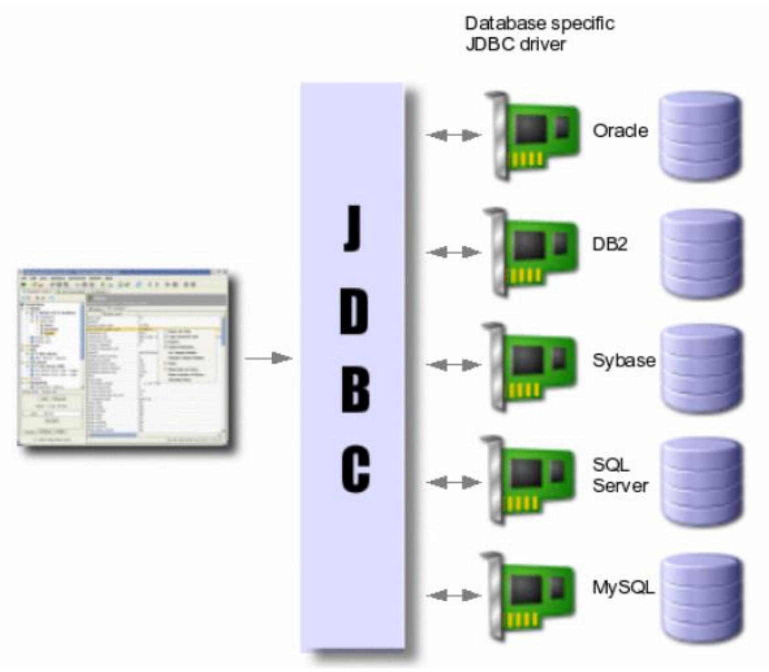
CSCD 327 Project (35 points)

Please submit via Canvas
See Canvas for due date



In this project, you will focus on writing SQL queries. In addition, you will embed your SQL queries into Java (using JDBC) to implement a standalone program that answers several queries about the university database (*database_4*).

1. Java Files

You are provided two java files: 'TestMyQuery.java' and 'MyQuery.java'.

TestMyQuery.java

This file provides the main function for running the program. You should only modify three variables (mydatabase, username, and password), replacing them with your own information.

```
String serverName = "localhost:3306";
String mydatabase = "DatabaseName";
String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
String username = "YourUsername";
String password = "YourPassword";
```

MyQuery.java

This is the file in which you need to implement the query functions. Feel free to make any modifications to the file.

2. Queries (35 points + 5 extra)

There are seven total queries (Query1 to Query7). The points are evenly distributed (5 points per query). However, the queries may vary in terms of difficulty. If you get stuck on a harder query, try an easier one first, and then come back to the tough one.

Query 1: Calculate GPA for each student

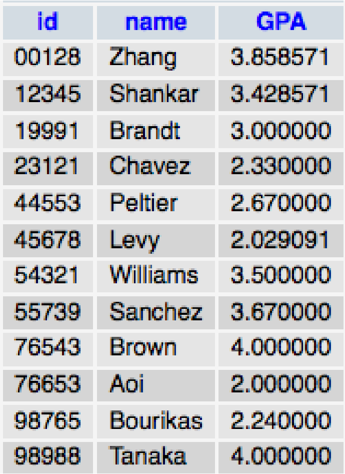
First you need to turn letter grade into numeric grade:

A: 4.0; A-: 3.67; B+:3.33; B: 3; B-: 2.67; C+: 2.33; C: 2; C-: 1.67; D+: 1.33; D: 1; D-: 0.67; F: 0

The GPA is calculated with the following formula (Don't include *null* grade in the calculation):

*GPA = sum(numerical grade * credit) / sum(credit)*

Here is the correct query result for your reference:

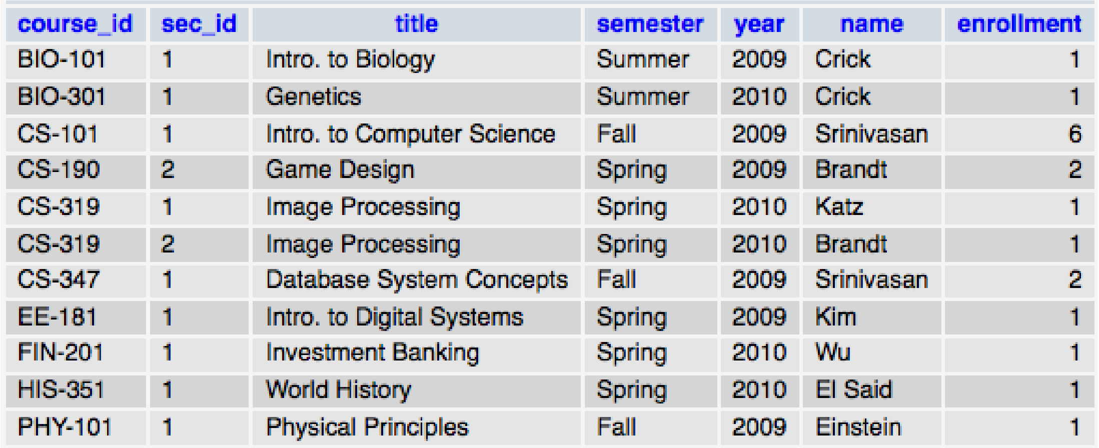


id	name	GPA
00128	Zhang	3.858571
12345	Shankar	3.428571
19991	Brandt	3.000000
23121	Chavez	2.330000
44553	Peltier	2.670000
45678	Levy	2.029091
54321	Williams	3.500000
55739	Sanchez	3.670000
76543	Brown	4.000000
76653	Aoi	2.000000
98765	Bourikas	2.240000
98988	Tanaka	4.000000

