CMPS 180, Midterm Exam, Fall 2016, Shel Finkelstein

Student Name: ______________________________

Student ID: ______________________________

UCSC Email: ______________________________

Midterm Points

Part	Max Points	Points
I	30	
II	25	
III	27	
IV	20	
Total	102	

Please show your **UCSC ID** when you turn in your exam.

Part I: (30 points, 6 each):

Question 1: Assume that R is a relation that has 5 tuples in it, S is a relation that has 4 tuples in it, and T is a relation that has 3 tuples in it. How many tuples are there in the result of the following SQL query?

```
SELECT *
FROM R, S, T;
```

Answer 1: 5 * 4 * 3 = 60

Question 2: Let R(A,B,C,D) be a relation, where attributes C and D can't be NULL, and where (A, B) is the primary key for that relation. Assume that A's domain has 10 different values, B's domain has 5 different values, C's domain has 4 different values, and D's domain has 3 different values. What is the maximum number of different tuples that can be in R?

Answer 2: 10 * 5 = 50. Since primary key must be unique, the number of different tuples must be less than or equal to the maximum number of primary key values.

Question 3: For the relation StarsIn(<u>movieTitle, movieYear,</u> starName), where movieTitle and starName are character, movieYear is integer, and(movieTitle, movieYear) is the primary key, write a SQL query that finds all the different titles of movies starring Harrison Ford.

Answer 3:

```
SELECT DISTINCT movieTitle
FORM StarsIn
WHERE starName = ‘Harrison Ford’;
```

Question 4: What are **two major advantages** of using transactions, rather than just executing SQL statements one at a time? Explain your two answers briefly.

Answer 4:

Here are several advantages of transactions. Question 4 only asks for two of these ACID properties.

#1: Atomicity. All of the transaction is executed, or none of it is. This is important for transactions such as moving money from one account to another.

#2: Isolation. A transaction (running with an isolation level other than Read Uncommitted, such as Serializability) doesn't see partial results of another transaction. Partial results may reflect work that might be rolled back, or incomplete work, a state of the data that never existed.

#3: Consistency. Operating with serializability, business rules (such as having a business' books balanced across debits, credits and balances) are preserved. That is, if the rules were true before a set of transactions started, and each transaction preserve the rules, then with serializability it's "as if" transactions were executed in some serial order, then at the end of execution of those transactions, the business rules are still true.

#4: Durability. With transactions, you're guaranteed that data that has been committed will not go away, even if there's a failure.

Question 5: SQL uses 3-valued logic, with TRUE, FALSE and UNKNOWN.

Rows don't appear in a query result if the value of the WHERE clause condition is FALSE, and they also don't appear in the result if the value of the WHERE clause is UNKNOWN. Perhaps UNKNOWN could be eliminated from SQL, replaced by FALSE? Nope. Explain why eliminating UNKNOWN that way would change SQL semantics.

Answer 5:

In SQL, the negation of TRUE is FALSE, and the negation of FALSE is TRUE, but the negation of UNKNOWN is still UNKNOWN. If salary is NULL for an employee named Smith, then the value of the WHERE condition "salary > 5" in the query:

```
SELECT name
FROM Employees
WHERE salary > 5;
```

will be UNKNOWN, and Smith won't be in the result.

But for the second query:

```
SELECT name
FROM Employees
WHERE NOT (salary > 5);
```

the value of the WHERE condition "NOT (salary > 5)" is also UNKNOWN, since negation of UNKNOWN is UNKNOWN, so once again Smith won't be in the result.

But if we replaced UNKNOWN with FALSE, the value of "salary > 5" would be FALSE if salary was NULL, and the value of "NOT (salary > 5)" would be TRUE, so for the second query, Smith's name would be in the result, a change to the semantics of SQL.

Part II: (25 points, 5 each)

The following questions concern a table that was created by the statement:

```
CREATE TABLE Students (
        student_id          INTEGER PRIMARY KEY,
        name                VARCHAR(20) ,
        address             VARCHAR(30),
        age                 INTEGER,
        major               CHAR(4) NOT NULL DEFAULT ('CMPS'),
        UNIQUE (name, address)
        );
```

Answer each question in this Part of the Midterm with **TRUE** or **FALSE**.

