CMPT 355 Assignment 1
Due at 11:00am September 29, 2017

## Description/Instructions

For this assignment we will be using the idea of a university course database. A university can have several departments, each of which can have a number of courses, such as CMPT 355. Some of these courses have multiple sections. These sections can be of different types, such as a lecture, lab, or tutorial. Assume that a section can only exist in one term and that each section has only one instructor. Multiple students can be enrolled in multiple sections.

There are four scripts included for you to use with this assignment. You can open them in DBVisualizer and run them using the **arrow button with the red dot**. They need to be run in the following order: assgn1-ddl.sql, assgn1-dml.sql, assgn1-dml2.sql. Running these scripts will create the university course data model described above, and populate it with some fake data. This will let you test your queries in part two, and add new tables to the database in part three. After you are done this assignment, you can use the assgn1-dropTables.sql script to remove these tables from your database.

As with all assignments in this class, make sure you keep a script of everything you do. We will be marking the assignments based on the scripts and reports you hand in, as well as what is in your database. Applying the scripts you hand in should be the equivalent of what is in your database. For this assignment, you can hand in Part 1 in a .txt file (since you are not applying anything to the database). Parts 2 and 3 can be handed in as an .sql file. Remember that when you have multiple SQL statements in a row, you need to use the ; symbol after each statement as the statement terminator.

## Part 1 – Analyzing the Data Model /30

Review the data model found in Assignment1-Part1.png and use it to answer the following questions. Keep in mind that there might be multiple ways to solve a problem. You only need to provide one approach, unless otherwise stated.

1) List all the possible candidate key(s) for the **departments** table. **/2**
   - If we didn't use a surrogate key for the primary key on the **enrollments** table, what would you have used as a natural primary key instead and why? **/4**
2) We have a gender column on the **students** table. The type on this column is a CHAR(1), but any single character value could be input into that field right now.
   **a)** Name two different approaches you could use to make sure that only the ISO gender codes (M,F,U,N) are valid values to insert into the gender column. **/4**
   b) Write the DDL you would use to implement both of these solutions (but don't actually change anything in the database). **/10**

3) Currently in the **sections** table, it is possible for the num_enrolled to be higher than the max_enrollment. What could you add to make sure this doesn't happen? Write the code that would accomplish this. **/10**

## Part 2 – Querying the Data Model /40

Write queries to return the requested information. Make sure that the results are formatted so that an outside user could understand what each field means. You can use your choice of implicit or explicit joins, except where otherwise specified. Remember that when writing queries, a good first step is to look at all the information you need to return and identify the tables (or functions you could use) to get that data.

1) Return the university name, department name, course code, course name, course description, and credit units of the courses offered at the University of Saskatchewan. **/5**
2) We want to check to see if the number of students enrolled in the **enrollments** table matches the number of students in **sections**.num_enrolled. Write a query to return a count of the number of sections that have a matching number of enrollments. **/5** (**HINT:** you'll want to use a subquery in the WHERE clause to aggregate data.)
3) Display all the sections of the CMPT **courses**. Include the course code, course name, lecture type, maximum enrollment, number currently enrolled, remaining number of available enrollments, the name of the instructor of the section, the start and end date of the term, and the start and end time of the section. **Write this query twice – once using implicit joins and again using explicit joins. /10 (5 each)**
4) With the help of an inline view, return the number of **sections** for each lecture type of each **course**. You should return the following information: course code, course name, lecture type, number of sections. Sort your query first by the course code, and then by the number of sections. **/10** (**HINT:** try writing the inline view to aggregate the section information first and then create the rest of the query. See the inline view example in slide set 5 for an example.)
5) We want to report a section status for each **section** in our database. Based on how many students are enrolled in the section, the section status can be one of three values: room available, section full, or section over-filled. With the help of a case statement, write a query to return the course code, course name, section code, lecture type, and section status. **/10**

## Part 3 – Changing the Data Model /30

So far, our data model has a field for the student's overall grade in the **enrollments** table. However, there are a lot of factors that go into that final grade. We're going to add an assessment portion to our structure. See Assignment1-AssessmentModel.png for the updated data model - note the addition of **enrollment_assessments** and **assessments**.

1) You'll notice there are a number of blank spaces in this model, labeled 1-7. Fill in these blanks with appropriate values. Some of the values are foreign keys, and some are data types. Assume it's possible to receive partial marks on an assessment (such as 75.5/100). **/7**
2) Write and apply the DDL statements required to add these tables to the database. Remember to include constraints (such as primary and foreign keys) where appropriate. **/10**
3) Using INSERT statements, add records for our five assignments, one midterm, and one final into the **assessments** table for the CMPT 355 lecture section. Remember that each assignment is worth 10%, the midterm is worth 20%, and the final is worth 30%. You can assume that all assessments have a total_points value of 100, and you can make up your own due date. **/5**
    a) **Can you do this using only one insert statement with a SELECT clause? If you can, you only have to use this one query to get full marks on the whole question. /3**
4) Let's say that the instructor for lectures of CMPT 355 has suddenly quit. Using the appropriate DML statements, remove the current instructor record from the database completely and assign me (Ellen) to teach the class. Feel free to make up whatever data you need to make my instructor record. **/5**