1 Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Piazza (under Resources->General Resources). If you did Lab1, you should already know how to log in to the class PostgreSQL server. You'll get help on Lab2 in your Lab Section, not the Lectures, so *be sure to attend Lab Sections*.

2 Goal

The goal of the second assignment is to create a PostgreSQL data schema with 5 tables that are very similar to the tables that you created in Lab1. The tables have the same names, attributes and data types as the tables of Lab1, and the same primary keys but there are some UNIQUE constraints and some restrictions on NULLs.

After you create the data schema with the 5 tables, you will be required to write some SQL statements that use those tables. Under Resources→Lab2, we have provided you with data that you can load into your tables so that you can test the results of your queries. Testing can prove that a query is wrong, but not that it is right, so be careful. We will not give you with the results of these queries on the load data; you should be able to figure out the results on that data yourselves. You can also test your queries on your own data. In the "real world", you have to make and check your own tests.

Lab2 is due in two weeks, so you will have an opportunity to discuss the assignment during the Discussion Section in the first week of the assignment, and to discuss issues you have had in writing a solution to the assignment during the Discussion Section of the second week. Instructions for submitting the assignment appear at the end of this document.

3 Lab2 Description

3.1 Create PostgreSQL Schema Lab2

You will create a Lab2 schema to set apart the database tables created in this lab from ones you will create in future, as well as from tables (and other objects) in the default (public) schema. Note that the meaning of schema here is specific to PostgreSQL and different from the general meaning. See here for more details on PostgreSQL schemas. You create the Lab2 schema like this:

```
CREATE SCHEMA Lab2;
```

Now that you have created the schema, you want to set Lab2 to be your default schema when you use psql. If you do not set Lab2 as the default schema, then you will have to qualify your table names with the schema name (e.g. Lab2.Customers). To set the default schema, you modify your search path. (For more details, see here.)

```
ALTER ROLE username SET SEARCH_PATH to Lab2;
```

You will need to log out and log back in to the server for this default schema change to take effect. (Students often forget to do this.)

You do not have to include the CREATE SCHEMA or ALTER ROLE statements in your solution.

3.2 Create tables

You will create tables in schema Lab2 for the tables Persons, Teams, Players, Games and GamePlayers. The attributes of the 5 tables are the same as the tables of Lab1. Data types for the other attributes in these tables are the same as the ones specified for the tables of Lab1, and the Primary Keys and other Foreign Keys are also the same. You should use the create.sql file that was our solution to Lab1 (which we provide on Piazza for Lab2) as the basis for your create.sql file for Lab2. However, the tables must have additional constraints which are described in the next section.

3.2.1 Constraints

The following attributes cannot be NULL. All other attributes can be (but remember that attributes in Primary Keys also cannot be NULL).

- In Persons: name
- In Players: joinDate
- In Teams: teamColor

Also, the following must be unique for the specified table. That is, there cannot be identical rows in that table that have exactly the same (non-NULL) values for all of those attributes (composite unique constraint).

- In Teams: the attribute teamCity
- In Games: the 2 attributes gameDate and homeTeam
- In Games: the 2 attributes gameDate and visitorTeam

For example, the first constraint says that there can't be two rows in Teams that have the same teamCity. And the second constraint says that there can't be two rows in Games that have the same values for both gameDate and homeTeam (if both gameDate and homeTeam are not NULL). Think of this as saying that there can't be two games that are on the same gameDate that have the same homeTeam.

You will write a CREATE TABLE command for each of the 5 tables that has these additional constraints. Save the commands in the file *create.sql*

4 SQL Queries

Below are English descriptions of the five SQL queries that you need to write for this assignment, which you will include in files queryX.sql, where X is the number of the query, e.g., your SQL statement for Query 1 will be in the file query1.sql, and so forth. Follow the directions as given. You will lose points if you give extra tuples or attributes in your results, if you give attributes in with the wrong names or in the wrong order, or if you have missing or wrong results. You will also lose points if your queries are unnecessarily complex, even if they are correct. Grading is based on correctness of queries on all data, not just the load data that we have provided.

Remember the Referential Integrity constraints from Lab1, which should be retained for Lab2. For example, if a homeTeam appears in a Games tuple, then there must be a tuple in Teams that has that homeTeam value as its teamID value..

Attributes should have their original names unless an alias is requested. And if a query asks that several attributes appear in the result, the first attribute mentioned should appear first, the second attribute mentioned should appear second, etc.