Question 6: major can never have the value NULL, but any other attribute could be NULL.

Answer 6: FALSE. Primary Key (student_id) can never be NULL.

Question 7: The answers to the following 3 queries might <u>all be different</u>.

```
SELECT COUNT(*)
FROM Students;

SELECT COUNT(name)
FROM Students;

SELECT COUNT(DISTINCT name)
FROM Students;
```

Answer 7: TRUE. COUNT(*) counts tuples where name is NULL. COUNT(name) only counts tuples where name is not NULL. COUNT(DISTINCT name) counts duplicate names only once.

Question 8: The following is a legal SQL query.

```
SELECT MAX(age) - MIN(age), major
FROM Students
WHERE age > 18
GROUP BY major;
```

Answer 8: TRUE

Question 9: If there are no tuples in Students, and you execute the statement:

```
INSERT INTO Students(student_id, name, age)
   VALUES (12345, 'John Smith', 21);
```

then after executing that INSERT statement, there will be one row in Students, with student_id 12345, name 'John Smith', address NULL, age 21, and major 'CMPS'.

Answer 9: TRUE. When values aren't supplied, DEFAULT values are used. For attributes that don't have a DEFAULT value and that also don't forbid NULL, the DEFAULT value used is NULL.

Question 10:

If (98765, 'Eliza Doolittle', 'Higgins Place', 21, 'ENGL') is a tuple in the Students table, and the following is executed, with no other work going on:

```
BEGIN TRANSACTION;

UPDATE Students
SET age = age + 1
WHERE name = 'Eliza Doolittle';

UPDATE Students
SET major = 'CMPS'
WHERE age = 22;

ROLLBACK;
```

then afterwards, for the tuple with student_id 98765, the age will be 22 and the major will be 'CMPS'.

Answer 10: FALSE. When a transaction is rolled back, its database modifications aren't executed.

Part III: (27 points, 9 each):

Questions 11-13 are about the instances of the tables Customers, Slopes and Activities on the sheet at the back of the test.

Show attribute names at the top for all SQL results.

Question 11: What is the result of the following SQL query?

```
SELECT a.day
FROM Slopes s, Activities a
WHERE s.color = 'Blue'
AND s.slope_id = a.slope_id;
```

Answer 11:

```
  a.day
-------------
01/05/09
01/06/09
01/07/09
01/07/09
```

Question 12: What is the result of the following SQL query?

```
SELECT DISTINCT c.cname
FROM Customers c
WHERE c.age <= ALL ( SELECT c2.age
                     FROM Customers c2
                     WHERE c.type = c2.type
                      AND c2.level = 'Advanced' )
ORDER BY c.cname;
```

Answer 12:

```
c.cname
--------
 Cho
 Ice
 Luke
```

Cho and Luke appear because the age of each is less than the age for all the rows in the empty set. (Recall our example: I'm taller than all the Martians on Earth.)

Ice's age is less than or equal to his own. But Paul's age isn't less than or equal to Ice's age, so Paul doesn't appear.

The cnames in the answer appear in ascending order.

Question 13: What is the result of the following SQL query?

```
SELECT c.cname, s.sname, COUNT(a.day),
FROM Activities a, Customers c, Slopes s
WHERE a.cid = c.cid
AND a.slope-id = s.slope-id
GROUP BY c.cname, s.sname;
```

Answer 13:

```
c.cname          s.sname          COUNT(a.day)
---------------------------------------------------------------
Cho          Magic Carpet          1
Cho          Mountain Run          2
Ice          Olympic Lady          1
Ice          Mountain Run          1
Luke         Olympic Lady          1
```

Cho went on the slope named Mountain Run on two days.

Part IV: (<u>20</u> points, 10 each):

Question 14-15 are also about the relations Customers, Activities and Slopes that appear at the end of the test. For these questions, you should write SQL queries that are correct for <u>any instances</u> of the relations, not just for the data shown.

(If you want to create and then use views to answer these questions, that's okay, but it's not required.)

