# wrangle_act

April 5, 2019

## 0.1 Project: WeRateDogs - Wrangle and Analyze Data

### 0.1.1 Presented by: Paula Munoz

### 0.1.2 Introduction

Throughout this project I will gather, assess and clean data related to the Twitter account @dog_rates, also knowns as WeRateDogs to create an interesting and trustworthy analysis and visualizations.

I will be using the Twitter's API to obtain the retweet and favorite count to complement the analysis.

**About WeRateDogs** WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog, and these ratings almost always have a denominator of 10. and the numerator is almost always greater than 10. Some examples of ratings are: 11/10, 12/10, 13/10, etc.

## 0.2 Data Wrangling

### 0.2.1 Gathering Data

Importing Libraries and loading the data

```
In [27]: #Importing Libraries
         import pandas as pd
         import numpy as np
         import requests
         import tweepy
         import json
         import warnings
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

```
In [28]: #Load twitter Archive dataset
         df = pd.read_csv('twitter-archive-enhanced.csv')
```

```
In [29]: #Download tweet image predictions tsv file from Internet by using requests library
         url= "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictic
```

```
response = requests.get(url)

with open('image_predictions.tsv', mode = 'wb') as file:
    file.write(response.content)

#Load tsv file
df_images = pd.read_csv('image_predictions.tsv', sep = '\t')
```

In [612]:
```
#Personal Twitter API Key and tokens (Information removed for Project submission)

 consumer_key = 'MY_CONSUMER_KEY'
 consumer_secret = 'MY_CONSUMER_SECRET'
 access_token = 'MY_ACCESS_TOKEN'
 access_secret = 'MY_ACCESS_SECRET'

 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
 auth.set_access_token(access_token, access_secret)

 api = tweepy.API(auth, wait_on_rate_limit= True, wait_on_rate_limit_notify= True)
```

In [30]:
```
#Retweet and favorite count for tweet_ids
with open ('tweet_json.txt', 'a', encoding = 'utf-8') as file2:
    for id in df['tweet_id']:
        try:
            tweet = api.get_status(id, tweet_mode = 'extended')
            json.dump(tweet._json, file2)
            file2.write('\n')
        except:
            continue
```

In [31]:
```
#Each tweet's JSON data should be written to its own line
tweets_list = []
with open('tweet_json.txt', 'r') as file3:
    for line in file3:
        try:
            tweet = json.loads(line)
            tweets_list.append(tweet)
        except:
            continue
```

In [32]:
```
#Creating Dataframe with tweet's information
df_tweets2 = pd.DataFrame(tweets_list)
```

In [33]:
```
df_tweets2.to_csv('df_tweets_api_data2.csv')
```

In [34]:
```
#Creating Dataframe with tweet's information
df_tweets = pd.DataFrame(tweets_list, columns = ['id', 'retweet_count', 'favorite_count
df_tweets.to_csv('df_tweets_api_data.csv')
```

### 0.3 Assessing Data

I will make an in depth assessment for the three following datasets:

- df: Contains the twitter Archive data
- df_images: Contains Image Predictions data
- df_tweets: Contains retweet and favorite counts data

The assesment will be done *Visually* and *Programatically*

#### 0.3.1 Visual Assessment

By quickly visually inspecting the datasets I was able to identify some quality and tidiness issues, such as:

**Quality** o "Source" field from archived dataset (df) is difficult to read, it has unnecessary html tags

**Tidiness** o Dog stages are listed separately instead of having one column called "Stage"

#### 0.3.2 Programmatic Assessment

```
In [7]: #Checking the first five rows of df dataframe
        df.head()
```

```
Out[7]:              tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
        0  892420643555336193                    NaN                  NaN
        1  892177421306343426                    NaN                  NaN
        2  891815181378084864                    NaN                  NaN
        3  891689557279858688                    NaN                  NaN
        4  891327558926688256                    NaN                  NaN

                         timestamp  \
        0  2017-08-01 16:23:56 +0000
        1  2017-08-01 00:17:27 +0000
        2  2017-07-31 00:18:03 +0000
        3  2017-07-30 15:58:51 +0000
        4  2017-07-29 16:00:24 +0000

                                                   source  \
        0  <a href="http://twitter.com/download/iphone" r...
        1  <a href="http://twitter.com/download/iphone" r...
        2  <a href="http://twitter.com/download/iphone" r...
        3  <a href="http://twitter.com/download/iphone" r...
        4  <a href="http://twitter.com/download/iphone" r...

                                                     text  retweeted_status_id  \
        0  This is Phineas. He's a mystical boy. Only eve...                  NaN
        1  This is Tilly. She's just checking pup on you...                  NaN
```

```
2  This is Archie. He is a rare Norwegian Pouncin...                  NaN
3  This is Darla. She commenced a snooze mid meal...                  NaN
4  This is Franklin. He would like you to stop ca...                  NaN

   retweeted_status_user_id retweeted_status_timestamp  \
0                       NaN                        NaN
1                       NaN                        NaN
2                       NaN                        NaN
3                       NaN                        NaN
4                       NaN                        NaN

                                       expanded_urls  rating_numerator  \
0  https://twitter.com/dog_rates/status/892420643...                13
1  https://twitter.com/dog_rates/status/892177421...                13
2  https://twitter.com/dog_rates/status/891815181...                12
3  https://twitter.com/dog_rates/status/891689557...                13
4  https://twitter.com/dog_rates/status/891327558...                12

   rating_denominator      name doggo floofer pupper puppo
0                  10   Phineas  None    None   None  None
1                  10     Tilly  None    None   None  None
2                  10    Archie  None    None   None  None
3                  10     Darla  None    None   None  None
4                  10  Franklin  None    None   None  None
```

In [8]:
```
#Checking the first five rows of df_images dataframe
df_images.head()
```

Out[8]:
```
             tweet_id                                          jpg_url  \
0  666020888022790149  https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg
1  666029285002620928  https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg
2  666033412701032449  https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg
3  666044226329800704  https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg
4  666049248165822465  https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg

   img_num                     p1   p1_conf  p1_dog                 p2  \
0        1  Welsh_springer_spaniel  0.465074    True             collie
1        1                redbone  0.506826    True  miniature_pinscher
2        1         German_shepherd  0.596461    True            malinois
3        1     Rhodesian_ridgeback  0.408143    True             redbone
4        1      miniature_pinscher  0.560311    True          Rottweiler

    p2_conf  p2_dog                   p3   p3_conf  p3_dog
0  0.156665    True    Shetland_sheepdog  0.061428    True
1  0.074192    True  Rhodesian_ridgeback  0.072010    True
2  0.138584    True           bloodhound  0.116197    True
3  0.360687    True   miniature_pinscher  0.222752    True
4  0.243682    True             Doberman  0.154629    True
```

```
In [9]: #Checking the first five rows of df_tweets dataframe
        df_tweets.head()
```

Out[9]:

| | id | retweet_count | favorite_count |
|---|---|---|---|
| 0 | 892420643555336193 | 8279 | 37912 |
| 1 | 892177421306343426 | 6117 | 32564 |
| 2 | 891815181378084864 | 4051 | 24517 |
| 3 | 891689557279858688 | 8421 | 41266 |
| 4 | 891327558926688256 | 9121 | 39440 |

```
In [10]: #Checking the last five rows of df dataframe
         df.tail()
```

Out[10]:

| | tweet_id | in_reply_to_status_id | in_reply_to_user_id | timestamp | source | text | retweeted_status_id | retweeted_status_user_id | retweeted_status_timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 2351 | 666049248165822465 | NaN | NaN | 2015-11-16 00:24:50 +0000 | <a href="http://twitter.com/download/iphone" r... | Here we have a 1949 1st generation vulpix. Enj... | NaN | NaN | NaN |
| 2352 | 666044226329800704 | NaN | NaN | 2015-11-16 00:04:52 +0000 | <a href="http://twitter.com/download/iphone" r... | This is a purebred Piers Morgan. Loves to Netf... | NaN | NaN | NaN |
| 2353 | 666033412701032449 | NaN | NaN | 2015-11-15 23:21:54 +0000 | <a href="http://twitter.com/download/iphone" r... | Here is a very happy pup. Big fan of well-main... | NaN | NaN | NaN |
| 2354 | 666029285002620928 | NaN | NaN | 2015-11-15 23:05:30 +0000 | <a href="http://twitter.com/download/iphone" r... | This is a western brown Mitsubishi terrier. Up... | NaN | NaN | NaN |
| 2355 | 666020888022790149 | NaN | NaN | 2015-11-15 22:32:08 +0000 | <a href="http://twitter.com/download/iphone" r... | Here we have a Japanese Irish Setter. Lost eye... | NaN | NaN | NaN |

```
                                                expanded_urls  rating_numerator  \
2351  https://twitter.com/dog_rates/status/666049248...                 5
2352  https://twitter.com/dog_rates/status/666044226...                 6
2353  https://twitter.com/dog_rates/status/666033412...                 9
2354  https://twitter.com/dog_rates/status/666029285...                 7
2355  https://twitter.com/dog_rates/status/666020888...                 8

      rating_denominator  name doggo floofer pupper puppo
2351                  10  None  None    None   None  None
2352                  10     a  None    None   None  None
2353                  10     a  None    None   None  None
2354                  10     a  None    None   None  None
2355                  10  None  None    None   None  None
```

In [11]:
```
#Checking the last five rows of df_images dataframe
df_images.tail()
```

Out[11]:
```
                tweet_id                                          jpg_url  \
2070  891327558926688256  https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg
2071  891689557279858688  https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg
2072  891815181378084864  https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg
2073  892177421306343426  https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg
2074  892420643555336193  https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg

      img_num           p1   p1_conf  p1_dog                 p2   p2_conf  \
2070        2       basset  0.555712    True    English_springer  0.225770
2071        1  paper_towel  0.170278   False  Labrador_retriever  0.168086
2072        1    Chihuahua  0.716012    True            malamute  0.078253
2073        1    Chihuahua  0.323581    True            Pekinese  0.090647
2074        1       orange  0.097049   False               bagel  0.085851

      p2_dog                         p3   p3_conf  p3_dog
2070    True  German_short-haired_pointer  0.175219    True
2071    True                    spatula  0.040836   False
2072    True                     kelpie  0.031379    True
2073    True                   papillon  0.068957    True
2074   False                     banana  0.076110   False
```

In [12]:
```
#Checking the last five rows of df_tweets dataframe
df_tweets.tail()
```

Out[12]:
```
                    id  retweet_count  favorite_count
2335  666049248165822465             42             106
2336  666044226329800704            136             292
2337  666033412701032449             43             123
2338  666029285002620928             46             126
2339  666020888022790149            498            2532
```

In [13]:
```
#Checking the dimensions of df dataframe
df.shape
```

```
Out[13]: (2356, 17)

In [14]: #Checking the dimensions of df_images dataframe
         df_images.shape

Out[14]: (2075, 12)

In [15]: #Checking the dimensions of df_tweets dataframe
         df_tweets.shape

Out[15]: (2340, 3)

In [16]: #Checking summary of df dataframe as well as the number of non-Null values and Datatype
         df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB


In [17]: #Checking summary of df_images dataframe as well as the number of non-Null values and D
         df_images.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
```

```
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

```
In [18]: #Checking summary of df_tweets dataframe as well as the number of non-Null values and I
         df_tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2340 entries, 0 to 2339
Data columns (total 3 columns):
id                2340 non-null int64
retweet_count     2340 non-null int64
favorite_count    2340 non-null int64
dtypes: int64(3)
memory usage: 54.9 KB
```

```
In [19]: #Checking summary statistics of df dataframe
         df.describe()
```

Out[19]:

| | tweet_id | in_reply_to_status_id | in_reply_to_user_id | \ |
|---|---|---|---|---|
| count | 2.356000e+03 | 7.800000e+01 | 7.800000e+01 | |
| mean | 7.427716e+17 | 7.455079e+17 | 2.014171e+16 | |
| std | 6.856705e+16 | 7.582492e+16 | 1.252797e+17 | |
| min | 6.660209e+17 | 6.658147e+17 | 1.185634e+07 | |
| 25% | 6.783989e+17 | 6.757419e+17 | 3.086374e+08 | |
| 50% | 7.196279e+17 | 7.038708e+17 | 4.196984e+09 | |
| 75% | 7.993373e+17 | 8.257804e+17 | 4.196984e+09 | |
| max | 8.924206e+17 | 8.862664e+17 | 8.405479e+17 | |

| | retweeted_status_id | retweeted_status_user_id | rating_numerator | \ |
|---|---|---|---|---|
| count | 1.810000e+02 | 1.810000e+02 | 2356.000000 | |
| mean | 7.720400e+17 | 1.241698e+16 | 13.126486 | |
| std | 6.236928e+16 | 9.599254e+16 | 45.876648 | |
| min | 6.661041e+17 | 7.832140e+05 | 0.000000 | |
| 25% | 7.186315e+17 | 4.196984e+09 | 10.000000 | |
| 50% | 7.804657e+17 | 4.196984e+09 | 11.000000 | |
| 75% | 8.203146e+17 | 4.196984e+09 | 12.000000 | |
| max | 8.874740e+17 | 7.874618e+17 | 1776.000000 | |

| | rating_denominator |
|---|---|

```
count         2356.000000
mean            10.455433
std              6.745237
min              0.000000
25%             10.000000
50%             10.000000
75%             10.000000
max            170.000000
```

In [20]:
```
#Checking summary statistics of df_images dataframe
df_images.describe()
```