4.1 Query 1

A person is a coach if there is a team in Teams whose coachID is that person's personID. A person is a player if there is a player in Players whose playerID is that person's personID.

Find the name, address and salary of every person who is both a coach and a player. (Note that a person who is both a coach and a player could coach any team, not just the team that they play on.)

No duplicates should appear in your result.

(You should not use the canBePlayer or canBeCoach attributes of Persons in Query 1.)

4.2 Query 2

The team with the highest score in a game is the winner of that game. Find the names of the coaches whose teams have won at least one game against a team whose teamColor is red.

No duplicates should appear in your result.

4.3 Query 3

We'll say that a particular player has played in a particular game if there's a tuple in the GamePlayers table for that player and that game. (Doesn't matter what the value of minutesPlayed is.)

Output the personID, name and salary for every person who can be a player (canbePlayer), but who has never played in any game. The tuples in your result should appear in decreasing order of name (reverse alphabetical order), and when two tuples have the same name, in increasing salary order.

No duplicates should appear in your result.

4.4 Query 4

A team wins a game against another team if it had a higher number of points in that game than the other team. Sometimes the homeTeam (which is a teamID) wins the game, and sometimes the visitingTeam (which is also a teamID) wins the game. (There are no tie games.)

Find each pair of teamID's team1 and team2 (and their cities) such that:

- team1 wins against team2 in at least one game in which team1 was the home team,
- but team2 never wins a game against team1 in which team2 was the home team.

The attributes in your result should be called team1, team1City, team2 and team2City, in that order. No duplicates should appear in your result.

4.5 Query 5

Find the name and joinDate of the players for whom all of the following are true:

- The player plays on a team whose teamCity is not New York.
- The player joined their team before January 31, 2019.
- The player's salary is more than 78765.43
- The player is a starter (use isStarter).
- The player's name has the string 'er' (lowercase) appearing somewhere within their name as consecutive letters.

No duplicates should appear in your result.

5 Testing

While your solution is still a work in progress, it is a good idea to drop all objects from the database every time you run the script, so you can start fresh. Of course, dropping each object may be tedious, and sometimes there may be a particular order in which objects must be dropped. The following commands (which you can put at the top of create.sql if you want, but you don't have to), will drop your Lab2 schema (and all objects within it), and then create the (empty) schema again:

```
DROP SCHEMA Lab2 CASCADE;
CREATE SCHEMA Lab2;
```

Before you submit, login to your database via psql and execute your script. As you've learned already, the command to execute a script is: \i <filename>.

Under Resources→Lab2 on Piazza, we have provided a load script named *lab2_load_data.sql* that loads data into the 5 tables of the database. You can execute that script with the command:

\i *lab2_load_data.sql*.

You can test your 5 queries using that data, but you will have to figure out on your own whether your query results are correct. We won't provide answers, and students should not share answers with other students. Also, your queries must be correct on any database instance, not just on the data that we provide. You may want to test your SQL statements on your own data as well.

6 Submitting

1. Save your scripts for table creations and query statements as create.sql and query1.sql through query5.sql You may add informative comments inside your scripts if you want (the server interprets lines that start with two hyphens as comment lines).

2. Zip the file(s) to a single file with name Lab2_XXXXXXX.zip where XXXXXXX is your 7-digit student ID. For example, if a student's ID is 1234567, then the file that this student submits for Lab2 should be named Lab2_1234567.zip

   To generate the zip file you can use the Unix command:

   zip Lab2_1234567 create.sql query1.sql query2.sql query3.sql query4.sql query5.sql

   (Of course, you use your own student ID, not 1234567.)

3. You should already know how to transfer the files from the UNIX timeshare to your local machine before submitting to Canvas. If you are still not familiar with the process, use the instructions we provided at the Lab1 assignment.

4. Lab2 is due by 11:59pm on Sunday, April 25. Late submissions will not be accepted, and there will be no make-up Lab assignments.

5. You can get always rid of duplicates by using DISTINCT in your SELECT. In CSE 182 (unlike CSE 180), we do not deduct points if students use DISTINCT and it wasn't necessary to use it. (Using DISTINCT is unnecessary in a query if the query couldn't have duplicates even without DISTINCT appearing.) However, we will also deduct if you were told not to eliminate duplicates but you did.

6. Be sure to follow directions about Academic Integrity that are in the Syllabus. If you have any questions about those directions, please speak to the instructor as soon as possible.