Project: Wrangling and Analyze Data

In [3]:
```
import requests
import json
import os
import re
import warnings
#import tweepy
#from tweepy import OAuthHandler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')

from timeit import default_timer as timer
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

Data Gathering

In the cell below, gather **all** three pieces of data for this project and load them in the notebook. **Note:** the methods required to gather each data are different.

1. Directly download the WeRateDogs Twitter archive data (twitter_archive_enhanced.csv)

In [4]:
```
twitterArchive = pd.read_csv('data/twitter-archive-enhanced-2.csv')
```

1. Use the Requests library to download the tweet image prediction (image_predictions.tsv)

In [ ]:
```
url = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predi
response = requests.get(url)

with open(url.split('/')[-1], mode='wb') as file:
    file.write(response.content)
```

1. Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

In [ ]:
```
# Query Twitter API for each tweet in the Twitter archive and save JSON in a text fi
# These are hidden to comply with Twitter's API terms and conditions
consumer_key = os.environ.get('CONSUMER_KEY')
consumer_secret = os.environ.get('CONSUMER_SECRET')
access_token = os.environ.get('ACCESS_TOKEN')
access_secret = os.environ.get('ACCESS_SECRET')


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

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

In [7]:
```
# NOTE I HAD MOBILE VERIFICATION ISSUES:
```

```
# NOTE TO REVIEWER: this student had mobile verification issues so the following
# Twitter API code was sent to this student from a Udacity instructor
# Tweet IDs for which to gather additional data via Twitter's API
tweet_ids = twitterArchive.tweet_id.values
print(len(tweet_ids))

#Type of data
type(tweet_ids)
```

Out[7]:
```
2356
numpy.ndarray
```

The nature of data returned by twitter's API

In [ ]:
```
for tweet_id in tweet_ids:
    # the .get_status() method of the API helps get all the information about the tw
    tweet = api.get_status(tweet_id, tweet_mode='extended')
    print(tweet)
    # this next line helps us get the attributes in the python object returned
    print(f'The attributes in this python object are: {dir(tweet)}')
    break
```

In [ ]:
```
# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
count = 0
fails_dict = {}
start = timer()
# Save each tweet's returned JSON as a new line in a .txt file
with open('tweet_json.txt', 'w') as outfile:
    # This loop will likely take 20-30 minutes to run because of Twitter's rate limi
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Success")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except tweepy.TweepError as e:
            print("Fail")
            fails_dict[tweet_id] = e
            pass
end = timer()
print(end - start)
print(fails_dict)
```

In [ ]:
```
tweets = []
with open('tweet-json.txt','r') as file:
    for line in file:
        #json.loads() method to convert the json string to a python dictionary
        #print(type(json.loads(line)))
        tweet = json.loads(line)
        twitter_id = tweet['id']
        retweet_count = tweet['retweet_count']
        favourite_count = tweet['favorite_count']
        tweets.append({'tweet_id': twitter_id, 'retweet_count': retweet_count, 'favo
```

Assessing Data

In this section, detect and document at least **eight (8) quality issues and two (2) tidiness issue**. You must use **both** visual assessment programmatic assessement to assess the data.

Note: pay attention to the following key points when you access the data.

- You only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.
- Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling. Therefore, the requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned. This unique rating system is a big part of the popularity of WeRateDogs.
- You do not need to gather the tweets beyond August 1st, 2017. You can, but note that you won't be able to gather the image predictions for these tweets since you don't have access to the algorithm used.

Twitter Archive

In [5]:
```
twitterArchive.head()
```

Out[5]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.com/d
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/d
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/d
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.com/d
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.com/d

Programmatic Inspection

In [4]:

```
twitterArchive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 220.9+ KB
```

In [5]:
```
twitterArchive.describe()
```

Out[5]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweeted_status_
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	1.8100
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	1.2416
std	6.856705e+16	7.582492e+16	1.252797e+17	6.236928e+16	9.5992
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	7.8321
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	4.1969
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	4.1969
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	4.1969
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	7.8746

