Lab Report

Assignment 4
Data Base

Software Design Lab

Fall 2020

By: Tanvir Youhana

Table of Contents

Objective:

Developed an application that created a database which consisted of:

- **Students** (empID, firstName, lastName, email, sex)
- **Courses** (courseID, courseTitle, department)
- **Classes** (courseID, studentID, sectionNo, year, semester, grade)

Then obtain the grades from these tables and create a pie chart using JavaFX to display how many of each grade was given out to the students in the class.

(The RDBMS that I used was MySQL and used the MySQL Workbench to verify connection established as well as tables integrity)

Creating the database

For us to create the databases we first need to establish a connection to the database that we created on the MySQL Workbench. We connect to the database using the JDBC URL and changing the path directory to the database of our desire. We will also be needing the user id and the password to be able to access the database and make edits to it.

Once the connection is made, we can start making the tables. To make the tables we can a prepared statement that we code in java and this prepared statement is then pushing to the database where it takes the command and creates the tables of our desire. For this database we created three tables. These three tables are Students, Courses and Classes. The create these tables we use the command "CREATE TABLE IF NOT EXISTS (table-name)". We use the CREATE TABLE command to name the table columns as well as state their data types.

To insert into the database we use the SQL command "INSERT INTO (Table_name)(column_name1,column_name2,column_name3…)VALUES(name1,name2…)."To show the tables with all the data we can type the command "SELECT * FROM (Table_name)." Finally, to get number of students with their corresponding grades we use the command "SELECT (column1), column2 FROM (Table_name) WHERE (column3)."

Code developed from MySQL:

Creating the Student Table

```
CREATE TABLE IF NOT EXISTS Students (empl_ID INT NOT NULL,
PRIMARY KEY (empl_ID),
firstName VARCHAR(255),
lastName VARCHAR(255),
Email VARCHAR(255),
sex CHAR(1),
CHECK (sex = 'F' OR sex = 'M'OR sex= ‘U’))
```

Creating the Courses table

```
CREATE TABLE IF NOT EXISTS Courses (Course_ID INT NOT NULL,
PRIMARY KEY (Course_ID),
CourseTitle VARCHAR(255),
Department VARCHAR(255))
```

Creating the Classes table

```
CREATE TABLE IF NOT EXISTS Classes (Course_ID INT NOT NULL,
student_ID INT UNSIGNED NOT NULL,
section_No VARCHAR(255),
year INT UNSIGNED,
grade CHAR(1),
PRIMARY KEY (course_ID, student_ID, section_No),
CHECK (grade= 'A' OR grade = 'B' OR grade = 'C' OR grade = 'D' OR grade = 'F' OR grade
= 'W'))
```

Inserting information into Student Table

```
INSERT INTO students(empl_ID, firstName,lastName,Email,sex)
VALUE(‘empl_ID’, ‘firstName’, ‘lastName’, ‘Email’, ‘sex’)
```

Inserting information into courses Table

```
INSERT INTO courses(Course_ID,CourseTitle,Department)
VALUE(‘Course_ID’, ‘CourseTitle’, ‘Department’)
```

Insert information into Classes Table

```
INSERT INTO classes(Course_ID, student_ID,section_No,year,semester,grade)
VALUE(‘Course_ID’, ‘student_ID’, ‘section_No’, ‘year’, ‘semester’, ‘grade’)
```

Show Student Table

```
SELECT * FROM students
```

Show Courses Table

```
SELECT * FROM course
```

Show Classes Table

```
SELECT * FROM classes
```

Update Student Table

```
UPDATE students
SET firstName=’firstName’, lastName= ‘lastName’
WHERE student_ID= ‘student_ID’
```

Update Courses Table

```
UPDATE courses
SET courseTitle=’courseTitle’, department= ‘department’
WHERE course_ID= ‘course_ID’
```

Update Classes Table

```
UPDATE students
SET section_No=’section_No’, year= ‘year’, ‘semester= ‘semester’, grade= ‘grade’
WHERE course_ID= ‘course_ID’, student_ID= ‘student_ID’
```

Drop the Students Table

```
DROP TABLE Students
```

Drop the Course Table

```
DROP TABLE Courses
```

Drop the Classes Table

```
DROP TABLE Classes
```

Java Code Developed

The java code that was developed comprises of the classes Course.Java, Student.Java, Classes.Java, MyPieChart.Java, Main.Java, and all the other classes from previous projects

A.) Course. Java

This class had the Course constructor that was used to insert into the table

```
package sample;

public class Courses {
    int course_ID;
    String courseTitle, department;

    public Courses(int course_ID, String Coursetitle, String dept){
        this.course_ID = course_ID;
        this.courseTitle = Coursetitle;
        this.department = dept;
    }
}
```