Query 2: Find all the course sections offered in the morning. Ignore those sections having zero enrollments.

Include all the course sections which have start_hr <= 12 in the result. List *course_id*, *title*, *section_id*, *semester*, *year*, *instructor name*, and *the enrollment number* for each course.

Here is the correct query result for your reference:

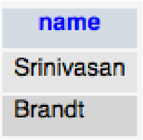


course_id	sec_id	title	semester	year	name	enrollment
BIO-101	1	Intro. to Biology	Summer	2009	Crick	1
BIO-301	1	Genetics	Summer	2010	Crick	1
CS-101	1	Intro. to Computer Science	Fall	2009	Srinivasan	6
CS-190	2	Game Design	Spring	2009	Brandt	2
CS-319	1	Image Processing	Spring	2010	Katz	1
CS-319	2	Image Processing	Spring	2010	Brandt	1
CS-347	1	Database System Concepts	Fall	2009	Srinivasan	2
EE-181	1	Intro. to Digital Systems	Spring	2009	Kim	1
FIN-201	1	Investment Banking	Spring	2010	Wu	1
HIS-351	1	World History	Spring	2010	El Said	1
PHY-101	1	Physical Principles	Fall	2009	Einstein	1

Query 3: Find the busiest instructor(s)

Find the name(s) of instructor(s) who has (or have) taught the most number of courses (there may be more than one such instructor). Y**ou cannot use *ORDER BY* and *LIMIT* in your SQL**.

Here is the correct query result for your reference:

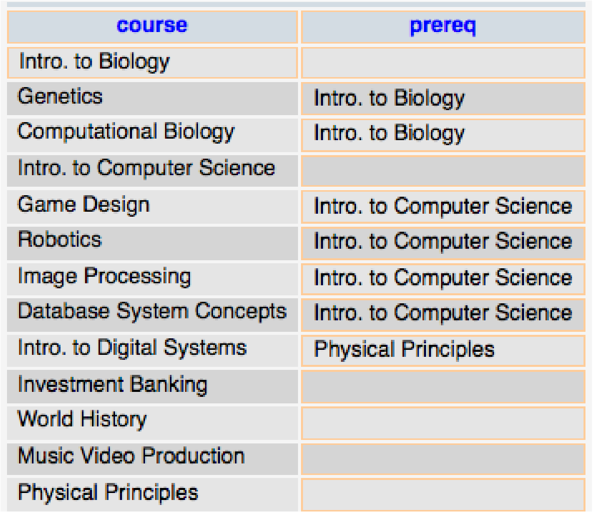


name
Srinivasan
Brandt

Query 4: Find the prereq for each course

For each course, list the name of the course and the name of its prereq course. You need to include those courses that don't have any prereqs (Leave the prereq field **empty** for those courses).

Here is the correct query result for your reference:

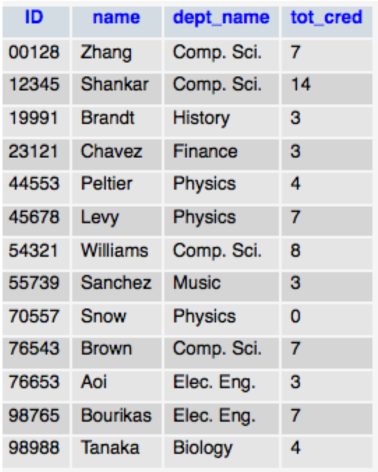


course	prereq
Intro. to Biology	
Genetics	Intro. to Biology
Computational Biology	Intro. to Biology
Intro. to Computer Science	
Game Design	Intro. to Computer Science
Robotics	Intro. to Computer Science
Image Processing	Intro. to Computer Science
Database System Concepts	Intro. to Computer Science
Intro. to Digital Systems	Physical Principles
Investment Banking	
World History	
Music Video Production	
Physical Principles	

Query 5: Update Tot_Cred

When you look at the *student* relation you will find that the *tot_cred* field provides incorrect information. Now you are going to update this field with the real total credits the students received. To maintain the cleanness of the original data, however, instead of making changes to the *student* relation, I would like you to make a copy of the *student* table, and name it as *studentCopy*, and update the *tot_cred* field in *studentCopy*. Note that if a student got an *F* or the grade is *null*, he/she got 0 credits for that course. Display the *studentCopy* table after the update.

