Homework #7: Databases

In this homework, you will be populating a database table with information about Pokemon (we have provided the cache data in pokemon.json) and writing code to fetch data from the table that you create.

Pokemon, short for ‘pocket monsters’ (TIL), is a video game series from Nintendo with a corresponding card game, television show, and several movies. Pokemon are creatures with special abilities that can be caught and taught to battle by Pokemon Trainers. Each pokemon has a unique level of stats that affect their fighting ability, including hit points (HP), speed, attack power, and defense power. Many Pokemon can evolve into new forms from gaining battle experience. Every pokemon has a type, some have more than one, that defines what type of abilities they can use (such as water, fire, or grass). Pokemon are most often caught by players to battle other pokemon, but other activities exist for players and pokemon to take part in such as fashion shows!

We have provided code for the following:

1. **read_data_from_file():** To read the cache data in pokemon.json
2. **set_up_database():** To create an SQLite database and set up the ‘connection’ and ‘cursor’
3. **set_up_types_table():** To create/setup one of the SQLite tables for you. This table is called ‘Types’. Run the starter code and then check the structure of the ‘Types’ table using the DB Browser!

When done with the assignment, your database will have two tables, including the one already provided for you. The other one, you will write the code for the create and fill!

We have also provided test cases that will pass if the functions are written correctly. Do **NOT** edit the test cases in any way.

Note: For extra credit, you will have to uncomment the **test_get_special_attack_pokemon_of_type()** function, and pass all the test cases within it.

Tasks

1. **set_up_pokemon_table(data, cur, conn):** This function takes 3 arguments as input: the data (written in JSON format), the database cursor, and database connection object and returns nothing. It iterates through the data present in pokemon.json, creates a table Pokemon with the following columns:

   - name (datatype: text and Primary key)

   - type_id (datatype: integer)

   - HP (datatype: integer)

   - attack (datatype: integer)

   - defense (datatype: integer)

   - speed (datatype: integer)

   - spl_attack (datatype: integer)

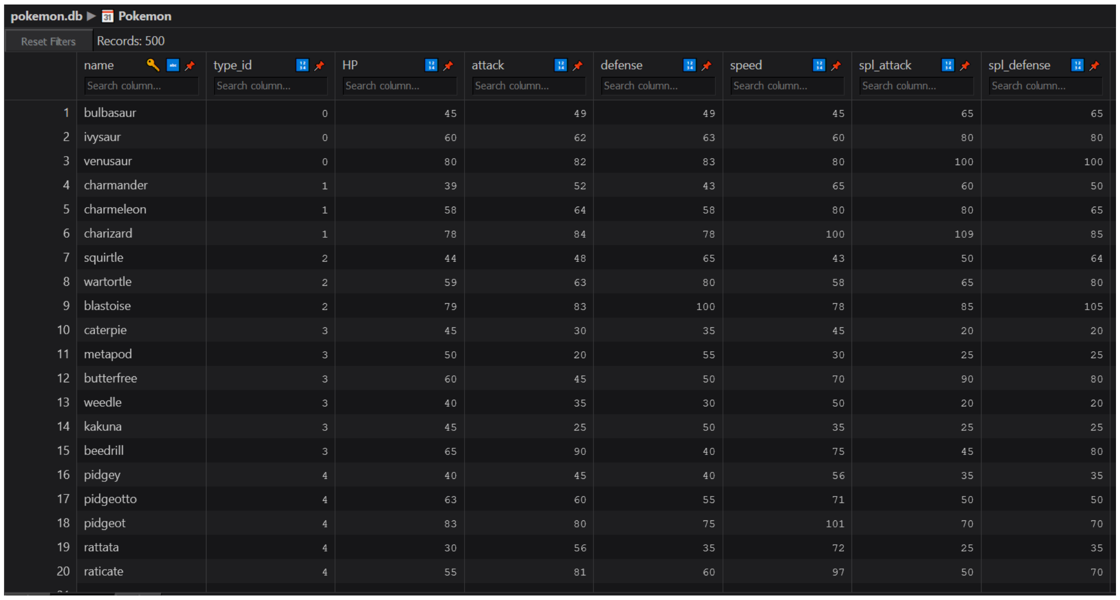
   - spl_defense (datatype: integer)

   The function then loads all of the data present in pokemon.json into this table.