Out[20]:

| | tweet_id | img_num | p1_conf | p2_conf | p3_conf |
|---|---|---|---|---|---|
| count | 2.075000e+03 | 2075.000000 | 2075.000000 | 2.075000e+03 | 2.075000e+03 |
| mean | 7.384514e+17 | 1.203855 | 0.594548 | 1.345886e-01 | 6.032417e-02 |
| std | 6.785203e+16 | 0.561875 | 0.271174 | 1.006657e-01 | 5.090593e-02 |
| min | 6.660209e+17 | 1.000000 | 0.044333 | 1.011300e-08 | 1.740170e-10 |
| 25% | 6.764835e+17 | 1.000000 | 0.364412 | 5.388625e-02 | 1.622240e-02 |
| 50% | 7.119988e+17 | 1.000000 | 0.588230 | 1.181810e-01 | 4.944380e-02 |
| 75% | 7.932034e+17 | 1.000000 | 0.843855 | 1.955655e-01 | 9.180755e-02 |
| max | 8.924206e+17 | 4.000000 | 1.000000 | 4.880140e-01 | 2.734190e-01 |

In [21]:
```
#Checking summary statistics of df_tweets dataframe
df_tweets.describe()
```

Out[21]:

| | id | retweet_count | favorite_count |
|---|---|---|---|
| count | 2.340000e+03 | 2340.000000 | 2340.000000 |
| mean | 7.422176e+17 | 2915.453846 | 7934.817094 |
| std | 6.832564e+16 | 4912.093787 | 12291.490833 |
| min | 6.660209e+17 | 0.000000 | 0.000000 |
| 25% | 6.783394e+17 | 584.500000 | 1369.750000 |
| 50% | 7.186224e+17 | 1361.000000 | 3450.500000 |
| 75% | 7.986954e+17 | 3395.250000 | 9709.500000 |
| max | 8.924206e+17 | 83264.000000 | 163731.000000 |

In [22]:
```
#Checking the number of unique values in each column of df dataframe
df.nunique()
```

Out[22]:
```
tweet_id                    2356
in_reply_to_status_id         77
in_reply_to_user_id           31
timestamp                   2356
source                         4
text                        2356
retweeted_status_id          181
retweeted_status_user_id      25
retweeted_status_timestamp   181
expanded_urls               2218
rating_numerator              40
```

```
rating_denominator              18
name                           957
doggo                            2
floofer                          2
pupper                           2
puppo                            2
dtype: int64
```

In [23]:
```
#Checking the number of unique values in each column of df_images dataframe
df_images.nunique()
```

Out[23]:
```
tweet_id    2075
jpg_url     2009
img_num        4
p1           378
p1_conf     2006
p1_dog         2
p2           405
p2_conf     2004
p2_dog         2
p3           408
p3_conf     2006
p3_dog         2
dtype: int64
```

In [24]:
```
#Checking the number of unique values in each column of df_tweets dataframe
df_tweets.nunique()
```

Out[24]:
```
id                2340
retweet_count     1718
favorite_count    2001
dtype: int64
```

In [25]:
```
#Checking for duplicates in df dataframe
sum(df.duplicated())
```

Out[25]:
```
0
```

In [26]:
```
#Checking for duplicates in df_images dataframe
sum(df_images.duplicated())
```

Out[26]:
```
0
```

In [27]:
```
#Checking for duplicates in df_tweets dataframe
sum(df_tweets.duplicated())
```

Out[27]:
```
0
```

In [28]:
```
#Checking value counts of name from df dataframe
df.name.value_counts()
```

Out[28]:

| | |
|---|---|
| None | 745 |
| a | 55 |
| Charlie | 12 |
| Cooper | 11 |
| Oliver | 11 |
| Lucy | 11 |
| Tucker | 10 |
| Penny | 10 |
| Lola | 10 |
| Winston | 9 |
| Bo | 9 |
| the | 8 |
| Sadie | 8 |
| Buddy | 7 |
| an | 7 |
| Daisy | 7 |
| Toby | 7 |
| Bailey | 7 |
| Jack | 6 |
| Scout | 6 |
| Koda | 6 |
| Stanley | 6 |
| Bella | 6 |
| Leo | 6 |
| Oscar | 6 |
| Rusty | 6 |
| Dave | 6 |
| Jax | 6 |
| Milo | 6 |
| Larry | 5 |
| ... | |
| Jiminus | 1 |
| Kobe | 1 |
| Wishes | 1 |
| Glacier | 1 |
| Asher | 1 |
| Georgie | 1 |
| Tess | 1 |
| Marq | 1 |
| Joey | 1 |
| Chesterson | 1 |
| Gert | 1 |
| Chesney | 1 |
| Tedders | 1 |
| Jeffrie | 1 |
| Nico | 1 |
| Bobbay | 1 |
| Bauer | 1 |

```
Nugget          1
Tycho           1
Rooney          1
Alfy            1
Marvin          1
Sage            1
Raphael         1
Walker          1
Yoda            1
old             1
Wiggles         1
Hubertson       1
Tito            1
Name: name, Length: 957, dtype: int64
```

```
In [29]: #More Checking on name field df dataframe
         df.name.value_counts().sort_index()
```

```
Out[29]: Abby            2
         Ace             1
         Acro            1
         Adele           1
         Aiden           1
         Aja             1
         Akumi           1
         Al              1
         Albert          2
         Albus           2
         Aldrick         1
         Alejandro       1
         Alexander       1
         Alexanderson    1
         Alf             1
         Alfie           5
         Alfy            1
         Alice           2
         Amber           1
         Ambrose         1
         Amy             1
         Amélie          1
         Anakin          2
         Andru           1
         Andy            1
         Angel           1
         Anna            1
         Anthony         1
         Antony          1
         Apollo          1
```

```
                ..
Ziva             1
Zoe              1
Zoey             3
Zooey            1
Zuzu             1
a               55
actually         2
all              1
an               7
by               1
getting          2
his              1
incredibly       1
infuriating      1
just             4
life             1
light            1
mad              2
my               1
not              2
officially       1
old              1
one              4
quite            4
space            1
such             1
the              8
this             1
unacceptable     1
very             5
Name: name, Length: 957, dtype: int64
```

In [30]:
```
#Checking value counts for rating_numerator from df dataframe
df.rating_numerator.value_counts().sort_index()
```

Out[30]:
```
0         2
1         9
2         9
3        19
4        17
5        37
6        32
7        55
8       102
9       158
10      461
11      464
```

```
12      558
13      351
14       54
15        2
17        1
20        1
24        1
26        1
27        1
44        1
45        1
50        1
60        1
75        2
80        1
84        1
88        1
99        1
121       1
143       1
144       1
165       1
182       1
204       1
420       2
666       1
960       1
1776      1
Name: rating_numerator, dtype: int64
```

In [31]:
```
#Checking value counts for rating_denominator from df dataframe
df.rating_denominator.value_counts().sort_index()
```

Out[31]:
```
0         1
2         1
7         1
10     2333
11        3
15        1
16        1
20        2
40        1
50        3
70        1
80        2
90        1
110       1
120       1
```

```
130       1
150       1
170       1
Name: rating_denominator, dtype: int64
```

#### 0.3.3 Quality

**df dataframe**

- name field has entries that are not real names such as "a", "actually", "the"
- There are tweets not related to dogs, but instead related to other animals and other things.
- Several records showing "None" for name field
- rating_numerator has values with less than "10" or too high (with three or more digits)
- rating_denominator has values different than "10"
- Data contains retweets which means there are duplicates
- Source field difficult to read/ understand

**df_images dataframe**

- Values showing under P1, P2, and P3 which are meant to be dog's breed may not be an actual dog breed

**df_tweets dataframe**

- Column referring to tweet id is called "id" instead of "tweet_id"

#### 0.3.4 Tidiness

**df dataframe**

- Dog stages such as: doggo, floofer, pupper and puppo showing in separate columns

**df_images dataframe**

- P1, P2, and P3 which are meant to be dog's breed are in different columns.

# 1 Cleaning Data

```
In [136]: #Before starting with cleaning tasks, I will make a copy of all three datasets to pres

          df_clean = df.copy()
          df_images_clean = df_images.copy()
          df_tweets_clean = df_tweets.copy()
```

## 1.1 Quality - 1

**Define** Correct column name from df_tweets_clean referring to tweet id which is currently called "id" to "tweet_id" to keep consistency with other two datasets and help the join process to be smoother

**Code**

```
In [137]: #renaming id to tweet_id for consistency with other datasets
          df_tweets_clean = df_tweets_clean.rename(columns={"id": "tweet_id"})
```

**Test**

```
In [138]: df_tweets_clean.head()
```

Out[138]:

| | tweet_id | retweet_count | favorite_count |
|---|---|---|---|
| 0 | 892420643555336193 | 8279 | 37912 |
| 1 | 892177421306343426 | 6117 | 32564 |
| 2 | 891815181378084864 | 4051 | 24517 |
| 3 | 891689557279858688 | 8421 | 41266 |
| 4 | 891327558926688256 | 9121 | 39440 |

### 1.2 Tidiness - 1

**Define** Join/ merge all three dataframes for better understanding of the data

**Code**

```
In [139]: #Joining df_clean with df_images_clean and dropping records with missing images
          df_clean = pd.merge(df_clean, df_images_clean, on = 'tweet_id', how = 'inner')

          #Joining df_clean with df_tweets_clean and dropping records with missing images
          df_clean = pd.merge(df_clean, df_tweets_clean, on = 'tweet_id', how = 'inner')
```

**Test**

```
In [140]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2067 entries, 0 to 2066
Data columns (total 30 columns):
tweet_id                    2067 non-null int64
in_reply_to_status_id       23 non-null float64
in_reply_to_user_id         23 non-null float64
timestamp                   2067 non-null object
source                      2067 non-null object
text                        2067 non-null object
retweeted_status_id         75 non-null float64
retweeted_status_user_id    75 non-null float64
retweeted_status_timestamp  75 non-null object
expanded_urls               2067 non-null object
rating_numerator            2067 non-null int64
rating_denominator          2067 non-null int64
name                        2067 non-null object
doggo                       2067 non-null object
```

```
floofer                     2067 non-null object
pupper                      2067 non-null object
puppo                       2067 non-null object
jpg_url                     2067 non-null object
img_num                     2067 non-null int64
p1                          2067 non-null object
p1_conf                     2067 non-null float64
p1_dog                      2067 non-null bool
p2                          2067 non-null object
p2_conf                     2067 non-null float64
p2_dog                      2067 non-null bool
p3                          2067 non-null object
p3_conf                     2067 non-null float64
p3_dog                      2067 non-null bool
retweet_count               2067 non-null int64
favorite_count              2067 non-null int64
dtypes: bool(3), float64(7), int64(6), object(14)
memory usage: 458.2+ KB
```

### 1.3 Tidiness - 2

**Define** Create a *breed* Column to remove individual columns related to breed, such as p1, p1_conf, p1_dog...

#### Code

```
In [141]: #Creating separate breeds_df dataframe to investigate different breeds columns
          #to later keep the best match
          #The best match will be determined based on 'px_dog' == True

          breeds_df = df_clean

          breeds_df['breed'] = [i['p1'] if i['p1_dog'] == True
                                else i['p2'] if i['p2_dog'] == True
                                else i['p3'] if i['p3_dog'] == True
                                else 'None' for index, i in breeds_df.iterrows() ]
```

```
In [142]: #Cheking best_breed values
          breeds_df.breed.value_counts()
```

```
Out[142]: None                  323
          golden_retriever      173
          Labrador_retriever    113
          Pembroke               95
          Chihuahua              93
          pug                    65
          toy_poodle             52
```

| | |
|---|---|
| chow | 51 |
| Samoyed | 45 |
| Pomeranian | 42 |
| malamute | 34 |
| cocker_spaniel | 33 |
| Chesapeake_Bay_retriever | 31 |
| French_bulldog | 30 |
| miniature_pinscher | 26 |
| Cardigan | 23 |
| Eskimo_dog | 22 |
| Staffordshire_bullterrier | 22 |
| beagle | 21 |
| German_shepherd | 21 |
| Shih-Tzu | 20 |
| Siberian_husky | 20 |
| Lakeland_terrier | 19 |
| Maltese_dog | 19 |
| Shetland_sheepdog | 19 |
| Rottweiler | 19 |
| kuvasz | 19 |
| Italian_greyhound | 17 |
| basset | 17 |
| American_Staffordshire_terrier | 16 |
| | ... |
| Weimaraner | 4 |
| Tibetan_terrier | 4 |
| Rhodesian_ridgeback | 4 |
| keeshond | 4 |
| Scottish_deerhound | 4 |
| giant_schnauzer | 4 |
| Afghan_hound | 4 |
| komondor | 3 |
| Brabancon_griffon | 3 |
| curly-coated_retriever | 3 |
| toy_terrier | 3 |
| cairn | 3 |
| Leonberg | 3 |
| Irish_water_spaniel | 3 |
| briard | 3 |
| Greater_Swiss_Mountain_dog | 3 |
| Sussex_spaniel | 2 |
| black-and-tan_coonhound | 2 |
| Australian_terrier | 2 |
| groenendael | 2 |
| Appenzeller | 2 |
| wire-haired_fox_terrier | 2 |
| silky_terrier | 1 |
| Irish_wolfhound | 1 |

```
Bouvier_des_Flandres                1
standard_schnauzer                  1
EntleBucher                         1
Scotch_terrier                      1
Japanese_spaniel                    1
clumber                             1
Name: breed, Length: 114, dtype: int64
```

```
In [143]: #Updating df_clean dataframe to keep only relevant records
          df_clean= pd.concat([df_clean, breeds_df]).drop_duplicates(['tweet_id'],keep='last')
```

```
In [144]: #Keep records where breed is different to "None"
          df_clean = df_clean[df_clean.breed != 'None']
```

```
In [145]: #Drop individual columns related to breed
          columns = ['p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_
          df_clean = df_clean.drop(columns, axis = 1)
```

