The first step in my wrangling process was gathering the data. This was achieved through a variety of methods, including manual downloading, programmatic downloading via the requests library, and downloading from the Twitter API via Tweepy. After downloading the data in JSON format from Tweepy I had issues parsing the JSON file. I found a solution in a Stack Overflow answer that utilized the "partial" function from functools to read the JSON objects one at a time. I then loaded the API data in to a Pandas dataframe and moved on to the assessment step of the process.

Assessing the data showed that all three dataframes had quality issues. Several of these issues were simple data type issues that were fixed by casting certain columns as different data types. Other issues included one column with irrelevant html tags in it, data with no images in them, and incorrect rating numerators and denominators. Data tidiness issues included the need to concatenate all three dataframes with each other, fix a few columns that represented data rather than a variable.

I started cleaning the quality issues first. The html tag issue was fixed using a regular expression to grab the content between the tags. The rating numerator and denominator issues were fixed by forcing the denominator to be a 10 (the previously implemented solution found the first '/' in the text of the tweet and assumed that was the rating, when this was often not the case). Data with no images were dropped by filtering the tweet URLs for those that had the word 'photo' in them.

Tidiness issues were a little trickier to solve. I consulted Stack Overflow to find a way to programmatically merge the three dataframes using the 'reduce' function from functools. The most challenging part of this project was melting the "doggo," "floofer," "pupper," and "puppo" columns. These columns represented data rather than variables. Only 293 out of the 1886 rows left in my data had non-null entries for any one of these four columns. Simply using pandas melt function caused issues – it was impossible to figure out which rows were redundant nulls and which ones were true nulls after melting the entire dataframe. My solution involved splitting the dataframe in to two dataframes: one composed of rows with at least one non-null entry under the four aforementioned columns, and one composed of rows with null entries under all four aforementioned columns. I used the melt function on the dataframe with at least one non-null entry, and simply dropped the columns in question in the dataframe with all nulls. I merged these two dataframes back together to create a tidy dataframe.

It's important to note that there are other issues with this data that weren't cleaned. The cleaning that *was* done is enough to proceed with analysis though!