Assignment #6 - Single Table, Importing Data, SQL

👀 Click on this link: https://classroom.github.com/a/5v6uVzcd (https://classroom.github.com/a/5v6uVzcd) to accept this assignment in GitHub classroom. This will create your homework repository. Clone your new repository.

In this homework, you'll:

1. install postgres (use `brew`, or on windows, download the installer from postgresql.org (https://www.postgresql.org/download/)
2. convert data from homework 3 into a csv file - you should already have a Python script to do this from your previous work! (`name_your_file.csv`)
3. design a table based on your data (`create_table.sql`)
4. import your csv into your database (`import_csv.sql`)
5. run a few simple queries (`queries.sql`)
6. **write about the results of your queries (`report.md`)**

Reuse or generate a csv File

1. copy over your data set (the data set of your choosing, not an assigned one) from any of your previous homeworks
   - if your data did not come in a csv or if you'd like to do some preprocessing before you import it into a database, it's ok to use `pandas` (for example to_csv (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_csv.html)) to do so
   - save any script or notebook that you used to do this within your repository as `convert.py` or `convert.ipynb`
   - save the result (the new csv) of running your script or notebook in your repository - choose a descriptive name for the resulting `.csv` `(` students.csv `,` dog_bites.csv`)
   - the csv should be contain your original data set or a cleaned/transformed data set
   - if the resulting csv is over 100mb, use a subset of the data
   - finally, it is possible to import JSON objects (one object perf line) into a `jsonb` field (but you'd have to consult the documentation for dealing with these objects in queries)
2. **in your `report.md`, under a heading called `# Overview`, once again, copy over your documentation citing the source of the data and what it contains based on the previous homeworks (columns, expected types, etc.)**

Design and Create a Table

1. based on the your csv, design a table to import your data into (see the slides on data types and creating a table)
   - determine what field should be a primary key (unique, something that will not be changed)
   - if no field is a good candidate for a primary key, use a `serial` numeric type (sequence) as a surrogate / artificial primary key
   - think of the appropriate data type for all of the other fields
   - consider whether or not you'd like to allow null values, duplicates, or set default values
2. **in `report.md`, under a heading called `# Table Design`, write up your decisions for each field** (for example, you can say that you have allowed null values for a particular field because there is missing data in your csv, or if there's missing data ... you'll fill with a default value)
3. write the *actual* SQL to make this table, and put it into your `create_table.sql`
   - prior to the `CREATE TABLE` statement, make sure to try to `DROP` the table first, but only if it exists
   - make sure that the appropriate constraints are set (`NOT NULL`, `UNIQUE`, etc.)
   - use descriptive names for the table and columns
   - make sure all names are lowercase and separated with underscore
4. create and connect to a database called `homework06`
5. run the SQL in your `create_table.sql` file by using copying and pasting, using `\i`, or using psql dbname < `/path/to/create_table.sql`

Import Your .csv

1. Now for the moment of truth! Try importing your csv using the `COPY` command (check the slides for different options)2. If there are errors, **note the errors in your `report.md` and how you fixed them under a heading called `# Import`**
    - (if there are no errors, just note that the import succeeded without error)
    - you can continue to refine your create table sql file and your csv
    - but you can only do this (^^^^) as long as you **add notes about your refinements under `# Import`**
    - ...and as long as all the other related files are updated
    - (for example `create_table.sql` should contain the new `CREATE` statement if you had to change the way your table was made... or if you needed to clean up the csv a little more, you'd have to drop in the new csv)
    - best would be to commit to your repo after each modification
2. add the command that you used to do the import into the `import_csv.sql` file

Run Some Simple Queries

Run commands and queries on your newly imported data. For the assignments, you can test out your work in any client that you like, but for the purposes of recording what you've done please run (or re-run) everything in the commandline client.

First, run some commands to see what's in your table

1. show all of databases in your postgres instance
2. (making sure you're connected to `homework06` database) show all of the tables in your database
3. finally, show some information about the table you created and imported data into: find a way to show its columns, types and constraints (hint: describe the table)

Once you've run the commands for the above, copy and paste the results into your `report.md` file under a header called `# Database Information`.

Now, run some queries to determine / calculate / show some information

As you go through writing your queries:

- **add your query to `queries.sql` with a comment above each statement reflecting the number and information you are extracting**

```
-- 2. number and description from below
YOUR QUERY UNDERNEATH;
```

- **add the results of each query to your `report.md`**
    - **this should go under a heading called `# Query Results`**
    - remember to add the number and description of the results based on the specifications below
    - drop this into a markdown "code" block (surround with triple backticks)

```
```
2. number and description from below
Your | Results | Here
-----+---------+-----
foo | bar | baz
```
```

Write queries to do / show the following:

1. the total number of rows in the database
2. show the first 15 rows, but only display 3 columns (your choice)
3. do the same as above, but chose a column to sort on, and sort in descending order
4. add a new column without a default value

5. set the value of that new column
6. show only the unique (non duplicates) of a column of your choice
7. group rows together by a column value (your choice) and use an aggregate function to calculate *something* about that group (count of all members of the group, the average of a column of the members of the group)
    - show the column that is used for grouping AND the result of the aggregate function (so, 2 columns minimum on output)
8. now, using the same grouping query or creating another one, find a way to filter the query results based on the values for the groups (for example, show all genres that have more than 2 movies in it and only show the genre and the number of movies for that genre).
    - you'll use a `HAVING` clause to do this (covered in the slides on group by)

Finally, add another 4 queries of your own that helps you gain some insight on your data (ok to reproduce the work that you did previously for homework 1 or 2... but using sql instead of python)

Document each query (9 - 12) in your `report.md` and `queries.sql` file

Conclusion

At the end of the homework, you should have 5 files:

1. `name_your_file.csv`
2. `create_table.sql`
3. `import_csv.sql`
4. `queries.sql`
5. `report.md` (note that report should contain the following headers)
    - Overview
    - Table Design
    - Import
    - Database Information
    - Query Results