## WRANGLE REPORT

The information was gathered from various sources, including the "WeRateDogs" twitter archive, tweet image prediction, and additional information from the Twitter API, and saved into their respective pandas' data frames. We viewed the individual and distinct properties of these individual data frames to get a sense of the nature of the data we are working on before proceeding to the wrangling phase. The wrangling of the data was done with the aid of a single wrangle function titled "wrangle" which takes the three individual data frames from above to produce a single cleaned and uniquely workable dataset. The sub-codes in this wrangle function will be discussed in detail in the following section of this note.

Firstly, we needed to have a connecting point between the three distinct data frames, and from earlier observation, the "tweet_id" of the first two data frames ( labelled df2) was similar to the "id" column in the third data frame (labelled df3), and for ease, the column was renamed from "id" to "tweet_id". It was immediately followed by a "merge" command, which combined the three data frames into one labelled "df" and set the "tweet_id" column as the index.

This data frame, df, would now be used for the subsequent wrangling, analysis, and visualization portion of this project. On closer inspection of this data frame, it was noticed that some columns had a lot of missing values, which would constitute a major problem during the analysis, so it was necessary to drop such columns. So with the limit at 20%, all columns with null values greater than 20% of the total number of rows were dropped, and these included *'in_reply_to_status_id_x', 'in_reply_to_user_id_x', 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp', 'in_reply_to_status_id_y', 'in_reply_to_status_id_str', 'in_reply_to_user_id_y', 'in_reply_to_user_id_str', 'in_reply_to_screen_name', 'geo', 'coordinates', 'place', 'contributors', 'retweeted_status', 'quoted_status_id', 'quoted_status_id_str', and 'quoted_status'.*

We needed a distinct column for the dog breed, so we extracted the rows where the dog breed was identified in either "p1" or "p2" or "p3" and, using a conditional state and looping through the various rows to extract the dog breed for each unique "tweet_id", saved it to column "breed". *'p1_dog', 'p2', 'p2_dog', 'p3', 'p3_dog'* were dropped as they constituted leakage to the new column created "breed"

The next step was to drop columns with a high cardinality value, that is, a high number of distinct values in the column, as this would cause problems during the data analysis and visualization phase. So the limit on the number of distinct was set at 120 and the following g columns labelled *'timestamp', 'text', 'expanded_urls', 'name', 'jpg_url', 'p1', 'p1_conf', 'p2_conf', 'p3_conf', 'created_at', 'id_str', 'full_text', 'entities', 'extended_entities', 'retweet_count', and 'favorite_count'.*

On observation with the histogram, it was discovered that the data in the rating numerator column was highly skewed and, on a closer look with the ".describe()", it was seen that 75% of the values were less than 450 and it had a max of 1000+ and 50% within the range of 10, so it was necessary to cut off these outliers. The data frame was sliced with the rating numerator column dropping the upper limit of the top 1% of values, producing more evenly distributed data.

So columns such *as 'source_x','source_y', 'is_quote_status', 'favorite', 'display_text_range', and 'user'* were not needed and were dropped while the dog stages were changed to 1.s and 0.s to reflect true or false for ease. Finally, columns with low cardinality such *as 'truncated','retweeted', 'possibly_sensitive', 'possibly_sensitive', 'possibly_sensitive', 'possibly_sensitive', 'possibly_appealable','lang'* were dropped as they provide the least information with little or no variation.