# HW-3

36-650 – Statistical Computing

Deadline: October 3rd, 11:59PM EST

You must submit **your own** HW as series of SQL scripts (and screenshots) stored in a folder on your GitHub repository.

Your GitHub repository should contain a folder named HW-3. Inside this folder, there should be a folder for each question.

On GitHub, Check-in the answer for each question in a separate SQL file in the corresponding folder. Name the SQL file with the question number (e.g. q1.sql)

If you were asked to submit a copy of the output, take a screenshot and check in the screenshot into your GitHub repository and name it with the question number (e.g. q7.png)

*If the output has many rows, it is sufficient to take a screenshot of the first five, unless otherwise instructed.*

Check in all your scripts and screenshots under a folder named HW2 in your GitHub repository. Make sure your GitHub repository is publicly viewable.

On Canvas, only submit a URL to your GitHub repository.

Not following the submission guidelines may result in grading penalties.

---

In this homework, you will be working with National Basketball Association (NBA) player data. In the `Modules` section on Canvas, under `DATA`, you will find two files: `README` and `player-stats.sql`. The former defines the table columns that appear in the latter. It is provided for completeness; you should not have to refer to it.

---

## Question 1

*(10 points)*

Download `player-stats.sql` and read it into your `postgres` session using the `\i` command. Show that you have changed the working directory inside `postgres` to where your downloaded data are stored. As a final step, display the current list of tables in your database.

```
FILL ME IN
```

## Question 2

*(10 points)*

Use an aggregate function from Notes 13B to determine how many unique colleges there are among the 476 players in the database. (You might want to look up the documentation for `distinct`, which is `SQL`'s analogue to `R`'s `unique()`.)

```
FILL ME IN
```

## Question 3

*(10 points)*

Create a table called `teams`, which will have five columns: an `id` column of type `char(3)` that will be the table's primary key; `location` and `name` columns of type `text` (e.g., Boston for location, Celtics for name); a column showing the `division` in which the team plays (e.g., Atlantic); and a column showing the `conference` the team plays in (e.g., Eastern). Note that `division` and `conference` should be of type `DivisionType` and `ConferenceType` respectively; these types should be created as enumerated variables. (See the `postgres` documentation for `CREATE TYPE` and look specifically for the variant that contains the word `ENUM`; essentially, you are defining your own factor variables, with defined levels.)

```
FILL ME IN
```

---

Run the following code in your `postgres()` session to populate the `teams` table:

```
insert into teams VALUES
  ('BOS', 'Boston', 'Celtics', 'Atlantic', 'Eastern'),
  ('BKN', 'Brooklyn', 'Nets', 'Atlantic', 'Eastern'),
  ('NYK', 'New York', 'Knicks', 'Atlantic', 'Eastern'),
  ('PHI', 'Philadelphia', '76ers', 'Atlantic', 'Eastern'),
  ('TOR', 'Toronto', 'Raptors', 'Atlantic', 'Eastern'),
  ('CHI', 'Chicago', 'Bulls', 'Central', 'Eastern'),
  ('CLE', 'Cleveland', 'Cavaliers', 'Central', 'Eastern'),
  ('DET', 'Detroit', 'Pistons', 'Central', 'Eastern'),
  ('IND', 'Indiana', 'Pacers', 'Central', 'Eastern'),
  ('MIL', 'Milwaukee', 'Bucks', 'Central', 'Eastern'),
  ('ATL', 'Atlanta', 'Hawks', 'Southeast', 'Eastern'),
  ('CHA', 'Charlotte', 'Bobcats', 'Southeast', 'Eastern'),
  ('MIA', 'Miami', 'Heat', 'Southeast', 'Eastern'),
  ('ORL', 'Orlando', 'Magic', 'Southeast', 'Eastern'),
  ('WAS', 'Washington', 'Wizards', 'Southeast', 'Eastern'),
  ('DEN', 'Denver', 'Nuggets', 'Northwest', 'Western'),
  ('MIN', 'Minnesota', 'Timberwolves', 'Northwest', 'Western'),
  ('OKC', 'Oklahoma City', 'Thunder', 'Northwest', 'Western'),
  ('POR', 'Portland', 'Trail Blazers', 'Northwest', 'Western'),
  ('UTA', 'Utah', 'Jazz', 'Northwest', 'Western'),
  ('GSW', 'Golden State', 'Warriors', 'Pacific', 'Western'),
  ('LAC', 'Los Angeles', 'Clippers', 'Pacific', 'Western'),
  ('LAL', 'Los Angeles', 'Lakers', 'Pacific', 'Western'),
  ('PHX', 'Phoenix', 'Suns', 'Pacific', 'Western'),
```

```
('SAC', 'Sacramento', 'Kings', 'Pacific', 'Western'),
('DAL', 'Dallas', 'Mavericks', 'Southwest', 'Western'),
('HOU', 'Houston', 'Rockets', 'Southwest', 'Western'),
('MEM', 'Memphis', 'Grizzlies', 'Southwest', 'Western'),
('NOP', 'New Orleans', 'Hornets', 'Southwest', 'Western'),
('SAS', 'San Antonio', 'Spurs', 'Southwest', 'Western');
```

## Question 4

*(20 points)*

None of the available datasets list player positions. In the NBA, there are five commonly acknowledged positions: center, power forward, small forward, shooting guard, and point guard. One can map information in the `more_player_stats` table to these positions via the empirical formula

```
prl = per - 67*va/(gp*minutes)
```

where ranges of `prl` map directly to positions (see below). Create a new table (just call it `new_table`) that has three columns: `player`, of type `integer references more_player_stats` (it's a foreign key, something we haven't explicitly mentioned yet, but is useful for joining tables as we will see next week); `prl`, of type `numeric`; and `position`, of type `text`. Use `insert` to populate this table: you'd `select` the `id` from `more_player_stats`, and you'd select the equation above. That sounds a bit weird. Let's say table `foo` has three columns, `id`, `x`, and `y`. Now you create `bar`, that has `new_id` and `z`, which is defined as `x/y`.

- `insert into bar (new_id,z) (select id,round(x/y,1) from foo);`

(Here I decided to round off the quotient. You should do the same with `prl`.) The last thing to do is to `update` your new table by setting the position on the basis of the following criteria: `'PF'` where `prl` is greater than or equal to 11.3; `'PG'` where `prl` is [10.8,11.3); `'C'` for [10.6,10.8); and `'SG/SF'` for [0,10.6). Display the first 10 rows of your new table.

```
FILL ME IN
```

## Question 5

*(20 points)*

Take the `position` column you've just defined and add it to the `player_bios` table. (Remember: `alter table` and `update`.)

You'll take advantage of the fact that `player` in `new_table` references `id` in `more_player_stats`... which means it references `id` in `player_bios` as well. After you are done, display the first five rows of `player_bios` (selecting just the `firstname`, `lastname`, and `position` columns).

```
FILL ME IN
```

## Question 6

*(20 points)*

Now we will convert the heights given in `player_bios` from feet-inches format to simply inches. This involves `alter`ing the `player_bios` table to add a new column of type `numeric`, then using `update` to set the values of the height in inches. You may wish to use the following:

```
12*split_part(height,'-',1)::integer + split_part(height,'-',2)::integer
```

This splits the height string on '-'; the first output is feet, so we multiply that by 12 (and cast to `integer`), and the second output is inches, which is simply cast to `integer`. Finally, use two `alter` commands to drop the old height column and to rename your new column 'height'. As you did in the last question, display the first five rows of `player_bios` (but this time: `firstname`, `lastname`, and `height`).

```
FILL ME IN
```