**Centennial College**

**COMP 228: Java Programming**

**LAB #5 - Developing Database Applications using JDBC.**

**Student:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Due Date: Week 11.

Purpose: The purpose of this Lab assignment is to:

1. Practice JDBC in Java Applications
2. Develop a GUI Java application with data access capabilities

References: Read the textbook, ppt slides, and consult references (if any).

This material provides the necessary information you need to complete the exercises.

Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students.

YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED LAB5:

1. THE FIRST ONE IS A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF EACH EXERCISE (If there are more than 1 exercise). DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD FOLLOW EACH QUESTION AND COVER ALL THE ASPECTS/FUNCTIONALITIES OF EACH EXERCISE. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.

2. SUBMIT ALSO ONE ZIPPED PROJECT FILE THAT CONTAINS ALL THE EXERISES SEPARATELY INTO THE SAME DROP BOX.

This material provides the necessary information you need to complete the exercises.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohSmith\_COMP228Lab5**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohSmith\_COMP228Lab5.zip**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

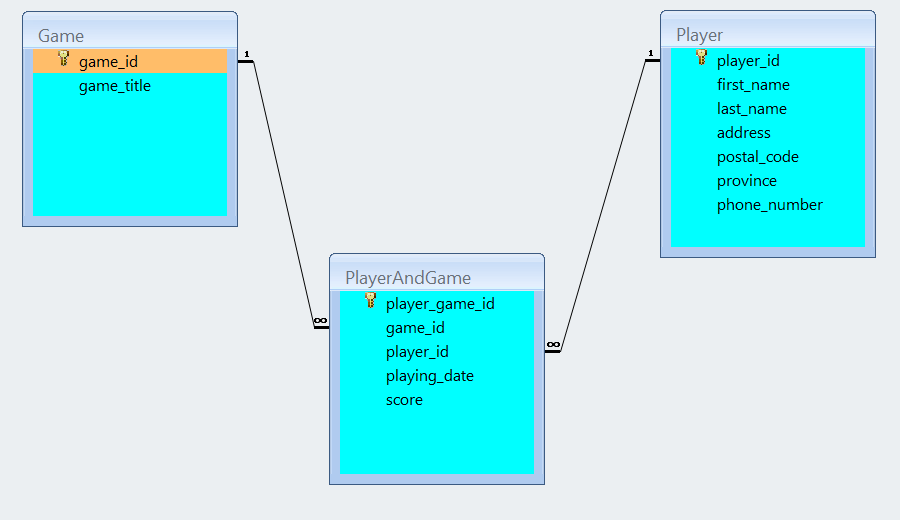
- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

### **Exercise 1:**

Develop a GUI Java application that will allow the players to submit information about themselves and the games that they are playing on-line. The information will be stored in a simple Oracle database. The database tables are shown in the following picture:



You can use SQL Developer to create your database in Oracle server.

You should populate the table *Game* with titles of games that you have "played" during this semester.

Your GUI should provide the necessary SWING or JavaFX components that will allow the user to enter and display the data. You will use JDBC to provide the following operations:

1. *Insert* game and player information into the database.
2. *Update* the existing player information.
3. *Display* reports with player and played games information. You may use a *JTable or other components to display the reports. Allow the user to select player\_id*.

Use prepared statements to implement all database operations.

(10 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of UI and event handling. | 35% |
| Correct implementation of JDBC | 50% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 10% |
| **Total** | 100% |

**The UI may look like the following:**

**[IN YOUR APPLICATION DO NOT USE THE SAME DATA. USE YOUR FULL NAME AS ONE OF THE PLAYERS. YOU WILL BE GRADED 0 IF YOU USE THE SAME DATA AS SHOW HERE]**