B.) Students.Java

This class had the Student constructor that was used to take in the first, last, email, and sex. Here I decided to generate a random student ID and used the method generateID to get a random value that is 4 digits long. We also have "gets" that returned specific detail of each student when needed.

```
package sample;

import java.util.Random;

public class Student {
    String firstName, lastName, sex,email;
    int studentID;

    public Student(String first, String last, String E_mail,String sex){
        this.firstName = first;
        this.lastName = last;
        this.sex = sex;
        this.studentID = generateID();
        this.email=E_mail;
    }

    public static int generateID(){
        Random random = new Random();
         return Integer.parseInt(String.format("%04d", 1000+random.nextInt(999)));

    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getSex() {
        return sex;
    }

    public String getEmail() { return email;}

    public int getStudentID() {
        return studentID;
    }
}
```

C.) Classes.Java

The Classes.Java class has a constructor for inserting the information for classes table as well as “getters” to get the information desired.

```
D.) package sample;
   public class Classes {
       String semester, section, grade, lastName;
       int course_ID, student_ID, year;

       public Classes(int course_ID, int student_ID, String section, int year,
   String semester, String grade){
           this.course_ID = course_ID;
           this.student_ID = student_ID;
           this.section = section;
           this.year = year;
           this.semester = semester;
           this.grade = grade;
       }

       public void setLastName(String lastName){
           this.lastName = lastName;
       }

       public String getLastName(){
           return lastName;
       }

       public String getSemester() {
           return semester;
       }

       public String getSection() {
           return section;
       }

       public String getgrade() {
           return grade;
       }

       public int getCourse_ID() {
           return course_ID;
       }

       public int getStudent_ID() {
           return student_ID;
       }

       public int getYear() {
           return year;
       }
   }
```

MyPieChart.Java

The MyPieChart.Java class was reformatted for this assignment. Took the hash map of all the grades with its respective frequency and converted it into different array lists that was used to draw the Pie Chart. The legend is also created based off the array lists. Used the enum MyColor.Java from previous projects to generate random colors for each section of the pie.

```
package sample;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;
import java.util.*;


public class MyPieChart {

    double other, total;
    HashMap<String, Double> pieMap = new HashMap<String, Double>();
    ArrayList<Color> colors = new ArrayList<>();
    ArrayList<String> labels = new ArrayList<>();
    ArrayList<Double> probabilites = new ArrayList<>();


    public MyPieChart(HashMap<String, Integer> hashMap, int total) {
        this.total = total;
```

```java
        // Populate pie chart Hashmap with probabilities
        hashMap.forEach((key,value) -> {
            pieMap.put(key, (Double.valueOf(value)/total));
        });

        // Create the labels and add probability for chart
        pieMap.forEach((key,value)->{
            String students = ( (int)( value * total ) ) + "";
            double number = value;
            number = Math.round(number * 100);
            number = number/100;
            probabilites.add(value);
            labels.add(key +" ["+students+"] : "+ number);
        });

    }



    // Create legend with equal spaces with color and label
    public void drawLegend( double x, double y, GraphicsContext gc){
        gc.strokeText("LEGEND", x, y);

        for(int i=0; i<probabilites.size(); i++) {
            gc.setStroke(Color.BLACK);
            gc.fillText( labels.get(i), x + 30, y + 30);
            gc.setFill(colors.get(i));
            gc.fillRect(x, y + 15, 20, 20);
            y+=30;
        }


    }

    public void drawPieChart(double totalWidth, double totalHeight, GraphicsContext
graphicsContext) {

        // Position away from center and spaced next to legend
        double x = totalWidth / 2D - 20;
        double y = totalHeight / 2D - 90 ;
        // Starting and ending for each arc
        double start = 0;
        double end;


        Color color;

        // Start where the previous arc ends, and end at an angle calculated by
probability
        end = (probabilites.get(0))*360.00;
        for(int i=0; i<probabilites.size(); i++) {
            color = MyColor.getRandomColor();
            graphicsContext.setFill(color);
            colors.add(color);
            graphicsContext.fillArc(x, y,300 ,300,start,end,ArcType.ROUND);
            if(i == probabilites.size()-1){
                break;
            }
            start = start+end; // Get location for previous arc ended
            end = (probabilites.get(i+1))*360.00; // end at the next angle from
probability
        }

        // After pie chart complete, draw legend
        drawLegend( 50, 100, graphicsContext);
    }
}
```