Question 14: Write a SQL query that determines the total number of activities that each customer did. (A customer did an activity if there's a row with that customer's cid in the Activities table.)

Your result should show the customer's cid and cname, and the activity count. But don't include any customers who engaged in no activities.

Answer 14:

Here are 3 solutions. The first is a simple GROUP BY, the second uses a View, and the third uses a subquery in the FROM Clause (Lecture 4, slide 30).

There's no need to include a HAVING Clause that gets rid of customer who engaged in no activities, since there won't be groups for customers who have no activities.

Solution 1:

```
SELECT c.cid, c.cname, COUNT(*)
FROM Customers c, Activities a
WHERE c.cid = a.cid
GROUP BY c.cid, c.cname;
```

Solution 2:

```
CREATE VIEW ActivityCount(cid, act_count) AS
  SELECT a.cid, COUNT(*)
  FROM Activities a
  GROUP BY a.cid;

SELECT c.cid, c.cname, k.act_count
FROM Customers c, ActivityCount k
WHERE c.cid = k.cid;
```

Solution 3:

```
SELECT c.cid, c.cname, k.act_count
FROM Customers c,
        ( SELECT a.cid AS cid, COUNT(*) AS act_count
          FROM Activities a
          GROUP BY a.cid) k
WHERE c.cid = k.cid;
```

Question 15: Write a SQL query whose result is the name of Slopes where at least two different customers did Activities on that slope. Only the name of the slope should appear, and no slope name should appear more than once.

Answer 15:

Here are 4 solutions. The first has a subquery with EXISTS, checking for existence of two rows in activities on a specific slope that have different customers.

The second is a GROUP BY, using COUNT(DISTINCT a.cid) >=2. (Checking for activities where the number of different customers on a specific slope is 2 or more.)

The third (View) and fourth solutions (subquery in the FROM clause) are variations of the second solution. Solutions 2, 3 and 4 for Q15 and similar to solutions 1, 2 and 3 (respectively) for Q14.

Solution 1:

```
SELECT DISTINCT s.sname
FROM Slopes s
WHERE EXISTS ( SELECT *
               FROM Activities a1, Activities a2
               WHERE s.slope_id = a1.slope_id
                  AND s.slope_id = a2.slope_id
                  AND a1.cid <> a2.cid
              );
```

Solution 2:

```
SELECT DISTINCT s.sname
FROM Slopes s, Activities a
WHERE s.slope_id = a.slope_id
GROUP BY s.slope_id, s.sname
HAVING COUNT(DISTINCT a.cid) >=2;
```

Solution 3:

```
CREATE VIEW SlopeCustCount(slope_id, cust_count) AS
    SELECT a.slope_id, COUNT(DISTINCT a.cid)
    FROM Activities a
    GROUP BY a.slope_id;

SELECT DISTINCT s.sname
FROM Slopes s, SlopeCustCount scc
WHERE s.slope_id = scc.slope_id
AND scc.cust_count >= 2;
```

Solution 4:

```
SELECT DISTINCT s.sname
FROM Slopes s,
            ( SELECT a.slope_id AS slope_id, COUNT(DISTINCT a.cid) AS cust_count
              FROM Activities a
              GROUP BY a.slope_id ) scc
WHERE s.slope_id = scc.slope_id
AND scc.cust_count >= 2;
```

For Parts III-IV (Questions 11-15), here are the relations Customers, Activities and Slopes that describe customer participation in winter activities on slopes.
cid is the primary key for Customers, slope-id is the primary key for Slopes, and (cid, slope-id, day) is the primary key for Activities.

Customers

cid	cname	level	type	age
36	Cho	Beginner	snowboard	18
34	Luke	Inter	snowboard	25
87	Ice	Advanced	ski	20
39	Paul	Beginner	ski	33

Activities

cid	slope-id	day
36	s3	01/05/09
36	s1	01/06/09
36	s1	01/07/09
87	s2	01/07/09
87	s1	01/07/09
34	s2	01/05/09

Slopes

slope-id	sname	Color
s1	Mountain Run	Blue
s2	Olympic Lady	Black
s3	Magic Carpet	Blue
s4	KT-22	green