The assignment is to be done by yourself.

The goal of this assignment is to get you (re-)familiarized with relational databases, and specifically the standard query language for databases, SQL. We will use the open source PostgreSQL database system for this purpose. Installing the system on your machine will be an important part of the assignment.

Software installs often do not go smoothly; we strongly recommend you at least install and play with PostgreSQL well in advance. Generally speaking, we will likely not be able to help you with any installation issues, but sooner you contact us, better the chances.

1 Installing and Using PostgreSQL

The PostgreSQL website has binary packages and source code for installing it on a variety of platforms. You will be asked for the “superuser” password along the way during install – make sure to remember that. The current version of PostgreSQL is 9.0.2. You will find the detailed documentation at: http://www.postgresql.org/docs/9.0/static/index.html

You will be using PostgreSQL in a client-server mode. First you have to start the server in the background. Recall that the server is a continuously running process that listens on a specific port (the actual port would differ, and you can usually choose it when starting the server). In order to connect to the server, the client will need to know the port. The client and server are often on different machines, but for you, it may be easiest if they are on the same machine.

Using the “psql” client is the easiest – it provides a commandline access to the database. But there are other clients too. We will assume “psql” here.

After installing PostgreSQL, before starting the server, you must first “initialize” a database storage area on disk. This is the area where all the data files are stored (called “data directory” in PostgreSQL). This can be done using the “initdb” command, which takes in a data directory as the input. See initdb documentation for more details: http://www.postgresql.org/docs/current/static/app-initdb.html

After initializing the database storage area, you can start the server. There are different ways to do it depending on the installation and the platform. On UNIX platforms, the command:
    postgres -D /usr/local/pgsql/data
will start the server, with the data directory ”/usr/local/pgsql/data”. See the documentation for more details: http://www.postgresql.org/docs/9.0/static/server-start.html

Note: Both the above steps may be done during the install itself. E.g., on Macs, the 1-click installer asks for a location for the data directory (Default: /Library/PostgreSQL/9.0/data), initializes it, and starts the server.

After the server has started, you have to “create” a database. This is done using the “createdb” command. PostgreSQL automatically creates one database for its own purpose, called “postgres”. It is preferable you create a different database for your data. Here are more details on “createdb”: http://www.postgresql.org/docs/9.0/static/tutorial-createdb.html

For the assignment, you can create a database called “soccer” (you can name it something else if you want). The rest of the commands will assume the database is called “soccer”.

Important: PostgreSQL server has a default superuser called “postgres”. You can do everything under that username, or you can create a different username for yourself. If you run a command (say createdb) without any options, it uses
the same username that you are logged in under. However, if you haven’t created a PostgreSQL user with that name, the command will fail. You can either create a user (by logging in as the superuser), or run everything as a superuser (typically with the option: “-U postgres”).

Once the database is created, you can connect to it. There are many ways to connect to the server. The easiest is to use the commandline tool called “psql”. Start it by:

```bash
psql -U postgres soccer
```

The above command gives the username “postgres” to connect to the database. You may be prompted for a password. “psql” takes quite a few other options: you can specify different user, a specific port, another server etc. See documentation: [http://www.postgresql.org/docs/9.0/static/app-psql.html](http://www.postgresql.org/docs/9.0/static/app-psql.html)

Now you can start using the database.

- The psql program has a number of internal commands that are not SQL commands; such commands are often client and database specific. For psql, they begin with the backslash character, “\”. For example, you can get help on the syntax of various PostgreSQL SQL commands by typing: “\h”.
- “\d”: lists out the tables in the database.
- All commands like this can be found at: [http://www.postgresql.org/docs/9.0/static/app-psql.html](http://www.postgresql.org/docs/9.0/static/app-psql.html). “\?” will also list them out.
- To populate the database using the provided soccer dataset, use the following: “\i populate-soccer.sql”. For this to work, the populate-soccer.sql file must be in the same directory as the one where you started psql. This commands creates the tables, and inserts the tuples. We will discuss the schema of the dataset in the next section.

## 2 Soccer Dataset

The dataset contains the details of the 2006 Soccer World Cup. In particular, it contains the information about players, goalkeepers, teams, and the scores of the matches. The schema of the tables should be self-explanatory. The data was collected from the FIFA website: [http://fifaworldcup.yahoo.com/06/en/w/schedule.html](http://fifaworldcup.yahoo.com/06/en/w/schedule.html)

There are two stages in the world-cup. In the first stage, also called “group play”, teams from the same group play with each other. There are 8 groups, named “A” to “H”. The top two teams from each group advance to the Stage 2, which is a knock-out round. There might be ties in the first stage; there are no ties in the second stage.

## 3 SQL

Queries in psql must be terminated with a semicolon. After populating the database, you can test it by running simple queries like:

```sql
select * from goalkeepers;
```

We will not cover SQL in the class.

- The website for CMSC 424 ([https://sites.google.com/site/spring2010cmsc424/](https://sites.google.com/site/spring2010cmsc424/)) contains my slides for SQL, along with a set of sample queries.
- You can also look at an undergraduate textbook. Keep in mind the syntax of the commands can slightly vary, especially commands that use any advanced features.
There are numerous online resources. PostgreSQL manual is a good place for introduction to SQL. http://sqlzoo.net also has many examples.