Main.Java

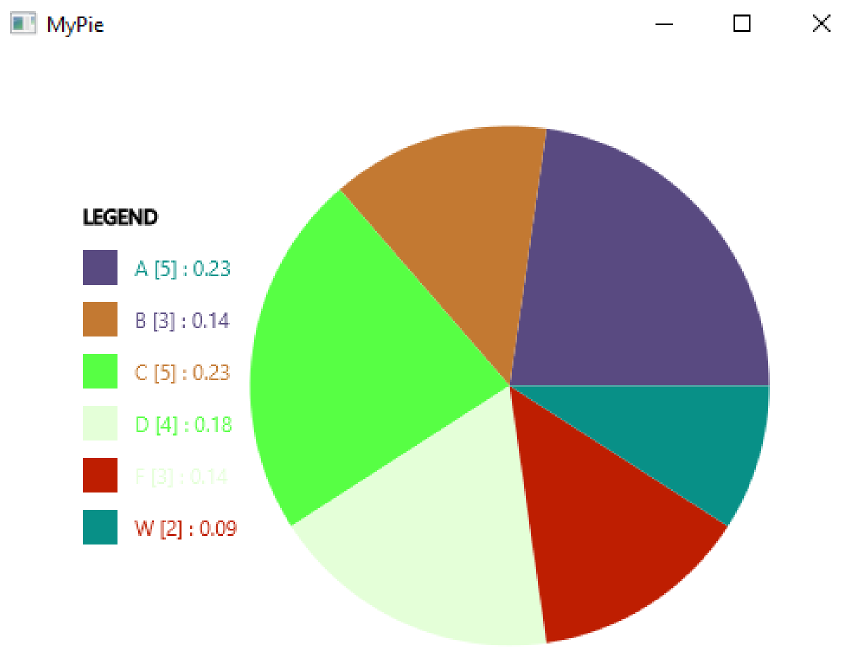
In the Main.Java is where we established the connection with the database as well as have the different methods that contained prepared statements that were passed to the SQL database to create, populate and update the database as we please. All the prepared statements for the SQL database are mentioned above using the insert methods we were able to populate the tables with student information from the Fall 2020 Algorithms class. We also use array lists to store all the students and classes. The main also servs as the place where we create the canvas and used it to draw the pie chart with a legend showing how many students got each grade. We also used the main to output all the data tables that were created on the databases.