Here is the correct query result for your reference:

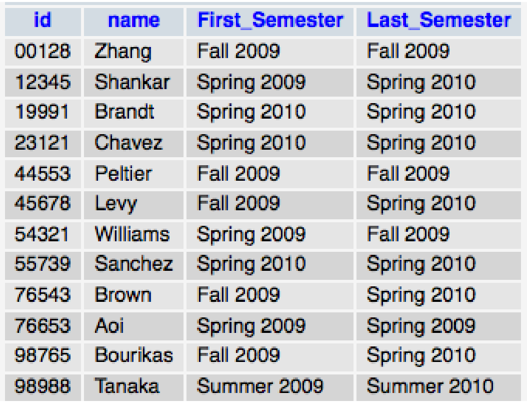


ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	7
12345	Shankar	Comp. Sci.	14
19991	Brandt	History	3
23121	Chavez	Finance	3
44553	Peltier	Physics	4
45678	Levy	Physics	7
54321	Williams	Comp. Sci.	8
55739	Sanchez	Music	3
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	7
76653	Aoi	Elec. Eng.	3
98765	Bourikas	Elec. Eng.	7
98988	Tanaka	Biology	4

Query 6: Find the first and the last semesters

For each student, find the first semester and the last semester in which he/she has taken a course. The key to this query is to find a way to compare three different semesters (Spring, Summer, and Fall) in a year. Combine the semester and the year information in one field in your result.

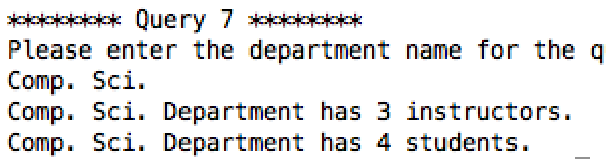
Here is the correct query result for your reference:



id	name	First_Semester	Last_Semester
00128	Zhang	Fall 2009	Fall 2009
12345	Shankar	Spring 2009	Spring 2010
19991	Brandt	Spring 2010	Spring 2010
23121	Chavez	Spring 2010	Spring 2010
44553	Peltier	Fall 2009	Fall 2009
45678	Levy	Fall 2009	Spring 2010
54321	Williams	Spring 2009	Fall 2009
55739	Sanchez	Spring 2010	Spring 2010
76543	Brown	Fall 2009	Spring 2010
76653	Aoi	Spring 2009	Spring 2009
98765	Bourikas	Fall 2009	Spring 2010
98988	Tanaka	Summer 2009	Summer 2010

Query 7: Write a stored procedure to get the head count

First you define a stored procedure in *database_4* named *getNumbers*. This procedure takes department name as input, and returns the number of instructors and the number of students in the department as outputs. In other words, *getNumbers*() has one input parameter and two output parameters. Test this procedure in MySQL to make sure it functions properly. Next, your application program asks the user to enter a department name (e.g., "Comp. Sci."), and the program should return the number of instructors and the number of students in the department accordingly. Here is a snapshot of the output.



```
******** Query 7 ********
Please enter the department name for the query:
Comp. Sci.
Comp. Sci. Department has 3 instructors.
Comp. Sci. Department has 4 students.   _
```

3. Compiling and Running Your Code

It is highly recommended that your program should access the database server via **localhost**. This means to access your databases, your application program must be on the same machine as the database server (i.e., *cscd-327-dev.eastern.ewu.edu*). Use your **Eastern credential** to log onto the server. I have already installed JDK and JDBC on the server for your application development, but you need to complete one extra task before you start working on your code, i.e. set up the environmental variable *CLASSPATH* properly.

Use *vim* or other editor (e.g. nano) to edit your profile file named *.profile* under your home directory. Copy and paste the following line to the bottom of your *.profile* file.

```
export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar
```

Save the file. Logout the server, and log back in. Type *javac* to make sure the system recognizes this command. Now you are ready to edit, compile, and run your application code on the server.

- **How to edit**: Your Eastern NetStorage folder has been mapped onto our server. I would suggest you to put all your project files on NetStorage so that you can easily access and edit your source files on a Windows machine (I assume Windows is your preferred operating system).
- **How to compile**: ssh to the server and enter "*javac TestMyQuery.java*".
- **How to run**: ssh to the server and enter "*java TestMyQuery*".

4. Submission

You need to submit your work through Blackboard online submission system. Include the following files into a single .zip file, name it as YourFirstName_YourLastName.zip, and submit this file:

- TestMyQuery.java
- MyQuery.java
- result.txt, result.doc, result.pdf, or result.jpg

The result file should show the test results you have obtained. If you have worked on the extra credit for Query 6, please also include the SQL query in the result file. You don't need to provide the results in any fancy format, but I hope the results are organized neatly.

5. One Last Note

For each of these queries, think carefully about what computation should be done in SQL, and what computation should be done in Java. Of course, you can always compute the "correct" answer by fetching all of the data from MySQL, and loading it into Java virtual memory, but this is very inefficient! Instead, most of the computation can (and should) be done in SQL, and you should only retrieve the minimum amount of data necessary.