In [6]:
```
twitterArchive.duplicated().any()
```

Out[6]:
```
False
```

In [7]:
```
twitterArchive.source.value_counts()
```

Out[7]:
```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>
```

```
11
Name: source, dtype: int64
```

In [23]:
```
twitterArchive.rating_numerator.value_counts().sort_index()
```

Out[23]:
```
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 [24]:
```
twitterArchive.rating_denominator.value_counts().sort_index()
```

Out[24]:
```
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
```

In [22]:
```
# Checking records that are retweet and not retweets
(twitterArchive.retweeted_status_id.isnull() == False).value_counts()
```

Out[22]:
```
False    2175
True      181
Name: retweeted_status_id, dtype: int64
```

Quality issues

1. ID should be a string datatype not integers
2. Timestamp and retweet_timestamp should be datetime variable
3. Retweets are present in the data
4. in_reply and retweet_status should be a string datatype not integers
5. Unexplainable column names
6. Missing values (NaN)
7. Source values have html <'*a*'> tag
8. Dog name are not standardized

Tidiness issues

1. Columns doggo, flooferm, pupper, and puppo should be values for dog_type
2. More than one dog type is filled for a particular dog

Image Prediction

In [6]:
```
imagePrediction = pd.read_csv('data/image-predictions.tsv', sep='\t')
imagePrediction.head()
```

Out[6]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_s
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_she
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridg
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pi

Programmatic Inspection

In [8]:
```
imagePrediction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   tweet_id  2075 non-null   int64
 1   jpg_url   2075 non-null   object
 2   img_num   2075 non-null   int64
 3   p1        2075 non-null   object
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool
 6   p2        2075 non-null   object
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool
 9   p3        2075 non-null   object
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 119.6+ KB
```

In [9]:
```
imagePrediction.duplicated().any()
```

Out[9]:
```
False
```

In [28]:
```
print(imagePrediction.jpg_url.duplicated().any())
print(imagePrediction.jpg_url.duplicated().sum())
```

```
True
66
```

In [25]:
```
imagePrediction.img_num.value_counts()
```

Out[25]:
```
1    1780
2     198
3      66
4      31
Name: img_num, dtype: int64
```

In [29]:
```
imagePrediction.describe()
```

Out[29]:

	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

Quality issues

1. TweetID should be a string datatype not integers

2. p1, p2, p3 contains values with underscore

3. 2075 tweetsid present instead of 2356, some ids are missing

Twitter's API

In [7]:
```
#Read with read_json method
df = pd.read_json('data/tweet-json.txt', lines=True)
df_tweetApi = df[['id', 'retweet_count', 'favorite_count']]
df_tweetApi.head()

