CSE 132A Midterm – Fall 2013

NAME: ________________________________ ID: ____________

This is an open-book book exam. You may use any paper materials you have brought, but no electronic devices are allowed. Write your name and student id on every page. Good luck!

Problem 1 True or False? Circle your answer. No justification required.

1. Recall that in the SQL expression SELECT r.A FROM R r, we call "r" an *alias* (a.k.a. *tuple variable* because r ranges over the tuples in R).

   Consider a schema in which each pair of distinct tables has disjoint column names. Then every SQL query Q with aliases over this schema can be rewritten to an SQL query Q' without aliases, over the same schema, such that Q' always returns the same answer as Q on every input database).

   True [✓ **False**]

2. SELECT * FROM R WHERE R.A <= 13 OR R.A > 13

   always returns the same result as

   SELECT * FROM R

   True [✓ **False**]

3. NATURAL LEFT OUTER JOIN can be expressed in SQL without using any of the keywords NATURAL, LEFT, OUTER and JOIN, and without relying on user-defined functions (UDFs)[1], either.

   [✓ **True**] **False**

4. SELECT DISTINCT R.A FROM R can be expressed without using the DISTINCT keyword (or any user-defined functions).

   [✓ **True**] **False**

[1] The SQL standard allows users to define functions in a general-purpose programming language, and call them within an SQL expression just like any other built-in function. Do not rely on UDFs for your answer.

5. SELECT MAX (R.A) FROM R can be expressed without the MAX built-in aggregate and without UDFs.

   [✓ **True**] **False**

6. Consider a relational schema consisting of tables R($\underline{\text{A}}$,B) and S($\underline{\text{B}}$,C) where R.A and S.B are primary keys. R.B is not null, and it is a foreign key referencing S.

   SELECT r.A FROM R r, S s WHERE r.B = s.B

   always returns the same answer as

   SELECT A FROM R

   [✓ **True**] **False**

7. Recall that the EXCEPT operator in SQL denotes the difference of two relations with compatible schema.

   EXCEPT can be expressed in SQL without using the EXCEPT keyword or UDFs.

   [✓ **True**] **False**

8. In SQL, all nested queries without correlated variables can be unnested.

   True **False**

9. Let R and S be relations of schema (A,B). Then

   SELECT A FROM (R UNION S)

   always returns the same result as

   (SELECT A FROM R) UNION (SELECT A FROM S).

   [✓ **True**] **False**

10. Let R(A,B) be a relation with primary key A and numeric, not-null B. Then SELECT A, MAX(B) FROM R GROUP BY A returns R.

   [✓ **True**] **False**

Problem 2 Consider the following schema modeling information about a soccer tournament. In this tournament, each pair of teams (X,Y) faces each other twice (once at the home stadium of team X, once at the home stadium of team Y). In a match, we call the team whose stadium hosts the match the *home* team, while the other team is the *away* team.

Teams (name, coach)
Matches (hTeam, aTeam, hScore, aScore)

where name is the primary key for table Teams and coach is a candidate key for the same table. Attributes hTeam and aTeam denote the home, respectively away team. Each of aTeam, hTeam are foreign keys referencing the Teams table. Their value cannot be null. The pair hTeam, aTeam is the primary key for table Matches. hScore/aScore denote the score of the home/away team, respectively. Their values cannot be null, as the Matches table refers only to completed matches whose final scores are known.

Express the following query in SQL:

Count the victories of team "LA Galaxy". Return a table with a single column called "victories".

```
SELECT count(*) AS victories
FROM   Matches m
WHERE  m.hTeam = 'LA Galaxy' AND hScore > aScore OR
       m.aTeam = 'LA Galaxy' AND aScore > hScore
```

Problem 3 In the tournament of Problem 2, each team earns per-match points as follows: a defeat results in 0 points, a tie in 1 point, a victory at home in 2 points, and a victory away in 3 points.

Express the following query in SQL:

For each team, return its name and total number of points. The result of the query will be a table with two columns: name and points.

```
CREATE VIEW Standings (name, points) AS

SELECT     name, SUM(pts) AS points
FROM       (SELECT aTeam AS name, 3 AS pts FROM Matches WHERE aScore > hScore
           UNION ALL
           SELECT hTeam AS name, 2 AS pts FROM Matches WHERE hScore > aScore
           UNION ALL
           SELECT aTeam AS name, 1 AS pts FROM Matches WHERE hScore = aScore
           UNION ALL
           SELECT hTeam AS name, 1 AS pts FROM Matches WHERE hScore = aScore
           UNION ALL
           SELECT name, 0 AS points FROM Teams)
GROUP BY name
```

Problem 4 For the schema of Problem 2, express the following query in SQL:

Return the names of undefeated coaches (that is, coaches whose teams have lost no match). The result of the query will be a table with a single column called "coach".

```
SELECT t.coach
FROM   Teams t
WHERE  NOT EXISTS
         ( SELECT *
           FROM Matches m
           WHERE m.aTeam = t.name AND m.aScore < m.hScore
                 OR
                 m.hTeam = t.name AND m.hScore < m.aScore )
```

Problem 5 For the schema of Problem 2, express the following query in SQL:

Return the teams defeated only by the leading teams. The leading teams are the teams with the highest number of points. The result of the query will be a table with a single column called "name".

Note that teams that were never defeated are part of the answer. "Defeated only by leaders" means "if defeated, then the winner is a leader" (if never defeated, then the condition on winners being leaders is vacuously satisfied). Some of you have restricted the answer to only the teams that indeed suffered at least one such defeat. This earned full points.

The solution uses the view Standings defined in the solution for Problem 3.

```
CREATE VIEW Leaders (name) AS
SELECT s.name
FROM   Standings s
WHERE  NOT EXISTS
       (SELECT * FROM Standings s1 WHERE s.points < s1.points)


SELECT name FROM Teams WHERE name NOT IN
       (SELECT t.name
       FROM    Teams t, Matches m
       WHERE   t.name = m.aTeam AND m.aScore < m.hScore
               AND m.hTeam NOT IN Leaders
               OR
               t.name = m.hTeam AND m.hScore < m.aScore
               AND m.aTeam NOT IN Leaders)
```