Graphical user interface

Description automatically generated

**Display All Players**

Table

Description automatically generated

**Update Operation:**

Graphical user interface, website

Description automatically generated

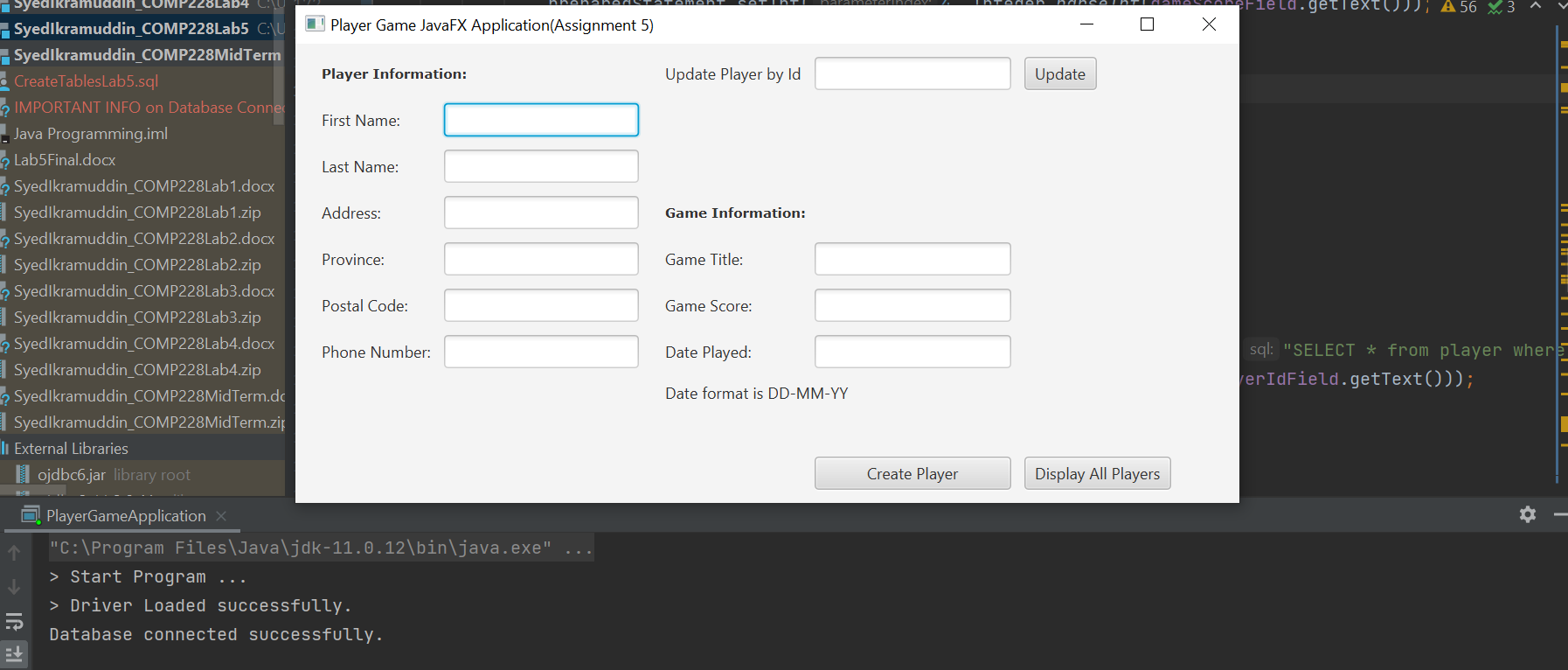
**After Updating Display All Players:**

Table

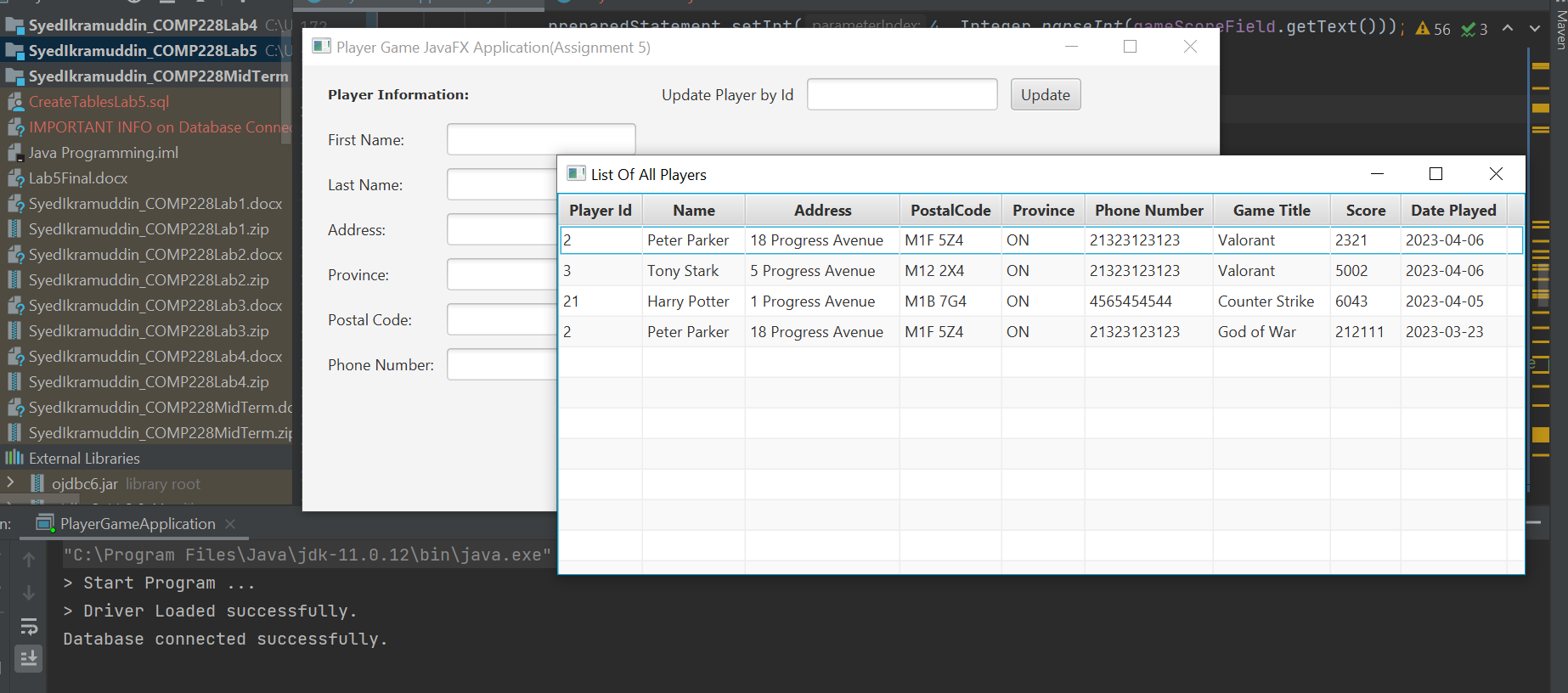
Description automatically generated

**Output:**

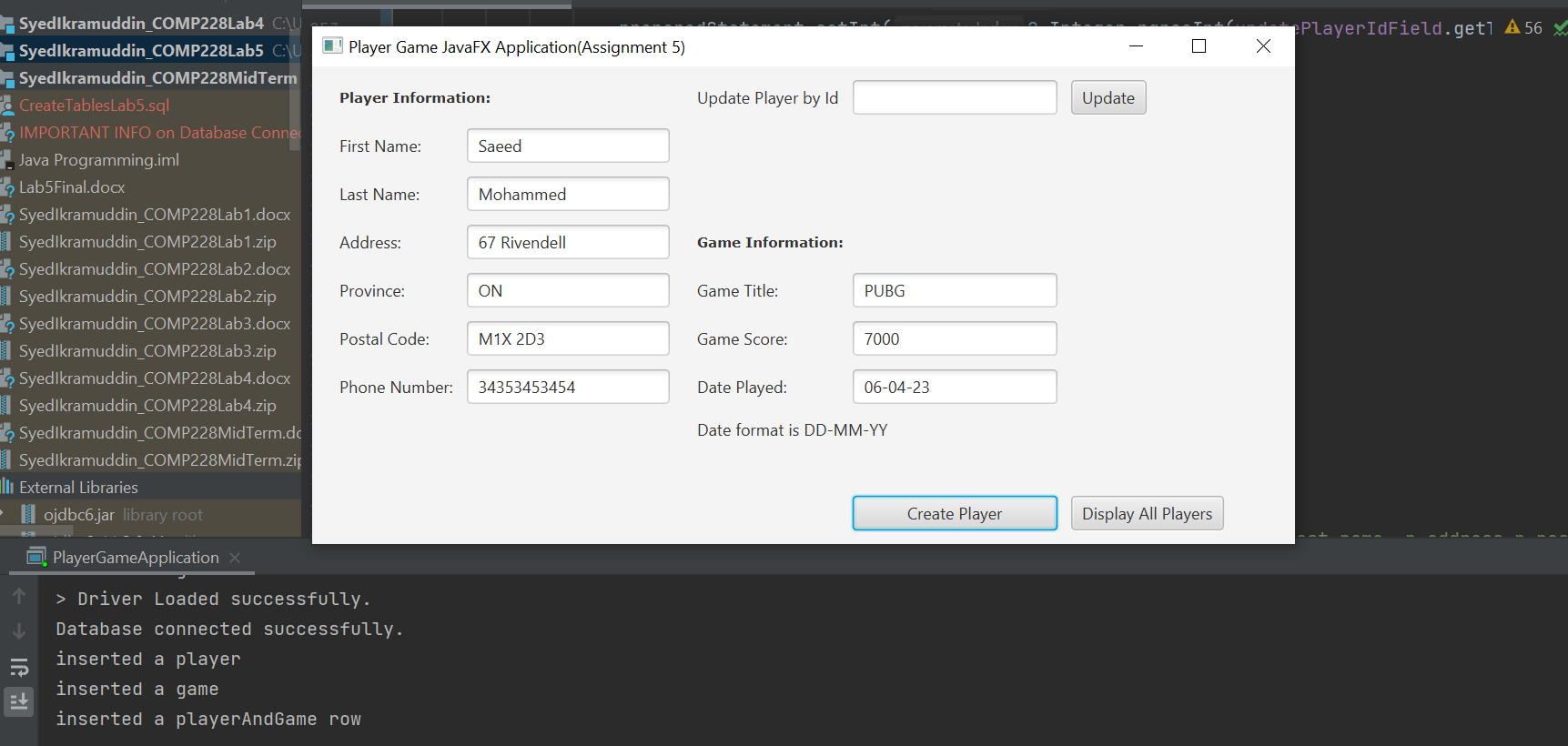
UI:

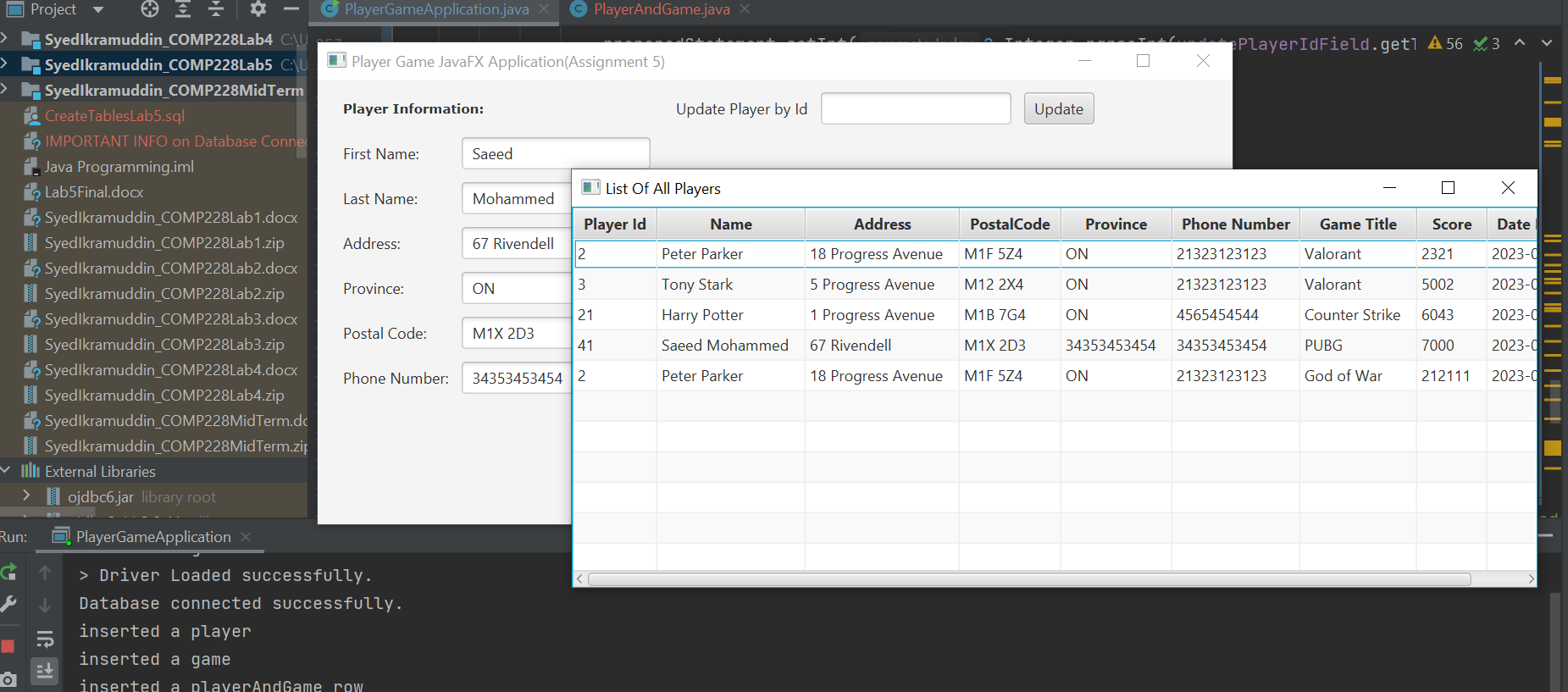


Display All players:

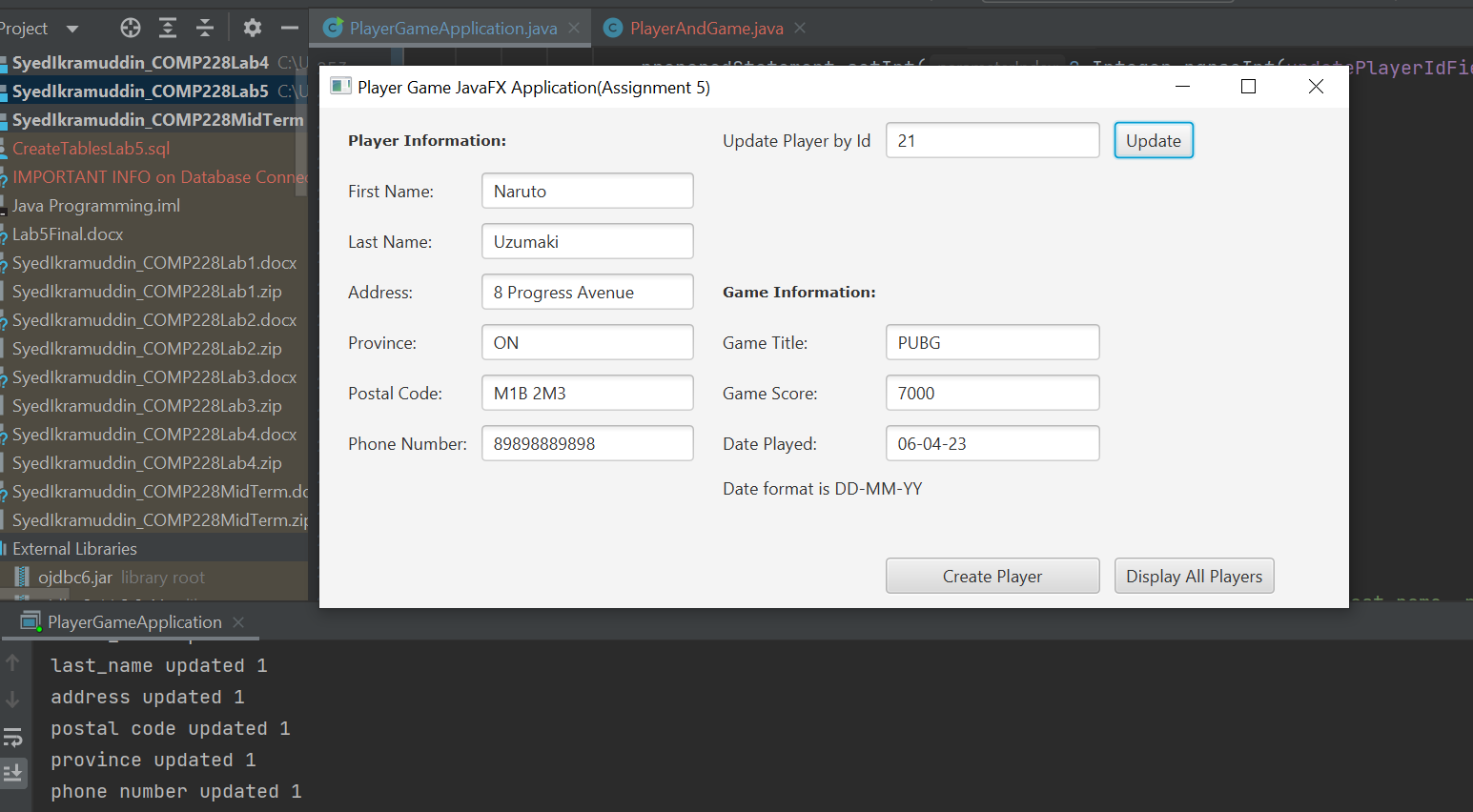


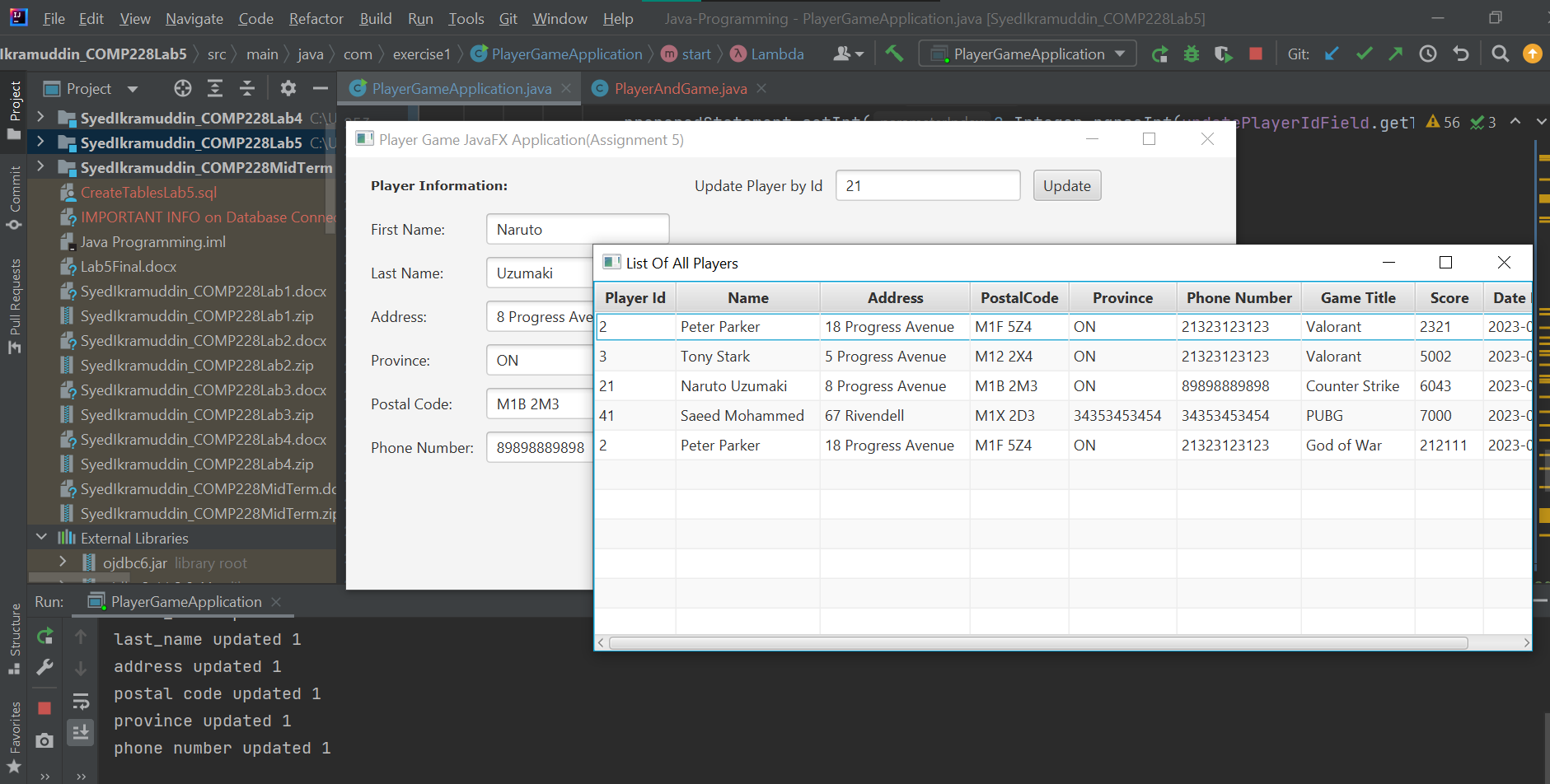
Create Player:





Update Player Id 21:





**Code:**

**PlayerAndGame.java**

package com.exercise1;

public class PlayerAndGame {

Integer id;

String name;

String address;

String postalCode;

String province;

String phoneNumber;

String gameTitle;

Integer score;

String datePlayed;

PlayerAndGame(){}

public PlayerAndGame(Integer id, String name, String address, String postalCode, String province, String phoneNumber, String gameTitle, Integer score, String datePlayed) {

this.id = id;

this.name = name;

this.address = address;

this.postalCode = postalCode;

this.province = province;

this.phoneNumber = phoneNumber;

this.gameTitle = gameTitle;

this.score = score;

this.datePlayed = datePlayed;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPostalCode() {

return postalCode;

}

public void setPostalCode(String postalCode) {

this.postalCode = postalCode;

}

public String getProvince() {

return province;

}

public void setProvince(String province) {

this.province = province;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

public String getGameTitle() {

return gameTitle;

}

public void setGameTitle(String gameTitle) {

this.gameTitle = gameTitle;

}

public Integer getScore() {

return score;

}

public void setScore(Integer score) {

this.score = score;

}

public String getDatePlayed() {

return datePlayed;

}

public void setDatePlayed(String datePlayed) {

this.datePlayed = datePlayed;

}

}

**PlayerGameApplication.java**

package com.exercise1;

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.StackPane;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

import oracle.jdbc.\*;

import oracle.sql.\*;

import java.sql.\*;

import java.time.LocalDate;

public class PlayerGameApplication extends Application {

Label playerInformation;

Label gameInformation;

TextField firstNameField;

TextField lastNameField;

TextField addressField;

TextField provinceField;

TextField postalCodeField;

TextField phoneNumberField;

TextField updatePlayerIdField;

TextField gameTitleField;

TextField gameScoreField;

TextField datePlayedField;

DatePicker;

Button updatePlayerButton;

Button createPlayerButton;

Button displayAllPlayerButton;

TableView<PlayerAndGame> playerAndGameTable;

ObservableList<PlayerAndGame> data;

public Connection getDatabaseConnection(){

Connection;

try {

System.out.println("> Start Program ...");

Class.forName("oracle.jdbc.driver.OracleDriver");

System.out.println("> Driver Loaded successfully.");

connection = DriverManager.getConnection("jdbc:oracle:thin:@199.212.26.208:1521:SQLD"," COMP228\_W23\_sy\_139", "password");

System.out.println("Database connected successfully.");

Statement = connection.createStatement();

return connection;

}

catch(Exception e)

{

e.printStackTrace();

}

return null;

}

@Override

public void start(Stage stage) throws Exception {

Connection = getDatabaseConnection();

if(connection == null){

System.out.println("Database not connected");

}

GridPane = new GridPane();

gridPane.setPadding(new Insets(20,20,20,20));

gridPane.setHgap(10);

gridPane.setVgap(10);

playerInformation= new Label("Player Information:");

gridPane.add(playerInformation,0,0,2,1);

playerInformation.setFont(Font.font("Verdana", FontWeight.BOLD,10));

gridPane.add(new Label("First Name:"),0,1);

firstNameField = new TextField();

gridPane.add(firstNameField,1,1);

gridPane.add(new Label("Last Name:"),0,2);

lastNameField = new TextField();

gridPane.add(lastNameField,1,2);

gridPane.add(new Label("Address:"),0,3);

addressField = new TextField();

gridPane.add(addressField,1,3);

gridPane.add(new Label("Province:"),0,4);

provinceField = new TextField();

gridPane.add(provinceField,1,4);

gridPane.add(new Label("Postal Code:"),0,5);

postalCodeField = new TextField();

gridPane.add(postalCodeField,1,5);

gridPane.add(new Label("Phone Number:"),0,6);

phoneNumberField = new TextField();

gridPane.add(phoneNumberField,1,6);

gridPane.add(new Label("Update Player by Id"),3,0);

updatePlayerIdField = new TextField();

gridPane.add(updatePlayerIdField,4,0);

updatePlayerButton = new Button("Update");

gridPane.add(updatePlayerButton,5,0);

