This project gathered data from WeRateDogs account on Twitter. There were three different datasets that needed to be assessed for quality and tidiness. These datasets were tweet-json.txt, image-predictions.tsv and twitter-archive-enhanced.csv.

Quality Issue 1, 8 and 11

There are three occassions where we would need to convert a column data type from integer (int) to strings (str). To reduce the time needed to code for every single column as well as repeat long code repetition, we created a function to deal with this quality issue. The code is as shown below:

```
# We will write a def function here to make all other conversions simpler by calling it
def conv_to_str(dataset,column):
    dataset[column] = dataset[column].astype(str)
```

```
conv_to_str(twitter_archive_clean,'tweet_id')
```

We test our cleaning attempt with the .type() method like so:

Test

```
type(twitter_archive_clean.tweet_id[0])
```

```
str
```

Quality Issue 2

We also convert the timestamp in the twitter archive dataframe from a string to a dateTime type. This is to make sure that dateTime analysis can be computed successfully. The code to enable this is executed using pd.to_dateTime() and also tested with the .type() method

Code

```
In [51]: twitter_archive_clean['timestamp'] = pd.to_datetime(twitter_archive_clean['timestamp'])
```

Test

```
In [52]: type(twitter_archive_clean['timestamp'][0])
Out[52]: pandas._libs.tslibs.timestamps.Timestamp
```

Quality Issue 3 and 4

With this project, we required only original tweets and not retweets so we dropped all columns that had anything to do with retweets. These columns were found in the twitter_archive dataset during assessment. The respective columns were dropped using the .drop() method. One way of confirming if our method worked is to check the column information the twitter archive dataset holds. This is also done by using the .info() method. Both methods are executed as shown;

Code

```
In [53]: twitter_archive_clean = twitter_archive_clean.drop(columns=
             ['retweeted_status_id',
             'retweeted_status_user_id',
             'retweeted_status_timestamp']
                                                         )
```

Test

```
In [54]: twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   tweet_id               2356 non-null   object
 1   in_reply_to_status_id  78 non-null     float64
 2   in_reply_to_user_id    78 non-null     float64
 3   timestamp              2356 non-null   datetime64[ns, UTC]
 4   source                 2356 non-null   object
 5   text                   2356 non-null   object
 6   expanded_urls          2297 non-null   object
 7   rating_numerator       2356 non-null   int64
 8   rating_denominator     2356 non-null   int64
 9   name                   2356 non-null   object
 10  doggo                  2356 non-null   object
 11  floofer                2356 non-null   object
 12  pupper                 2356 non-null   object
 13  puppo                  2356 non-null   object
dtypes: datetime64[ns, UTC](1), float64(2), int64(2), object(9)
memory usage: 257.8+ KB
```

We repeated the same action for other column features we won't need in our analysis

Code

```
In [55]: twitter_archive_clean = twitter_archive_clean.drop(columns=
                                                            ['in_reply_to_status_id', 'in_reply_to_user_id',
                                                             'expanded_urls',
                                                             'source' ]
                                                           )
```

Test

```
In [56]: twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2356 non-null   object
 1   timestamp           2356 non-null   datetime64[ns, UTC]
 2   text                2356 non-null   object
 3   rating_numerator    2356 non-null   int64
 4   rating_denominator  2356 non-null   int64
 5   name                2356 non-null   object
 6   doggo               2356 non-null   object
 7   floofer             2356 non-null   object
 8   pupper              2356 non-null   object
 9   puppo               2356 non-null   object
dtypes: datetime64[ns, UTC](1), int64(2), object(7)
memory usage: 184.2+ KB
```

Quality Issue 5 and 7

Column names should be made easy to understand at first glance by the analyst so there isn't the need for constant referencing. Constant referencing wastes time. A few columns in the imgage prediction dataset would need their column names renamed. For this purpose we would deploy the .rename() method and check with the info() method as shown;

Code

```
In [57]: image_predictions_clean.rename(columns={'p1':'algor_1',
                                    'p2':'algor_2',
                                    'p3':'algor_3',
                                   'p1_conf':'algor_1_conf',
                                   'p2_conf':'algor_2_conf',
                                   'p3_conf':'algor_3_conf',
                                   'p1_dog':'algor_1 _bool',
                                   'p2_dog':'algor_2_bool',
                                   'p3_dog': 'algor_3_bool'}, inplace=True);
```

Test

```
In [58]: image_predictions_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   tweet_id       2075 non-null   int64
 1   jpg_url        2075 non-null   object
 2   img_num        2075 non-null   int64
 3   algor_1        2075 non-null   object
 4   algor_1_conf   2075 non-null   float64
 5   algor_1 _bool  2075 non-null   bool
 6   algor_2        2075 non-null   object
 7   algor_2_conf   2075 non-null   float64
 8   algor_2_bool   2075 non-null   bool
 9   algor_3        2075 non-null   object
 10  algor_3_conf   2075 non-null   float64
 11  algor_3_bool   2075 non-null   bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

We also name the 'id' column in the tweet json dataset to 'tweet_id' so we could merge by that column with other dataframes

Code

