Data Gathering:

Three datasets are gathered containing data pertaining to the WeRateDogs twitter page. 'archive_df' is read from the given 'twitter-archive-enhanced.csv' file. 'image_predictions_df' is downloaded and saved as a tsv file using the requests python library and read using pandas.read_csv. Finally the twitter API is queried with the .get_status of the 'tweet_id's found in the 'archive_df' DataFrame. Each tweet's _json attribute is written line by line to 'tweet_json.txt'. For tweets where .get_status failed, the tweet id and error code are written to 'failed_tweets.txt'. The 'tweet_json.txt' file is read line by line and written to a dictionary with the .loads method of the json python library and the dictionary is converted to a DataFrame called 'tweet_json_df' using 'from_dict'. The 'id', 'retweet_count' and 'favorite_count' columns are then used to make up the 'tweet_json_filtered_df' DataFrame.

Assessing Data:

The three DataFrames are visually assessed by displaying them. 'archive_df' has 2 columns for the rating namely 'rating_numerator' and 'rating_denominator', it would be possible to have a single column for a score out of 10. There are four columns for the 'dog stages' where only one column is needed. There is a 'timestamp' column which combines the post date and time that can be converted to 2 columns . The 'source' column is cluttered by html tags and the 'name' column contains incorrect dog names such as 'a'.

In 'image_predictions_df' there are three predictions per tweet id with confidence levels an indication of whether the prediction is dog breed. Some of these have either no predictions that are dog breeds or it is the second or third prediction that is a dog breed. The text in of the dog breed predictions consistently starts with upper or lowercase letters with underscores instead of spaces.

The assessment is continued programmatically. The .info() method gives an overview of the column names, null values and data types. 'Tweet id' is an int64 data type instead of an object/string data type. The 'retweeted_status_id' and 'in_reply_to_status_id' columns of 'archive_df' show that there are retweets and replies.

The three tables need to be combined as they represent a single set of observations and columns that won't be used in analysis need to be dropped. These observations lead to the issues summarised in 'wrangle_act.ipynb'.

Cleaning Data:

First copies of the DataFrames are made to perform cleaning. The tweet_json_df_filtered_copy 'id' column is renamed 'tweet_id' and the data type is converted to string for all 3 DataFrames with .astype(str).

The three DataFrames are combined using merge to join the DataFrames on the 'tweet_id' column.

The master data set is copied to continue cleaning. Using the numpy select method the 4 columns for dog stages are converted to a single column dog is inserted for rows without a stage. The data type is also converted to category.

The 'rating_numerator' column is converted to 'rating_out_of_10' by dividing the 'rating_numerator' by the 'rating_denominator' and multiplying by 10.

Incorrect dog names are replaced with 'None' and not dropped as these rows are still useful for analysis. The names could be corrected manually but there are 657 missing which makes this impractical.

Using nested np.where functions the incorrect dog breeds are replaced with the second or third prediction if these are a dog breed but the first isn't. 'none found' is inserted where all three are not dog breeds. Lambda functions with str.lower() and str.replace() are used to convert the predictions to lowercase and replace the underscores with spaces.

Rows with a non_null value for 'in_reply_to_status_id' or 'retweeted_status_id ' are dropped to remove retweets and replies.

The timestamp column is changed to datetime and the date and time parts are extracted to 'post_date' and 'post_time'.

With 'np.select' the html tags are removed from the source column.