CMPS 182, Midterm Exam, Winter 2016, Shel Finkelstein

Student Name: _____

Student ID: _____

UCSC Email: _____

Midterm Points

Part	Max Points	Points
I	36	
II	20	
III	32	
IV	18	
Total	106	

Part I: (36 points, 6 each):

Question 1: Assume that R is a relation that has 10 tuples in it, and S is a relation that has 8 tuples in it. How many tuples are there in the result of the following SQL query?

SELECT *
FROM R, S;

Answer 1: ___80_____

Question 2: Let R(A,B,C) be a relation, where attributes A, B and C can't be NULL, and where A is the primary key for that relation. Assume that A's domain has 12 different values, B's domain has 5 different values, and C's domain has 2 different values. What is the maximum number of different tuples that can be in R?

Answer 2: _____12____

Question 3: What do the following statements do?

DELETE FROM Employees;

DROP TABLE Employees;

Answer 3:

DELETE statement does the following:

Deletes all the tuples from the Employees Table. However, the Employees Table is still there, and tuples can be inserted in it.

DROP statement does the following:

Drops the Employees Table so that it no longer exists. Not only are the tuples gone, but also since the table is gone, no tuples can be inserted in it.

Question 4: We discussed the ACID properties for transactions. What do the A, C, I and D in ACID stand for? Also, explain what one of those 4 properties means. (If you explain more than one, only the first will be graded.)

Answer 4:

Atomicity: All of the transaction happened or none of it happened.

Consistency: Some business property is always maintained for the contents of the database, such as having column values from a domain, or not null, or having value of a deptno attribute for an Employees table come from the department number key of a Departments table.

Isolation: Limiting the ways in which running transactions can affect each other, such as serializability, which means transactions run "as if" they were run one-at-a-time in some order, even though many transactions are executed simultaneously.

Durability: Changes made by a committed transaction last, even though there might be failures even the transaction committed. Of course, there may be further updates that are made by other committed transactions.

Explanation of one of the above: See above for all four.

Question 5:

a) Give one reason why primary keys of relations have indexes.
b) There could be indexes on all attributes of relations, but there aren't. Give one reason why aren't all attributes indexed?

Answer 5a):

Primary keys (and other unique attributes) have to be unique. Having index on primary makes checking uniqueness fast. Searching through entire relation would be much slower. Another reason for indexing primary keys is to speed up queries looking up tuple matching primary key (or range of keys).

Answer 5b): Answer could be any of the following:

- Indexes take extra space.
- Caching indexes takes extra memory.
- Updating indexes when relations are modified taking time, slowing query/transaction execution.

[There were two questions labeled Question 6 on the exam, so the first is now labeled Question 6A, and the second is labeled Question 6B.]

Question 6A: SQL uses 3-valued logic, with TRUE, FALSE and UNKNOWN. Fill in the truth table for OR.

Answer 6A:

P	Q	P or Q
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	UNKNOWN	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE
FALSE	UNKNOWN	UNKNOWN
UNKNOWN	TRUE	TRUE
UNKNOWN	FALSE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN

Note that FALSE is not the same as UNKNOWN, since negation (NOT) of FALSE is TRUE, but negation of UNKNOWN is still UNKNOWN.

Part II: (20 points, 5 each)

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

```
CREATE TABLE Employees (
        name            CHAR(30) ,
        salary          INTEGER,
        age             INTEGER,
        department      CHAR(5) NOT NULL DEFAULT ('Sales');
        PRIMARY KEY (name)
        );
```

Answer each questions with **TRUE** or **FALSE**.

[There were two questions labeled Question 6 on the exam, so the first is now labeled Question 6A, and the second is labeled Question 6B.]

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

Answer 6B: ___FALSE__

Question 7: The answer to the query:

```
SELECT COUNT(*)
FROM Employees;
```

might be a larger number than the answer to the query:

```
SELECT COUNT(salary)
FROM Employees;
```

Answer 7: ___TRUE____

Question 8: The following is a legal SQL query:

```
SELECT department, MIN(age), MAX(age)
FROM EMPLOYEES
WHERE salary > 9000
GROUP BY department;
```

Answer 8: ___TRUE____

Question 9: If ('Smith', 6000, 21, 'Sales') is a tuple in the Employees table, and the following is executed, with no other work going on:

```
BEGIN TRANSACTION;

UPDATE Employees
SET salary = salary + 1000
WHERE name = 'Smith';

UPDATE Employees
SET salary = salary + 500
WHERE department = 'Sales';

ROLLBACK TRANSACTION;
```

then afterwards, Smith's salary will be 7500.

