I used two sets of data about the different types Pokémon you can catch in Pokémon Go. One data set is relatively organized while the other needs to be transformed more. To make this process easier I referred to the datasets by the name of the web site I go the off of, so the organized data set will be called Kaggle and the unorganized data will be called RapidAPI. For the Kaggle dataset I read the CSV file into a pandas data frame. The RapidAPI dataset was much harder, since it was stored in an API and I had to run multiple API calls on it to get the information I needed. I created a for loop to call the different types of APIs which the data is stored in.

I took the Kaggle dataset in a pandas data and started with making another column to sum up total stats, or total strength of each Pokémon. I then made the legendary column a Boolean to simplify data. For the RapidAPI I started with a definition that pulled the API, grabbed the API key, and put it all into a data frame. I then create a variable that held all the different API calls that I wanted and passed it through a for loop and stored all the data frames into a list of data frames. I then renamed columns and removed duplicate rows, the Pokémon have different forms, and unless the stats differed, I removed the other forms. I then organized the tables reworked the rows and created a combined data frame with all the columns I wanted. I then did the following to both the Kaggle and RapidAPI data sets. I reordered the columns and split the data frame into three data frames; name, catch, and battle. I then removed duplicates from the catch and battle data frames and created a primary key. Next, I ran a for loop to place a foreign key in the name data frame from the catch and battle data frames. I then checked all the data frames and then created a connection to pgadmin and pushed the data onto the tables.

I then created six tables in a pgAdmin SQL database, there is a name, catch, and battle table for the RapidAPI dataset and Kaggle dataset. I chose Pgadmin because I thought it would hold the information best. All the columns needed to be the same and there needed to be values for each row and column. With the stricter rules of pgAdmin this made it very easy. All that was difficult was creating the foreign keys and knowing that the different joins would work.