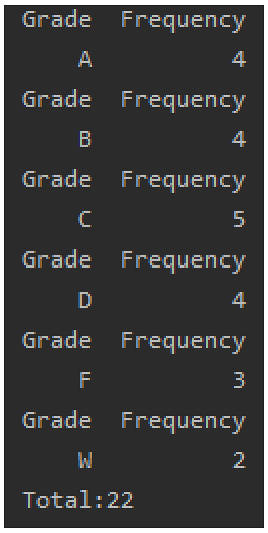
```
package sample;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.stage.Stage;

import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class Main extends Application {
    static int total = 0;

    @Override
    public void start(Stage primaryStage) throws Exception {
        //Creating the canvas
        Group root = new Group();
        Scene shape = new Scene(root);
        primaryStage.setTitle("MyPie");
        int w = 500, h = 400;
        Canvas canvas = new Canvas(w, h);
        GraphicsContext Graphics = canvas.getGraphicsContext2D();
        root.getChildren().add(canvas);
        primaryStage.setScene(shape);
        MyPoint p = new MyPoint(w / 4, h / 7);

        Connection conn = null;
        try {
// Loads the class object for the mysql driver into the DriverManager.
            Class.forName("com.mysql.cj.jdbc.Driver");
            //  Attempt to establish a connection to the specified database via the
            //  DriverManager
            conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/java", "root",
"Stillheart11432!");
            // Check the connection
            if (conn != null) {
                System.out.println("We have connected to our database!");
                //Drop table
                dropTables(conn);
                //Create the table and show the table structure
                PreparedStatement table_1 = conn.prepareStatement("CREATE TABLE IF NOT EXISTS
Students" +
                        "(empl_ID INT NOT NULL, " +
                        " PRIMARY KEY (empl_ID)," +
                        "firstName VARCHAR(255)," +
                        "lastName VARCHAR(255)," +
                        "Email VARCHAR(255)," +
                        "sex CHAR(1)," +
                        "CHECK (sex ='F'OR sex='M'OR sex='U'))");
                table_1.execute();

                PreparedStatement table_2 = conn.prepareStatement("CREATE TABLE IF NOT EXISTS
Courses" +
                        "(Course_ID INT NOT NULL," +
                        "PRIMARY KEY (Course_ID)," +
                        "CourseTitle VARCHAR(255)," +
                        "Department VARCHAR(255))");
                table_2.execute();
                PreparedStatement table_3 = conn.prepareStatement("CREATE TABLE IF NOT EXISTS
Classes" +
                        " (Course_ID INT NOT NULL, " +
                        " student_ID INT UNSIGNED NOT NULL, " +
                        " section_No VARCHAR(255), " +
                        " year INT UNSIGNED, " +
                        " semester VARCHAR(255), " +
                        " grade CHAR(1), " +
                        " PRIMARY KEY (course_ID, student_ID, section_No)," +
                        " CHECK (grade= 'A' OR grade = 'B' OR grade = 'C' OR grade = 'D' OR
grade = 'F' OR grade = 'W'))");
                table_3.execute();
                //create a array list to populate the database with students information
                ArrayList<Student> Student = new ArrayList<>();
                ArrayList<Classes> Classes = new ArrayList<>();
                Student.add(new Student("Tanvir", "Youhana", "tyouhan000@citymail.cuny.edu",
"M"));
                Student.add(new Student("Bob", "Ross", "Bobrossn045@citymail.cuny.edu", "M"));
                Student.add(new Student("Nicolas", "Cage", "Ncage054@citymail.cuny.edu",
"U"));
                Student.add(new Student("Donald", "Trump", "Dtrump056@citymail.cuny.edu",
"U"));
                Student.add(new Student("Robert", "Harris", "Rharrisn002@citymail.cuny.edu",
"M"));
                Student.add(new Student("Barrack", "Obama", "Bobama023@citymail.cuny.edu",
"M"));
                Student.add(new Student("Daniel", "Yoon", "Dyoon045@citymail.cuny.edu", "U"));
                Student.add(new Student("Gabyr", "Wurt", "Gwurt078@citymail.cuny.edu", "F"));
                Student.add(new Student("Cindy", "Ortiz", "Cortiz0076@citymail.cuny.edu",
"F"));
```

```
                Student.add(new Student("Bernard", "Ko", "Bko001@citymail.cuny.edu", "M"));
                Student.add(new Student("Angelica", "Ortiz", "Aortiz094@citymail.cuny.edu",
"F"));
                Student.add(new Student("Ronald", "Smith", "Rsmith345@citymail.cuny.edu",
"M"));
                Student.add(new Student("Jason", "Williams", "Jwilliams343@citymail.cuny.edu",
"M"));
                Student.add(new Student("Brandon", "King", "Bking346@citymail.cuny.edu",
"U"));
                Student.add(new Student("Lisa", "Green", "Lgreen294@citymail.cuny.edu", "F"));
                Student.add(new Student("Richard", "Reed", "Rreed786@citymail.cuny.edu",
"M"));