Answer 9: __FALSE___

Part III: (32 points, 8 each):

Questions 10-13 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 10: What is the result of the following SQL query:

```
SELECT *
FROM Customers
WHERE age >= 20;
```

Answer 10:

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

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

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

```
SELECT DISTINCT color
FROM Slopes
WHERE EXISTS ( SELECT *
                FROM Activities
                WHERE Slopes.slope-id = Activities.slope-id );
```

Answer 11:

color
blue
black

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

→ Color
Blue
black

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

```
SELECT c.cname, s.sname
FROM Activities a, Customers c, Slopes s
WHERE a.cid = c.cid
AND a.slope-id = s.slope-id
AND a.day = '01/07/09';
```

Answer 12:

cname	sname
Cho	Mountain Run
Ice	Olympic Lady
Ice	Mountain Run

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

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

```
SELECT type, MAX(age)
FROM Customers
GROUP BY type
ORDER BY type;
```

Answer 13:

type	MAX(age)
ski	33
snowboard	25

Customers

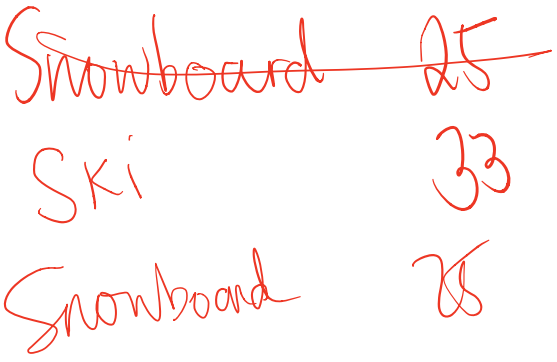
cid	cname	level	type	age
36	Cho	Beginner	snowboard	18
34	Luke	Inter	snowboard	25
87	Ice	Advanced	ski	20
39	Paul	Beginner	ski	33

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



Part IV: (18 points, 9 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 any instances 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 whose result is the age for customers whose level is Beginner and who had an activity on 01/06/09. The result should only include age, and shouldn't include any age more than once.

Answer 14:

```
SELECT DISTINCT age
FROM Customers
WHERE level = 'Beginner'
AND EXISTS
    ( SELECT *
      FROM Activities
       WHERE Activities.cid  = Customers.cid
       AND Activities.day = '0/1/06/09' );
```

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

```
SELECT DISTINCT AGE
FROM Customers C
WHERE c.level = 'Beginner'
AND EXISTS
    (SELECT * FROM Activities a
     WHERE a.cid = c.cid
     AND a.day = '01/06/09');
```

Question 15: Write a SQL query that determines the total number of activities that took place on each slope. Your result should show the slope-id, the name of the slope and the total number of activities on it. But don't include any slopes that had no activities on them.

Answer 15: Three different correct answers are shown below.

Answer #1, using GROUP BY:

```
SELECT Slopes.slope-id, Slopes.sname, COUNT(*)
FROM Slopes, Activities
WHERE Slopes.slope-id = Activities.slope-id
GROUP BY Slopes.slope-id, Slopes.sname;
```

There's no need for a HAVING clause to check for COUNT(*) > 0, although it wouldn't be incorrect to include it, just extraneous.
Non-aggregate attributes in SELECT have to appear in GROUP BY. Hence:

- Must GROUP BY sname as well as slope-id, since otherwise sname couldn't appear in the SELECT.
- Also, Couldn't have Activities.slope-id in the SELECT, since we did GROUP BY Slopes.slope-id.

Answer #2, using a View:

```
CREATE VIEW SlopeCounts AS
    SELECT slope_id, COUNT(*) AS number_of_slopes
    FROM Activities
    GROUP BY slope_id;
```

Note that once more we didn't need a HAVING clause to check for COUNT(*) > 0, since we're adding up counts for each slope within activity. Therefore, the value of number_of_slopes can never be 0. However, putting in the HAVING clause wouldn't be incorrect, just extraneous.

```
SELECT Slopes.slope-id, Slopes.sname, SlopeCounts.number_of_slopes
FROM Slopes, SlopeCounts
WHERE Slopes.slope-id = SlopeCounts.slope-id;
```

Answer #3, using a Subquery in the FROM clause.
See 6.3.5 of textbook; also mentioned in class.

```
SELECT Slopes.slope-id, Slopes.sname, SlopeCounts.number_of_slopes
 FROM Slopes, ( SELECT slope_id, COUNT(*) AS number_of_slopes
                  FROM Activities
                  GROUP BY slope_id;
               ) SlopeCounts
WHERE Slopes.slope-id = SlopeCounts.slope-id;
```

Don't expect many people to answer this way, but it's correct, and very similar to the view approach.

For Part III-IV (Questions 10-15), 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) 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