CIS761– Database System Concepts
Name:_______________

SQL Assignment 2 (20 points) – due Friday, September 13th at 11:59PM

***Collaboration policy**: This is an individual assignment. You are allowed to discuss the assignment with your colleagues, but your submission should reflect your own work. Sharing or copying code is not permitted.*

Assignment objectives: create a relational **social network** database, populate the database and practice advanced SQL queries, triggers and data modification statements on this database. The database and its schema are described in what follows.

Students at your hometown high school have decided to organize their social network using databases. Here's the schema they have used, together with the corresponding English description for each relation:

Highschooler (ID, name, grade) -- There is a high school student with unique *ID* and a given *first name* in a certain *grade.*

Friend (ID1, ID2) --The student with *ID1* is friends with the student with *ID2*. Friendship is mutual, so if (123, 456) is in the Friend table, so is (456, 123).

Likes (ID1, ID2) --The student with *ID1* likes the student with *ID2*. Liking someone is not necessarily mutual, so if (123, 456) is in the Likes table, there is no guarantee that (456, 123) is also present.

An SQL script for creating and populating the database is provided in the file **social-network.sql**, available on KSOL. Answer the following questions:

Advanced SQLizing

Write an SQL query for each of the following questions (run these queries on the original sample of data):

- (3p) Find the name and grade of the student(s) with the greatest number of friends. (This is a “witness” type query.)

```
select name, grade from Highschooler
where Highschooler.ID in (select ID1 from Friend group by ID1
having count(*) >= all (select count(*) from Friend group by ID1) );


select s.name, s.grade
```

```
from Highschooler as s, Friend as f
where f.id1=s.id group by f.id1 having count(*) >= all (select
count(*) from Friend group by ID1);
```

```
select name, grade
from (select id, name, grade, count(f.id2) as NumberFriends
      from Highschooler h, Friend f
      where h.id=f.id1
      group by h.id) as FullTable
where NumberFriends = (
         select max(numberfriends)
         from (select id1, count(*) as numberfriends
               from Friend
               group by id1) as CountsTable);
```

- (3p) What is the average number of friends per student? (Your result should be just one number.)

```
select avg(numFriendsID1)
from (select count(Friend.ID2) as numFriendsID1
from Friend group by Friend.ID1) as FriendsID1;

select (V1/V2)
from
(select count(F.ID1) as V1 from Friend F) as CountFriend,
(select count(H.ID) as V2 from Highschooler H) as
CountHighschooler;
```

(3p) For each student A who likes a student B where the two are not friends, find if they have a friend C in common (who can introduce them!). For all such trios, return the name and grade of A, B, and C.

```
select H1.name, H1.grade, H2.name, H2.grade, H3.name, H3.grade
   from (select L.ID1 as A, L.ID2 as B, commonFriend.C as C
   from Likes L, (select F1.ID1 as C, F1.ID2 as A, F2.ID2 as B from
   Likes L1 inner join Friend F1 inner join Friend F2 on F1.ID2 =
   L1.ID1 and F2.ID2 = L1.ID2 where F1.ID1 = F2.ID1) as commonFriend
   where L.ID2 not in (select ID2 from Friend F where F.ID1 = L.ID1)
      and commonFriend.A=L.ID1 and commonFriend.B = L.ID2 ) Love inner
      join Highschooler H1 on H1.ID = Love.A inner join Highschooler H2
      on H2.ID = Love.B inner join Highschooler H3 on H3.ID = Love.C;
```

```
select A.name, A.grade, B.name, B.grade, C.name, C.grade
from Highschooler A, Highschooler B, Highschooler C, Likes, Friend
where A.ID = Likes.ID1 and B.ID = Likes.ID2 and C.ID = Friend.ID2
and
Likes.ID2 not in (select ID2 from Friend where ID1=Likes.ID1) and
Likes.ID1=Friend.ID1 and Friend.ID2 in (select ID2 from Friend where
ID1 = Likes.ID2);
```

- (3p) Find the number of students who are either friends with Cassandra or are friends of friends of Cassandra. Do not count Cassandra, even though technically she is a friend of a friend.

  ```
  select count(distinct f1.id2) from
  friend f1
  where (f1.id1 = 1709 or
  f1.id1 in
  (
        select f2.id2
        from friend f2
        where f2.id1 = 1709
  )) and (f1.id2<>1709) ;
  ```

Triggers

- (2p) Write a trigger that makes new students named 'Friendly' automatically like everyone else in their grade. That is, after the trigger runs, we should have ('Friendly', A) in the Likes table for every other Highschooler A in the same grade as 'Friendly'.

  ```
  DELIMITER $
  CREATE TRIGGER Friendly
  AFTER INSERT ON Highschooler
  FOR EACH ROW
  IF (new.name = 'Friendly') THEN
  INSERT INTO Likes
  SELECT new.ID, h.ID
  FROM Highschooler h
  WHERE h.grade = new.grade;
  END IF;
  $
  DELIMITER ;
  ```

- (2p) Write a trigger that automatically deletes students when they graduate, i.e., when their grade is updated to exceed 12.

  ```
  DROP TRIGGER IF EXISTS graduate;
  DELIMITER $
  CREATE trigger graduate
  AFTER UPDATE on Highschooler
  FOR EACH ROW
  begin
  IF (NEW.grade > 12) THEN
  DELETE FROM Highschooler
  WHERE ID = NEW.ID;
  END IF;
  END;
  $
  DELIMITER ;
  ```

Data modification statements

- (2p) For all cases where A is friends with B, and B is friends with C, add a new friendship for the pair A and C. Do not add duplicate friendships, friendships that already exist, or friendships with oneself.

  ```
  INSERT INTO Friend (ID1, ID2)
  SELECT DISTINCT f1.ID1, f2.ID2
  FROM Friend AS f1, Friend AS f2
  WHERE f1.ID2 = f2.ID1 and f1.ID1 != f2.ID2 and NOT EXISTS
     (SELECT f3.ID1, f3.ID2
     FROM Friend f3
     WHERE f3.ID1 = f1.ID1 and f3.ID2 = f2.ID2
     );
  ```

- (2p) If two students A and B are friends, and A likes B but not vice-versa, remove the Likes tuple.

  ```
    DELETE FROM Likes
    USING (SELECT h1.ID AS ID1, h2.ID AS ID2
          FROM Highschooler h1, Highschooler h2, Likes L
          WHERE h1.ID = L.ID1 AND h2.ID = L.ID2 AND
            h2.ID NOT IN (
            SELECT ID1 FROM Likes WHERE h1.ID = ID2)
            AND h1.ID NOT IN (
            SELECT ID2 FROM Likes WHERE h2.ID = ID1)
    ) AS temp, Likes
    WHERE Likes.ID1 = temp.ID1 AND Likes.ID2 = temp.ID2 AND
     temp.ID1 IN (
        SELECT ID1 FROM Friend WHERE temp.ID2 = ID2)
    AND temp.ID2 IN (
        SELECT ID2 FROM Friend WHERE temp.ID1 = ID1);
  ```

What to turn it: A *.txt* file with all SQL queries (together with their results), triggers and data modification statements. Submit this file using the course Dropbox.