CMPS 182, Final Exam, Spring 2018, Shel Finkelstein

Student Name: ______________________________________

Student ID: ______________________________________

UCSC Email: ______________________________________

Part	Max Points	Points
I	42	
II	24	
III	36	
Total	102	

The first Section (Part I) of the Spring 2018 CMPS 182 Final is multiple choice and is double-sided. Answer all multiple choice questions on your Scantron sheet. You do not have to hand in the first Section of the Exam, but you must hand in the Scantron sheet, with your name, email and student id on that Scantron sheet. Please be sure to use a #2 pencil to mark your choices on this Section of the Final.

This separate second Section (Parts II and III) of the Final is not multiple choice and is single-sided, so that you have extra space to write your answers. If you use that extra space, please be sure to write the number of the problem that you're solving next to your answer. Please write your name, email and student id on this second Section of the Exam, which you must hand in. You may use any writing implement on this Section of the Exam.

At the end of the Final, please be sure to hand in your **Scantron sheet** for the first Section of the Exam and also **this second Section of the Exam**, and show your **UCSC id** when you hand them in. Also hand in your **8.5 x 11 "cheat sheet"**, with your name written in the upper right corner of the sheet.

Part II: (24 points, 6 points each)

Question 22: Here are statements creating two relations R and S:

```
CREATE TABLE S (
   c INT PRIMARY KEY,
   d INT
   );
```

```
CREATE TABLE R (
   a INT PRIMARY KEY,
   b INT,
   FOREIGN KEY(b)
         REFERENCES S(c)
            ON UPDATE CASCADE
            ON DELETE SET NULL
   );
```

Relation R(a,b) currently contains the four tuples, (0,4), (1,5), (2,4), and (3,5).
Relation S(c,d) currently contains the four tuples, (2,80), (3,81), (4,82), and (5,83).

Indicate all changes that happen to both R and S when the following SQL statements are executed upon the R and S instances that are shown above. That is, you should ignore changes made by earlier parts of this question when you answer later parts.

22a): UPDATE R set b=5 WHERE a=0;

Answer 22a):
Changes made to the original R:
(0,4) becomes (0,5).

Changes made to the original S:
No changes

22b): DELETE FROM S WHERE c=4;

Answer 22b):
Changes made to the original R:
(0,4) becomes (0, NULL), and (2,4) becomes (2, NULL).

Changes made to the original S:
(4,82) is deleted.

22c): UPDATE S SET c=9 WHERE d=83;

Answer 22c):
Changes made to the original R:
(1,5) becomes (1,9), and (3,5) becomes (3,9).

Changes made to the original S:
(5,83) becomes (9,83).

Question 23a): Why are advantages of having Stored Procedures in a database? Give two different clear reasons. If you give more than two, only first two will be graded.

Answer 23a):

Any two of the following three reasons are good:

- Performance: Having a stored procedure means that code can be executed directly in the database, without having to move data between the database and the client.

- Reuse: Once a Stored Procedure has been rewritten, it can be re-used by anyone who's allowed to use it, without having to write it again.

- Security: A user may be granted the right to run a Stored Procedure, even though that user is not allowed to access the data that the Stored Procedure uses.

Question 23b): Since Stored Procedures have the advantages that you listed in part a), why are there also other approaches that combine programming language constructs with SQL?

Answer 23b):

Stored Procedures can't do everything that (for example) a C or Java program can do. For example, they can't handle user interactions, and they don't have access to all the libraries that C and Java programs can employ.

Also, putting more functionality into the database increases the workload of the database, whereas they can be many instances of clients running C/Java programs.

Question 24: Customer(cid, name, level, type, age) is a table, in which type can have values ‘snowboard’, ‘ski’ and ‘skate’. If there are no skiers in an instance, but there are other customers, what are the results of the following queries? Each of your answers must be one of the following; an answer can be used more than once.

i. Result is always the Empty Set
ii. Result is always all the Customers
iii. Result is always an Error
iv. None of the Above

24a):
```
SELECT c.name
FROM Customers c
WHERE c.age > ALL  ( SELECT c2.age
                     FROM Customers c2
                     WHERE c2.type = ‘ski’ );
```

Answer 24a): ii. Result is always all the Customers

24b):
```
SELECT c.name
FROM Customers c
WHERE c.age > ANY  ( SELECT c2.age
                     FROM Customers c2
                     WHERE c2.type = ‘ski’ );
```

Answer 24b): i. Result is always the Empty Set or iii. Result is always an Error
WHERE was omitted before c.age, so either of these answers will be accepted.

24c):
```
SELECT c.name
FROM Customers c
WHERE c.age > ( SELECT MAX(c2.age)
                FROM Customers c2
                WHERE c2.type = ‘ski’ );
```

Answer 24c): i. Result is always the Empty Set or iii. Result is always an Error
WHERE was omitted before c.age, so either of these answers will be accepted.