                Student.add(new Student("Brooke", "Long", "Blong123@citymail.cuny.edu", "F"));
                Student.add(new Student("John", "Fisher", "Jfisher412@citymail.cuny.edu",
"M"));
                Student.add(new Student("Angel", "Ortiz", "Aortiz094@citymail.cuny.edu",
"U"));
                Student.add(new Student("Mary", "Ann", "Mann534@citymail.cuny.edu", "F"));
                Student.add(new Student("Melanie", "Beni", "Mbeni345@citymail.cuny.edu",
"F"));
                Student.add(new Student("Justin", "Rice", "Jrice121@citymail.cuny.edu", "M"));


                // adds Algo 220 into course table
                Courses Algo = new Courses(22000, "Algorithms", "Computer Science");
                createNewCourse(conn, Algo);
                //Test to update courses
                updateCourse(conn, 22000, "Linear",  "Electrical Engineering");


                Classes.add(new Classes(Algo.course_ID, Student.get(0).studentID, "T", 2020,
"Fall", "A"));
                Classes.add(new Classes(Algo.course_ID, Student.get(1).studentID, "T", 2020,
"Fall", "B"));
                Classes.add(new Classes(Algo.course_ID, Student.get(2).studentID, "T", 2020,
"Fall", "C"));
                Classes.add(new Classes(Algo.course_ID, Student.get(3).studentID, "T", 2020,
"Fall", "F"));
                Classes.add(new Classes(Algo.course_ID, Student.get(4).studentID, "T", 2020,
"Fall", "F"));
                Classes.add(new Classes(Algo.course_ID, Student.get(5).studentID, "T", 2020,
"Fall", "W"));
                Classes.add(new Classes(Algo.course_ID, Student.get(6).studentID, "T", 2020,
"Fall", "A"));
                Classes.add(new Classes(Algo.course_ID, Student.get(7).studentID, "T", 2020,
"Fall", "C"));
                Classes.add(new Classes(Algo.course_ID, Student.get(8).studentID, "T", 2020,
"Fall", "A"));
                Classes.add(new Classes(Algo.course_ID, Student.get(9).studentID, "T", 2020,
"Fall", "D"));
                Classes.add(new Classes(Algo.course_ID, Student.get(10).studentID, "T", 2020,
"Fall", "D"));
                Classes.add(new Classes(Algo.course_ID, Student.get(11).studentID, "T", 2020,
"Fall", "B"));
                Classes.add(new Classes(Algo.course_ID, Student.get(12).studentID, "T", 2020,
"Fall", "C"));
                Classes.add(new Classes(Algo.course_ID, Student.get(13).studentID, "T", 2020,
"Fall", "A"));
                Classes.add(new Classes(Algo.course_ID, Student.get(14).studentID, "T", 2020,
"Fall", "C"));
                Classes.add(new Classes(Algo.course_ID, Student.get(15).studentID, "T", 2020,
"Fall", "F"));
                Classes.add(new Classes(Algo.course_ID, Student.get(16).studentID, "T", 2020,
"Fall", "W"));
                Classes.add(new Classes(Algo.course_ID, Student.get(17).studentID, "T", 2020,
"Fall", "D"));
                Classes.add(new Classes(Algo.course_ID, Student.get(18).studentID, "T", 2020,
"Fall", "B"));
                Classes.add(new Classes(Algo.course_ID, Student.get(19).studentID, "T", 2020,
"Fall", "D"));
                Classes.add(new Classes(Algo.course_ID, Student.get(20).studentID, "T", 2020,
"Fall", "C"));
                Classes.add(new Classes(Algo.course_ID, Student.get(21).studentID, "T", 2020,
"Fall", "A"));


                //take all students from arraylist and add it to the database
                for (int i = 0; i < Student.size(); i++) {
                    createNewStudent(conn, Student.get(i));
                    insertIntoClasses(conn, Student.get(i), Classes.get(i));

                }


                displaySemestergrade(conn, "Fall");

                // Create pie chart
                MyPieChart myPieChart = new MyPieChart(displaySemestergrade(conn, "Fall"),
total);
                //draws pie chart
                myPieChart.drawPieChart(w / 1.5, h / 1.5, Graphics);
                primaryStage.show();

                //Hashmap that keeps track of the grades and frequency
                HashMap<String, Integer> hashMap = displaySemestergrade(conn, "Fall");
                // prints out how many of each character was counted
                for (Map.Entry<String, Integer> entry : hashMap.entrySet()) {
                    System.out.printf("%5s %10s", "Grade", "Frequency");
                    System.out.println();
                    System.out.format("%5s %10s", entry.getKey(), entry.getValue());
                    System.out.println();
                }
                System.out.println("Total:" + total);


                //Print out the table
                System.out.println("Students Table");
                readStudentsList(conn);
                System.out.println();
                System.out.println("Course Table");
                readCoursesList(conn);