```
In [62]: tweet_df_clean.rename(columns={'id':'tweet_id'}, inplace=True)
```

Test

```
In [63]: tweet_df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2354 non-null   int64
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
dtypes: int64(3)
memory usage: 55.3 KB
```

Quality Issue 6

It is important that all null values are correctly represented otherwise they could be categorized as something which might affect our the quality of our analysis. It can be seen from the twitter archive datastet that the null values are represented by 'None' instead of 'NaN'. We correct this issue by replacing all 'None' values with 'NaN' using numpy's np.nan and check for nulls using the .isnull() method.

Code

```
In [59]: twitter_archive_clean.name = twitter_archive_clean.name.replace('None', value = np.nan)
```

Test

```
In [60]: twitter_archive_clean.name.isnull().sum()
Out[60]: 745
```

Quality Issue 9

Some breed names in the predicitions column in the image prediction dataset starts with an upper-case while others start with a lower case showing non-uniformity. These columns are represented by algor_1, algor_2 and algor_3. All names starting with lower-case letters were converted to start with upper-case by deploying the .str.title() method. This was tested by calling a sample and visually assessing.

Code

```
In [67]: image_predictions_clean.algor_1 = image_predictions_clean.algor_1.str.title()
         image_predictions_clean.algor_2 = image_predictions_clean.algor_2.str.title()
         image_predictions_clean.algor_3 = image_predictions_clean.algor_3.str.title()
```

Test

```
In [68]: image_predictions_clean.sample(5)
```

Out[68]:

jpg_url	img_num	algor_1	algor_1_conf	algor_1_bool	algor_2	algor_2_conf	algor_2_bool	algor_3	algor_3_conf	algor_3_bool
gWIAAwOUy.jpg	1	Shih-Tzu	0.225848	True	Norfolk_Terrier	0.186873	True	Irish_Terrier	0.106987	True
LWEAEnSk7.jpg	1	Microwave	0.981946	False	Rotisserie	0.007472	False	Television	0.005881	False
7UcAANzmK.jpg	1	Soft-Coated_Wheaten_Terrier	0.403496	True	Cocker_Spaniel	0.135164	True	Golden_Retriever	0.088719	True
odXAAAj1de.jpg	1	Hamster	0.132440	False	Toy_Poodle	0.123962	True	Bubble	0.056212	False
WMAAaoG5.jpg	1	Llama	0.379624	False	Triceratops	0.162761	False	Hog	0.084251	False

Quality Issue 10

According to WeRateDogs, all dogs are awesome so we cant have a rating lower than the perfect score, 10/10. This means we need to make sure at least all ratings numerator in the twitter archive dataset is at least 10. We first query tha dataset for all ratings numerator values less than 10 and add 10 to them using the .add() method. By deploying the .value_counts() method, we can see that our lowest value is now 10.

Code

```
In [70]: twitter_archive_clean.rating_numerator[twitter_archive_clean['rating_numerator'] <= 10 ] =
         twitter_archive_clean.rating_numerator[twitter_archive_clean['rating_numerator'] <= 10 ].add(10)
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_6440\3417593879.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  twitter_archive_clean.rating_numerator[twitter_archive_clean['rating_numerator'] <= 10 ] = twitter_archive_clean.rating_numerator[twitter_archive_clean['rating_numerator'] <= 10 ].add(10)
```

```
In [289]: twitter_archive_clean.rating_numerator.value_counts()
```

```
Out[289]: 12     567
          11     473
          20     462
          13     370
          19     158
          18     102
          14      71
          17      56
          15      39
          16      32
          420      2
          10       2
          75       2
          45       1
          60       1
          44       1
          121      1
          143      1
          99       1
```

Tidiness Issue 1

To be able to do relational analysis between each dataframe easily we would need to merge the three dataframes. The .merge() method was called here. We first merged the image predictions and twitter archive into one master dataframe by the 'tweet_id' in both dataframes. Then we merged that master dataframe with the tweet json dataframe by the same 'tweet_id'.

Code

In [73]:
```
master_data_clean = []

