**Extraction – Ally Qi**

We extracted the two datasets we found on Kaggle.com. One was in the form of a csv file and the second was in json format. These two datasets were related, so we planned to read them into pandas dataframes and merge them.

At the bottom, an overview of the json data set can be found. For the csv file, there were many duplicate fields so we only planned on using four: Name (for merging purposes), Legendary, Generation, and Tier.

**Transformation – Kyle Kinney**

After extracting the data we planned to merge our two related datasets. The first step was to reduce one dataframe to only the four columns we were interested in adding. This set contained a lot of duplicate columns that we didn’t want to have to delete later.

We performed an inner join on the “Name” column, since the datasets both centered on individual Pokemon. Once merged, we dropped additional columns that were unrelated to the tiers and competitive viability of the Pokemon, since this was the aim of our database.

We also added a column that combined the two potential types a Pokemon has. For Pokemon that only had one type, their full type would consist only of that type. If the Pokemon had 2 types, the full type would display in the format “type1/type2”.

One problem we ran into was that mongo did not accept columns with periods in the name, so we had to rename two columns in order to remove the periods. From there, this completed our first table which contained a full list of all the Pokemon, their stats in each category, their typing, a list of possible abilities, a dictionary of possible moves, which generation they are in, whether they are “legendary” or not, and their competitive tier.

For our second table/collection we used the previous dataframe and grouped by typing. The main feature of this table was a list of average stats, which would highlight which types were superior in each category. We also attempted to add a count of how many Pokemon of each type appeared in the highest level tiers, but ran into issues getting this to work.

For our final table, we again used the original dataframe and grouped by tier level. This proved to be more fruitful as we were able to list average stats again, but also included the most common typing in each tier and the most commonly represented generation. This allowed some insights into which types and generations were found at a higher or lower levels competitively.

**Load – Garet Douglass**

Even though our data was relational, we decided to load our data into a mongo database so that it could more easily be used in an API due to its format. We utilized pymongo do establish a connection to a json database, built our three collections (pokemons, type, tier) inside our database (pokemon\_db) and inserting the data. We accomplished this with the “.to\_dict” function, applying it to each dataframe and then using “insert\_many” to add it to mongo.

**Final Analysis and Thoughts**

With our data, the goal was to create a database that would allow individuals to pull information on competitive Pokemon tiers to assist in building a team of Pokemon. The first table will likely be one of the most useful resources, as individuals can pull lists of the highest tiered Pokemon as a starting point. The other two tables are more suited for interesting facts and information for fans of Pokemon, including which generation or types typically have the best stats or ranking.

Our database does contain slightly outdated information for this purpose however, since the csv file scraped from Smogon is from the prior generation and tier rankings have changed as new Pokemon have been introduced.

Pokemon JSON:

* ID: (National) Pokedex number.
* Name: Pokemon's English name.
* Type 1: Pokemon's main typing.
* Type 2: Pokemon's secondary typing.
* Abilities: List of all abilities that Pokemon species can have. Includes hidden/special ones.
* Category: The "species" of that Pokemon as given in the Sun/Moon Pokedex.
* Height (ft)
* Height (m)
* Weight (lbs)
* Weight (kg)
* [Capture Rate:](https://bulbapedia.bulbagarden.net/wiki/Catch_rate) Quantifies how easy to catch the Pokemon is.
* Egg Steps: Number of in-game steps require for an egg of this Pokemon to hatch.
* Exp Group: All Pokemon fall into one of six "Exp Groups" that determine how much experience points are required for it to level up.
* Total: The sum of the base stats.
* HP: Hit Points, or Health Points.
* Attack
* Defense
* Sp. Attack: Special Attack
* Sp. Defense: Special Defense
* Speed
* Moves: A list (python dict) of all the moves that Pokemon can learn:
  + Type: The move's typing.
  + Level: The level at which the Pokemon learns this move.
  + Power
  + Accuracy: Base accuracy as a percentage.
  + PP: Power points, or how many times the move can be used per battle.
  + Effect: If the move has a secondary effect, the percentage of it being activated.
  + Description: Some flavour text from the game