In this project, I gathered, assessed and cleaned data from the WeRateDogs Twitter archive data, using 3 datasets: 'twitter-archive-enhanced.csv', 'image-predictions.tsv' and 'tweet-json.txt'.

1. Gathering
    a) I loaded the datasets to 3 dataframes called df1, df2 and df3, respectively.
    b) For df3, I didn't directly gather the data from Twitter's API because the code failed to authenticate my keys and access tokens. I, however, included the code for querying the API and gathered the data from a readily available json text file. I saved the tweet_id, retweet_count and favorite_count columns to df3.

2. Assessment and Cleaning
   I assessed the data visually and programmatically, to identify 8 quality and 3 tidiness issues, made copies of the original data, then cleaned and merged the copies, so that data is in line with the 3 rules of data. Below is a summary of my cleaning efforts, including key issues I faced, and how I approached them:

    a) df1_clean['tweet_id'] was in scientific notation, which truncates the decimals in the notation and produces an empty table when merging with df2_clean and df3_clean, because there are no common values along which to merge data. My computer automatically converts the ids to scientific notation after downloading the file, with no way to disable this function. Converting the values to strings before merging is not helpful as it still produces an empty dataset.

       I tried to fix the exponent issue by setting the column format to 0 decimals, but all values were rounded off to the nearest thousand, which posed a similar problem as the exponent format when merging with df2_clean and df3_clean as their tweet_ids are not rounded off.

       I created a 'test' df by merging df1_clean and df2_clean. It only returned 11 entries, showing that there were only 11 common values in both entire datasets. I fixed this issue by replacing df1_clean['tweet_id'] column with one from another dataframe.

       I dropped the df1_clean['tweet_id'] column and then concatenated df3_clean['tweet_id'] because the 2 dataframes have similar numbers of rows (df1_clean has 2 more) and their ids are in the same order. Before dropping, I did some more assessment of df1_clean and df3_clean to identify the 2 extra rows in df1_clean and drop them, so that both data frames have the exact same ids. I then concatenated df3_clean['tweet_id'] to df1_clean, which fixed the problem.

    b) I converted master_df['tweet_id'] and df1_clean['timestamp'] to their correct datatypes of string and datetime, respectively.

c) I created one dog_stages column. Some dogs have more than one stage and most don't have a stage. I also turned 'None' entries into null values.

d) rating_denominator should be one value, so I dropped rows with values not equalling 10. Since rating_denominator was now a constant, I dropped the column.

e) I dropped unnecessary rows and columns e.g. columns in df1_clean like in_reply_to_status_id, with too many null values, and rows with the wrong data like retweets and replies in df1_clean and image predictions that aren't dogs in df2_clean. I also dropped rows with wrong dog names in df1_clean, which started with lowercase.

f) In df2_clean, I created a column (p_conf) for the highest confidence levels for dog breeds per row, and one for the corresponding dog breed (dog_breed), so that there is one observation per tweet_id. To ensure that the first instance in every row of (p_dog = True) had the highest confidence level for dogs, I queried the data to make sure that p1 > p2 > p3 for all rows. This made it easier to set conditions in the for loop when creating the columns. I then dropped the numbered p, p_conf and p_dog columns, and rows with no dog breeds.

3. Storing data
   After cleaning, I merged the data to one dataframe called master_df, as it forms one observational unit, and I stored it as a csv file called 'twitter_archive_master.csv'.