#Read using DataFrame()
#df_tweetApi = pd.DataFrame(tweets)
#df_tweetApi
```

Out[7]:

	id	retweet_count	favorite_count
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048

Programmatic Inspection

In [14]:
```
df_tweetApi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              2354 non-null   int64
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
dtypes: int64(3)
memory usage: 55.2 KB
```

In [30]:
```
df_tweetApi.describe()
```

Out[30]:

	id	retweet_count	favorite_count
count	2.354000e+03	2354.000000	2354.000000
mean	7.426978e+17	3164.797366	8080.968564
std	6.852812e+16	5284.770364	11814.771334
min	6.660209e+17	0.000000	0.000000
25%	6.783975e+17	624.500000	1415.000000
50%	7.194596e+17	1473.500000	3603.500000
75%	7.993058e+17	3652.000000	10122.250000
max	8.924206e+17	79515.000000	132810.000000

In [31]:
```
df_tweetApi.duplicated().any()
```

Out[31]:
```
False
```

Quality issues

1. There are 2 missing ids. 2356-2354=2

Cleaning Data

In this section, clean **all** of the issues you documented while assessing.

Note: Make a copy of the original data before cleaning. Cleaning includes merging individual pieces of data according to the rules of tidy data. The result should be a high-quality and tidy master pandas DataFrame (or DataFrames, if appropriate).

In [8]:
```
# Make copies of original pieces of data
tweet_archive_copy = twitterArchive.copy()
img_prediction_copy = imagePrediction.copy()
tweetAPI_copy = df_tweetApi.copy()
```

Twitter Archive

Issue : Data type

Define:

Convert 'timestamp' to date

Code

In [9]:
```
tweet_archive_copy.timestamp = pd.to_datetime(tweet_archive_copy.timestamp)
tweet_archive_copy.retweeted_status_timestamp = pd.to_datetime(tweet_archive_copy.re
```

Test

In [8]:
```
tweet_archive_copy.head(3)
```

Out[8]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56+00:00	href="http://twitter.co
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27+00:00	href="http://twitter.co

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03+00:00	href="http://twitter.co

- ### **Issue**: Replace inconstitent name

Define:

In [10]:
```
tweet_archive_copy[tweet_archive_copy.name.str.islower() == True].name.unique()
```

Out[10]:
```
array(['such', 'a', 'quite', 'not', 'one', 'incredibly', 'mad', 'an',
       'very', 'just', 'my', 'his', 'actually', 'getting', 'this',
       'unacceptable', 'all', 'old', 'infuriating', 'the', 'by',
       'officially', 'life', 'light', 'space'], dtype=object)
```

Code

In [11]:
```
for i in tweet_archive_copy[tweet_archive_copy.name.str.islower() == True].name.uniq
    tweet_archive_copy.replace(i, 'None', inplace= True)
```

Test

In [10]:
```
tweet_archive_copy.name.unique()
```

Out[10]:
```
array(['Phineas', 'Tilly', 'Archie', 'Darla', 'Franklin', 'None', 'Jax',
       'Zoey', 'Cassie', 'Koda', 'Bruno', 'Ted', 'Stuart', 'Oliver',
       'Jim', 'Zeke', 'Ralphus', 'Canela', 'Gerald', 'Jeffrey', 'Maya',
       'Mingus', 'Derek', 'Roscoe', 'Waffles', 'Jimbo', 'Maisey', 'Lilly',
       'Earl', 'Lola', 'Kevin', 'Yogi', 'Noah', 'Bella', 'Grizzwald',
       'Rusty', 'Gus', 'Stanley', 'Alfy', 'Koko', 'Rey', 'Gary', 'Elliot',
       'Louis', 'Jesse', 'Romeo', 'Bailey', 'Duddles', 'Jack', 'Emmy',
       'Steven', 'Beau', 'Snoopy', 'Shadow', 'Terrance', 'Aja', 'Penny',
       'Dante', 'Nelly', 'Ginger', 'Benedict', 'Venti', 'Goose', 'Nugget',
       'Cash', 'Coco', 'Jed', 'Sebastian', 'Walter', 'Sierra', 'Monkey',
       'Harry', 'Kody', 'Lassie', 'Rover', 'Napolean', 'Dawn', 'Boomer',
       'Cody', 'Rumble', 'Clifford', 'Dewey', 'Scout', 'Gizmo', 'Cooper',
       'Harold', 'Shikha', 'Jamesy', 'Lili', 'Sammy', 'Meatball',
       'Paisley', 'Albus', 'Neptune', 'Quinn', 'Belle', 'Zooey', 'Dave',
       'Jersey', 'Hobbes', 'Burt', 'Lorenzo', 'Carl', 'Jordy', 'Milky',
       'Trooper', 'Winston', 'Sophie', 'Wyatt', 'Rosie', 'Thor', 'Oscar',
       'Luna', 'Callie', 'Cermet', 'George', 'Marlee', 'Arya', 'Einstein',
       'Alice', 'Rumpole', 'Benny', 'Aspen', 'Jarod', 'Wiggles',
       'General', 'Sailor', 'Astrid', 'Iggy', 'Snoop', 'Kyle', 'Leo',
       'Riley', 'Gidget', 'Noosh', 'Odin', 'Jerry', 'Charlie', 'Georgie',
       'Rontu', 'Cannon', 'Furzey', 'Daisy', 'Tuck', 'Barney', 'Vixen',
       'Jarvis', 'Mimosa', 'Pickles', 'Bungalo', 'Brady', 'Margo',
       'Sadie', 'Hank', 'Tycho', 'Stephan', 'Indie', 'Winnie', 'Bentley',
       'Ken', 'Max', 'Maddie', 'Pipsy', 'Monty', 'Sojourner', 'Odie',
       'Arlo', 'Sunny', 'Vincent', 'Lucy', 'Clark', 'Mookie', 'Meera',
       'Buddy', 'Ava', 'Rory', 'Eli', 'Ash', 'Tucker', 'Tobi', 'Chester',
       'Wilson', 'Sunshine', 'Lipton', 'Gabby', 'Bronte', 'Poppy',
       'Rhino', 'Willow', 'Orion', 'Eevee', 'Smiley', 'Logan', 'Moreton',
       'Klein', 'Miguel', 'Emanuel', 'Kuyu', 'Dutch', 'Pete', 'Scooter',
       'Reggie', 'Kyro', 'Samson', 'Loki', 'Mia', 'Malcolm', 'Dexter',
```