Here are some example queries on the soccer dataset and the SQL for them.

- Report all ‘ties’ (matches where both teams scored identical number of goals).
  
  ```sql
  select * from Results where scoreone = scoretwo;
  ```

- Find the player who was on the field the longest.
  
  ```sql
  select * from Players where totalminutes =
  (select max(totalminutes) from Players);
  ```
  Note that using a nested "subquery" (which first finds the maximum value of the attribute "totalminutes") as above is the most compact way to write this query.

- Write a query to count the total number of matches that took place on June 21, 2006.
  
  ```sql
  select count(*) from Results where matchdate =
  to_date('06-21-2006', 'MM-DD-YYYY');
  ```
  There are many interesting and useful functions on the "date" datatype. See: http://www.postgresql.org/docs/9.0/interactive/functions-datetime.html

- Write a query to add a new column called “winner” to the Results table.
  
  ```sql
  alter table Results add winner char(3);
  ```

- Initially the “winner” column for all matches would be listed as empty. Write a query to update the “winner” column to be the winner of each match. Set it to ‘TIE’ if the match ended in a tie.
  
  ```sql
  update Results set winner =
  case
  when scoreone = scoretwo then 'TIE'
  when scoreone > scoretwo then teamone
  when scoreone < scoretwo then teamtwo
  end;
  ```
  This uses the "case" construct.

- (WITH) In many cases you might find it easier to create temporary tables, especially for queries involving finding "max" or "min". This also allows you to break down the full query and makes it easier to debug. It is preferable to use the WITH construct for this purpose. The syntax appears to differ across systems, but here is the link to PostgreSQL: http://www.postgresql.org/docs/9.0/static/queries-with.html

- (LIMIT) PostgreSQL allows you to limit the number of results displayed which is useful for debugging etc. Here is an example:
  
  ```sql
  select * from Goalkeepers limit 5;
  ```

4 Assignment Submission

For each of the queries below, submit the query to do the task as well as the query result. Please use the provided text file for writing your submissions. The text file should be self-explanatory. The submission will be through submit server.

1. Figure out how to find out all the attributes and their types for a given table in “psql”, and write down the statement and its result on the Teams table. This kind of a statement is usually unique to the client, and is not part of standard SQL.
2. Count the total number of teams that used only one goalkeeper throughout the tournament.

3. Find the “best” goalkeeper, defined to be the goalkeeper with the highest saves to goals-against ratio, with at least 10 saves.

4. Write a query to find out if any teams that were in the same “group” met during the postseason. Use the fact that the group play ended on June 23, 2006.

5. List the top 10 players with the highest total minutes played, along with their rank. The output would have schema: (name, total minutes, rank). The player with the highest total minutes will have rank = 1, player with next highest will have rank = 2 etc. If there is a tie, then both players must get the same rank, and the next rank must be skipped. E.g., if two players tie for the highest total minutes, then they both get rank 1, and there is no player with rank 2.

6. The results table is not in a nice format to answer many queries (such as who won or who lost). Write a query to create a table: WinLossKO(winner, loser, matchdate) using the Results table, for matches that took place in Stage 2 (the “knockout round”). The knockout round started on June 24, 2006.

7. Write a trigger to keep the WinLossKO table updated when a new entry is added to the Results table. Remember that the new tuple added to the Result table may or may not be from the knockout round, so you must check for that (i.e., you shouldn’t add those tuples to the WinLossKO table). See here for detailed trigger documentation: http://www.postgresql.org/docs/9.0/static/plpgsql-trigger.html
   Database systems tend to be very picky about the trigger syntax, so be careful.

8. Write a query to find the number of points for each team during the “group play”. A team gets 3 points for a win, 1 point for a tie, and none for a loss. The output should be simply a list of team names, along with their points. Note that this is to be computed only for matches played on or before June 23, 2006. Verify against the webpage to confirm your answers. http://fifaworldcup.yahoo.com/06/en/w/group/index.html
   This is a somewhat harder problem, and you should use temporary tables (i.e., WITH) liberally.

9. Some tasks are hard to write as SQL queries. For example, consider the above task of computing the total points in the group play, and (taking it further) finding the top 2 in each group and printing out the first set of knockout round matches.
   However, this is much easier to do in an embedded language like PL/pgSQL. Documentation for this is at: http://www.postgresql.org/docs/9.0/interactive/plpgsql.html
   You are to write a procedure in PL/pgSQL to do the above task. Specifically, the output should be the list of countries, and the number of points they accumulated during group play.
   Here is an example “anonymous” function (that is executed directly in psql):

```sql
DO language plpgsql $$
DECLARE
vr record;
BEGIN

FOR vr IN SELECT name FROM goalkeepers LOOP
   RAISE NOTICE 'Name is %', vr.name;
END LOOP;

END $$;
```

Note the "$$": they are used as string delimiters (easier to use than using quotes). The “DO” at the beginning makes this be executed immediately. You can also create a function instead.