DSE 201 Final – Winter 2017

Good luck!

Problem 1 True or False (no justification required)? User-defined functions (UDFs) are not allowed in any of your solutions.

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

2. SELECT * FROM T WHERE T.A <= 39 OR T.A > 39

   always returns the same result as SELECT * FROM T.

3. NATURAL LEFT JOIN is SQL-expressible without the JOIN keyword.

4. SELECT DISTINCT T.A FROM T is SQL-expressible without the DISTINCT keyword.

5. SELECT MAX (R.A) FROM R can be expressed without the MAX built-in aggregate, ORDER BY, LIMIT, TOP K, WINDOW and without UDFs.

6. Let R($\underline{A}$,B) and S($\underline{B}$,C) be tables whose underlined attributes are primary keys. Attribute 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.

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

8. In SQL, all nested queries without correlated variables can be unnested (without creating views or auxiliary tables).

9. Consider tables R(A,B) and S(A,B). Then

   SELECT A FROM (R UNION S)

   always returns the same result as

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

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.

Problem 2 In a soccer league each team has a home stadium, and each pair of teams faces each other twice during the season (once at the home stadium of each team). For a given match, the team whose stadium hosts the match is the *home* team, while the other team is the *visitors* team. We model information about the league using the schema

$$\text{Teams (\underline{name}, coach)}$$
$$\text{Matches (\underline{hTeam, vteam}, hScore, vScore)}$$

where name is the primary key for table Teams and coach is a candidate key for the same table. Attributes hTeam and vteam denote the home, respectively visitors team. They are foreign keys referencing the Teams table. Their value cannot be null. The pair hTeam, vteam is the primary key for table Matches. hScore/vScore denote the score of the home/visitors team, respectively. The Matches table refers only to completed matches, listing their final scores which are not null.

Express the following in SQL:

(i) *Count the victories of team "San Diego Sockers". Return a single column called "wins".*

(ii) According to league rules, 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.
For each team, return its name and total number of points earned. Output a table with two columns: name and points.

(iii) *Return the names of undefeated coaches (that is, coaches whose teams have lost no match). Output a table with a single column called "coach".*

(iv) *Return the teams defeated only by the scoreboard leaders (i.e. "if defeated, then the winner is a leader"). The leaders are the teams with the highest number of points (several leaders can be tied). Output a single column called "name".*

(v) For each query in Problems (i) through (iv), create useful indexes or explain why there are none.

(vi) Assume that the result of the query in Problem (ii) is materialized in a table called Scoreboard. Write triggers to keep the Scoreboard up to date when the Matches table is inserted into, respectively updated. The resulting Scoreboard updates should be incremental (i.e. do not recompute Scoreboard from scratch).