24d):
```
SELECT c.name
FROM Customers c
WHERE c.age > ( SELECT MIN(c2.age)
                FROM Customers c2
                WHERE c2.type = ‘ski’ );
```

Answer 24d): i. Result is always the Empty Set

Question 25: You have a relation R(A, B, C, D), with the following non-trivial Functional Dependencies:

$AB \rightarrow C$
$AB \rightarrow D$
$CD \rightarrow A$

25a): Is R in BCNF? Justify your answer.

Answer 25a)

No, R is not in BCNF. For the FD $CD \rightarrow A$

- The Functional Dependency isn't non-trivial, and
- $(CD)^+$ is just CD, just the left-hand side isn't a superkey.

Hence R is not in BCNF.

25b): Is R in 3NF? Justify your answer.

Answer 25b)

Yes, R is in 3NF. Trivial FDs always hold, and they are obviously trivial. Just as obviously, none of the non-trivial FDs provided are trivial. But:

- For the FDs $AB \rightarrow C$ and $AB \rightarrow D$, the left-hand side is a superkey, since $(AB)^+$ is ABCD.
- For the FD $CD \rightarrow A$, the right-hand side is part of a key, namely AB. We already know that AB is a superkey, since $(AB)^+$ is ABCD. Is there any subset of AB that is also a key?
    - A^+ is just A.
    - B^+ is just B

  Since there is no subset of superkey AB that is a superkey, then AB is a key. So the right-hand side of the FD $CD \rightarrow A$ is part of a key, AB.

So all of the non-trivial FDs provided for R are "dogs" (left-side is a superkey) or "horses" (right-hand side is part of a key), we've shown that R is in 3NF.

Part III: (36 points, 9 points each)

Some familiar tables appear below, with Primary Keys underlined. **These tables also appear on the last page of the Final, which you can tear off to help you do questions in Part III of the Final.** You don't have to hand in that last page at the end of the Exam.

ChirpUsers(<u>userID,</u> userPassword, userName, joinDate, address, education, income, spouseID, active)

ChirpPosts(<u>posterID, postNum</u>, thePost, censored, postDate)

ChirpFollowers(<u>userID, followerID</u>, followStartDate)

ChirpReads(<u>posterID, postNum, postReader</u>, timesRead, latestReadDate)

Assume that <u>no attributes</u> can be NULL, and that there are no UNIQUE constraints.

Assume Referential Integrity as follows:
- Each posterID in ChirpPosts appears as a userID in ChirpUsers.
- Each followerID in ChirpFollowers appears as a userID in ChirpUsers.
- Each userID in ChirpFollowers appears as a userID in ChirpUsers.
- Each postReader in ChirpReads appears as a userID in ChirpUsers.
- Each (posterID, postNum) in ChirpReads appears as a (posterID, postNum) in ChirpPosts

Write legal SQL queries for Questions 25-28. If you want to create and then use views to answer these questions, that's okay, but views are not required unless the question asks for them.

Don't use DISTINCT in your queries unless it's necessary; 0.5 points will be deducted. Some points will also be deducted for unnecessarily complex queries.

Question 26: Find the posterID, postNum and thePost for each post that was read by at least one active user. (active is a Boolean attribute in ChirpUsers.) Duplicates should not appear in your answer.

Answer 26:

There are multiple correct ways to write each of the SQL queries requested in Part III. We've provided at least one correct way of answering each question.

```
SELECT DISTINCT posterID, postNum, thePost
FROM ChirpPosts p, ChirpReaders r, ChirpUsers u
WHERE p.posterID = r.posterID
    AND p.postNum = r.postNum
    AND r.readerID = u.userID
    AND u.active;
```

Could also write the last line as: `AND u.active = TRUE;`

DISTINCT is needed, since there might be multiple active users who read a particular post.

```
SELECT posterID, postNum, thePost
FROM ChirpPosts p,
WHERE EXISTS ( SELECT *
                FROM ChirpReaders r, ChirpUsers u
                WHERE p.posterID = r.posterID
                    AND p.postNum = r.postNum
                    AND r.readerID = u.userID
                    AND u.active );
```

DISTINCT is not needed, since we're going through each different post to decide if there exists any reader who's an active user.

Some students also included the condition:

```
        r.timeRead > 0
```

or the equivalent. That was fine, but we didn't require it.

Question 27: ChirpUsers whose education is ‘C’ are college students. Find the userID and userPassword for each user who is a college student, but who isn’t followed by any college student. No duplicates should appear in your answer.

Answer 27:

```
SELECT userID, userPassword
FROM ChirpUsers u
WHERE u.education = ‘C’
    AND NOT EXISTS ( SELECT *
                            FROM ChirpFollowers f, ChirpUsers u2
                            WHERE f.userID = u.userID
                                AND f.followerID = u2.userID
                                AND u2.education = ‘C’);
```

DISTINCT is not needed, since userID is the primary key of ChirpUsers.

This query can also be written using NOT IN.

```
SELECT userID, userPassword
FROM ChirpUsers u
WHERE u.education = ‘C’
    AND u.userID NOT IN ( SELECT f.userID
                               FROM ChirpFollowers f, ChirpUsers u2
                               WHERE f.followerID = u2.userID
                                   AND u2.education = ‘C’);
```

Meaning: The user has a college education, but isn’t in the set of users who are followed by someone with a college education.