                System.out.println();
                System.out.println("Classes Table");
                readClassesList(conn);

                //close the connection to the database
                conn.close();
            }
        } catch (SQLException ex) {
            System.out.println("SQLException: " + ex.getMessage());
            ex.printStackTrace();
        } catch (Exception ex) {
            System.out.println("Exception: " + ex.getMessage());
            ex.printStackTrace();
        }

    }


    public static void main(String[] args) {
        launch(args);
    }


    // Returns a HashMap with frequency for each grade
    public static HashMap<String, Integer> displaySemestergrade(Connection connection, String 
semester) {
        String sql = "SELECT student_ID, grade FROM classes WHERE semester = \"" + semester + 
"\"";
        HashMap<String, Integer> hashMap = new HashMap<>();
        int mtotal = 0;

        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                // How many times it occurs and corresponding letter
                String grade = rs.getString("grade");
                String studentID = rs.getString("student_ID");

                if (hashMap.containsKey(grade)) {
                    hashMap.put(grade, hashMap.get(grade) + 1); // Increment
                } else {
                    hashMap.put(grade, 1); // Unique
                }
                mtotal++;
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
        total = mtotal;
        return hashMap;
    }


    // Pass in student and class objects to add student to a class
    public static void insertIntoClasses(Connection connection, Student student, Classes 
classes) {
        ResultSet rs = null;
        String sql = "INSERT INTO classes(Course_ID, student_ID, section_No, year, 
semester,grade) " + "VALUES(?,?,?,?,?,?)";

        try {
            // Creates the student in the database
            createNewStudent(connection, student);
            // Enters the student into the class
            PreparedStatement pstmt = connection.prepareStatement(sql, 
Statement.RETURN_GENERATED_KEYS);
            pstmt.setInt(1, classes.course_ID);
            pstmt.setInt(2, classes.student_ID);
            pstmt.setString(3, classes.section);
            pstmt.setInt(4, classes.year);
            pstmt.setString(5, classes.semester);
            pstmt.setString(6, classes.grade);

            int rowAffected = pstmt.executeUpdate();
            if (rowAffected == 1) {
                rs = pstmt.getGeneratedKeys();
                if (rs.next())
                    System.out.println("Successfully added " + rs.getInt(1));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        } finally {
            try {
                if (rs != null) rs.close();
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }


    public static void createNewStudent(Connection connection, Student student) {
        ResultSet rs = null;
        String sql = "INSERT INTO students(empl_ID,firstName,lastName,Email,sex) " +
"VALUES(?,?,?,?,?)";

        try {
            PreparedStatement pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setInt(1, student.studentID);
            pstmt.setString(2, student.firstName);
            pstmt.setString(3, student.lastName);
            pstmt.setString(4, student.email);
            pstmt.setString(5, student.sex);


            int rowAffected = pstmt.executeUpdate();
            if (rowAffected == 1) {
                rs = pstmt.getGeneratedKeys();
                if (rs.next())
                    System.out.println("Successfully added student " + rs.getInt(1));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        } finally {
            try {
                if (rs != null) rs.close();
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }


    public static void createNewCourse(Connection connection, Courses course) {
        ResultSet rs = null;
        String sql = "INSERT INTO courses(Course_ID,CourseTitle,Department) " +
"VALUES(?,?,?)";

        // Using prepared statements before inserting
        try {
            PreparedStatement pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setInt(1, course.course_ID);
            pstmt.setString(2, course.courseTitle);
            pstmt.setString(3, course.department);

            int rowAffected = pstmt.executeUpdate();
            if (rowAffected == 1) {
                rs = pstmt.getGeneratedKeys();
                if (rs.next())
                    System.out.println("Successfully added course " + rs.getInt(1));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        } finally {
            try {
                if (rs != null) rs.close();