**Test**

```
In [146]: df_clean.breed.value_counts()
```

```
Out[146]: golden_retriever                  173
          Labrador_retriever                113
          Pembroke                           95
          Chihuahua                          93
          pug                                65
          toy_poodle                         52
          chow                               51
          Samoyed                            45
          Pomeranian                         42
          malamute                           34
          cocker_spaniel                     33
          Chesapeake_Bay_retriever           31
          French_bulldog                     30
          miniature_pinscher                 26
          Cardigan                           23
          Eskimo_dog                         22
          Staffordshire_bullterrier          22
          beagle                             21
          German_shepherd                    21
          Shih-Tzu                           20
          Siberian_husky                     20
          Lakeland_terrier                   19
          Maltese_dog                        19
          Shetland_sheepdog                  19
          Rottweiler                         19
          kuvasz                             19
          Italian_greyhound                  17
```

```
basset                              17
American_Staffordshire_terrier      16
West_Highland_white_terrier         16
                                   ...
Weimaraner                           4
Tibetan_terrier                      4
Rhodesian_ridgeback                  4
keeshond                             4
Scottish_deerhound                   4
giant_schnauzer                      4
Afghan_hound                         4
komondor                             3
Brabancon_griffon                    3
curly-coated_retriever               3
toy_terrier                          3
cairn                                3
Leonberg                             3
Irish_water_spaniel                  3
briard                               3
Greater_Swiss_Mountain_dog           3
Sussex_spaniel                       2
black-and-tan_coonhound              2
Australian_terrier                   2
groenendael                          2
Appenzeller                          2
wire-haired_fox_terrier              2
silky_terrier                        1
Irish_wolfhound                      1
Bouvier_des_Flandres                 1
standard_schnauzer                   1
EntleBucher                          1
Scotch_terrier                       1
Japanese_spaniel                     1
clumber                              1
Name: breed, Length: 113, dtype: int64
```

```
In [147]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1744 entries, 1 to 2066
Data columns (total 22 columns):
tweet_id                    1744 non-null int64
in_reply_to_status_id       20 non-null float64
in_reply_to_user_id         20 non-null float64
timestamp                   1744 non-null object
source                      1744 non-null object
text                        1744 non-null object
retweeted_status_id         60 non-null float64
```

```
retweeted_status_user_id      60 non-null float64
retweeted_status_timestamp    60 non-null object
expanded_urls                 1744 non-null object
rating_numerator              1744 non-null int64
rating_denominator            1744 non-null int64
name                          1744 non-null object
doggo                         1744 non-null object
floofer                       1744 non-null object
pupper                        1744 non-null object
puppo                         1744 non-null object
jpg_url                       1744 non-null object
img_num                       1744 non-null int64
retweet_count                 1744 non-null int64
favorite_count                1744 non-null int64
breed                         1744 non-null object
dtypes: float64(4), int64(6), object(12)
memory usage: 313.4+ KB
```

### 1.4 Quality - 2

**Define** Remove records from df_clean dataframe that contains retweets, thus will help to remove duplicate records

#### Code

```
In [148]: #Checking the number of records that contains data on retweeted_status_id
          len(df_clean[df_clean.retweeted_status_id.isnull() == False])
```

```
Out[148]: 60
```

```
In [149]: #Removing the records that contains data on retweeted_status_id
          df_clean = df_clean[df_clean.retweeted_status_id.isnull()]
```

#### Test

```
In [150]: #Checking to make sure that there are no records that contains data on retweeted_statu
          len(df_clean[df_clean.retweeted_status_id.isnull() == False])
```

```
Out[150]: 0
```

```
In [151]: #Checking info
          df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1684 entries, 1 to 2066
Data columns (total 22 columns):
tweet_id                      1684 non-null int64
in_reply_to_status_id         20 non-null float64
```

```
in_reply_to_user_id           20 non-null float64
timestamp                     1684 non-null object
source                        1684 non-null object
text                          1684 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 1684 non-null object
rating_numerator              1684 non-null int64
rating_denominator            1684 non-null int64
name                          1684 non-null object
doggo                         1684 non-null object
floofer                       1684 non-null object
pupper                        1684 non-null object
puppo                         1684 non-null object
jpg_url                       1684 non-null object
img_num                       1684 non-null int64
retweet_count                 1684 non-null int64
favorite_count                1684 non-null int64
breed                         1684 non-null object
dtypes: float64(4), int64(6), object(12)
memory usage: 302.6+ KB
```

### 1.5 Quality - 3

**Define** After removing retweets records the fields: *retweeted_status_id*, *retweeted_status_user_id* and *retweeted_status_timestamp* now contains no information, thus we should proceed to remove these columns, since they are no longer relevant.

#### Code

```
In [152]: #Dropping Columns
          df_clean.drop(['retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_ti
```

#### Test

```
In [153]: #Checking Info to make sure the three columns were removed
          df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1684 entries, 1 to 2066
Data columns (total 19 columns):
tweet_id                1684 non-null int64
in_reply_to_status_id   20 non-null float64
in_reply_to_user_id     20 non-null float64
timestamp               1684 non-null object
source                  1684 non-null object
```

```
text                    1684 non-null object
expanded_urls           1684 non-null object
rating_numerator        1684 non-null int64
rating_denominator      1684 non-null int64
name                    1684 non-null object
doggo                   1684 non-null object
floofer                 1684 non-null object
pupper                  1684 non-null object
puppo                   1684 non-null object
jpg_url                 1684 non-null object
img_num                 1684 non-null int64
retweet_count           1684 non-null int64
favorite_count          1684 non-null int64
breed                   1684 non-null object
dtypes: float64(2), int64(6), object(11)
memory usage: 263.1+ KB
```

### 1.6 Quality - 4

**Define** Clean the *source* field from df_clean dataframe to remove URLs and tags and make the *source* field is more readable

#### Code

```
In [154]: #Checking source field from df_clean
          df_clean['source'].value_counts()
```

```
Out[154]: <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
          <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
          <a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>
          Name: source, dtype: int64
```

```
In [155]: #Clean Twitter for iPhone records
          df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitter.com/downl

          #Clean Twitter Web Client records
          df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitter.com" rel=

          #Clean Twieet Deck records
          df_clean['source'] = df_clean['source'].str.replace('<a href="https://about.twitter.co
```

#### Test

```
In [156]: #Veryfying Source fields are now clean
          df_clean['source'].value_counts()
```

```
Out[156]: Twitter for iPhone    1654
          Twitter Web Client      22
          TweetDeck                8
          Name: source, dtype: int64
```

### 1.7 Quality - 5

**Define** Clean *name* field which has entries that are not real names such as "a", "actually", "the", and do some research by looking at the *text* field to find out if there are real Dog names

#### Code

```
In [157]: #Number of records that contain names with lowercase which may be incorrect names
          #such as "a", "actually", "the"
          len(df_clean[df_clean.name.str.islower()])
```

```
Out[157]: 80
```

```
In [158]: #Save these specific records (with lowercase names) to a dataframe for further analysi
          lowercase_names = df_clean[df_clean.name.str.islower()]

          #Modifying max_colwidth to read sample text to see if there is a different way to iden
          pd.options.display.max_colwidth =180

          #Get sample tweets to analyze text and search for patterns that would help
          #us identify the real dog name
          lowercase_names['text'].sample(10)
```

```
Out[158]: 833       This is an Iraqi Speed Kangaroo. It is not a dog. Please only send in dogs.
          2059    Here is a Siberian heavily armored polar bear mix. Strong owner. 10/10 I would
          686           This is my dog. Her name is Zoey. She knows I've been rating other dogs.
          897     We only rate dogs. Please stop sending in non-canines like this Alaskan Flop T
          810     This is one of the most reckless puppers I've ever seen. How she got a license
          817     This is a mighty rare blue-tailed hammer sherk. Human almost lost a limb tryin
          1347    This is the newly formed pupper a capella group. They're just starting out but
          1999          This is a Dasani Kingfisher from Maine. His name is Daryl. Daryl doesn't
          1781    This is a Helvetica Listerine named Rufus. This time Rufus will be ready for t
          1281                                Stop sending in lobsters. This is the final warning
          Name: text, dtype: object
```

```
In [159]: #By looking at the sample texts, we can see that there could be tweets with real Dog n
          #and most of these tweets contained the words "name" or "named"

          #Retrieving the records from lowercase_names that contain words "name", this will also

          lowercase_names[lowercase_names.text.str.contains("name")]
```

```
Out[159]:                tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
          686   765395769549590528                    NaN                  NaN
          1574  675706639471788032                    NaN                  NaN
          1671  673636718965334016                    NaN                  NaN
          1750  671743150407421952                    NaN                  NaN
          1781  671147085991960577                    NaN                  NaN
          1831  670427002554466305                    NaN                  NaN
```

```
1843  670303360680108032                    NaN                  NaN
1875  669564461267722241                    NaN                  NaN
1904  668955713004314625                    NaN                  NaN
1917  668636665813057536                    NaN                  NaN
1930  668507509523615744                    NaN                  NaN
1947  668171859951755264                    NaN                  NaN
1961  667861340749471744                    NaN                  NaN
1967  667773195014021121                    NaN                  NaN
1976  667538891197542400                    NaN                  NaN
1985  667470559035432960                    NaN                  NaN
1999  667177989038297088                    NaN                  NaN
2022  666781792255496192                    NaN                  NaN
2025  666701168228331520                    NaN                  NaN

                     timestamp              source  \
686   2016-08-16 03:52:26 +0000  Twitter for iPhone
1574  2015-12-12 15:59:51 +0000  Twitter for iPhone
1671  2015-12-06 22:54:44 +0000  Twitter for iPhone
1750  2015-12-01 17:30:22 +0000  Twitter for iPhone
1781  2015-11-30 02:01:49 +0000  Twitter for iPhone
1831  2015-11-28 02:20:27 +0000  Twitter for iPhone
1843  2015-11-27 18:09:09 +0000  Twitter for iPhone
1875  2015-11-25 17:13:02 +0000  Twitter for iPhone
1904  2015-11-24 00:54:05 +0000  Twitter for iPhone
1917  2015-11-23 03:46:18 +0000  Twitter for iPhone
1930  2015-11-22 19:13:05 +0000  Twitter for iPhone
1947  2015-11-21 20:59:20 +0000  Twitter for iPhone
1961  2015-11-21 00:25:26 +0000  Twitter for iPhone
1967  2015-11-20 18:35:10 +0000  Twitter Web Client
1976  2015-11-20 03:04:08 +0000  Twitter Web Client
1985  2015-11-19 22:32:36 +0000  Twitter Web Client
1999  2015-11-19 03:10:02 +0000  Twitter for iPhone
2022  2015-11-18 00:55:42 +0000  Twitter for iPhone
2025  2015-11-17 19:35:19 +0000  Twitter for iPhone


686         This is my dog. Her name is Zoey. She knows I've been rating other dogs. S
1574  This is a Sizzlin Menorah spaniel from Brooklyn named Wylie. Lovable eyes. Chill
1671   This is a Lofted Aphrodisiac Terrier named Kip. Big fan of bed n breakfasts. Fi
1750      This is a Tuscaloosa Alcatraz named Jacob (Yacb). Loves to sit in swing. Ste
1781  This is a Helvetica Listerine named Rufus. This time Rufus will be ready for the
1831   This is a Deciduous Trimester mix named Spork. Only 1 ear works. No seat belt.
1843    This is a Speckled Cauliflower Yosemite named Hemry. He's terrified of intrude
1875    This is a Coriander Baton Rouge named Alfredo. Loves to cuddle with smaller we
1904  This is a Slovakian Helter Skelter Feta named Leroi. Likes to skip on roofs. Goo
1917   This is an Irish Rigatoni terrier named Berta. Completely made of rope. No eyes
1930    This is a Birmingham Quagmire named Chuk. Loves to relax and watch the game wh
1947              This is a Trans Siberian Kellogg named Alfonso. Huge ass eyeballs. Ac
```

```
1961  This is a Shotokon Macadamia mix named Cheryl. Sophisticated af. Looks like a di
1967      This is a rare Hungarian Pinot named Jessiga. She is either mid-stroke or go
1976                        This is a southwest Coriander named Klint. Hat looks expen
1985    This is a northern Wahoo named Kohl. He runs this town. Chases tumbleweeds. Dr
1999        This is a Dasani Kingfisher from Maine. His name is Daryl. Daryl doesn't l
2022                                  This is a purebred Bacardi named Octaviath. Can
2025   This is a golden Buckminsterfullerene named Johm. Drives trucks. Lumberjack (?)

                                                      expanded_urls  \
686   https://twitter.com/dog_rates/status/765395769549590528/photo/1
1574  https://twitter.com/dog_rates/status/675706639471788032/photo/1
1671  https://twitter.com/dog_rates/status/673636718965334016/photo/1
1750  https://twitter.com/dog_rates/status/671743150407421952/photo/1
1781  https://twitter.com/dog_rates/status/671147085991960577/photo/1
1831  https://twitter.com/dog_rates/status/670427002554466305/photo/1
1843  https://twitter.com/dog_rates/status/670303360680108032/photo/1
1875  https://twitter.com/dog_rates/status/669564461267722241/photo/1
1904  https://twitter.com/dog_rates/status/668955713004314625/photo/1
1917  https://twitter.com/dog_rates/status/668636665813057536/photo/1
1930  https://twitter.com/dog_rates/status/668507509523615744/photo/1
1947  https://twitter.com/dog_rates/status/668171859951755264/photo/1
1961  https://twitter.com/dog_rates/status/667861340749471744/photo/1
1967  https://twitter.com/dog_rates/status/667773195014021121/photo/1
1976  https://twitter.com/dog_rates/status/667538891197542400/photo/1
1985  https://twitter.com/dog_rates/status/667470559035432960/photo/1
1999  https://twitter.com/dog_rates/status/667177989038297088/photo/1
2022  https://twitter.com/dog_rates/status/666781792255496192/photo/1
2025  https://twitter.com/dog_rates/status/666701168228331520/photo/1

      rating_numerator  rating_denominator name doggo floofer pupper puppo  \
686                 13                  10   my  None    None   None  None
1574                10                  10    a  None    None   None  None
1671                10                  10    a  None    None   None  None
1750                11                  10    a  None    None   None  None
1781                 9                  10    a  None    None   None  None
1831                 9                  10    a  None    None   None  None
1843                 9                  10    a  None    None   None  None
1875                10                  10    a  None    None   None  None
1904                10                  10    a  None    None   None  None
1917                10                  10   an  None    None   None  None
1930                10                  10    a  None    None   None  None
1947                 7                  10    a  None    None   None  None
1961                 9                  10    a  None    None   None  None
1967                 8                  10    a  None    None   None  None
1976                 9                  10    a  None    None   None  None
1985                11                  10    a  None    None   None  None
1999                 8                  10    a  None    None   None  None
2022                10                  10    a  None    None   None  None
```

```
2025                8                  10     a  None    None   None  None

                                                  jpg_url  img_num  retweet_count  \
686   https://pbs.twimg.com/media/Cp87Y0jXYAQyjuV.jpg        1          3663
1574  https://pbs.twimg.com/media/CWCXj35VEAIFvtk.jpg        1           102
1671  https://pbs.twimg.com/media/CVk9ApFWUAA-S1s.jpg        1           371
1750  https://pbs.twimg.com/media/CVKC1IfWIAAsQks.jpg        1           236
1781  https://pbs.twimg.com/media/CVBktzQXAAAPpUA.jpg        1           230
1831  https://pbs.twimg.com/media/CU3VzVwWwAAAsst.jpg        1           167
1843  https://pbs.twimg.com/media/CU1lWFaVAAAlOHG.jpg        1           139
1875  https://pbs.twimg.com/media/CUrFUvDVAAA9H-F.jpg        1           127
1904  https://pbs.twimg.com/media/CUibq3uVAAAup_O.jpg        1            73
1917  https://pbs.twimg.com/media/CUd5gBGWwAAOIVA.jpg        1           499
1930  https://pbs.twimg.com/media/CUcECBYWcAAzFRg.jpg        1           111
1947  https://pbs.twimg.com/media/CUXSwy8W4AA6uet.jpg        1           199
1961  https://pbs.twimg.com/media/CUS4WJ-UsAEJj10.jpg        1            78
1967  https://pbs.twimg.com/media/CURoLrOVEAAaWdR.jpg        1            57
1976  https://pbs.twimg.com/media/CUOTFZOW4AABsfW.jpg        1            65
1985  https://pbs.twimg.com/media/CUNU78YWEAECmpB.jpg        1           100
1999  https://pbs.twimg.com/media/CUJK18UWEAEg7AR.jpg        1            55
2022  https://pbs.twimg.com/media/CUDigRXXIAATI_H.jpg        1           192
2025  https://pbs.twimg.com/media/CUCZLHlUAAAeAig.jpg        1           219

      favorite_count                      breed
686            27941                   Pembroke
1574             662            English_springer
1671            1127                        pug
1750             747                toy_poodle
1781             683           Yorkshire_terrier
1831             525                toy_terrier
1843             433           Shetland_sheepdog
1875             391                toy_poodle
1904             283              cocker_spaniel
1917            1052                   komondor
1930             330                    basenji
1947             498                   Chihuahua
1961             244                    malamute
1967             236  West_Highland_white_terrier
1976             204           Yorkshire_terrier
1985             258                toy_poodle
1999             190                      vizsla
2022             383           Italian_greyhound
2025             425          Labrador_retriever
```

In [160]:
```
#Assign to new lowercase_names dataframe only the 22 records that contain a name:

lowercase_names = lowercase_names[lowercase_names.text.str.contains("name")]
```

```
In [160]: len(lowercase_names)

Out[160]: 19

In [161]: #ignore warnings
          warnings.filterwarnings("ignore")

In [162]: #Extracting dog names where text contains the word "named"
          extract_name = lowercase_names[lowercase_names.text.str.contains("name")]

          named =extract_name['text'].str.extract(r"named\s(\w+)")
          named =named[named.isnull()==False]
          named

Out[162]: 1574        Wylie
          1671          Kip
          1750        Jacob
          1781        Rufus
          1831        Spork
          1843        Hemry
          1875      Alfredo
          1904        Leroi
          1917        Berta
          1930         Chuk
          1947      Alfonso
          1961       Cheryl
          1967      Jessiga
          1976        Klint
          1985         Kohl
          2022    Octaviath
          2025         Johm
          Name: text, dtype: object

In [163]: #Extracting dog names where text contains "name is"
          name_is =extract_name['text'].str.extract(r"name is\s(\w+)")
          name_is = name_is[name_is.isnull()== False]
          name_is

Out[163]: 686      Zoey
          1999    Daryl
          Name: text, dtype: object

In [164]: #Appending name and name is results to new name
          new_name =named.append(name_is)
          new_name

Out[164]: 1574        Wylie
          1671          Kip
          1750        Jacob
```

```
1781        Rufus
1831        Spork
1843        Hemry
1875      Alfredo
1904        Leroi
1917        Berta
1930         Chuk
1947      Alfonso
1961       Cheryl
1967      Jessiga
1976        Klint
1985         Kohl
2022    Octaviath
2025         Johm
686          Zoey
1999        Daryl
Name: text, dtype: object
```

In [165]:
```
#In lowercase_names dataframe replace "name" values with newly found Dog names
lowercase_names['name'] = new_name
lowercase_names.head()
```

Out[165]:

| | tweet_id | in_reply_to_status_id | in_reply_to_user_id | timestamp | source | text | expanded_urls |
|---|---|---|---|---|---|---|---|
| 686 | 765395769549590528 | NaN | NaN | 2016-08-16 03:52:26 +0000 | Twitter for iPhone | This is my dog. Her name is Zoey. She knows I've been rating other dogs. S | https://twitter.com/dog_rates/status/765395769549590528/photo/1 |
| 1574 | 675706639471788032 | NaN | NaN | 2015-12-12 15:59:51 +0000 | Twitter for iPhone | This is a Sizzlin Menorah spaniel from Brooklyn named Wylie. Lovable eyes. Chill | https://twitter.com/dog_rates/status/675706639471788032/photo/1 |
| 1671 | 673636718965334016 | NaN | NaN | 2015-12-06 22:54:44 +0000 | Twitter for iPhone | This is a Lofted Aphrodisiac Terrier named Kip. Big fan of bed n breakfasts. Fi | https://twitter.com/dog_rates/status/673636718965334016/photo/1 |
| 1750 | 671743150407421952 | NaN | NaN | 2015-12-01 17:30:22 +0000 | Twitter for iPhone | This is a Tuscaloosa Alcatraz named Jacob (Yacb). Loves to sit in swing. Ste | https://twitter.com/dog_rates/status/671743150407421952/photo/1 |
| 1781 | 671147085991960577 | NaN | NaN | 2015-11-30 02:01:49 +0000 | Twitter for iPhone | This is a Helvetica Listerine named Rufus. This time Rufus will be ready for the | |

```
1781  https://twitter.com/dog_rates/status/671147085991960577/photo/1

      rating_numerator  rating_denominator   name doggo floofer pupper puppo  \
686                 13                  10   Zoey  None    None   None  None
1574                10                  10  Wylie  None    None   None  None
1671                10                  10    Kip  None    None   None  None
1750                11                  10  Jacob  None    None   None  None
1781                 9                  10  Rufus  None    None   None  None

                                              jpg_url  img_num  retweet_count  \
686   https://pbs.twimg.com/media/Cp87Y0jXYAQyjuV.jpg        1           3663
1574  https://pbs.twimg.com/media/CWCXj35VEAIFvtk.jpg        1            102
1671  https://pbs.twimg.com/media/CVk9ApFWUAA-S1s.jpg        1            371
1750  https://pbs.twimg.com/media/CVKC1IfWIAAsQks.jpg        1            236
1781  https://pbs.twimg.com/media/CVBktzQXAAAPpUA.jpg        1            230

      favorite_count             breed
686            27941          Pembroke
1574             662  English_springer
1671            1127               pug
1750             747        toy_poodle
1781             683  Yorkshire_terrier
```

In [166]:
```
#Update dog names in df_clean dataframe for tweet_ids where a new dog name was found
df_clean= pd.concat([df_clean, lowercase_names]).drop_duplicates(['tweet_id'],keep='la
```

**Test**

In [167]:
```
#Checking individual records where we know a new dog name should be assigned to make s
df_clean[df_clean.tweet_id == 765395769549590528 ]
```

Out[167]:
```
               tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
686  765395769549590528                    NaN                  NaN

                     timestamp              source  \
686  2016-08-16 03:52:26 +0000  Twitter for iPhone


686  This is my dog. Her name is Zoey. She knows I've been rating other dogs. She's no

                                                    expanded_urls  \
686  https://twitter.com/dog_rates/status/765395769549590528/photo/1

     rating_numerator  rating_denominator  name doggo floofer pupper puppo  \
686                13                  10  Zoey  None    None   None  None

                                              jpg_url  img_num  retweet_count  \
686  https://pbs.twimg.com/media/Cp87Y0jXYAQyjuV.jpg        1           3663
```

```
             favorite_count     breed
686           27941  Pembroke
```

```
In [168]: #Checking again for lowercase names the count has been reduced from 98 to 76
          len(df_clean[df_clean.name.str.islower()])
```

```
Out[168]: 61
```

```
In [169]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1684 entries, 1 to 2025
Data columns (total 19 columns):
tweet_id                 1684 non-null int64
in_reply_to_status_id    20 non-null float64
in_reply_to_user_id      20 non-null float64
timestamp                1684 non-null object
source                   1684 non-null object
text                     1684 non-null object
expanded_urls            1684 non-null object
rating_numerator         1684 non-null int64
rating_denominator       1684 non-null int64
name                     1684 non-null object
doggo                    1684 non-null object
floofer                  1684 non-null object
pupper                   1684 non-null object
puppo                    1684 non-null object
jpg_url                  1684 non-null object
img_num                  1684 non-null int64
retweet_count            1684 non-null int64
favorite_count           1684 non-null int64
breed                    1684 non-null object
dtypes: float64(2), int64(6), object(11)
memory usage: 263.1+ KB
```

```
In [170]: #Checking names
          df_clean['name'].value_counts()
```

```
Out[170]: None          419
          a              29
          Cooper         10
          Lucy           10
          Charlie         9
          Tucker          9
          Oliver          9
          Penny           8
          Daisy           7
```

```
the             7
Sadie           7
Winston         7
Toby            6
Lola            6
Koda            6
Jax             6
Oscar           5
Rusty           5
Leo             5
Stanley         5
Bella           5
Bo              5
Larry           4
Dave            4
Cassie          4
Gus             4
Duke            4
Dexter          4
Bentley         4
Maggie          4
              ...
Mason           1
Goliath         1
Yoda            1
Ruffles         1
Ace             1
Jeffri          1
Cali            1
Terrance        1
Reagan          1
Jazz            1
Rumble          1
Quinn           1
Kevon           1
incredibly      1
Mingus          1
Dawn            1
Florence        1
Johm            1
Ginger          1
Timison         1
Stella          1
Ben             1
Bobb            1
Lassie          1
Berta           1
Jamesy          1
```

```
Odin            1
Penelope        1
Kramer          1
Chuk            1
Name: name, Length: 867, dtype: int64
```

### 1.8 Quality - 6

**Define** Identify and remove tweets that may no be related to Dogs.

While analyzing the text to find valid dog names, I found many tweets that were not related to dogs, but instead related to other animals and other things.

