new Menu("Sponsor");

MenuItem item10, item11, item12;

sponsor.add(item10 = new MenuItem("Insert sponsor"));

sponsor.add(item11 = new MenuItem("Update sponsor"));

sponsor.add(item12 = new MenuItem("Delete sponsor"));

mbar.add(sponsor);

Menu city = new Menu("City");

MenuItem item13, item14, item15;

city.add(item13 = new MenuItem("Insert City"));

city.add(item14 = new MenuItem("Update City"));

city.add(item15 = new MenuItem("Delete City"));

mbar.add(city);

Menu match = new Menu("Match");

MenuItem item16, item17, item18;

match.add(item16 = new MenuItem("Insert match"));

match.add(item17 = new MenuItem("Update match"));

match.add(item18 = new MenuItem("Delete match"));

mbar.add(match);

// 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);

addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent we)

{

System.exit(0);

}

});

//Frame properties

setTitle("IPL database");

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

setBackground(clr);

setFont(new Font("SansSerif", Font.CENTER\_BASELINE, 18));

setSize(900, 1000);

setVisible(true);

}

public void actionPerformed(ActionEvent ae)

{

String arg = ae.getActionCommand();

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

{

cardLO.show(home, "insertPlayer");

}

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

{

cardLO.show(home, "updatePlayer");

}

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

{

cardLO.show(home, "deletePlayer");

}

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

{

cardLO.show(home, "insertTeam");

}

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

{

cardLO.show(home, "updateTeam");

}

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

{

cardLO.show(home, "deleteTeam");

}

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

{

cardLO.show(home, "insertOwner");

}

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

{

cardLO.show(home, "updateOwner");

}

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

{

cardLO.show(home, "deleteOwner");

}

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

{

cardLO.show(home, "insertSponsor");

}

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

{

cardLO.show(home, "updateSponsor");

}

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

{

cardLO.show(home, "deleteSponsor");

}

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

{

cardLO.show(home, "insertCity");

}

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

{

cardLO.show(home, "updateCity");

}

else if(arg.euqals("Delete City"))

{

cardLO.show(home, "deleteCity");

}

else if(arg.euqals("Insert Match"));

{

cardLO.show(home, "insertmatch");

}

else if(arg.equals("view Match"))

{

cardLO.show(home, "viewmatch");

}

else

{

cardLO.show(home, "deletematch");

}

}

public static void main(String ... args)

{

new Ipldatabase();

}

}

**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","sandali","sandali");

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/sandalinemmaniwar/dbms-assignment-2.git

**TESTING**

The program runs for execution of three basic operations of insertion, update and delete on 6 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.

**DISCUSSIONS**

The application “IPL Database” helps to find the details about the players ,teams and various attributes in involved in the ipl. A user can check the details he/she requires but access to any confidential details is provided only if the user has his/her specific identity which needs to be recorded in the database.

The details of the teams after every match is played is recorded in the database and can be accessed whenever required.

The data entered in the database is highly secure and can only be altered by the authorized person through their own identity. The data entered is stored into the database immediately to avoid loss or tampering of data.

The update choice is only provided to corresponding coordinator who is given the access to the database.

**REFERENCES**

1. <https://github.com/sandalinemmaniwar/DBMS-ASSIGNMENT1>
2. <https://en.wikipedia.org/wiki/List_of_Indian_Premier_League_seasons_and_results>