# CSCD 327 Project (35 points) Due: 11:59pm, November 24, 2020

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 crime database (First, you need to create a new database named *crimeDB*; then you need to import the tables using *database\_crime.sql*). Please refer to *schema.pdf* to find the partial description of the database.

#### 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 one variable (*password*), replacing it with the actual root password.

```
String serverName = "localhost";
String mydatabase = "crimeDB";
String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
String username = "root";
String password = "root password"; //change needed
```

### MyQuery.java

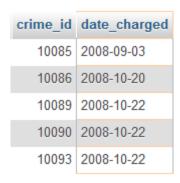
This is the file in which you need to implement the query functions. Feel free to make any modifications to the file.

## 2. Queries (35 points)

There are 7 queries (Queries 1-7) in this assignment (Query 0 is a sample solution provided by the instructor). 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.

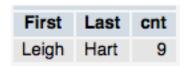
### Query 0: List all the crimes that have a charge date before October 23, 2008.

Here is the correct query result for your reference:



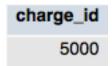
Query 1: List the name of each officer who has reported more than the average number of crimes officers have reported.

Here is the correct query result for your reference:



Query 2: List the information on crime charges for each charge that has had a fine (*fine\_amount*) above average and a paid amount (*amount\_paid*) below average.

Here is the correct query result for your reference:



Query 3: List all the names of all criminals who have had any of the crime code charges involved in crime ID 10089.

Here is the correct query result for your reference:

first	last
Sam	Phelps
Dave	Caulk
Tommy	Cat
Tim	Simon
Reed	Pints
Nancy	Mansville
Cart	Perry
Penny	Statin
Lee	Panner

Query 4: List criminals (ID and name) who have multiple sentences assigned.

Here is the correct query result for your reference:

criminal_id	last	first	cnt_sentence
1030	Panner	Lee	2

Query 5: List the total number of crime charges successfully defended (i.e, charge\_status = 'GL') by precinct. Include only precincts with at least seven guilty charges.

Here is the correct query result for your reference:

precinct	charge_cnt
WAVE	8

Query 6: For each criminal, list the start\_date of the first sentence, and the end date of the last sentence.

criminal_id	first	last	earliest_start_date	latest_end_date
1020	Sam	Phelps	2008-09-15	2010-09-15
1021	Tammy	Sums	2008-12-05	2009-06-05
1022	Dave	Caulk	2009-03-20	2009-08-20
1024	Cart	Perry	2008-12-20	2009-03-20
1025	Tommy	Cat	2008-12-20	2009-03-20
1026	Tim	Simon	2008-12-20	2009-03-20
1027	Reed	Pints	2008-12-20	2009-03-20
1028	Nancy	Mansville	2008-12-20	2009-03-20
1029	Penny	Statin	2008-12-20	2009-02-05
1030	Lee	Panner	2008-12-20	2009-07-06

Query 7: Write a stored procedure to get the number of crimes reported by an officer.

First you define a stored procedure in *database\_crime* named *getNumber*. This procedure takes an *officer\_id* as input, and returns the number of crimes reported by the officer as output. In other words, *getNumber()* has one input parameter and one output parameter. Test this procedure in MySQL to make sure it functions properly. Next, your application program asks the user to enter an officer\_id (e.g., "111115"), and the program should return the number of crimes reported by the officer accordingly. Here is a snapshot of the output.

```
******** Query 7 *******

Please enter the officer_id for the query:
111115

Officer 111115 has reported 9 crimes.
```

## 3. Compiling and Running Your Code

## **Eclipse Users**

- 1) Download three files from Canvas:
  - a. TestMyQuery.java
  - b. MyQuery.java
  - c. mysql-connector-java-8.0.19.jar
- 2) Open TestMyQuery.java using Eclipse
  - a. Edit variable *password* in TestMyQuery.java
  - b. Go to Project → Properties → Library → Add External Jar, and add *mysql-connector-java-8.0.19.jar* file
  - c. Now you should be able to compile and run TestMyQuery without any error messages.

## jGRASP Users

- 1) Download three files from Canvas:
  - a. TestMyQuery.java
  - b. MyQuery.java
  - c. mysql-connector-java-8.0.19.jar
- 2) Open TestMyQuery.java using jGRASP
  - a. Edit variable *password* in TestMyQuery.java
  - b. Go to Settings→PATH/CLASSPATH→Workspace, select CLASSPATHS tab, and add a new class path pointing to *mysql-connector-java-8.0.19.jar* file
  - c. Now you should be able to compile and run TestMyQuery without any error messages.

## 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

You don't need to provide the results in any fancy format, but I hope the results are organized clearly and neatly. Here's the sample output from Query 0:

***** Q1	ery 0 ******
Crime_ID	Charge_Date
10085	2008-09-03
10086	2008-10-20
10089	2008-10-22
10090	2008-10-22
10093	2008-10-22