These tweets contained phrases such as: "don't rate", "stop sending", "only send dogs", "only rate dos", "whithout dog"

#### Code

```
In [171]: #Save these specific records to their own dataframes
          no_dog = df_clean[df_clean.text.str.contains("don't rate|stop sending|only send dogs|o

          #sample text to make sure these are irrelevant tweets
          no_dog['text'].sample(5)
```

```
Out[171]: 1000                              This is a taco. We only rate dogs. Please only send i
          583                                          Who keeps sending in pictures without
          1131                               Really guys? Again? I know this is a rare Albanian
          56               Please don't send in photos without dogs in them. We're not @porch_ra
          153     Guys, we only rate dogs. This is quite clearly a bulbasaur. Please only send d
          Name: text, dtype: object
```

```
In [172]: #Assign name: "no_dog" to records no relevant to dogs
          no_dog['name'] = 'no_dog'

          #Number of records
          len(no_dog)
```

```
Out[172]: 68
```

```
In [173]: #Update dog names in df_clean dataframe for tweet_ids no relevant to dogs
          df_clean= pd.concat([df_clean, no_dog]).drop_duplicates(['tweet_id'],keep='last')
```

```
In [174]: #Check name value counts to make sure names were updated to "no_dog",
          #to later remove these records
          df_clean['name'].value_counts()
```

```
Out[174]: None        371
          no_dog       68
          a            22
          Lucy         10
          Cooper       10
```

```
Oliver        9
Tucker        9
Charlie       9
Penny         8
Winston       7
Sadie         7
Daisy         7
the           6
Jax           6
Toby          6
Koda          6
Lola          6
Stanley       5
Rusty         5
Bella         5
Bo            5
Oscar         5
Leo           5
Oakley        4
Maggie        4
Brody         4
Bentley       4
Dexter        4
Larry         4
Chester       4
              ...
Mason         1
Goliath       1
Yoda          1
Ruffles       1
Ace           1
Jeffri        1
Buckley       1
Jazz          1
Rolf          1
Kramer        1
Callie        1
Reagan        1
Rumble        1
Quinn         1
Kevon         1
Mingus        1
Dawn          1
Florence      1
Johm          1
Ginger        1
Timison       1
Stella        1
```

```
Ben           1
Bobb          1
Lassie        1
Berta         1
Jamesy        1
Odin          1
Penelope      1
Chuk          1
Name: name, Length: 860, dtype: int64
```

```
In [175]: #Keep records where name is different to "no_dog"
          df_clean = df_clean[df_clean.name != 'no_dog']
```

**Test**

```
In [176]: #Veryfing there are no longer "no_dog" names
          df_clean['name'].value_counts()
```

```
Out[176]: None        371
          a            22
          Cooper       10
          Lucy         10
          Charlie       9
          Oliver        9
          Tucker        9
          Penny         8
          Sadie         7
          Daisy         7
          Winston       7
          Jax           6
          the           6
          Toby          6
          Koda          6
          Lola          6
          Leo           5
          Bella         5
          Oscar         5
          Bo            5
          Rusty         5
          Stanley       5
          George        4
          Brody         4
          Dexter        4
          Larry         4
          Maggie        4
          Reggie        4
          Bentley       4
          Cassie        4
```

```
                ...
Mason           1
Goliath         1
Yoda            1
Ruffles         1
Ace             1
Jeffri          1
Buckley         1
Jazz            1
Rolf            1
Kramer          1
Callie          1
Reagan          1
Rumble          1
Quinn           1
Kevon           1
Mingus          1
Dawn            1
Florence        1
Johm            1
Ginger          1
Timison         1
Stella          1
Ben             1
Bobb            1
Lassie          1
Berta           1
Jamesy          1
Odin            1
Penelope        1
Chuk            1
Name: name, Length: 859, dtype: int64
```

```
In [177]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1616 entries, 1 to 2025
Data columns (total 19 columns):
tweet_id                1616 non-null int64
in_reply_to_status_id   19 non-null float64
in_reply_to_user_id     19 non-null float64
timestamp               1616 non-null object
source                  1616 non-null object
text                    1616 non-null object
expanded_urls           1616 non-null object
rating_numerator        1616 non-null int64
rating_denominator      1616 non-null int64
name                    1616 non-null object
```

```
doggo                    1616 non-null object
floofer                  1616 non-null object
pupper                   1616 non-null object
puppo                    1616 non-null object
jpg_url                  1616 non-null object
img_num                  1616 non-null int64
retweet_count            1616 non-null int64
favorite_count           1616 non-null int64
breed                    1616 non-null object
dtypes: float64(2), int64(6), object(11)
memory usage: 252.5+ KB
```

### 1.9 Quality - 7

**Define** Further analysis on tweets with name "a" since this is the second highest name in dataset, and doesn't seen to be a valid name

#### Code

```
In [178]: #Records which have "a", "an", "all" as name
          df_clean[df_clean.name.str.match('a')]
```

```
Out[178]:                tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
          50    881536004380872706                    NaN                  NaN
          817   747885874273214464                    NaN                  NaN
          819   747816857231626240                    NaN                  NaN
          833   746369468511756288                    NaN                  NaN
          854   743222593470234624                    NaN                  NaN
          937   728035342121635841                    NaN                  NaN
          1117  704859558691414016                    NaN                  NaN
          1126  704054845121142784                    NaN                  NaN
          1136  703079050210877440                    NaN                  NaN
          1142  702539513671897089                    NaN                  NaN
          1154  700864154249383937                    NaN                  NaN
          1257  692187005137076224                    NaN                  NaN
          1512  677644091929329666                    NaN                  NaN
          1599  675047298674663426                    NaN                  NaN
          1640  674082852460433408                    NaN                  NaN
          1710  672604026190569472                    NaN                  NaN
          1911  668815180734689280                    NaN                  NaN
          2038  666407126856765440                    NaN                  NaN
          2044  666337882303524864                    NaN                  NaN
          2046  666287406224695296                    NaN                  NaN
          2058  666057090499244032                    NaN                  NaN
          2059  666055525042405380                    NaN                  NaN
          2061  666050758794694657                    NaN                  NaN
          2063  666044226329800704                    NaN                  NaN
```

```
2064  666033412701032449                    NaN                   NaN
2065  666029285002620928                    NaN                   NaN

                      timestamp              source  \
50    2017-07-02 15:32:16 +0000  Twitter for iPhone
817   2016-06-28 20:14:22 +0000  Twitter for iPhone
819   2016-06-28 15:40:07 +0000  Twitter for iPhone
833   2016-06-24 15:48:42 +0000  Twitter for iPhone
854   2016-06-15 23:24:09 +0000  Twitter for iPhone
937   2016-05-05 01:35:26 +0000  Twitter for iPhone
1117  2016-03-02 02:43:09 +0000  Twitter for iPhone
1126  2016-02-28 21:25:30 +0000  Twitter for iPhone
1136  2016-02-26 04:48:02 +0000  Twitter for iPhone
1142  2016-02-24 17:04:07 +0000  Twitter for iPhone
1154  2016-02-20 02:06:50 +0000  Twitter for iPhone
1257  2016-01-27 03:26:56 +0000  Twitter for iPhone
1512  2015-12-18 00:18:36 +0000  Twitter for iPhone
1599  2015-12-10 20:19:52 +0000  Twitter for iPhone
1640  2015-12-08 04:27:30 +0000  Twitter for iPhone
1710  2015-12-04 02:31:10 +0000  Twitter for iPhone
1911  2015-11-23 15:35:39 +0000  Twitter for iPhone
2038  2015-11-17 00:06:54 +0000  Twitter for iPhone
2044  2015-11-16 19:31:45 +0000  Twitter for iPhone
2046  2015-11-16 16:11:11 +0000  Twitter for iPhone
2058  2015-11-16 00:55:59 +0000  Twitter for iPhone
2059  2015-11-16 00:49:46 +0000  Twitter for iPhone
2061  2015-11-16 00:30:50 +0000  Twitter for iPhone
2063  2015-11-16 00:04:52 +0000  Twitter for iPhone
2064  2015-11-15 23:21:54 +0000  Twitter for iPhone
2065  2015-11-15 23:05:30 +0000  Twitter for iPhone


50    Here is a pupper approaching maximum borkdrive. Zooming at never before seen spe
817                  This is a mighty rare blue-tailed hammer sherk. Human almost lost
819                    Viewer discretion is advised. This is a terrible attack in prog
833                    This is an Iraqi Speed Kangaroo. It is not a dog. Please only s
854                    This is a very rare Great Alaskan Bush Pupper. Hard to stumble
937                                                                  This is a
1117                                    Here is a heartbreaking scene of an inc
1126                                                                   Here is
1136                 This is a Butternut Cumberfloof. It's not windy they just look l
1142                    This is a Wild Tuscan Poofwiggle. Careful not to startle. Rar
1154                                                "Pupper is a present to world
1257                 This is a rare Arctic Wubberfloof. Unamused by the happenings. N
1512                                                 This is a dog swinging. I re
1599                                    This is a fluffy albino Bacardi Columbia m
1640                                                         This is a Sagitario
1710                                                 This is a baby Rand Paul.
```

```
1911                    This is a wild Toblerone from Papua New Guinea. Mouth always o
2038                  This is a southern Vesuvius bumblegruff. Can drive a truck (wow)
2044                  This is an extremely rare horned Parthenon. Not amused. Wears sh
2046                     This is an Albanian 3 1/2 legged  Episcopalian. Loves well-po
2058                                 My oh my. This is a rare blond Canadian terrier o
2059                Here is a Siberian heavily armored polar bear mix. Strong owner.
2061                This is a truly beautiful English Wilson Staff retriever. Has a n
2063                   This is a purebred Piers Morgan. Loves to Netflix and chill. A
2064                          Here is a very happy pup. Big fan of well-maintained de
2065                 This is a western brown Mitsubishi terrier. Upset about leaf. Ac


50
817                                                    https://twitter.com/dog_rate
819
833
854
937                                                    https://twitter.com/dog_rate
1117
1126
1136                                                   https://twitter.com/dog_rate
1142  https://twitter.com/dog_rates/status/702539513671897089/photo/1,https://twitter.
1154
1257  https://twitter.com/dog_rates/status/692187005137076224/photo/1,https://twitter.
1512
1599
1640
1710
1911
2038
2044
2046
2058
2059
2061
2063
2064
2065

      rating_numerator  rating_denominator name doggo floofer  pupper puppo  \
50                  14                  10    a  None    None  pupper  None
817                  8                  10    a  None    None    None  None
819                  4                  10    a  None    None    None  None
833                  9                  10   an  None    None    None  None
854                 12                  10    a  None    None  pupper  None
937                 12                  10  all  None    None  pupper  None
1117                10                  10    a  None    None  pupper  None
1126                60                  50    a  None    None    None  None
```

```
1136               11                10    a  None    None    None  None
1142               12                10    a  None    None    None  None
1154               12                10    a  None    None  pupper  None
1257               12                10    a  None    None    None  None
1512               11                10    a  None    None    None  None
1599               11                10    a  None    None    None  None
1640               11                10    a  None    None    None  None
1710               11                10    a  None    None    None  None
1911                7                10    a  None    None    None  None
2038                7                10    a  None    None    None  None
2044                9                10   an  None    None    None  None
2046                1                 2   an  None    None    None  None
2058                9                10    a  None    None    None  None
2059               10                10    a  None    None    None  None
2061               10                10    a  None    None    None  None
2063                6                10    a  None    None    None  None
2064                9                10    a  None    None    None  None
2065                7                10    a  None    None    None  None


50    https://pbs.twimg.com/ext_tw_video_thumb/881535971568889856/pu/img/9bawiZ--8FKyw
817                                       https://pbs.twimg.com/media/CmEGMSvUYAAl
819                                       https://pbs.twimg.com/media/CmDHdCoWkAAC
833                                       https://pbs.twimg.com/media/ClujESVXEAA4
854                                       https://pbs.twimg.com/media/ClBO9zOWYAAA
937                                       https://pbs.twimg.com/media/ChqARqmWsAEI
1117                                      https://pbs.twimg.com/media/CcgqBNVW8AE7
1126                                      https://pbs.twimg.com/media/CcVOJEcXEAMO
1136                                      https://pbs.twimg.com/media/CcHWqQCW8AEb
1142                                      https://pbs.twimg.com/media/Cb_r8qTUsAAS
1154                                      https://pbs.twimg.com/media/Cbn4OqKWwAAD
1257                                      https://pbs.twimg.com/media/CZskaEIWIAUe
1512  https://pbs.twimg.com/ext_tw_video_thumb/677644010865999872/pu/img/zVHEMYnJKzq1S
1599                                      https://pbs.twimg.com/media/CV4_8FgXAAQO
1640                                      https://pbs.twimg.com/media/CVrSxy7WsAAF
1710                                      https://pbs.twimg.com/media/CVWRyylWIAAM
1911                                      https://pbs.twimg.com/media/CUgb21RXIAAl
2038                                      https://pbs.twimg.com/media/CT-NvwmW4AAu
2044                                      https://pbs.twimg.com/media/CT9OwFIWEAMu
2046                                      https://pbs.twimg.com/media/CT8g3BpUEAAu
2058                                      https://pbs.twimg.com/media/CT5PY9OWoAAQ
2059                                      https://pbs.twimg.com/media/CT5N9tpXIAAi
2061                                      https://pbs.twimg.com/media/CT5Jof1WUAEu
2063                                      https://pbs.twimg.com/media/CT5Dr8HUEAA-
2064                                      https://pbs.twimg.com/media/CT4521TWwAEv
2065                                      https://pbs.twimg.com/media/CT42GRgUYAA5

      img_num  retweet_count  favorite_count                    breed
```

```
50          1          15570          48394                  Samoyed
817         1           1050           3084                   kuvasz
819         1           1240           5074                Pembroke
833         1           1751           6452          German_shepherd
854         1           2022           6468                   kuvasz
937         1           1762           4727               Pomeranian
1117        1            578           2346                      pug
1126        1            964           3040           Great_Pyrenees
1136        2           3286           7723                Pembroke
1142        3           1011           2995               Pomeranian
1154        1            651           2701                   kuvasz
1257        2            866           2636           Siberian_husky
1512        1            840           1929                Chihuahua
1599        1            342           1085                  Samoyed
1640        1            176            767               Pomeranian
1710        1            407           1126               toy_poodle
1911        1            278            580                  redbone
2038        1             40            106  black-and-tan_coonhound
2044        1             90            190              Newfoundland
2046        1             63            143               Maltese_dog
2058        1            138            289         golden_retriever
2059        1            235            428                     chow
2061        1             57            130     Bernese_mountain_dog
2063        1            136            292      Rhodesian_ridgeback
2064        1             43            123          German_shepherd
2065        1             46            126                  redbone
```