It can be written similarly using <> ALL, with the same subquery. Writing it using <> ANY would be an error.

```
SELECT userID, userPassword
FROM ChirpUsers u
WHERE u.education = ‘C’
    AND u.userID <> ALL ( SELECT f.userID
                               FROM ChirpFollowers f, ChirpUsers u2
                               WHERE f.followerID = u2.userID
                                   AND u2.education = ‘C’);
```

A somewhat common error was finding college students who had at least one follower who isn’t a college student. But Question 27 asked for college students who didn’t have any followers who were college students.

Question 28: Several users might live at the same address. Find the userID, userName, address and income for each individual whose income is the largest of any individual who lives at their address. Your result should be ordered by income, with larger income values appearing ahead of smaller income values.

Answer 28:

```
SELECT u.userID, u.userName, u.address, u.income
FROM ChirpUsers u
WHERE u.income = ( SELECT MAX(u2.income)
                   FROM ChirpUsers u2
                   WHERE u.address = u2.address )
ORDER BY u.income DESC;
```

Note that multiple users could have the same income, so multiple users could have the largest income of any individual who lives at their address.

There are many ways of writing this query incorrectly, such as trying to put MAX into the outer SELECT, which won't give you the individuals who have the maximum income.

Question 29: This question has two parts; be sure to answer both.

29a): Create a view named ManyFollowers. For each user in ChirpUsers, this view should provide userID and the number of followers that the user has. Don't count followers if 'Potter' appears at the end of their names. In your result, the second attribute should be called numberOfFollowers. But only include a tuple for a user if that user has at least 5 followers (who don't have 'Potter' at the end of their names).

29b): If there's a tuple for a userID in ManyFollowers, then we'll say that the user has many followers. Write a query over the ManyFollowers view (and the ChirpFollowers table) to execute the following query:

For each user that has many followers, output that user's userID, that user's numberOfFollowers (as it appears in ManyFollowers), and the earliest followStartDate for any follower of that user. The attributes in your result should be called userID, numberOfFollowers and earliestFollowDate.

Answers 29a) and 29b):

This problem is very similar to the Lab3 createview and queryview problems.

29a):

```
CREATE VIEW ManyFollowers AS
        SELECT f.userID, COUNT(f.followerID) AS numberOfFollowers
        FROM ChirpFollowers f
        WHERE f.followerID NOT IN ( SELECT u.userID
                                    FROM ChirpUsers u
                                    WHERE u.userName LIKE '%Potter')
        GROUP BY f.userID
        HAVING COUNT(f.followerID) >= 5;
```

or

```
CREATE VIEW ManyFollowers AS
        SELECT f.userID, COUNT(f.followerID) AS numberOfFollowers
        FROM ChirpFollowers f, ChirpUsers u
        WHERE f.followerID = u.userID
            AND u.userName NOT LIKE '%Potter'
        GROUP BY f.userID
        HAVING COUNT(f.followerID) >= 5;
```

COUNT(*) would also be okay in either view definition, as would COUNT of any attribute that can't be NULL.

29b):

```
SELECT m.userID, m.numberOfFollowers,
                    MIN(f.followStartDate) AS earliestFollowDate
FROM ManyFollowers m, ChirpFollowers f
WHERE m.userID = f.userID
GROUP BY m.userID, m.numberOfFollowers;
```

Some students may have forgotten to include m.numberOfFollowers in the GROUP BY clause.

This is extra blank space, in case you need more paper to answer questions.
Write Question number clearly if you use blank pages.

You may tear off this page to help you do Part III of the Final.

Some familiar tables appear below, with Primary Keys underlined. **These tables also appear on the last page of the Final, which you can tear off to help you do questions in Part III of the Final.** You don't have to hand in that last page at the end of the Exam.

ChirpUsers(userID, userPassword, userName, joinDate, address, education, income, spouseID, active)

ChirpPosts(posterID, postNum, thePost, censored, postDate)

ChirpFollowers(userID, followerID, followStartDate)

ChirpReads(posterID, postNum, postReader, timesRead, latestReadDate)

Assume that no attributes can be NULL, and that there are no UNIQUE constraints.

Assume Referential Integrity as follows:

- Each posterID in ChirpPosts appears as a userID in ChirpUsers.
- Each followerID in ChirpFollowers appears as a userID in ChirpUsers.
- Each userID in ChirpFollowers appears as a userID in ChirpUsers.
- Each postReader in ChirpReads appears as a userID in ChirpUsers.
- Each (posterID, postNum) in ChirpReads appears as a (posterID, postNum) in ChirpPosts.

Write legal SQL queries for Questions 25-28. If you want to create and then use views to answer these questions, that's okay, but views are not required unless the question asks for them.

Don't use DISTINCT in your queries unless it's necessary; 0.5 points will be deducted. Some points will also be deducted for unnecessarily complex queries.