**DSE 201 Final – Winter 2017**

**Problem 1**

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

1. **False**
2. **False**

Consider non-numeric data (but nulls do become returned).

1. **True**

Can use ‘Union’.

1. **True**

Using ‘Groupby’ or a nested ‘Exists In’ statement.

1. **True**

You can use (-1 \* MIN(R.A.))\*-1.

1. **True**
2. **True**

Use ‘Not Exist’ instead.

1. **False**
2. **False**

If rewrite as

SELECT A From (SELECT \* FROM R UNION SELECT \* FROM S) a

1. **True**

**Problem 2**

**Initiating the database:**

CREATE SCHEMA sockers;

-- Create necessary tables

CREATE TABLE sockers.teams

(

name character primary key NOT NULL,

coach character varying(50) NOT NULL

);

CREATE TABLE sockers.matches

(

hteam character references sockers.teams(name) NOT NULL,

vteam character references sockers.teams(name) NOT NULL,

hscore integer NOT NULL,

vscore integer NOT NULL,

PRIMARY KEY(hteam, vteam)

);

COPY sockers.teams FROM '/Users/Orysya/Desktop/DSE201\_Database\_Management\_Systems/Final/teams.csv' DELIMITER ',' CSV;

COPY sockers.matches FROM '/Users/Orysya/Desktop/DSE201\_Database\_Management\_Systems/Final/matches.csv' DELIMITER ',' CSV;

Express the following in SQL

**/\* 1. Count the victories of team "San Diego Sockers". Return a single column called "wins". \*/**

CREATE VIEW sockers.homewins(team, hwins) AS

SELECT t1.team AS team, COALESCE(t2.hwins, 0) AS hwins

FROM (

SELECT t.name AS team, 0 AS hwins

FROM sockers.teams as t) AS t1

LEFT JOIN (

SELECT hteam AS team, COUNT(m.hteam) AS hwins

FROM sockers.matches AS m

WHERE m.hscore > m.vscore

GROUP BY hteam) AS t2

ON t1.team = t2.team;

CREATE VIEW sockers.awaywins(team, awins) AS

SELECT t1.team AS team, COALESCE(t2.awins, 0) AS awins

FROM (

SELECT t.name AS team, 0 AS awins

FROM sockers.teams as t) AS t1

LEFT JOIN (

SELECT vteam AS team, COUNT(m.vteam) AS awins

FROM sockers.matches AS m

WHERE m.vscore > m.hscore

GROUP BY vteam) AS t2

ON t1.team = t2.team;

PREPARE wins(char) AS

SELECT (home.hwins + away.awins) AS wins

FROM sockers.homewins AS home, sockers.awaywins AS away

WHERE home.team = away.team AND home.team = $1;

EXECUTE wins('G');

**/\* 2. 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 2 columns: name and points. \*/**

CREATE VIEW sockers.ties(team, ties) AS

SELECT t1.team AS team, (COALESCE(t2.hties,0) + COALESCE(t3.aties, 0)) AS ties

FROM (

SELECT t.name AS team, 0 AS awins

FROM sockers.teams as t) AS t1

LEFT JOIN (

SELECT hteam AS team, COUNT(m.hteam) AS hties

FROM sockers.matches AS m

WHERE m.hscore = m.vscore

GROUP BY hteam) AS t2

ON t1.team = t2.team

LEFT JOIN (

SELECT vteam AS team, COUNT(m.vteam) AS aties

FROM sockers.matches AS m

WHERE m.vscore = m.hscore

GROUP BY vteam) AS t3

ON t1.team = t3.team;

CREATE VIEW sockers.losses(team, losses) AS

SELECT t1.team AS team, (COALESCE(t2.hloss,0) + COALESCE(t3.aloss, 0)) AS losses

FROM (

SELECT t.name AS team, 0 AS aloss

FROM sockers.teams as t) AS t1

LEFT JOIN (

SELECT hteam AS team, COUNT(m.hteam) AS hloss

FROM sockers.matches AS m

WHERE m.hscore < m.vscore

GROUP BY hteam) AS t2

ON t1.team = t2.team

LEFT JOIN (

SELECT vteam AS team, COUNT(m.vteam) AS aloss

FROM sockers.matches AS m

WHERE m.vscore < m.hscore

GROUP BY vteam) AS t3

ON t1.team = t3.team;

CREATE VIEW sockers.totalpoints(team, points) AS

SELECT l.team AS teamname, (l.losses\*0 + t.ties\*1 + h.hwins\*2 + a.awins\*3) AS points

FROM sockers.losses AS l, sockers.ties AS t, sockers.homewins AS h, sockers.awaywins AS a

WHERE l.team = t.team AND t.team = h.team AND h.team = a.team;

SELECT \*

FROM sockers.totalpoints

ORDER BY points DESC;

**/\* 3. Return the names of undefeated coaches (that is, coaches whose teams have lost no match). Output a table with a single column called "coach" \*/**

SELECT t.coach AS coach

FROM sockers.teams AS t

WHERE t.name IN (

SELECT l.team

FROM sockers.losses AS l

WHERE l.losses = 0);

**/\* 4. Return the teams defeated only by the scoreboard leaders (ie. "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". \*/**

CREATE VIEW sockers.topteams(team, points, rnk) AS

SELECT \*, RANK() OVER (ORDER BY points DESC) AS rnk

FROM sockers.totalpoints;

SELECT t.name AS name

FROM sockers.teams AS t

WHERE t.name NOT IN (

SELECT tt.name

FROM sockers.teams AS tt, sockers.matches AS m

WHERE tt.name = m.hteam

AND m.vscore > m.hscore

AND m.vteam NOT IN (

SELECT tt.team

FROM sockers.topteams AS tt

WHERE rnk <= 1)

UNION

SELECT tt.name

FROM sockers.teams AS tt, sockers.matches AS m

WHERE tt.name = m.vteam

AND m.hscore > m.vscore

AND m.hteam NOT IN (

SELECT tt.team

FROM sockers.topteams AS tt

WHERE rnk <= 1));

**/\* 5. For each query in Problems (i) through (iv), create useful indexes or explain why there are none. \*/**

CREATE INDEX matches\_hscore\_id ON sockers.matches(hscore);

CREATE INDEX matches\_vscore\_id ON sockers.matches(vscore);

/\* The above indexes can be used for queries i-iv., since all the queries use vscore and hscore via the ‘Where’ clause. Still, the database will probably not become large enough for these indexes to influence querying performance. Unless the data size is large enough, no indexes should be used but if the data size is large the indexes above should be used. \*/

**/\* 6. 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). \*/**

CREATE TRIGGER teams\_trigger

AFTER INSERT ON teams

FOR EACH ROW

BEGIN

INSERT INTO scoreboard VALUES (:new.name, 0);

END;

CREATE TRIGGER matches\_trigger

AFTER INSERT ON matches

FOR EACH ROW

BEGIN

IF :new.hscore < :new.vscore

THEN

UPDATE scoreboard

SET points = points + 3

WHERE name = :new.vteam;

ELSEIF :new.hscore > :new.vscore

THEN

UPDATE scoreboard

SET points = points + 2

WHERE name = :new.hteam;

ELSE

UPDATE scoreboard

SET points = points + 1

WHERE name = :new.vteam;

UPDATE scoreboard

SET points = points + 1

WHERE name = :new.hteam;

END IF;

END;