In [179]:
```
#Save these specific records to a dataframe called a_names for further analysis
a_names = df_clean[df_clean.name.str.match('a')]

#read sample text to see if there is a different way to identify dog's name
a_names['text'].sample(10)
```

Out[179]:
```
1911       This is a wild Toblerone from Papua New Guinea. Mouth always open. Addicted
833       This is an Iraqi Speed Kangaroo. It is not a dog. Please only send in dogs.
854       This is a very rare Great Alaskan Bush Pupper. Hard to stumble upon without
1710                                          This is a baby Rand Paul. Curls for da
1257     This is a rare Arctic Wubberfloof. Unamused by the happenings. No longer has
2058                    My oh my. This is a rare blond Canadian terrier on wheels. Onl
817     This is a mighty rare blue-tailed hammer sherk. Human almost lost a limb tryin
1599                           This is a fluffy albino Bacardi Columbia mix. Excellent
2044     This is an extremely rare horned Parthenon. Not amused. Wears shoes. Overall
2038     This is a southern Vesuvius bumblegruff. Can drive a truck (wow). Made friend
Name: text, dtype: object
```

In [180]:
```
#After reading a couple of sample text, the pattern is that there is no
#real Dog name for this tweets, thus I will update the name to "None"
```

```
a_names['name'] = 'None'

#Update dog names in df_clean dataframe
df_clean= pd.concat([df_clean, a_names]).drop_duplicates(['tweet_id'],keep='last')
```

**Test**

```
In [181]: #Veryfing there are no longer names with a

          df_clean['name'].value_counts()
```

```
Out[181]: None        397
          Cooper       10
          Lucy         10
          Tucker        9
          Oliver        9
          Charlie       9
          Penny         8
          Winston       7
          Sadie         7
          Daisy         7
          Koda          6
          Toby          6
          Jax           6
          Lola          6
          the           6
          Bo            5
          Rusty         5
          Stanley       5
          Bella         5
          Leo           5
          Oscar         5
          Oakley        4
          Dexter        4
          Dave          4
          Bailey        4
          Jack          4
          Maggie        4
          Bentley       4
          Scooter       4
          Winnie        4
                      ...
          Crimson       1
          Mason         1
          Goliath       1
          Ruffles       1
          Ace           1
          Jeffri        1
```

```
Terrance      1
Kramer        1
Joey          1
Penelope      1
Rolf          1
Callie        1
Reagan        1
Rumble        1
Quinn         1
Kevon         1
Mingus        1
Dawn          1
Florence      1
Johm          1
Ginger        1
Timison       1
Stella        1
Ben           1
Bobb          1
Lassie        1
Berta         1
Jamesy        1
Odin          1
Chuk          1
Name: name, Length: 856, dtype: int64
```

### 1.10 Quality - 8

**Define** Investigate and clean *name* field which has "None" as the name to identify if there could be posibly real Dog names.

#### Code

```
In [182]: #Count the number of records which have "None" as name
          len(df_clean[df_clean.name.str.contains("None")])
```

```
Out[182]: 397
```

```
In [183]: #Save these specific records to a dataframe called none_names for further analysis
          none_names = df_clean[df_clean.name.str.contains("None")]

          #read sample text to see if there is a different way to identify dog's name
          none_names['text'].sample(10)
```

```
Out[183]: 766
          1398                                  Here we see a nifty leaping pupper. Feet look de
          2033                                       Oh boy what a pup! Sunglasses take this one
          343     We are proud to support @LoveYourMelon on their mission to put a hat on every
          931                                                                       When you're
```

```
157      Sometimes you guys remind me just how impactful a pupper can be. Cooper will
214                          Here we have some incredible doggos for #K9VeteransDay. A
2053                                        Here we have a well-established sunbloc
1117                                                  Here is a heartbreaking scen
1635
Name: text, dtype: object
```

In [184]:
```
#As with lowercase_names dataframe there could be real name dogs that can
#be extracted fron text field, I will look for records with the words "name" or "named


#Extracting dog names where text contains the word "named"
valid_none_names = none_names[none_names.text.str.contains("name")]

named_v = valid_none_names['text'].str.extract(r"named\s(\w+)")
named_v =named_v[named_v.isnull()==False]
named_v

#Extracting dog names where text contains "name is"
name_is_v =valid_none_names['text'].str.extract(r"name is\s(\w+)")
name_is_v = name_is_v[name_is_v.isnull()== False]
name_is_v

#Appending new name results
new_name_v =named_v.append(name_is_v)
new_name_v

#In new valid_none_names dataframe replace "name" values with newly found Dog names

valid_none_names['name'] = new_name_v

valid_none_names['name'].value_counts()
```

Out[184]:
```
Zoey          1
Sabertooth    1
Big           1
Tickles       1
Zeus          1
Guss          1
Name: name, dtype: int64
```

In [185]:
```
#Update dog names in df_clean dataframe for tweet_ids where a new dog name was found
df_clean= pd.concat([df_clean, valid_none_names]).drop_duplicates(['tweet_id'],keep='l
```

In [186]:
```
#Remove records with possible NaN
df_clean = df_clean.dropna(subset=['name'])
```

**Test**

```
In [187]: #Checking value counts for names, and veryfing "None" values have been reduced
          df_clean['name'].value_counts()
```

```
Out[187]: None        388
          Lucy         10
          Cooper       10
          Charlie       9
          Oliver        9
          Tucker        9
          Penny         8
          Winston       7
          Sadie         7
          Daisy         7
          Koda          6
          the           6
          Jax           6
          Toby          6
          Lola          6
          Bella         5
          Zoey          5
          Leo           5
          Stanley       5
          Bo            5
          Oscar         5
          Rusty         5
          Maggie        4
          Chester       4
          Scout         4
          Larry         4
          Dexter        4
          Brody         4
          Bentley       4
          Oakley        4
                       ...
          Mason         1
          Goliath       1
          Ruffles       1
          Ace           1
          Jeffri        1
          Kawhi         1
          Buckley       1
          Jazz          1
          Rolf          1
          Kramer        1
          Callie        1
          Reagan        1
          Rumble        1
          Quinn         1
```

```
Kevon         1
Mingus        1
Dawn          1
Florence      1
Johm          1
Ginger        1
Timison       1
Stella        1
Ben           1
Bobb          1
Lassie        1
Berta         1
Jamesy        1
Odin          1
Penelope      1
Chuk          1
Name: name, Length: 860, dtype: int64
```

```
In [188]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1613 entries, 1 to 1981
Data columns (total 19 columns):
tweet_id                 1613 non-null int64
in_reply_to_status_id    18 non-null float64
in_reply_to_user_id      18 non-null float64
timestamp                1613 non-null object
source                   1613 non-null object
text                     1613 non-null object
expanded_urls            1613 non-null object
rating_numerator         1613 non-null int64
rating_denominator       1613 non-null int64
name                     1613 non-null object
doggo                    1613 non-null object
floofer                  1613 non-null object
pupper                   1613 non-null object
puppo                    1613 non-null object
jpg_url                  1613 non-null object
img_num                  1613 non-null int64
retweet_count            1613 non-null int64
favorite_count           1613 non-null int64
breed                    1613 non-null object
dtypes: float64(2), int64(6), object(11)
memory usage: 252.0+ KB
```

### 1.11 Tidiness - 3

**Define** Create a *stage* variable to remove individual dog stage columns

```
In [189]: df_clean['doggo'].value_counts()

Out[189]: None     1551
          doggo      62
          Name: doggo, dtype: int64

In [190]: df_clean['floofer'].value_counts()

Out[190]: None       1606
          floofer       7
          Name: floofer, dtype: int64

In [191]: df_clean['pupper'].value_counts()

Out[191]: None      1438
          pupper     175
          Name: pupper, dtype: int64

In [192]: df_clean['puppo'].value_counts()

Out[192]: None     1591
          puppo      22
          Name: puppo, dtype: int64
```

**Code**

```
In [193]: #Create column stage with corresponding stage names
          df_clean['stage'] = df_clean[['doggo', 'floofer', 'pupper', 'puppo']].max(axis=1)

          #drop individual stage columns
          df_clean.drop(['doggo', 'floofer', 'pupper', 'puppo'], axis =1 , inplace = True)
```

**Test**

```
In [194]: df_clean['stage'].value_counts()

Out[194]: None       1355
          pupper      175
          doggo        54
          puppo        22
          floofer       7
          Name: stage, dtype: int64

In [195]: df_clean.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1613 entries, 1 to 1981
Data columns (total 16 columns):
tweet_id                 1613 non-null int64
in_reply_to_status_id    18 non-null float64
```

```
in_reply_to_user_id      18 non-null float64
timestamp                1613 non-null object
source                   1613 non-null object
text                     1613 non-null object
expanded_urls            1613 non-null object
rating_numerator         1613 non-null int64
rating_denominator       1613 non-null int64
name                     1613 non-null object
jpg_url                  1613 non-null object
img_num                  1613 non-null int64
retweet_count            1613 non-null int64
favorite_count           1613 non-null int64
breed                    1613 non-null object
stage                    1613 non-null object
dtypes: float64(2), int64(6), object(8)
memory usage: 214.2+ KB
```

### 1.12 Quality - 9

**Define** There are many records with "None" Stage, analyze text to identify possible stages for these records

#### Code

```
In [196]: df_clean['stage'].value_counts()
```

```
Out[196]: None       1355
          pupper      175
          doggo        54
          puppo        22
          floofer       7
          Name: stage, dtype: int64
```

```
In [197]: #Create separate dataframe to analyze None values for stage variable
          none_stage= df_clean[df_clean.stage.str.contains("None")]

          #Extracting the records that contain words "pupper"
          n_pupper = none_stage[none_stage.text.str.contains("pupper")]
          n_pupper['stage'] = 'pupper'

          #Extracting the records that contain words "doggo"
          n_doggo = none_stage[none_stage.text.str.contains("doggo")]
          n_doggo['stage'] = 'doggo'

          #Extracting the records that contain words "puppo"
          n_puppo = none_stage[none_stage.text.str.contains("puppo")]
          n_puppo['stage'] = 'puppo'
```

```
#Extracting the records that contain words "floofer" but there were none
n_floofer = none_stage[none_stage.text.str.contains("floofer")]
n_floofer['stage'] = 'floofer'
```

```
In [198]: len(n_pupper)
          len(n_doggo)
          len(n_puppo)
          len(n_floofer)
```

```
Out[198]: 0
```

```
In [199]: #Update stage in none_stage dataframe, I'm not updatting floofer since there were 0 up
          none_stage= pd.concat([none_stage, n_pupper]).drop_duplicates(['tweet_id'],keep='last'
          none_stage= pd.concat([none_stage, n_doggo]).drop_duplicates(['tweet_id'],keep='last')
          none_stage= pd.concat([none_stage, n_puppo]).drop_duplicates(['tweet_id'],keep='last')
```

```
In [200]: #Update stage on df_clean dataframe
          df_clean= pd.concat([df_clean, none_stage]).drop_duplicates(['tweet_id'],keep='last')
```

**Test**

```
In [201]: df_clean['stage'].value_counts()
```

```
Out[201]: None       1329
          pupper      192
          doggo        58
          puppo        27
          floofer       7
          Name: stage, dtype: int64
```

```
In [202]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1613 entries, 9 to 437
Data columns (total 16 columns):
tweet_id                 1613 non-null int64
in_reply_to_status_id    18 non-null float64
in_reply_to_user_id      18 non-null float64
timestamp                1613 non-null object
source                   1613 non-null object
text                     1613 non-null object
expanded_urls            1613 non-null object
rating_numerator         1613 non-null int64
rating_denominator       1613 non-null int64
name                     1613 non-null object
jpg_url                  1613 non-null object
img_num                  1613 non-null int64
retweet_count            1613 non-null int64
```

```
favorite_count           1613 non-null int64
breed                    1613 non-null object
stage                    1613 non-null object
dtypes: float64(2), int64(6), object(8)
memory usage: 214.2+ KB
```

