NYU Tandon School of Engineering
Computer Science and Engineering Department
CS-UY 3083 Section B
Professor Ratan Dey

Homework No. : 04
Total points: 100.
Due Date: **03/26/2017 11:59 pm**. Late Submission will be accepted until 03/28/2017 11.59 pm but for each day, homework will be graded on 20% less points than original.

Please submit electronically using NYU-Classes. You may work **individually or with a partner**. If you work with a partner, you may hand in one file for both of you. Make sure both of your names are in your file (near the top).

Teaching Assistant: Andrew Zarenberg (email: az1148@nyu.edu) and Chun Wu (cw1753@nyu.edu).

HOMEWORK #4
Hand in your solutions via NYU Classes in one file: The SQL queries should be in a plain text file named homework4-queries.txt or homework4-queries.sql so we can easily copy/paste your queries in order to test them.

You are strongly advised to add data to the University database in order to test your SQL queries thoroughly. Do not hand in the test data or the results of executing the queries; just use it to check your work. You might find it helpful to test the subqueries separately before testing the whole queries in which they are nested. Modify the data in the university database in order to test your queries thoroughly.

If you're using a DBMS other than MySQL, note which one near the top of the file. NOTE: MySQL doesn't have INTERSECT or EXCEPT. Use IN or NOT IN (subquery), instead.

For example:
```
SELECT a1, a2 FROM T1 WHERE predicate1
AND (a1, a2) NOT IN
(SELECT a1, a2 FROM T2 WHERE predicate2))
```

is equivalent to

```
(SELECT a1, a2 FROM T1 WHERE predicate1)
EXCEPT
(SELECT a1, a2 FROM T2 WHERE predicate2)
```

Problem 1
Write SQL queries for each of the following:

1. Find IDs of students who got an A in CS-101 and an A in CS-319.

2. Find IDs and names of students who got an A in CS-101 and an A in CS-319.

3. Find IDs and names of students who got an A in CS-319 and who took CS-101, but did not get an A in CS-101.

4. Find IDs and names of students who got an A in CS-319 and did not get an A in CS-101.

5. Find IDs of students who have repeated a course (i.e. taken the same course more than once).

6. Create a table gradepoint(letter,points) to associate letter grades with points and fill it with the appropriate values ('A', 4.0), ('A-' ,3.7), etc. Use it to find each student's ID and grade point average.

Note: If all courses had the same number of credits, you could compute gradepoint averages with a query involving a natural join of takes and gradepoint, along with the avg aggregation operator, grouping by ids. However, that solution doesn't take account of different courses having different number of credits. For full credit, write a more complex query to deal with that issue. We'll assume that all graded courses are included, even if a student repeats the same course.

7. When the database has the gradepoint table an additional foreign key constraint would be useful on one of the other tables. What is that constraint and which table does it belong on? (Optionally, you may add the constraint to your database.)

8. Create a view called deanslist with the ids and names of students whose gradepoint averages are over 3.0.

Note: Unfortunately, If your gpa calculation involves a subquery in the where clause, MySQL won't let you use it to define a view. If necessary, re-write the view definition so it doesn't involve a subquery in the where clause.

9. Use the deanslist view to find the number of deans list students in each department.

10. Find the ID and name of the 'Comp Sci' student with the highest grade point average (of any Comp Sci student).

11. Find IDs of students who got a higher grade in CS-101 than they got in CS-347.

12. Find IDs of students who've gotten no grades lower than A- (in other words, they have A- or A in every course for which they have a grade).

13. Find the course id of each course that has been offered two years in a row. Hint: Find the course ids of courses taught in year s.year and in year s.year+1, where s is a correlation variable representing a tuple of the section table.

14. Find instructors who have taught all 'Comp Sci' courses. Hint: Model this after the problem we did in class to find students who've taken all 'Biology' courses.

15. Find id and name of each student who has taken a course with course id LIKE 'CS%' every semester that he or she has been enrolled (i.e. every semester that he/she has taken any course.) Hint:

- Write a subquery to find all the (semester,year) pairs student s has taken any courses. (Here, s represents a correlation variable.)
- Write a subquery to find all the (semester, year) pairs when student s has taken a course with id LIKE 'CS%'.
- Nest these subqueries inside a main query involving the correlation variable, a set difference operation, and a test for an empty relation. (Remember that MySQL doesn't have the except operator, so the set difference operation requires an additional level of nesting.)

16. List all courses and the IDs of instructors who've taught them. Include courses that haven't been taught, with NULL in the ID column for such courses.

Problem 2
Consider the table definitions in the university DDL.sql file used in HW 1.

1. What tables could change and how would they change if an instructor is deleted from the instructor table? Explain.

2. Now suppose each ON DELETE SET NULL were changed to ON DELETE CASCADE. What tables could change and how would they change if an instructor is deleted from the instructor table? Explain.

3. Add a constraint to assure that a student cannot take a section of a course that no one teaches. (If you want to try this on MySQL you might need to create an additional index).

4. Consider the following foreign key constraints on the section table:

a) foreign key (building) references classroom(building)
b) foreign key (room_number) references classroom(room_number)
c) foreign key (building, room_number) references classroom(building, room_number)

Give a small example of data that satisfy constraint a and constraint b, but do not satisfy constraint c.