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }

    //NOT Neccesary
    // Print all the students
    public static String getStudentName(Connection connection, int studentID) {
        String sql = "SELECT * FROM students WHERE student_ID = " + studentID;
        String name = "";
        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                name = rs.getString("lastName");
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }

        return name;
    }


    // Print all the students
    public static void readStudentsList(Connection connection) {
        String sql = "SELECT * FROM students";

        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                System.out.println("--------------------------------|");
                System.out.println(rs.getString("empl_ID") + "\t" + rs.getString("lastName") +
                        "\t" + rs.getString("firstName") + "\t" + rs.getString("sex"));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }


    // Print all the classes
    public static void readClassesList(Connection connection) {
        String sql = "SELECT * FROM classes";
        ;
        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                System.out.println(rs.getInt("Course_ID") + "\t" + rs.getInt("student_ID")
                        + "\t" + rs.getString("section_No") + "\t" + rs.getInt("year") + "\t"
+ rs.getString("semester") + "\t" + rs.getString("grade"));

            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }


    // Print all the courses
    public static void readCoursesList(Connection connection) {
        String sql = "SELECT * FROM courses";

        try {
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                System.out.println(rs.getString("course_ID") + "\t" +
rs.getString("courseTitle")
                        + "\t" + rs.getString("department"));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }


    // Allows to update students name and sex. Student id should stay the same
    public static void updateStudent(Connection connection, int studentID, String firstName,
String lastName) {
        String sqlUpdate = "UPDATE students " + "SET firstName = ?, lastName = ? " + "WHERE
student_ID = ?";

        try {
            PreparedStatement pstmt = connection.prepareStatement(sqlUpdate);
            pstmt.setString(1, firstName);
            pstmt.setString(2, lastName);
            pstmt.setInt(3, studentID);

            int rowAffected = pstmt.executeUpdate();
            System.out.println(String.format("Row affected %d", rowAffected));
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
    //update the courses
    public static void updateCourse(Connection connection, int course_ID, String courseTitle,
String department) {
        String sqlUpdate = "UPDATE courses " + "SET courseTitle = ?, department = ? " + "WHERE
course_ID = ?";

        try {
            PreparedStatement pstmt = connection.prepareStatement(sqlUpdate);
            pstmt.setString(1, courseTitle);
            pstmt.setString(2, department);
            pstmt.setInt(3, course_ID);

            int rowAffected = pstmt.executeUpdate();
            System.out.println(String.format("Row affected %d", rowAffected));
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
    public static void updateClasses(Connection connection, int course_ID, int student_ID, int
section_No,int year, String semester, String grade) {
        String sqlUpdate = "UPDATE classes " + "SET section_No = ?, year = ?, semester=?,
grade= ?" + "WHERE course_ID = ? , student_ID=?";

        try {
            PreparedStatement pstmt = connection.prepareStatement(sqlUpdate);
            pstmt.setInt(1, section_No);
            pstmt.setInt(2, year);
            pstmt.setString(3, semester);
            pstmt.setString(4, grade);

            int rowAffected = pstmt.executeUpdate();
            System.out.println(String.format("Row affected %d", rowAffected));
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }


    public static void dropTables(Connection conn) {
        try {
            PreparedStatement drop1 = conn.prepareStatement("DROP TABLE Students");
            drop1.execute();

            PreparedStatement drop2 = conn.prepareStatement("DROP TABLE Courses");
            drop2.execute();

            PreparedStatement drop3 = conn.prepareStatement("DROP TABLE Classes");
            drop3.execute();

        } catch (SQLException ex) {
            System.out.println("SQLException: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

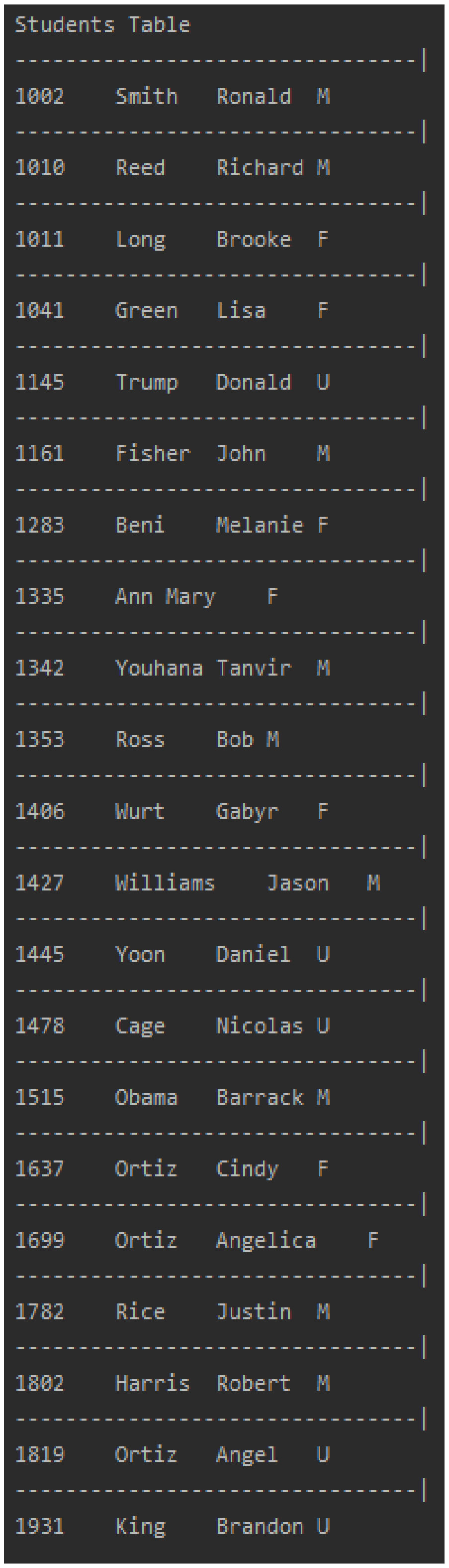
Outputs

Pie Chart

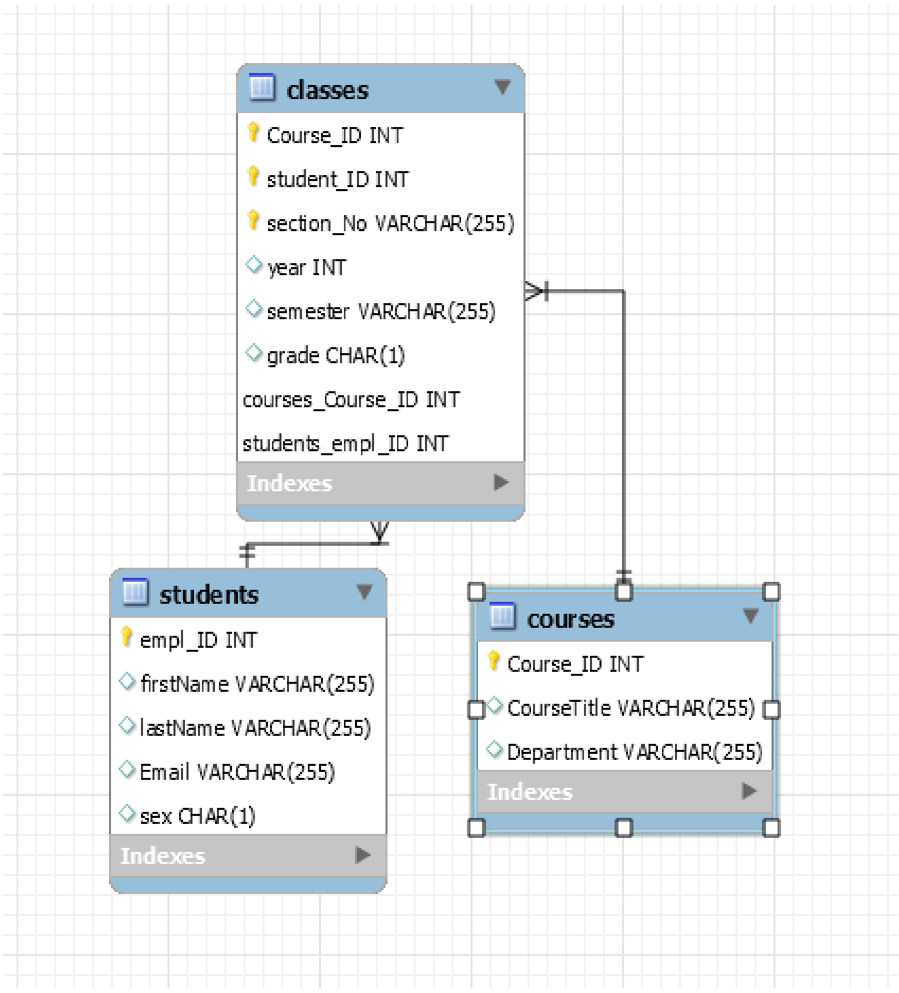


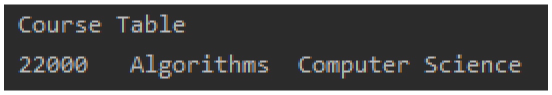