   Note that some pokemon have multiple types, for this case we only want their first type in the Pokemon table.

   Hint: To find the type_id for each pokemon, you will have to look up the first type of each pokemon in the types table we create for you. See setUpTypesTable() and the 'Types' table itself in the DB Browser to get a better understanding.

   Expected table in DB Browser:


pokemon.db ▶ Pokemon

Reset Filters Records: 500

	name	type_id	HP	attack	defense	speed	spl_attack	spl_defense
1	bulbasaur	0	45	49	49	45	65	65
2	ivysaur	0	60	62	63	60	80	80
3	venusaur	0	80	82	83	80	100	100
4	charmander	1	39	52	43	65	60	50
5	charmeleon	1	58	64	58	80	80	65
6	charizard	1	78	84	78	100	109	85
7	squirtle	2	44	48	65	43	50	64
8	wartortle	2	59	63	80	58	65	80
9	blastoise	2	79	83	100	78	85	105
10	caterpie	3	45	30	35	45	20	20
11	metapod	3	50	20	55	30	25	25
12	butterfree	3	60	45	50	70	90	80
13	weedle	3	40	35	30	50	20	20
14	kakuna	3	45	25	50	35	25	25
15	beedrill	3	65	90	40	75	45	80
16	pidgey	4	40	45	40	56	35	35
17	pidgeotto	4	63	60	55	71	50	50
18	pidgeot	4	83	80	75	101	70	70
19	rattata	4	30	56	35	72	25	35
20	raticate	4	55	81	60	97	50	70

2. **get_pokemon_by_HP(hp, cur):** This function takes two arguments as input: an HP and the database cursor. It selects all pokemon for a particular HP and returns a list of tuples. Each tuple contains the pokemon name, their type_id and HP.

3. **get_pokemon_above_HP_equal_speed_and_attack(hp, cur):** This function takes two arguments as input: the HP and the database cursor. It selects all pokemon **greater than** the HP passed as input to the function at a speed **equal to** the pokemon's attack value. For example, a pokemon named 'test', having an attack stat of 45 a speed stat of 45 and an HP value greater than the value passed into the function's arguments meets the criteria required to be added to the list of tuples being returned.
    - Eg of return value: [(pokemon name, speed, defense), ...]

4. **get_pokemon_above_speed_above_defense_of_type(speed, defense, type, cur):** This function takes four arguments as input: A speed value, a defense value, a pokemon type and the database cursor. It selects all pokemon **of a type**
    - At a speed **greater than** the speed passed to the function
    - At a defense greater than the defense passed to the function

   It returns a list of tuples, each tuple containing the pokemon name, type, speed and defense

   Hint: You have to use a JOIN for this task.

Grading Rubric

1. **set_up_pokemon_table() -** 25 Points
    a. 10 points for entering all 500 pokemon in the table
    b. 5 points for creating all 8 columns in the table
    c. 10 points for using the correct data type for each column
2. **get_pokemon_by_HP() -** 10 points
    a. 5 points for returning the correct number of pokemon by HP
    b. 5 points for returning the list of tuples with the correct column values in each tuple
3. **get_pokemon_above_HP_equal_speed_and_attack() -** 10 points
    a. 5 points for correctly selecting the pokemon based on stated criteria
    b. 5 points for correctly returning the list of tuples with the correct column values in each tuple.
4. **get_pokemon_above_speed_above_defense_of_type() -** 15 points
    a. 5 points for correctly using a JOIN to get each pokemon's type for their corresponding type_id.
    b. 5 points for correctly selecting the pokemon based on stated criteria
    c. 5 points for correctly returning the list of tuples with the correct column values in each tuple.

Extra Credit (5 points)

1. **get_special_attack_pokemon_of_type(type, cur):** This function takes two arguments as input: the pokemon type and the database cursor. It selects all pokemon **of a type** that have **their special attack values greater than the attack value by 20 or more. (i.e. special attack >= attack +20)**

    It returns (you guessed it) a list of tuples, each tuple containing the pokemon name, type, attack and their special attack values.

    Make sure to uncomment the function test_get_special_attack_pokemon_of_type() for this portion of the homework. Your extra credit code must pass the commented test cases to receive credit.