gameInformation= new Label("Game Information:");

gridPane.add(gameInformation,3,3,2,1);

gameInformation.setFont(Font.font("Verdana", FontWeight.BOLD,10));

gridPane.add(new Label("Game Title:"),3,4);

gameTitleField = new TextField();

gridPane.add(gameTitleField,4,4);

gridPane.add(new Label("Game Score:"),3,5);

gameScoreField = new TextField();

gridPane.add(gameScoreField,4,5);

gridPane.add(new Label("Date Played:"),3,6);

gridPane.add(new Label("Date format is DD-MM-YY"),3,7,2,1);

datePlayedField = new TextField();

gridPane.add(datePlayedField,4,6);

createPlayerButton = new Button("Create Player");

displayAllPlayerButton = new Button("Display All Players");

createPlayerButton.setPrefWidth(150);

gridPane.add(createPlayerButton, 4,11);

gridPane.add(displayAllPlayerButton, 5,11);

createPlayerButton.setOnAction( actionEvent -> {

try {

PreparedStatement selectPlayerId = connection.prepareStatement("SELECT player\_id from Player where first\_name = ? and last\_name = ?");

selectPlayerId.setString(1,firstNameField.getText());

selectPlayerId.setString(2,lastNameField.getText());

ResultSet rs1 = selectPlayerId.executeQuery();

if(rs1.next() == false){

PreparedStatement = connection.prepareStatement("Insert INTO Player (first\_name,last\_name,address,postal\_code,province,phone\_number)" +

"VALUES(?,?,?,?,?,?)");

preparedStatement.setString(1,firstNameField.getText());

preparedStatement.setString(2,lastNameField.getText());

preparedStatement.setString(3,addressField.getText());

preparedStatement.setString(4,postalCodeField.getText());

preparedStatement.setString(5,phoneNumberField.getText());

preparedStatement.setString(6,phoneNumberField.getText());

int count = preparedStatement.executeUpdate();

if(count == 1){

System.out.println("inserted a player");

}

}

ResultSet rs3 = selectPlayerId.executeQuery();

PreparedStatement selectGameId = connection.prepareStatement("SELECT game\_id from GAME where game\_title = ?");

selectGameId.setString(1,gameTitleField.getText());

ResultSet rs2 = selectGameId.executeQuery();

if(rs2.next() == false){

PreparedStatement = connection.prepareStatement("Insert INTO Game (game\_title) VALUES (?)");

preparedStatement.setString(1,gameTitleField.getText());

int count = preparedStatement.executeUpdate();

if(count == 1){

System.out.println("inserted a game");

}

}

ResultSet rs4 = selectGameId.executeQuery();

PreparedStatement = connection.prepareStatement("Insert INTO PlayerAndGame (player\_id,game\_id,playing\_date,score)" +

"VALUES(?,?,?,?)");

if(rs3.next() == true){

preparedStatement.setInt(1,rs3.getInt("player\_id"));

}

if(rs4.next() == true){

preparedStatement.setInt(2,rs4.getInt("game\_id"));

}

preparedStatement.setString(3,datePlayedField.getText());

preparedStatement.setInt(4, Integer.parseInt(gameScoreField.getText()));

int count = preparedStatement.executeUpdate();

if(count == 1){

System.out.println("inserted a playerAndGame row");

}

} catch (SQLException e) {

e.printStackTrace();

}

});

updatePlayerButton.setOnAction(actionEvent -> {

try {

PreparedStatement = connection.prepareStatement("SELECT \* from player where player\_id = ?");

preparedStatement.setInt(1,Integer.parseInt(updatePlayerIdField.getText()));

ResultSet rs = preparedStatement.executeQuery();

if(rs.next() == false){

System.out.println("player Id not found");

return;

}

} catch (SQLException e) {

e.printStackTrace();

}

if(!firstNameField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set first\_name = ? where player\_id = ?");

preparedStatement.setString(1,firstNameField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("first\_name updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

if(!lastNameField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set last\_name = ? where player\_id = ?");

preparedStatement.setString(1,lastNameField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("last\_name updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

if(!addressField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set address = ? where player\_id = ?");

preparedStatement.setString(1,addressField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("address updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

if(!postalCodeField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set postal\_code = ? where player\_id = ?");

preparedStatement.setString(1,postalCodeField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("postal code updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

if(!provinceField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set province = ? where player\_id = ?");

preparedStatement.setString(1,provinceField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("province updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

if(!phoneNumberField.getText().isEmpty()){

try {

PreparedStatement = connection.prepareStatement("Update Player set phone\_number = ? where player\_id = ?");

preparedStatement.setString(1,phoneNumberField.getText());

preparedStatement.setInt(2,Integer.parseInt(updatePlayerIdField.getText()));

int count = preparedStatement.executeUpdate();

System.out.println("phone number updated "+count);

} catch (SQLException e) {

e.printStackTrace();

}

}

});

displayAllPlayerButton.setOnAction(actionEvent -> {

playerAndGameTable = new TableView<>();