### 1.13 Quality - 10

**Define** Clean *rating_numerator* column since it has very low values (below 4) or extremely high values (Over 15)

#### Code

```
In [203]: #Checking rating_numerator values
          df_clean.rating_numerator.value_counts().sort_index()
```

```
Out[203]: 0        1
          1        1
          2        2
          3        4
          4        7
          5       12
          6       16
          7       31
          8       66
          9      121
          10     343
          11     341
          12     407
          13     223
          14      22
          24       1
          26       1
          27       1
          44       1
          45       1
          50       1
          60       1
          75       1
          80       1
          84       1
          88       1
          99       1
          121      1
          143      1
          144      1
          165      1
          Name: rating_numerator, dtype: int64
```

```
In [204]: #Reviewing text for rating_numerators > 14
          df_clean[df_clean.rating_numerator >14 ]['text']
```

```
Out[204]: 609            This is Sophie. She's a Jubilant Bush Pupper. Super h*ckin rare. Appea
          341                                                                    The floo
          411    Meet Sam. She smiles 24/7 &amp;amp; secretly aspires to be a reindeer. \nKeep Sam
          553                                              This is Logan, the Chow who lived. He
          730
          996                                                      This is Bluebert. He just
          1060                          From left to right:\nCletus, Jerome, Alejandro, Burp, &
          1202                                                                           H
          1506
          1017                                                                          Ha
          1042                                                        Here's a brigade
          1374                                                 Two sneaky puppers were not initial
          1375                                Someone help the girl is being mugged. Several ar
          1446                                                          Here we have un
          1565                                                          Here we have an
          1126
          Name: text, dtype: object
```

```
In [205]: #After analyzing the text for rating numerators greater than 14, I have decidied that
          #I will do the same with high denominator values
          df_clean['rating_numerator'] = [i['rating_numerator'] if i['rating_numerator'] <= 14
                                          else int(i['rating_numerator']/10) for index, i in df
```

```
In [206]: #Reviewing text for rating_numerators <=4
          df_clean[df_clean.rating_numerator <= 4]['text']
```

```
Out[206]: 609            This is Sophie. She's a Jubilant Bush Pupper. Super h*ckin rare. Appea
          245                                                        When you're so blinded
          411    Meet Sam. She smiles 24/7 &amp;amp; secretly aspires to be a reindeer. \nKeep Sam
          610                                  This is Wesley. He's clearly trespassing. Seems r
          962
          986                                        This is Alexanderson. He's got a weird as
          1037                                  What hooligan sent in pictures w/out a dog in
          1060                          From left to right:\nCletus, Jerome, Alejandro, Burp, &
          1083                                 This is Keurig. He's a rare dog. Laughs like an i
          1202                                                                           H
          1436
          1493                                 This is Crystal. She's a shitty fireman. No sense
          1785                                     Two miniature golden retrievers here. Webbed
          1897                                 This is Bernie. He's taking his Halloween costume
          1958                                  This is Tedrick. He lives on the edge. Needs som
          2000                                 These are strange dogs. All have toupees. Long ne
          2027                                   Cool dog. Enjoys couch. Low monotone bark. Very
          819                                    Viewer discretion is advised. This is a terribl
          2046                                     This is an Albanian 3 1/2 legged  Episcopalia
```

```
1446                                                                  Here we have un
Name: text, dtype: object
```

```
In [207]: #After reviewing text, I decided to remove records where rating_numerator is >= 4 sinc
          #with lower numerator ratins don't seem to be relevant

          df_clean = df_clean[df_clean.rating_numerator >= 4]
```

**Test**

```
In [208]: #Veryfing I no longer have rating_numerators with very low or very high values
          df_clean.rating_numerator.value_counts().sort_index()
```

```
Out[208]: 4       9
          5      13
          6      17
          7      32
          8      69
          9     122
          10    343
          11    341
          12    408
          13    223
          14     24
          16      1
          Name: rating_numerator, dtype: int64
```

### 1.14 Quality - 11

**Define** Clean *rating_denominator* column since it has values over 10

**Code**

```
In [209]: #Checking rating_numerator values
          df_clean.rating_denominator.value_counts().sort_index()
```

```
Out[209]: 10     1587
          11        2
          20        1
          40        1
          50        3
          70        1
          80        2
          90        1
          110       1
          120       1
          130       1
          150       1
          Name: rating_denominator, dtype: int64
```

```
In [210]: #Reviewing text for rating_denominators > 10
          df_clean[df_clean.rating_denominator >10 ]['text']
```

```
Out[210]: 341                                                The floofs have been released I
          730                                                                    Why does t
          871          After so many requests, this is Bretagne. She was the last surviving 9/1
          962
          996                                    This is Bluebert. He just saw that both #FinalFur
          1060    From left to right:\nCletus, Jerome, Alejandro, Burp, &amp;amp; Titson\nNone know
          1202                                                  Happy Wednesday here's a
          1400          This is Darrel. He just robbed a 7/11 and is in a high speed police cha
          1506                                                                        IT
          1017                                                 Happy Saturday here's 9 p
          1042                                        Here's a brigade of puppers. All look ve
          1374                          Two sneaky puppers were not initially seen, moving the rat
          1375          Someone help the girl is being mugged. Several are distracting her while
          1565                                         Here we have an entire platoon of puppe
          1126                                                              Here is a whol
          Name: text, dtype: object
```

```
In [211]: #After analyzing the text for the few ratings denominators greater than 10,
          #I have decidied that I will assing the value of 10 for such record
          df_clean['rating_denominator'] = [i['rating_denominator'] if i['rating_numerator'] ==
                                            else int(10) for index, i in df_clean.iterrows()]
```

**Test**

```
In [212]: df_clean.rating_denominator.value_counts()
```

```
Out[212]: 10    1602
          Name: rating_denominator, dtype: int64
```

```
In [213]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1602 entries, 9 to 437
Data columns (total 16 columns):
tweet_id                 1602 non-null int64
in_reply_to_status_id    18 non-null float64
in_reply_to_user_id      18 non-null float64
timestamp                1602 non-null object
source                   1602 non-null object
text                     1602 non-null object
expanded_urls            1602 non-null object
rating_numerator         1602 non-null int64
rating_denominator       1602 non-null int64
name                     1602 non-null object
jpg_url                  1602 non-null object
img_num                  1602 non-null int64
```

```
retweet_count            1602 non-null int64
favorite_count           1602 non-null int64
breed                    1602 non-null object
stage                    1602 non-null object
dtypes: float64(2), int64(6), object(8)
memory usage: 212.8+ KB
```

### 1.15 Quality - 11

**Define** Correct datatypes for *in_reply_to_status_id* and *in_reply_to_user_id* from float to integer, and convert timestamp to datetime data type.

#### Code

```
In [214]: #For variables in_reply_to_status_id and in_reply_to_user_id I will fill with Zeros th
          df_clean.in_reply_to_status_id = df_clean.in_reply_to_status_id.fillna(0)
          df_clean.in_reply_to_user_id = df_clean.in_reply_to_user_id.fillna(0)

          df_clean.in_reply_to_status_id = df_clean.in_reply_to_status_id.astype(np.int64)
          df_clean.in_reply_to_user_id = df_clean.in_reply_to_user_id.astype(np.int64)

          df_clean.timestamp = pd.to_datetime(df_clean.timestamp)
```

#### Test

```
In [215]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1602 entries, 9 to 437
Data columns (total 16 columns):
tweet_id                 1602 non-null int64
in_reply_to_status_id    1602 non-null int64
in_reply_to_user_id      1602 non-null int64
timestamp                1602 non-null datetime64[ns]
source                   1602 non-null object
text                     1602 non-null object
expanded_urls            1602 non-null object
rating_numerator         1602 non-null int64
rating_denominator       1602 non-null int64
name                     1602 non-null object
jpg_url                  1602 non-null object
img_num                  1602 non-null int64
retweet_count            1602 non-null int64
favorite_count           1602 non-null int64
breed                    1602 non-null object
stage                    1602 non-null object
dtypes: datetime64[ns](1), int64(8), object(7)
```

```
memory usage: 212.8+ KB
```

### 1.16 Storing

```
In [216]: #Creating a new dataframe with only the variables of interest to analyze

          df_clean_analysis = df_clean[['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_i
                                        'source', 'rating_numerator', 'rating_denominator', 'name
                                        'favorite_count', 'breed', 'stage']]

In [217]: #Veryfing new dataframe was created correctly
          df_clean_analysis.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1602 entries, 9 to 437
Data columns (total 12 columns):
tweet_id                 1602 non-null int64
in_reply_to_status_id    1602 non-null int64
in_reply_to_user_id      1602 non-null int64
timestamp                1602 non-null datetime64[ns]
source                   1602 non-null object
rating_numerator         1602 non-null int64
rating_denominator       1602 non-null int64
name                     1602 non-null object
retweet_count            1602 non-null int64
favorite_count           1602 non-null int64
breed                    1602 non-null object
stage                    1602 non-null object
dtypes: datetime64[ns](1), int64(7), object(4)
memory usage: 162.7+ KB


In [218]: #Saving to CSF file the clean dataframe
          df_clean_analysis.to_csv('twitter_archive_master.csv')
```

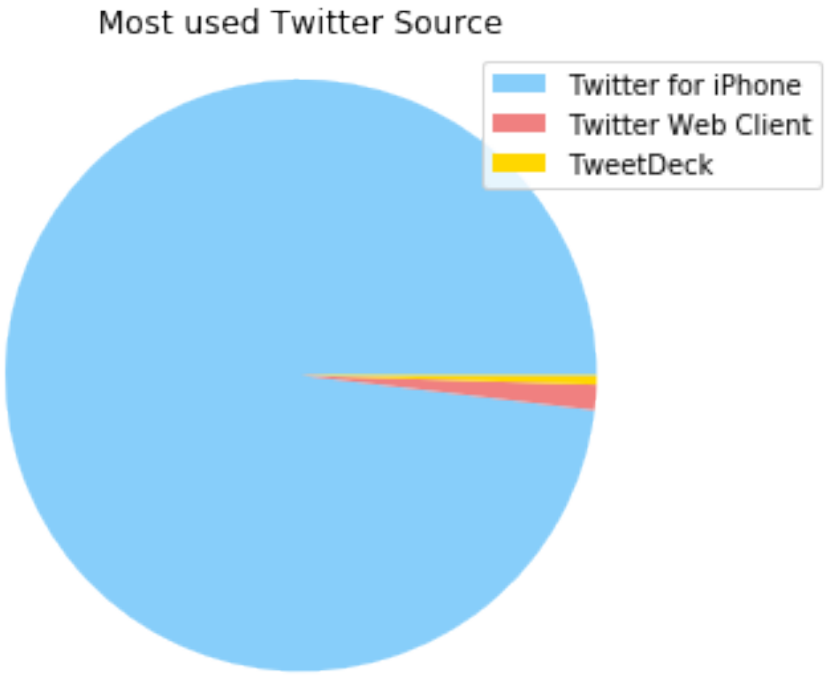
### 1.17 Analysis and Visualizations

#### 1.17.1 What is the most used Twitter Source?

```
In [219]: df_clean_analysis.source.value_counts()

Out[219]: Twitter for iPhone    1572
          Twitter Web Client      22
          TweetDeck                8
          Name: source, dtype: int64
```

*** Visualization**

```
In [220]: #Visualizing Twitter Sources
          labels = ['Twitter for iPhone', 'Twitter Web Client', 'TweetDeck']
          sizes = [1572,22,8]
          colors = ['lightskyblue', 'lightcoral','gold']
          patches, texts = plt.pie(sizes,colors=colors)
          plt.legend(patches,labels, loc ="best")
          plt.axis('equal')
          plt.title('Most used Twitter Source')
          plt.tight_layout()
          plt.show()
```




*Twitter for iPhone* is the most used Twitter source

### 1.17.2 Top 10 Dog Breeds

Let's find out what are the most common Dog Breeds based on twitter counts

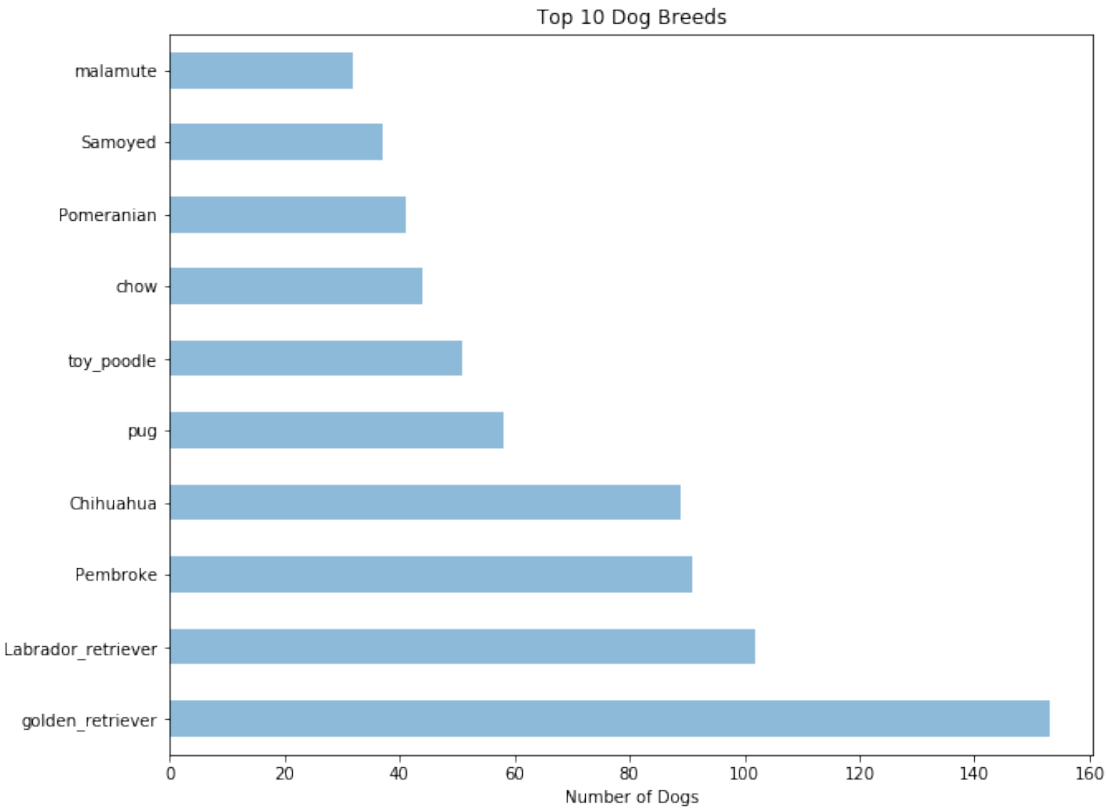
```
In [221]: df_clean_analysis.breed.value_counts()[0:10].sort_values(ascending = False)
```

```
Out[221]: golden_retriever      153
          Labrador_retriever    102
          Pembroke               91
          Chihuahua              89
```

```
pug                     58
toy_poodle              51
chow                    44
Pomeranian              41
Samoyed                 37
malamute                32
Name: breed, dtype: int64
```

**Dog Breeds Visualization**

```
In [222]: df_clean_analysis.breed.value_counts()[0:10].plot('barh', figsize = (10,8), title = "T
          plt.xlabel('Number of Dogs');
```




*Golden Retriever* is the top one breed that appears in more tweets, followed by *Labrador Retriever* Breed.

