**Describing the Dataset**

I have chosen the “Pokemon with stats” dataset from Kraggle. I am planning to use the name, type, hit\_points, attack\_strength, defensive\_strength, special\_attack\_strength, special\_defensive\_strength, pokedex number, and the two type columns.

I will be making an adjustment to the dataset, I will combine the two type columns so that they are one so I can deal with a many to many relationship.

I am going to have 4 tables, the main Pokemon table, the Pokedex table, the type table and the joiner table between pokemon and type table.

The Pokedex table will have and id and the proper pokedex number (I am not using the number as the id as it is not unique since there a few pokemon with the same size (there name is different but their pokedex number is the same))

The Pokemon table with have most of the start columns mentioned previously. Their primary key will be the id. It will also connect to the pokedex table using the pokedex id as a foreign key so that it has access to the pokedex number. The pokemon table will also be connected to the joiner table so that they have access to the multiple types a pokemon has (they can have one or two types).

The joiner table will have the id from the pokemon table as a foreign key as well as the id from the type table as a foreign key and there will bean new column for id which will be the primary key.

Finally, the Type table will have the id as primary key and then the type.

Model: Pokemon (map to the db Pokemon Table)

- name: string (varchar sql)

- type: int (FK)

- hit\_points: int

- attack\_strength: int

- defensive\_strength: int

- special\_attack\_strength: int

- special\_defensive\_strength: int

- number: int (FK)

- id: int (PK)

Model: Pokedex (map to pokedex Table (one to many))

- id: int (varchar sql PK)

- number: int

Pokemon\_Types Table (joiner table)

- pokemon\_name\_id: string (varchar sql FK)

- type\_type\_id: string (varchar sql FK)

- id: int (PK)

Types Table (many to many with pokemon)

-id: int (PK)

- type: string (varchar sql)

Routing Information

GET /Pokemon           =>pokemon#index (collection route, load and dispaly all pokemon)

GET /Pokemon/:name    =>pokemon#show (member route, load specific pokemon by name)

GET /Pokedex/:number  =>pokedex#show (member route, load specific pokemon by pokedex number)

GET /About =>about#index (load and display about page)

Controller: PokemonController

- action: index

- action: show

Controller: PokedexController

- action: show

Controller: About

-action: index

Views

- view: index /app/view/pokemon/index.html.erb

- view: show /app/view/pokemon/show.html.erb

- view: show /app/view/pokedex/show.html.erb

-view: index /app/view/about/index.html.erb