CSC 103 - HW 10

Complete each of the following exercises. To receive full credit please ensure the following:

- Submission structure follows the following:

```
HW10
    -> Q1
       -> Q1.sql
       -> Q1.py
       -> Q1.txt
    -> Q2
       -> Q2.sql
       -> Q2.db
       -> Q3.db
```

- Not every exercise may have each type of file extension as shown above, but each artifact (sql file, python file, etc.) will need to be gathered underneath that problem. There will be no shared `data` directory like we've had in past weeks.
- Exercise artifacts are uploaded to your GitHub repository under a folder called `HW10` (i.e. https://github.com/<username>/carlow-me)
- Exercise artifacts are uploaded to Brightspace as a compressed `.zip` file called `<LastName>_HW10.zip`.

It is crucial that you are naming artifacts and storing files in the format mentioned above. All files that need to reference datasets should use a relative reference that refers to it's location in the `data` directory and not an absolute reference that would only work on your machine. Failure to meet any of these requirements will result in loss of points for that and any following problems.

Exercises

1. Create the SQL to create a table called `pokemon`. The table should have 5 columns: id, name, height, weight, base_experience. The id field is the unique identifier for this table and we should ensure that the name field is not null and is unique. Each field should us an appropriate data type for the data it contains.

2. Assuming the table from Q1 has been created, write the python that would import the appropriate data from the `pokemon.csv` dataset into our database. A database file and a python file should be provided for this problem.
3. Assuming that all data has been imported into a file called `poke.db` (you can feel free to make a copy of the db file from Q2 and move it to your Q3 folder), write a query that returns all pokemon that have an ID between 10 and 35, including the pokemon with the id of `35`.
4. Utilizing the datasets `pokemon_types.csv` and `pokemon_type_names.csv`, create a single table called `pokemon_types` filled with data containing the following columns: id, name. The data that populates this table should only pull data from the `pokemon_type_names.csv` related to the `local_language_id` that is equal to 9. Use the database file from Q2 as the starting point for this problem so that at the end you have a table called `pokemon` that has been populated with our pokemon data and a table called `pokemon_types` that has been populated by the python code you write to populate the table.
5. Write a query that removes data from the pokemon table if the id is higher than 10 or if the weight is lower than 30. Use a copy of the database from Q4 as your starting point for this question and provide your query in a SQL file.
6. Starting fresh, provide the SQL that will create two tables `pokemon` and `pokemon_types`. They should have the same columns mentioned above, with similar constraints. However now the `pokemon` table should have a column named `type_id` and it should have a foreign key relationship with the `pokemon_types` table column `id`. You should pre-seed the `pokemon_types` table with a row with `id` of 999 that represents an `unknown` pokemon type. With that data pre-seeded, provide the code that will then insert that existing pokemon_types data as we did previously into the `pokemon_types` table.
7. Building from the previous problem (i.e. use a copy of the Q6.db as your starting Q7.db), write the python code to re-insert data from `pokemon.csv` into the `pokemon` table. You can take 1 of 2 approaches. If you'd prefer to create an initial table full of pokemon with unknown data types, you can insert data into the pokemon_table with this field all set to `999`. However, for option 2, you can use your pre-existing knowledge of the various tools that we've worked with up to this point to craft the appropriate dataset that would contain the pokemon data with the appropriate pokemon_type id and then insert it into the `pokemon` table. If you choose option 2, skip to question 9.
8. If you decide to take option 1 from Q7, use a copy of Q7.db as your starting point. Write the python that removes all entries from the pokemon table with an id higher than 50. Then write the code that looks up what the appropriate pokemon type is from the existing `pokemon_types.csv` dataset for any pokemon that remain in the `pokemon` table and then update the `pokemon_type_id` column to an appropriate value. Once run, there should be no pokemon that have `999` still set as the type.

9. Once our `pokemon` and `pokemon-types` table have been filled in appropriately, write the query that would return a pokemons name and the string representing their type.
10. Starting fresh, recreate the pokemon table using the queries we've used previously. However, add a constraint that checks to make sure that a pokemons height is more than 1 and their weight is less than 1000. Once this database table has been created, provide an additional python script that prompts a user to enter data for each of the available fields in the table. If for any reason data is not valid, you should prompt the user with a friendly message saying that it was unsuccessful along with the potential error that may have occurred to cause it to be unsuccessful. However, if everything is successful – let the user know it was inserted successfully.