```
Grade  Frequency
    A          4
Grade  Frequency
    B          4
Grade  Frequency
    C          5
Grade  Frequency
    D          4
Grade  Frequency
    F          3
Grade  Frequency
    W          2
Total:22
```

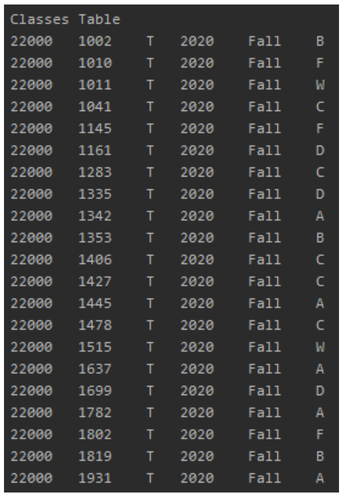


```
Students Table
--------------------------------|
1002    Smith   Ronald  M
--------------------------------|
1010    Reed    Richard M
--------------------------------|
1011    Long    Brooke  F
--------------------------------|
1041    Green   Lisa    F
--------------------------------|
1145    Trump   Donald  U
--------------------------------|
1161    Fisher  John    M
--------------------------------|
1283    Beni    Melanie F
--------------------------------|
1335    Ann Mary    F
--------------------------------|
1342    Youhana Tanvir  M
--------------------------------|
1353    Ross    Bob M
--------------------------------|
1406    Wurt    Gabyr   F
--------------------------------|
1427    Williams    Jason   M
--------------------------------|
1445    Yoon    Daniel  U
--------------------------------|
1478    Cage    Nicolas U
--------------------------------|
1515    Obama   Barrack M
--------------------------------|
1637    Ortiz   Cindy   F
--------------------------------|
1699    Ortiz   Angelica    F
--------------------------------|
1782    Rice    Justin  M
--------------------------------|
1802    Harris  Robert  M
--------------------------------|
1819    Ortiz   Angel   U
--------------------------------|
1931    King    Brandon U
```





```
Course Table
22000   Algorithms  Computer Science
```



```
Classes Table
22000   1002    T   2020    Fall    B
22000   1010    T   2020    Fall    F
22000   1011    T   2020    Fall    W
22000   1041    T   2020    Fall    C
22000   1145    T   2020    Fall    F
22000   1161    T   2020    Fall    D
22000   1283    T   2020    Fall    C
22000   1335    T   2020    Fall    D
22000   1342    T   2020    Fall    A
22000   1353    T   2020    Fall    B
22000   1406    T   2020    Fall    C
22000   1427    T   2020    Fall    C
22000   1445    T   2020    Fall    A
22000   1478    T   2020    Fall    C
22000   1515    T   2020    Fall    W
22000   1637    T   2020    Fall    A
22000   1699    T   2020    Fall    D
22000   1782    T   2020    Fall    A
22000   1802    T   2020    Fall    F
22000   1819    T   2020    Fall    B
22000   1931    T   2020    Fall    A
```