# Wrangle Report

## Introduction

Data wrangling mainly involves gathering data from different sources in different formats, assessing its quality and tidiness and then cleaning the data. The dataset used in this project is the twitter archive of Twitter user `@dog_rates`, also known as `WeRateDogs`. `WeRateDogs` is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10.

The goal of this project is to wrangle the WeRateDogs twitter data to create interesting and reliable analyses and visualizations. After effectively gathering, assessing, and cleaning the data, it is analysed to create informative visualizations and gain meaningful insights.

## Project Steps Overview

Project tasks are as follows:

### Gathering the Data

This project comprises of three datasets

`Twitter archive enhanced`: This dataset was downloaded by WeRateDogs from their twitter archive and provided exclusively for this project. The data was downloaded manually as a csv file. Using the read_csv() function, the data was read and imported from the csv file into a jupyter notebook as a DataFrame called `twitter_df`.

`Tweet image predictions`: This file was downloaded programmatically using the `Requests` library. Using the get() function of the requests library, the data was retrieved from its URL, hosted on Udacity's servers as a tsv file.

`Tweet Json`: Downloading this dataset was the most challenging as the data was obtained utilizing Twitter's API. First, a twitter developer account was created and the required keys and tokens (consumer key, consumer secret, access token, and access secret) were obtained to facilitate data query using Twitter's API. `Tweepy` was imported, `OAuthHandler` was used to authorize my consumer key and consumer secret , and `set_access_token` was used to set access to my access token and access secret. When calling the API, due to twitter's rate limit, the wait_on_rate_limit was set to True in the API parameter to wait after the tweet limit and continue automatically at the end of the waiting time.
The data was queried based on the tweet IDs provided in the `twitter archive enhanced` dataset and then an empty dictionary was initialized to house the data that failed. Using the python `open()` function, the successful query output was written to the created `tweet_json.txt`. Consequently, using the `open()` function and `for loop`, the tweet_json.txt was read line by line using `readlines()` and loaded the required data (tweet IDs, retweet count, and favorite count) to a pandas DataFrame called `tweet_attr`

### Assessing the Data

After gathering the data, all three datasets were assessed Visually and Programmatically. Visual assessment: each dataframe was displayed and assessed in the Jupyter Notebook Programmatic assessment: pandas function and methods such as `info(), describe(), nunique(), tail(), head(), duplicated(), isnull(), sample(), value_counts()` were used to assess the data

### Cleaning the Data

Before cleaning the data, a copy of each original dataframe was made to compare changes to the original data frame during the cleaning process. During the cleaning, the `Define-Code-Test` framework was utilized for documentation. The following cleaning decisions were made for proper analysis

- Remove rows with retweeted_status_id, retweeted_status_user _id, and retweeted_status _timestamp values as these are retweets and irrelevant to the analysis
- Dropped in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, and retweeted_status_timestamp columns due to missing data, and expanded_urls columns as irrelevant to analysis
- Created a new column called "dog_stage' and combined values in doggo, floofer, pupper, and puppo columns into the new column
- Replaced doggo,floofer", "doggo,puppo", and "doggo,pupper" with accurate dog_stages as defined in the Dogtionary
- Replaced all values starting with lowercase with NaN as they represented incorrect data, and replaced 'None' with NaN.
- Corrected and changed dog name "Al" to "Al Cabone" as indicated in the original tweet
- Converted timestamp datatype from string to datatime
- Replaced tweet ID 786709082849828864 rating "75" with "9.75" in rating_numerator column
- Replaced rating_numerator and rating_denominator of tweet ID 666287406224695296 with 9 and 10 respectively
- Converted all values in p1, p2, p3 columns to lowercase for consistency
- Joined the three dataframes to form a single dataframe.

### Storing the Data

After gathering, assessing, and cleaning the datasets, the merged dataset was saved in a csv file called `twitter_archive_master.csv`

In [ ]: