As per usual, I have provided a markdown template file for each problem on this assignment. The first problem will be involving the BASH shell, while the second two problems will deal with joins and self-joins. For the second two problems, running the provided `HW4.sql` file in the repository will create a handful of new tables in a *hw4* schema, which will show up to the left (separate from the *public* schema). As we get to using more and more tables, I think placing them in a container like this will help you track them and keep things organized. **The price you pay, though, is that when you refer to a table name not in the public schema, it must start with the schema name.** For instance, the superhero's table would be referred to as `hw4.superheros`. Given that you will likely be aliasing the table names anyway, this seems like a small price to pay for not having tables just scattered all throughout your database. Let me know if this is causing you issues.

In order to accept the assignment and get access to the repository, you should follow the link here:

Assignment link: https://classroom.github.com/a/UqZohHSB

1. (4 points) You will need a BASH or ZSH shell for this problem. On Unix type systems this should be simple, but in Windows you'll need to install either WSL2 to get a Unix shell or Git for Windows which comes with Git BASH which will work for everything we need.

   The file `Names.txt` in the repository has 1000 first names within it, one name per line. Write a BASH shell command which would determine the number of duplicated names (those that show up more than once) within a file and print that number (and only that number) to the shell. In addition to the programs introduced in class, you might find the BASH programs `uniq` and `sort` useful, and want to consult their documentation for proper use. Piping information from program to program will also be useful. If you are trying to check your command against the actual `Names.txt`, then you should get 197.

2. (10 points) Running the provided `HW4.sql` script in the repository will have created a selection of tables all related to superheroes. I'll include an image of the relationships between the tables in the repository, but you can also navigate here to see an interactive version. A description of the tables is below.

Table name	Description
`superheros`	The main list of superheros. Includes their superhero name, full/real name, a list of IDs linking to other tables, and their height (in centimeters) and weight (in kilograms).
`sexes`	The sex of the superhero: Male, Female, or N/A
`colors`	Contains references for eye color, skin color, and hair color.
`races`	Shows values for all the different races of superheros, such as Human.
`publishers`	List of all the publishers.
`alignments`	Shows three values to indicate how the superhero is aligned: Good, Neutral, or Bad.
`attributes`	Lists 6 different attributes (such as intelligence) that can describe a superhero.
`hero_attributes`	This table is an intersection of attributes and superheros. A superhero can have multiple attributes, and for each attribute, they can have a value from 0-100 indicating their rating. This table contains those ratings in the `attribute_value` column.
`superpowers`	List all the available abilities or superpowers that someone can have.
`hero_powers`	Lists the IDs of each power that each superhero can have, since a superhero can have more than one superpower.

Use these tables to answer the following questions:

(a) What percentage of bad or evil superheros have (purely) red eyes?

(b) What is the average intelligence of (purely) human superheros?

(c) Who is the heaviest superhero that can fly?

(d) What is the most common superpower for superheros with blond hair?

(e) How many unique combinations of race and sex have no corresponding superheros?

3. (8 points) Running the provided `HW4.sql` script in the repository will have created a `family_tree` table in your database (also in the hw4 schema). This table includes information from several generations of simulated families, including information about marriages and children. A short description of the columns is below:

Column	Description
`pid`	Unique personal identification number of an individual
`name`	The given name of the individual
`spouse_id`	The pid of this individuals spouse. To prevent duplication, this number is only assigned to the individual that marries *into* the family.
`parent1_id`	The pid of this individual's first parent. These are not provided for individuals marrying into the family.
`parent2_id`	The pid of this individual's second parent. These are not provided for individuals marrying into the family.
`yr_birth`	The year this individual was born.
`yr_death`	The year this individual died. Is `NULL` if they are still alive.
`yr_married`	The year this individual married. Is `NULL` if they never married.
`sex`	`M` for male or `F` for female.

Use this table to answer the following questions:

(a) How many married couples are represented in the data?

(b) What names were duplicated throughout the family tree, where individuals were given the same name despite being different individuals?

(c) What was the greatest age difference between partners at the time of their marriage?

(d) What is the youngest age at which someone became a grandparent?