### 1.17.3 Stage by Rating Numerator

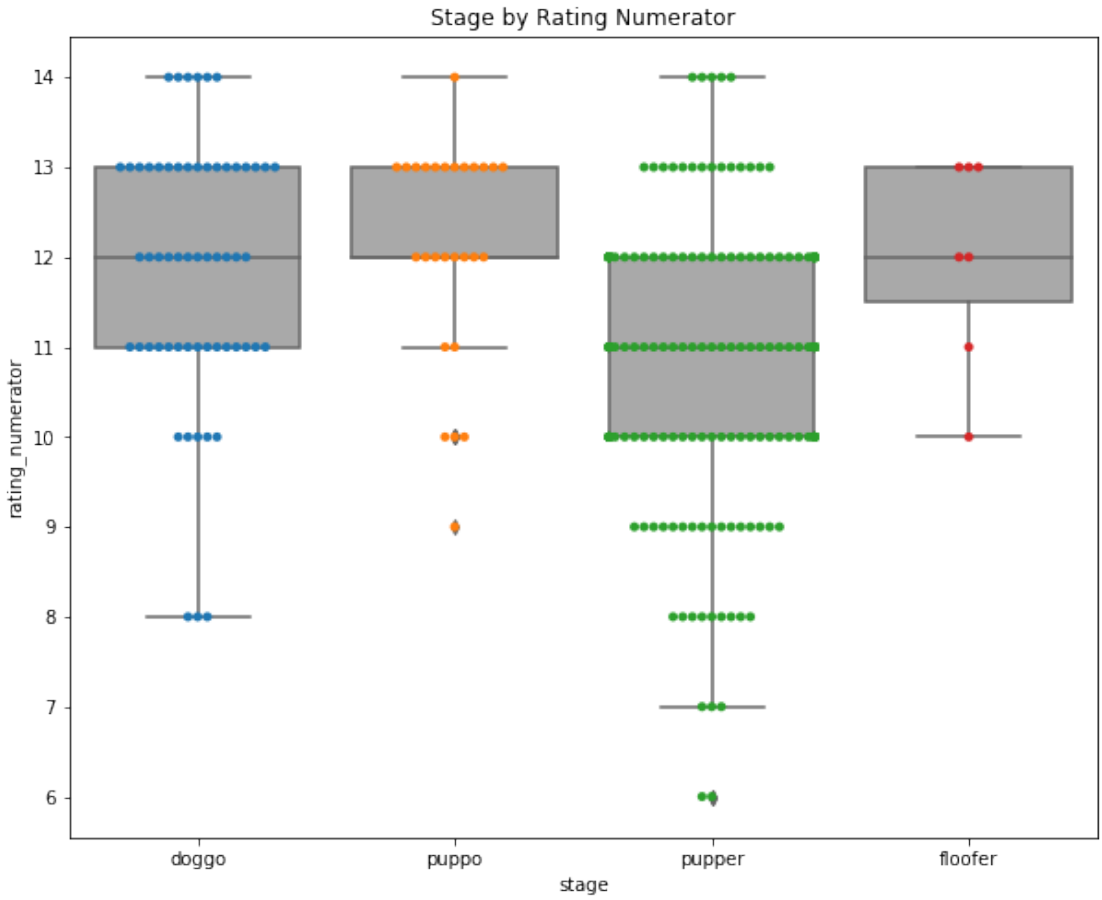
Considering that Rating Numerators scores, let's take a look at the different Dog Stages to find out how different Stages are rated.

```
In [223]: #Descriptive information about the dog's Stages
          df_clean_analysis['stage'].value_counts()
```

```
Out[223]: None       1320
          pupper      190
          doggo        58
          puppo        27
          floofer       7
          Name: stage, dtype: int64
```

### Stage by Rating Numerator Visualization

```
In [224]: import seaborn as sns
          #Replacing Stage with name "None" as NaN
          df_clean_analysis['stage'].replace('None', np.nan, inplace=True)

          #Creating a swarmplot over a boxplot to see where data points are concentrated
          plt.figure(figsize=(10,8))
          sns.boxplot(x='stage', y ='rating_numerator', data=df_clean_analysis, color='darkgray'
          sns.swarmplot(x='stage', y ='rating_numerator', data=df_clean_analysis);
```




*Pupper* is the Stage that shows more variation in numerator ratings.

*Puppo* and *floffer* are the stages with less count of ratings, however the few ratings given are high ratings, between 12 and 13.

*doggo* stage seems to have high ratings most of them are concentrated between 11 to 13, however there are also a good number or ratings with value 14

### 1.17.4 About Rating Numerator

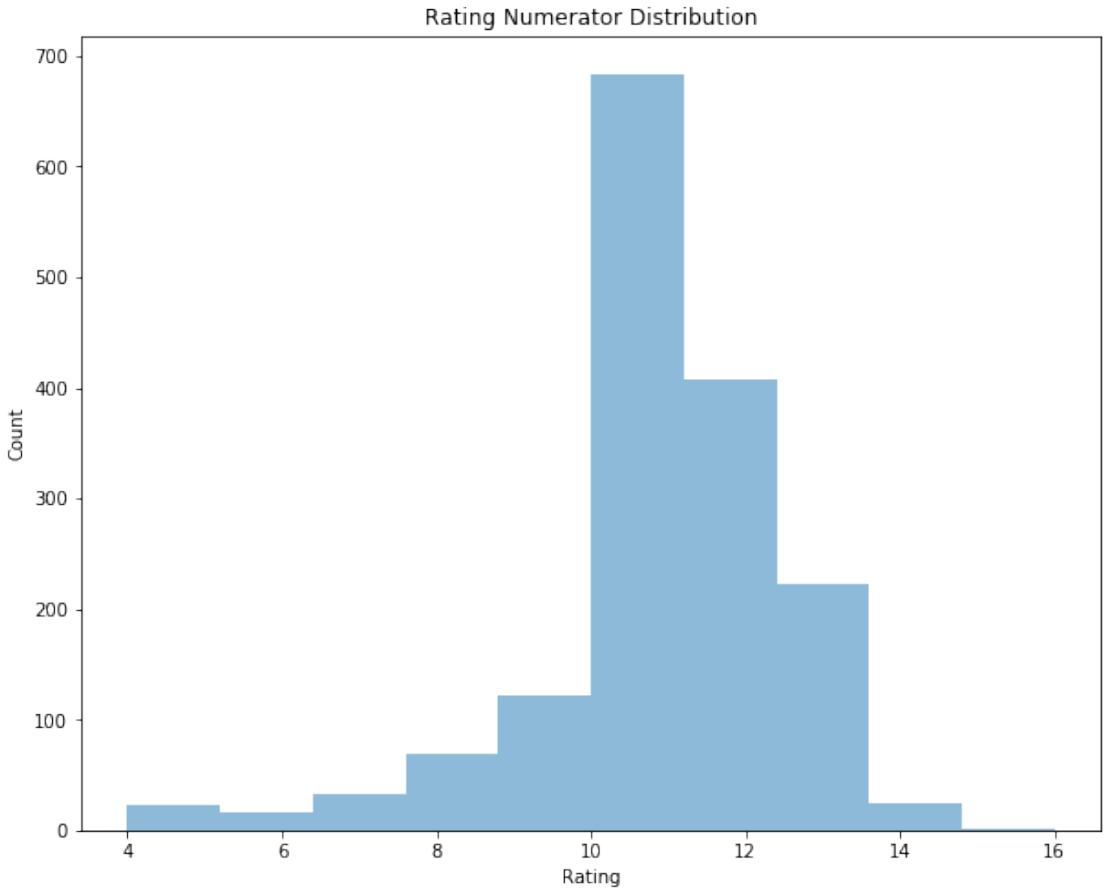
Considering that Rating Numerators vary from 4 - 16 where 4 is the lowest rating received and 16 is the highest, let's take a look at the distribution and statistics of Rating Numerator

```
In [225]: # Descriptive information about the dog's rating Numerator
          df_clean_analysis['rating_numerator'].describe()
```

```
Out[225]: count    1602.000000
          mean       10.864544
          std         1.733242
          min         4.000000
          25%        10.000000
          50%        11.000000
          75%        12.000000
          max        16.000000
          Name: rating_numerator, dtype: float64
```

#### Rating Numerator Visualization

```
In [226]: #Rating Numerator Distribution
          plt.figure(figsize=(10,8))
          plt.hist(df_clean_analysis['rating_numerator'], alpha=0.5)
          plt.xlabel('Rating')
          plt.ylabel('Count')
          plt.title('Rating Numerator Distribution');
```




*Rating Numerator* Distribution is a little bit Negative skewed where the great majority of counts of rating numerators are above 10, which makes sense based on the way Dogs get rated.

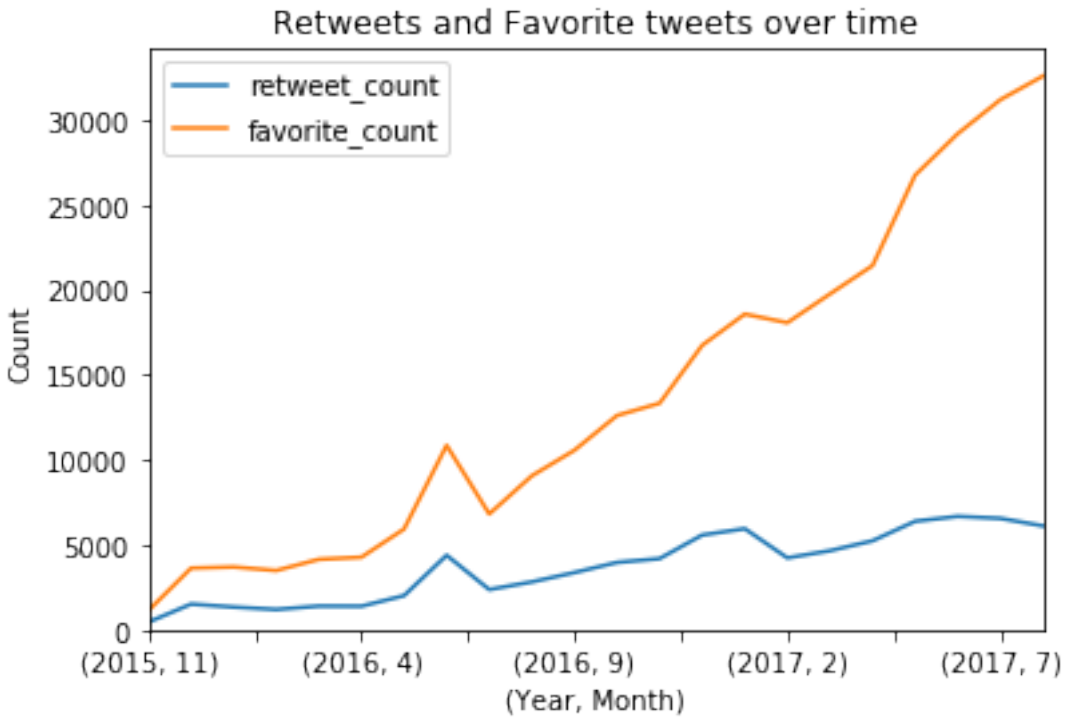
### 1.17.5 Retweets and Favorite Tweets over Time

```
In [227]: #Retweet Count Statistics
          df_clean_analysis.retweet_count.describe()

Out[227]: count     1602.000000
          mean      2630.825218
          std       4856.950015
          min         11.000000
          25%        592.250000
          50%       1305.000000
          75%       2954.500000
          max      83264.000000
          Name: retweet_count, dtype: float64

In [228]: #Favorite Count Statistics
          df_clean_analysis.favorite_count.describe()
```

```
Out[228]: count      1602.000000
          mean       8783.049313
          std       13072.335645
          min          78.000000
          25%        1991.250000
          50%        4004.500000
          75%       10862.500000
          max      163731.000000
          Name: favorite_count, dtype: float64
```

### Visualization

```
In [229]: #Plotting Retweets and Favorite
          df_clean_analysis.retweet_count.groupby([df_clean_analysis["timestamp"].dt.year, df_cl
          df_clean_analysis.favorite_count.groupby([df_clean_analysis["timestamp"].dt.year, df_c
          plt.title('Retweets and Favorite tweets over time')
          plt.ylabel('Count')
          plt.xlabel('(Year, Month)')
          plt.legend(loc ='best');
```




We can clearly see from above visualization how favorite counts grows almost exponentially over time, compared to retweet counts that even though still shows some increment, it is not as noticieable as favorite counts.

```
In [ ]:
```