Statement;

ResultSet rs;

data = FXCollections.observableArrayList();

try {

statement = connection.createStatement();

String selectListofPlayers = "select p.player\_id, p.first\_name,p.last\_name, p.address,p.postal\_code,p.province,p.phone\_number,\n" +

"g.game\_title, pg.playing\_date, pg.score\n" +

"from playerandgame pg join player p on pg.player\_id = p.player\_id\n" +

"join game g on pg.game\_id= g.game\_id";

rs = statement.executeQuery(selectListofPlayers);

while(rs.next()){

PlayerAndGame player = new PlayerAndGame(rs.getInt("player\_id"), rs.getString("first\_name")+" "+rs.getString("last\_name"),

rs.getString("address"), rs.getString("postal\_code"), rs.getString("province")

, rs.getString("phone\_number"), rs.getString("game\_title"), rs.getInt("score")

, rs.getDate("playing\_date").toString());

data.add(player);

}

} catch (SQLException e) {

e.printStackTrace();

}

TableColumn<PlayerAndGame, Integer> idCol = new TableColumn("Player Id");

idCol.setCellValueFactory( new PropertyValueFactory<>("id"));

TableColumn<PlayerAndGame, String> nameCol = new TableColumn("Name");

nameCol.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<PlayerAndGame, String> addressCol = new TableColumn("Address");

addressCol.setCellValueFactory(new PropertyValueFactory<>("address"));

TableColumn<PlayerAndGame, String> postalCodeCol = new TableColumn("PostalCode");

postalCodeCol.setCellValueFactory(new PropertyValueFactory<>("postalCode"));

TableColumn<PlayerAndGame, String> provinceCol = new TableColumn("Province");

provinceCol.setCellValueFactory(new PropertyValueFactory<>("province"));

TableColumn<PlayerAndGame, String> phoneNumberCol = new TableColumn("Phone Number");

phoneNumberCol.setCellValueFactory(new PropertyValueFactory<>("phoneNumber"));

TableColumn<PlayerAndGame, String> gameTitleCol = new TableColumn<>("Game Title");

gameTitleCol.setCellValueFactory(new PropertyValueFactory<>("gameTitle"));

TableColumn<PlayerAndGame, Integer> scoreCol = new TableColumn<>("Score");

scoreCol.setCellValueFactory(new PropertyValueFactory<>("score"));

TableColumn<PlayerAndGame, String> datePlayedCol = new TableColumn<>("Date Played");

datePlayedCol.setCellValueFactory(new PropertyValueFactory<>("datePlayed"));

playerAndGameTable.setItems(data);

playerAndGameTable.getColumns().addAll(idCol,nameCol,addressCol,postalCodeCol,provinceCol,phoneNumberCol,gameTitleCol,scoreCol,datePlayedCol);

StackPane secondaryLayout = new StackPane();

secondaryLayout.getChildren().add(playerAndGameTable);

Scene secondScene = new Scene(secondaryLayout, 760, 300);

// New window (Stage)

Stage newWindow = new Stage();

newWindow.setTitle("List Of All Players");

newWindow.setScene(secondScene);

// Set position of second window, related to primary window.

newWindow.setX(stage.getX() + 200);

newWindow.setY(stage.getY() + 100);

newWindow.show();

});

StackPane root=new StackPane();

root.getChildren().add(gridPane);

Scene scene=new Scene(root,720,350);

stage.setTitle("Player Game JavaFX Application(Assignment 5)");

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch(args);

}

}

SQL:

Create Tables

DROP TABLE GAME;

DROP TABLE PLAYER;

DROP TABLE PLAYERANDGAME;

--1

CREATE TABLE GAME(

game\_id int Generated Always as Identity,

game\_title varchar(30),

CONSTRAINT PK\_Game PRIMARY KEY (game\_id)

);

--2

CREATE TABLE PLAYER(

player\_id int Generated Always as Identity,

first\_name varchar(20),

last\_name varchar(20),

address varchar(20),

postal\_code varchar(20),

province varchar(15),

phone\_number varchar(20),

CONSTRAINT PK\_Player PRIMARY KEY (player\_id)

);

--3

CREATE TABLE PlayerAndGame(

player\_game\_id int Generated Always as Identity,

player\_id int,

game\_id int,

playing\_date date,

score int,

CONSTRAINT PK\_player\_game PRIMARY KEY (player\_game\_id),

CONSTRAINT FK\_Game FOREIGN KEY (game\_id)

REFERENCES GAME(game\_id),

CONSTRAINT FK\_Player FOREIGN KEY (player\_id)

REFERENCES player(player\_id)

);

/\*

--Select queries

select \* from player;

select \* from game;

select \* from PlayerAndGame;

select pg.player\_game\_id, p.first\_name,p.last\_name, p.address,p.postal\_code,p.province,p.phone\_number,

g.game\_title, pg.playing\_date, pg.score

from playerandgame pg join player p on pg.player\_id = p.player\_id

join game g on pg.game\_id= g.game\_id;

\*/

COMMIT;