NYU Tandon School of Engineering
Computer Science and Engineering
CS-UY 3083, Introduction to Database Systems, Fall 2018 Prof Frankl

HOMEWORK #4

Hand in your solutions via NYU GradeScope. In addition, hand in a *plain text file* named `hw4.txt` or `hw4.sql` vi NYU Classes, so we can easily copy/paste your queries in order to test them.

I strongly recommend that you execute them and check that the results are what you expect them to be. 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.

You may work with classmates (groups of up to 5 people). If you work with others, you should hand in one file for the whole group. Make sure all of your names are in your file (near the top.) **Use the group handin feature on GradeScope. Only one person from your group should hand in a file on NYU Classes.**

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 and names 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-319 and who took CS-101, but did not get an A in CS-101.
3. Find IDs of students who have repeated a course (i.e. taken the same course more than once.)
4. 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 create a view StudentGPA(id, gpa) showing 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. You may do that for partial credit. 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.

5. 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.)

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

7. 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).

8. 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.

9. Find IDs of instructors who have taught all 'Comp Sci' courses. Note that an instructor i is in this set if and only if the set of all 'Comp Sci' courses is a subset of the set of courses i taught.

    - Draw a Venn Diagram with two overlapping circles representing the set of all Comp Sci classes and the set of classes instructor i taught. Label the sets.
    - Add an element to one of the three sections of the diagram, representing a Comp Sci course that *i did not teach*.
    - Shade in the section of the diagram that will be empty if i taught every Comp Sci class. (This should be the section you just added an element to in the previous part.)
    - Write query representing the Comp Sci courses that instructor 12345 did not teach
    - Write a predicate that will be true if and only iff instructor 12345 taught all Comp Sci courses.
    - Now, instead of focussing on instructor 12345, enclose the predicate you just wrote in a query that will check whether an arbitrary instructor i taught all Comp Sci courses. If you've followed all the steps correctly, this query will be a solution to this problem.

    This is another approach to the same problem.

    - Write a query to find the number of Comp Sci courses that instructor 12345 taught.

- Write a query to find the total number of Comp Sci courses
- Write a predicate that uses the above two counting queries and that is true if and only if instructor 12345 taught all Comp Sci courses.
- Now, instead of focussing on instructor 12345, enclose the predicate you just wrote in a query that will check whether an arbitrary instructor i taught all Comp Sci courses. If you've followed all the steps correctly, this query will be a solution to this problem.

10. Following the approach above (labelled Venn diagram, development of predicate indicating that some set difference is empty OR develoment of predicate indicating that two sections of the diagram have the same number of elements), Find the names of People who Watch all the TV series that Gene Guo watches. (Suggestion: add some data to the database to test your query).

11. Find TV series that no one watches.

12. 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.)