CS 3380 Lab Assignment 10

Directions : This assignment must be submitted via Blackboard by **November 16 at 11:59 PM**. Your uploaded file should be named `lab10.sql`. **Do not** host your file in your public_html directory on babbage.cs.missouri.edu

This lab assignment deals with standings in the group stage for FIFA soccer tournaments such as the World Cup. Even if you know nothing about soccer, the ideas are simple and involve very basic math. A team earns three points in the group standings for a win, one point for a draw (a.k.a. a tie) and zero points for a loss. The total points earned by a team can be calculated as

$$3 \cdot wins + draws \tag{1}$$

With that as our introduction, complete the following steps.

1. Create a file called `lab10.sql` using your favorite editor (e.g. emacs, vim, nano, etc.) on babbage. All of the following steps involve adding SQL commands to this newly created file.

2. Write a SQL statement that drops the `lab10` schema if it exists.

3. Then write a SQL statement that creates a schema named `lab10`.

4. Add a statement to your file that sets the current search path to be `lab10`. This ensures that all tables and functions created below will exist within the `lab10` schema.

5. Write a `CREATE TABLE` statement to create a table named `group_standings` that matches the definition that follows. Be sure to include the `PRIMARY KEY` for your table and any NOT NULL constraints. Also, include `CHECK` constraints that enforce the range of possible values.

```
       Table "lab10.group_standings"
 Column |         Type          | Modifiers
--------+-----------------------+-----------
 team   | character varying(25) | not null
 wins   | smallint              | not null
 losses | smallint              | not null
 draws  | smallint              | not null
 points | smallint              | not null
Indexes:
    "group_standings_pkey" PRIMARY KEY, btree (team)
Check constraints:
    "group_standings_draws_check" CHECK (draws >= 0)
    "group_standings_losses_check" CHECK (losses >= 0)
    "group_standings_points_check" CHECK (points >= 0)
    "group_standings_wins_check" CHECK (wins >= 0)
```

6. Next, write a command that uses the psql `\copy` command to import data from the file found at `/facstaff/klaricm/public_cs3380/lab10/lab10_data.csv` After you load the data, there should be 4 records in your table.

7. Now, write a pure SQL (i.e. not a PL/pgSQL function) function named `calc_points_total` that takes two arguments that correspond to the number of wins and draws earned by a team. This function should return the total number of points earned based on the formula in Equation 1 above.

8. Create a PL/pgSQL function named `update_points_total` that is a trigger. This function should update the NEW record's points field using the `calc_points_total` function before any INSERT or UPDATE statement. Attach this function to the table as a trigger named `tr_update_points_total`. Test this trigger with a few UPDATE and/or INSERT statements. (You don't need to include these INSERT/UPDATE statements in your submission.)

9. Next, write a trigger function named `disallow_team_name_update` that compares the OLD and NEW records team fields. If they are different raise an exception that states that changing the team name is not allowed.

10. Then, attach this trigger to the table with the name `tr_disallow_team_name_update` and specify that it fires before any potential update of the team field in the table. Test this trigger with a few UPDATE statements to prove that it works. (You don't need to include these UPDATE statements in your submission.)

11. **Extra Credit:** Use the `ALTER TABLE` command to add a field to the table called `rank` that is of type smallint. We'll use this field to store a ranking of the teams. The team with the highest points value will be ranked number 1; the team with the second highest points value will be ranked number 2; etc. Write a PL/pgSQL function named `update_rank` that updates the `rank` field to contain the appropriate number for all teams. (There are both simple and complicated ways of doing this. Think about how it can be done with very little code.) Then, define a trigger named `tr_update_rank` that fires after an insert or update of any of the fields {`wins, draws`}. This trigger should be executed once per statement (not per row).