master_data_clean = pd.merge( image_predictions_clean, twitter_archive_clean, on='tweet_id', how="left")
```

In [74]:
```
master_data_clean = pd.merge( master_data_clean, tweet_df_clean, on='tweet_id', how="left")
```

Test

In [75]:
```
master_data_clean.shape
```

Out[75]:
```
(2075, 23)
```

In [76]:
```
master_data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2075 entries, 0 to 2074
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2075 non-null   object
 1   jpg_url             2075 non-null   object
 2   img_num             2075 non-null   int64
 3   algor_1             2075 non-null   object
 4   algor_1_conf        2075 non-null   float64
 5   algor_1 _bool       2075 non-null   bool
 6   algor_2             2075 non-null   object
 7   algor_2_conf        2075 non-null   float64
 8   algor_2_bool        2075 non-null   bool
 9   algor_3             2075 non-null   object
 10  algor_3_conf        2075 non-null   float64
 11  algor_3_bool        2075 non-null   bool
 12  timestamp           2075 non-null   datetime64[ns, UTC]
 13  text                2075 non-null   object
 14  rating_numerator    2075 non-null   int64
 15  rating_denominator  2075 non-null   int64
 16  name                1497 non-null   object
 17  doggo               2075 non-null   object
 18  floofer             2075 non-null   object
 19  pupper              2075 non-null   object
 20  puppo               2075 non-null   object
 21  retweet_count       2073 non-null   float64
 22  favorite_count      2073 non-null   float64
dtypes: bool(3), datetime64[ns, UTC](1), float64(5), int64(3), object(11)
memory usage: 346.5+ KB
```

Tidiness Issue 2

The doggo, floofer, pupper and puppo columns in twitter_archive should be categories in one column. We use the pd.melt() method to create a new column called 'stage' in the tweet json dataframe to house the dog stages and melt the four individual stages into a new 'stage' column in the master data and finally delete existing four stages column.

Code

In [77]:
```
df = pd.melt(master_data_clean, id_vars=['tweet_id'], value_vars=['doggo', 'floofer', 'pupper', 'puppo'],value_name='stage')

master_data_clean['stage'] = df['stage']
```

Test

[386]:
```
master_data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33200 entries, 0 to 33199
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    33200 non-null  object 
 1   jpg_url                     33200 non-null  object 
 2   img_num                     33200 non-null  int64  
 3   algor_1                     33200 non-null  object 
 4   algor_1_conf                33200 non-null  float64
 5   algor_1 _bool               33200 non-null  bool   
 6   algor_2                     33200 non-null  object 
 7   algor_2_conf                33200 non-null  float64
 8   algor_2_bool                33200 non-null  bool   
 9   algor_3                     33200 non-null  object 
 10  algor_3_conf                33200 non-null  float64
 11  algor_3_bool                33200 non-null  bool   
 12  in_reply_to_status_id       368 non-null    float64
 13  in_reply_to_user_id         368 non-null    float64
 14  timestamp                   33200 non-null  object 
 15  source                      33200 non-null  object 
 16  text                        33200 non-null  object 
 17  retweeted_status_id         1296 non-null   float64
 18  retweeted_status_user_id    1296 non-null   float64
 19  retweeted_status_timestamp  1296 non-null   object 
 20  expanded_urls               33200 non-null  object 
 21  rating_numerator            33200 non-null  int64  
 22  rating_denominator          33200 non-null  int64  
 23  name                        23952 non-null  object 
 24  retweet_count               33168 non-null  float64
 25  favorite_count              33168 non-null  float64
 26  stage                       1336 non-null   object 
dtypes: bool(3), float64(9), int64(3), object(12)
memory usage: 6.4+ MB
```

Tidiness Issue 3

We would finally extract day, month and year from the timestamp into indivual columns. We use the .dt method to create new variables called 'day', 'month' and 'year' and populate it with new columns in the master dataframe with the same names.

Code

In [102]:
```
day = master_data_clean['timestamp'].dt.day
month = master_data_clean['timestamp'].dt.month
year = master_data_clean['timestamp'].dt.year
```

In [104]:
```
master_data_clean['day'] = day
master_data_clean['month'] = month
master_data_clean['year'] = year
```

Test

In [103]:
```
master_data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2075 entries, 0 to 2074
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2075 non-null   object             
 1   jpg_url             2075 non-null   object             
 2   img_num             2075 non-null   int64              
 3   algor_1             2075 non-null   object             
 4   algor_1_conf        2075 non-null   float64            
 5   algor_1 _bool       2075 non-null   bool               
 6   algor_2             2075 non-null   object             
 7   algor_2_conf        2075 non-null   float64            
 8   algor_2_bool        2075 non-null   bool               
 9   algor_3             2075 non-null   object             
 10  algor_3_conf        2075 non-null   float64            
 11  algor_3_bool        2075 non-null   bool               
 12  timestamp           2075 non-null   datetime64[ns, UTC]
 13  text                2075 non-null   object             
 14  rating_numerator    2075 non-null   int64              
 15  rating_denominator  2075 non-null   int64              
 16  name                1497 non-null   object             
 17  doggo               2075 non-null   object             
 18  floofer             2075 non-null   object             
 19  pupper              2075 non-null   object             
 20  puppo               2075 non-null   object             
 21  retweet_count       2073 non-null   float64            
 22  favorite_count      2073 non-null   float64            
 23  stage               2075 non-null   object             
dtypes: bool(3), datetime64[ns, UTC](1), float64(5), int64(3), object(12)
memory usage: 362.7+ KB
```