Exam 2, Spring 2013
CSCI 4380 Database Systems
Time: 110 minutes

Name : ______________________

Q1.	**Q4.**
Q2.	
Q3.	**Total.**

Rules. Open book and notes. Do not use any electronic tools including your computer. Work alone. You **cannot** talk to anyone in class, or share notes or thoughts.

Question 1. Answer all parts of this question using the data model in the appendix. Write the following queries using SQL only (no procedural code is allowed). Make sure you use DISTINCT only if you have to.

(a) **(14 points)** Find all people who are friends with 'Clara Oswin Oswald' both on Facebook and Twitter since 2012. Return their names.

(b) **(14 points)** Find all people for whom the number of followers on Twitter is at least 10 times the number of friends on Twitter. Return the name of the people, the number of followers and the number friends on Twitter.

If the person has no friends, then return him if he has at least 10 followers. (**Hint: you can do this query with a single SELECT/FROM/WHERE/GROUP BY/HAVING clause with no nested subqueries.**)

(c) **(14 points)** Find pairs of people who are friends on Facebook, but are not friends on Twitter. Return their ids. (You can return one or two tuples for each pair of ids).

(d) **(10 points)** Write at most two update statements in a transaction block (no other procedural code is allowed) to update the type attribute for people to `'active'` if they have exchanged a message on Twitter in the last month, and `'passive'` otherwise.

Question 2 (18 points). We are now mining Facebook for some information. Suppose you are given the new table below:

```
CREATE TABLE fb_stats (
    state               VARCHAR(100)
    , agegroup          INT PRIMARY KEY
    , avg_ff_num        FLOAT
    , earliest_adoption  DATE
    , PRIMARY KEY (state, agegroup)
) ;
```

This table stores the earliest adoption date of Facebook of all people in that state and age group. The earliest adoption date for a person is the earliest day that they were linked to someone on Facbook (given by the `whenlinked` attribute).

The table also stores the average number of friends per person (`avg_ff_num`) in Facebook of all people in the same state and age group.

Write code in pseudo-code to populate this table based on data available in the database. You can use a single SQL query or any procedural code, however you must spell out any query in proper SQL.

Age group is a value of the form 10, 20, 30, etc. to denote people with ages 0-10, ages 11-20, ages 21-30, etc. Here is some help. The code below finds the age of a person born on January 1, 1985 (by finding the number of days between now and his birth day):

```
select trunc(extract(days from now() - date '01-01-1985')/365) ;
```

Question 3 (10 points). Assume you are already given the table `fb_stats` populated with the statistics from Question 2. Write the following query using SQL with or without procedural components.

We are computing some measure of hipness. For each agegroup, find the state with the earliest adoption date. Return the agegroup, the state with the earliest adoption date and the adoption date. If multiple states are tied for a specific age group, return all of them.

Question 4 (20 points). You are given the following table definitions and instances. Each of the statements below operate on the original tables. Write down the results of each operation below by listing only the changed tuples. Provide a short sentence of why these tuples were changed.

```
CREATE TABLE d (
    id     INT PRIMARY KEY
    , name VARCHAR(100) );
CREATE TABLE e (
    id     INT PRIMARY KEY
    , name VARCHAR(100)
    , did  INT NOT NULL FOREIGN KEY
      REFERENCES d(id) ON UPDATE CASCADE
    , pcnt INT ) ;
CREATE TABLE ep (
    pid     INT
    , eid   INT
    , PRIMARY KEY(pid, eid)
    , FOREIGN KEY (pid) REFERENCES p(id)
      ON DELETE CASCADE ON UPDATE CASCADE
    , FOREIGN KEY (eid) REFERENCES e(id)
      ON DELETE CASCADE ON UPDATE CASCADE ) ;
```

```
CREATE TABLE p (
    id     INT PRIMARY KEY
    , name VARCHAR(100) ) ;
CREATE TRIGGER epdel AFTER DELETE ON ep
FOR EACH ROW
REFERENCING OLD ROW AS old
DECLARE
   c int ;
BEGIN
   SELECT count(*) INTO c
   FROM ep
   WHERE pid = old.pid ;
   UPDATE e SET pcnt = (SELECT count(*)
       FROM ep WHERE ep.eid = old.eid);
   IF c <= 1 THEN
       DELETE FROM p WHERE id = old.pid ;
   END IF ;
END ;
```

id	name
1	'da'
2	'db'
3	'dc'

(d)

id	name	did	pcnt
1	'ea'	1	1
2	'eb'	1	2
3	'ec'	2	0
4	'ed'	1	2

(e)

id	name
1	'pa'
2	'pb'
3	'pc'

(p)

pid	eid
1	1
1	2
1	4
2	2
2	4

(ep)

(a) `DELETE FROM d WHERE d.name = 'db';`

(b) `DELETE FROM ep WHERE ep.pid = 2 and ep.eid=4;`

(c) `UPDATE e SET id = 22 WHERE id = 2;`

(d) `DELETE FROM e WHERE id < 3;`

Question 5 (0 points). What is good movie pitch based on SQL? Here is my pitch for "Expandables":

```
select
    distinct actors
from
    blockbusters
where
    year >= 1990
    and year < 2000
    and nr_order = 1
limit
    8
```

Appendix. This is a data model for keeping track of a set of people of interest to you in two different social networks, Facebook and Twitter.

```
CREATE TABLE people (
     fid           INT PRIMARY KEY
     , fuser       VARCHAR(20)  UNIQUE
     , tuser       VARCHAR(20)  UNIQUE
     , name        VARCHAR(100)  NOT NULL
     , dateofbirth DATE
     , state       VARCHAR(100)
     , type        VARCHAR(20)
) ;
```

Stores all the people in the database. Attributes `fuser` and `tuser` refer to the username of the user on Facebook and Twitter respectively. The `state` attribute stores where the person lives (if known), assuming of course that all the people in this database are from US. The `type` attribute will be described in one of the questions.

```
CREATE TABLE fb (
     user1              VARCHAR(20)
     , user2            VARCHAR(20)
     , how              VARCHAR(100)
     , whenlinked       DATE
     , lastcommunicated DATE
     , PRIMARY KEY(user1,user2)
     , FOREIGN KEY (user1) REFERENCES person(fuser)
       ON DELETE CASCADE ON UPDATE CASCADE
     , FOREIGN KEY (user2) REFERENCES person(fuser)
       ON DELETE CASCADE ON UPDATE CASCADE
) ;

CREATE TABLE tw (
     user               VARCHAR(20)
     , friend           VARCHAR(20)
     , whenlinked       DATE
     , lastcommunicated DATE
     , PRIMARY KEY(user, friend)
     , FOREIGN KEY (user) REFERENCES person(tuser)
       ON DELETE CASCADE ON UPDATE CASCADE
     , FOREIGN KEY (friend) REFERENCES person(tuser)
       ON DELETE CASCADE ON UPDATE CASCADE
) ;
```

The table `fb` stores who is friends with whom in Facebook, and the table `tw` stores the same in Twitter. Friendship in Facebook is symmetric, but only one tuple per pair is stored. If `(user1,user2)` is in `fb`, then `user1` friended `user2`, and `user2` approved it.

For Twitter, relationships are not necessarily symmetric. If A (`user`) is a friend of B (`friend`) on Twitter, then it means that B is a follower of A on Twitter. If A and B are mutual friends , then two tuples are stored.

For both tables, we store the date when the pair last communicated: i.e. they exchanged one message in each direction (A to B, and B to A) on the same day. Note that for two people to communicate on Twitter, they must be mutual friends.