Periodic Table Database

Instructions

You are started with a `periodic_table` database that has information about some chemical elements. You can connect to it by entering `psql --username=freecodecamp --dbname=periodic_table` in the terminal. You may want to get a little familiar with the existing tables, columns, and rows. Read the instructions below and complete user stories to finish the project. Certain tests may not pass until other user stories are complete. Good luck!

Part 1: Fix the database

There are some mistakes in the database that need to be fixed or changed. See the user stories below for what to change.

Part 2: Create your git repository

You need to make a small bash program. The code needs to be version controlled with `git`, so you will need to turn the suggested folder into a git repository.

Part 3: Create the script

Lastly, you need to make a script that accepts an argument in the form of an `atomic number`, `symbol`, or `name` of an element and outputs some information about the given element.

Notes:
If you leave your virtual machine, your database may not be saved. You can make a dump of it by entering `pg_dump -cC --inserts -U freecodecamp periodic_table > periodic_table.sql` in a bash terminal (not the psql one). It will save the commands to rebuild your database in `periodic_table.sql`. The file will be located where the command was entered. If it's anything inside the `project` folder, the file will be saved in the VM. You can rebuild the database by entering `psql -U postgres < periodic_table.sql` in a terminal where the `.sql` file is.

If you are saving your progress on freeCodeCamp.org, after getting all the tests to pass, follow the instructions above to save a dump of your database. Save the `periodic_table.sql` file, as well as the final version of your `element.sh` file, in a public repository and submit the URL to it on freeCodeCamp.org.

Complete the tasks below

- You should rename the `weight` column to `atomic_mass`
- You should rename the `melting_point` column to `melting_point_celsius` and the `boiling_point` column to `boiling_point_celsius`
- Your `melting_point_celsius` and `boiling_point_celsius` columns should not accept null values
- You should add the `UNIQUE` constraint to the `symbol` and `name` columns from the `elements` table
- Your `symbol` and `name` columns should have the `NOT NULL` constraint
- You should set the `atomic_number` column from the `properties` table as a foreign key that references the column of the same name in the `elements` table
- You should create a `types` table that will store the three types of elements

- Your `types` table should have a `type_id` column that is an integer and the primary key
- Your `types` table should have a `type` column that's a `VARCHAR` and cannot be `null`. It will store the different types from the `type` column in the `properties` table
- You should add three rows to your `types` table whose values are the three different types from the `properties` table
- Your `properties` table should have a `type_id` foreign key column that references the `type_id` column from the `types` table. It should be an `INT` with the `NOT NULL` constraint
- Each row in your `properties` table should have a `type_id` value that links to the correct type from the `types` table
- You should capitalize the first letter of all the `symbol` values in the `elements` table. Be careful to only capitalize the letter and not change any others
- You should remove all the trailing zeros after the decimals from each row of the `atomic_mass` column. You may need to adjust a data type to `DECIMAL` for this. Be careful not to change the value
- You should add the element with atomic number `9` to your database. Its name is `Fluorine`, symbol is `F`, mass is `18.998`, melting point is `-220`, boiling point is `-188.1`, and it's a `nonmetal`
- You should add the element with atomic number `10` to your database. Its name is `Neon`, symbol is `Ne`, mass is `20.18`, melting point is `-248.6`, boiling point is `-246.1`, and it's a `nonmetal`
- You should create a `periodic_table` folder in the `project` folder and turn it into a git repository with `git init`
- Your repository should have a `main` branch with all your commits
- Your `periodic_table` repo should have at least five commits
- You should create an `element.sh` file in your repo folder for the program I want you to make
- Your script (`.sh`) file should have executable permissions
- If you run `./element.sh`, it should output `Please provide an element as an argument` and finish running.
- If you run `./element.sh 1`, `./element.sh H`, or `./element.sh Hydrogen`, it should output `The element with atomic number 1 is Hydrogen (H). It's a nonmetal, with a mass of 1.008 amu. Hydrogen has a melting point of -259.1 celsius and a boiling point of -252.9 celsius.`
- If you run `./element.sh` script with another element as input, you should get the same output but with information associated with the given element.
- If the argument input to your `element.sh` script doesn't exist as an `atomic_number`, `symbol`, or `name` in the database, the output should be `I could not find that element in the database.`
- The message for the first commit in your repo should be `Initial commit`
- The rest of the commit messages should start with `fix:`, `feat:`, `refactor:`, `chore:`, or `test:`
- You should delete the non existent element, whose `atomic_number` is `1000`, from the two tables
- Your `properties` table should not have a `type` column
- You should finish your project while on the `main` branch. Your working tree should be clean and you should not have any uncommitted changes