CS411: Database Systems **Spring 2018**

Assignment 3

Due Monday, March 26 at 11:59pm

General Instructions

- Feel free to talk to other members of the class in doing the homework. You should, however, write down your solutions yourself. *List the names of everyone you worked with at the top of your submission.*
- Keep your solutions brief and clear.
- Please use Piazza if you have questions about the homework but do not post answers. Feel free to use private posts or come to the office hours.

Homework Submission

- We DO NOT accept late homework submissions.
- We will be using Compass for collecting the homework assignments. Please submit your answers via Compass. Hard copies are not accepted.
- Contact the TAs if you are having technical difficulties in submitting the assignment; attempt to submit well in advance of the due date/time.
- The homework must be submitted in **pdf** format. Scanned handwritten and/or hand-drawn pictures in your documents won't be accepted.
- Please do not zip the answer document (PDF) so that the graders can read it directly on Compass. You need to submit one answer document, named as **hw3_netid.pdf**.
- Please see the assignments page for more details. In particular, we will be announcing errata, if any, on this page.
- For this HW, please use syntax and ONLY syntax that was covered in class.

1 Single-Relation Queries (30 pts)

1. [10] Consider the following relation:

   `Graph(n1, n2)`

   A tuple (n1, n2) in Graph stores a directed edge from a node n1 to a node n2 in the corresponding graph. Your goal is to, for *every* node in the graph, count the number of outgoing edges of that node. Note that for nodes without any outgoing edges, their edge count would be zero; you need to output this as well.

   You can assume that (1) there are no duplicates or null values in the table; and (2) every node in the graph is involved in at least one edge.

   ANSWER:

```
SELECT n1 as n, COUNT(*) as cnt FROM Graph GROUP BY n1
UNION
SELECT n2 as n, 0 as cnt FROM Graph
WHERE n2 NOT IN (SELECT n1 FROM Graph)
GROUP BY n2
```

2. [10] Consider the following relation:

   `Trained(student, master, year)`

   A tuple (S, M, Y) in Trained specifies that a SQL Master M trained student S who graduated in year Y. Your goal is to find *the count of* distinct SQL Masters who trained a student who graduated in the same year that 'Alice' or 'Bob' graduated.

   ANSWER:

```
SELECT COUNT(DISTINCT(master))
FROM Trained
WHERE year IN
        (SELECT year FROM Trained
         WHERE student = 'Alice' OR student = 'Bob')
```

3. [10] Consider the following relation:

   `DBMS(operator, system, performance)`

   A tuple (O, S, P) in DBMS specifies an operator O in system S and has the performance value P. Your goal is to find those systems whose operators achieves a higher performance value on average than the average performance value in a system named 'PostgreSQL'.

   ANSWER:

```
SELECT system
FROM DBMS
GROUP BY system
```

```
HAVING AVG(performance)>
        (SELECT AVG(performance)
        FROM DBMS
        WHERE system = 'PostgreSQL')
```

ALTERNATIVE ANSWER:

```
SELECT system
FROM DBMS
GROUP BY system
HAVING AVG(performance)>
        (SELECT AVG(performance)
        FROM DBMS
        GROUP BY system
        HAVING system = 'PostgreSQL')
```

2 Multi-Relation Queries (20 pts)

Consider the following relations representing student information at UIUC:

```
Mentorship(mentee_sid, mentor_sid)
Study(sid, credits)
Enrollment(did, sid)
Student(sid, street, city)
```

- A tuple (M1, M2) in Mentorship specifies that M2 is a mentor of another student M1.
- A tuple (S, C) in Study specifies that the student S has taken C credits.
- A tuple in Enrollment (D, S) specifies that student S is enrolled in department D.
- A (ST, S, C) in Student specifies that student ST lives on street S in city C.

1. [10] Find all students who live in the same city and on the same street as their mentor.

   ANSWER:

   ```
   SELECT s1.sid
   FROM Student s1, Student s2, Mentorship m
   WHERE s1.sid = m.mentee_sid AND s2.sid = m.menter_sid AND
           s1.street = s2.street AND s1.city = s2.city
   ```

2. [10] Find all students (sid) who have taken more credits than the average credits of all of the students of their department.

   ANSWER:

```
SELECT DISTINCT(s1.sid)
FROM Study s1, (
        SELECT AVG(s.credits) AS avg_credits, e.did
        FROM Study s, Enrollment e
        WHERE e.sid = s.sid
        GROUP BY e.did) as s2,
        Enrollment e
WHERE s1.credits > s2. avg_credits
AND s1.sid = e.sid
AND e.did = s2.did
```

3 Database Manipulation and Views (25 pts)

1. [5] In the `Study` relation, insert a new student, whose id is 66666 and has 0 credits.

   ```
   INSERT INTO Study (sid, credits)
   VALUES (66666, 0);
   ```

2. [5] In the `Study` relation, delete students who have graduated (i.e., the ones who have more than 200 credits).

   ANSWER:

   ```
   DELETE FROM Study
   WHERE credits > 200;
   ```

3. [5] In the `Study` relation, add 2 credits for students who are mentors.

   ANSWER:

   ```
   UPDATE Study
   SET credits = credits + 2
   WHERE sid IN (SELECT mentor_sid FROM Mentorship);
   ```

4. [10] Incoming students are those who have been accepted (i.e., exist in the `Student` relation) but have not registered in any department (i.e., do not exist in the `Enrollment` relation). Create a View that contains `sid` of all incoming students.

   ANSWER:

   ```
   CREATE VIEW IncomingStudents AS
           SELECT sid FROM Student
           EXCEPT
           SELECT sid FROM Enrollment
   ```

4 Constraints and Triggers (25 pts)

1. [10] Consider the following relation:

   `Payment(salary, bonus)`

   Write a schema-level assertion using the "CREATE ASSERTION" statement to ensure that no bonus is larger than the maximum salary in the `Payment` relation.

   ANSWER:

```
CREATE ASSERTION BonusControl CHECK (
    NOT EXISTS (SELECT * FROM Payment WHERE bonus >
                          (SELECT MAX(salary) FROM Payment)))
```

2. [15] Consider the following relation:

   `Study(sid, major, GPA)`

   Write a trigger T1 that increases the GPA by 10% for those students who transform their major from any Non-CS major to 'CS'.

   ANSWER:

```
CREATE TRIGGER T1
        AFTER UPDATE OF major ON Study
        REFERENCING
                OLD ROW AS O, NEW ROW AS N
        FOR EACH ROW
        WHEN (N.major = 'CS' AND O.major <> 'CS')
        UPDATE Study
        SET GPA = 1.1 * GPA
        WHERE sid = N.sid
```