```
'Alfie', 'Fiona', 'Mutt', 'Bear', 'Doobert', 'Beebop', 'Alexander',
'Sailer', 'Brutus', 'Kona', 'Boots', 'Ralphie', 'Phil', 'Cupid',
'Pawnd', 'Pilot', 'Ike', 'Mo', 'Toby', 'Sweet', 'Pablo', 'Nala',
'Balto', 'Crawford', 'Gabe', 'Mattie', 'Jimison', 'Hercules',
'Duchess', 'Harlso', 'Sampson', 'Sundance', 'Luca', 'Flash',
'Finn', 'Peaches', 'Howie', 'Jazzy', 'Anna', 'Bo', 'Seamus',
'Wafer', 'Chelsea', 'Tom', 'Moose', 'Florence', 'Autumn', 'Dido',
'Eugene', 'Herschel', 'Strudel', 'Tebow', 'Chloe', 'Betty',
'Timber', 'Binky', 'Dudley', 'Comet', 'Larry', 'Levi', 'Akumi',
'Titan', 'Olivia', 'Alf', 'Oshie', 'Bruce', 'Chubbs', 'Sky',
'Atlas', 'Eleanor', 'Layla', 'Rocky', 'Baron', 'Tyr', 'Bauer',
'Swagger', 'Brandi', 'Mary', 'Moe', 'Halo', 'Augie', 'Craig',
'Sam', 'Hunter', 'Pavlov', 'Maximus', 'Wallace', 'Ito', 'Milo',
'Ollie', 'Cali', 'Lennon', 'Major', 'Duke', 'Reginald', 'Sansa',
'Shooter', 'Django', 'Diogi', 'Sonny', 'Philbert', 'Marley',
'Severus', 'Ronnie', 'Anakin', 'Bones', 'Mauve', 'Chef', 'Doc',
'Sobe', 'Longfellow', 'Mister', 'Iroh', 'Baloo', 'Stubert',
'Paull', 'Timison', 'Davey', 'Pancake', 'Tyrone', 'Snicku', 'Ruby',
'Brody', 'Rizzy', 'Mack', 'Butter', 'Nimbus', 'Laika', 'Dobby',
'Juno', 'Maude', 'Lily', 'Newt', 'Benji', 'Nida', 'Robin',
'Monster', 'BeBe', 'Remus', 'Mabel', 'Misty', 'Happy', 'Mosby',
'Maggie', 'Leela', 'Ralphy', 'Brownie', 'Meyer', 'Stella', 'Frank',
'Tonks', 'Lincoln', 'Oakley', 'Dale', 'Rizzo', 'Arnie', 'Pinot',
'Dallas', 'Hero', 'Frankie', 'Stormy', 'Mairi', 'Loomis', 'Godi',
'Kenny', 'Deacon', 'Timmy', 'Harper', 'Chipson', 'Combo', 'Dash',
'Bell', 'Hurley', 'Jay', 'Mya', 'Strider', 'Wesley', 'Solomon',
'Huck', 'O', 'Blue', 'Finley', 'Sprinkles', 'Heinrich',
'Shakespeare', 'Fizz', 'Chip', 'Grey', 'Roosevelt', 'Gromit',
'Willem', 'Dakota', 'Dixie', 'Al', 'Jackson', 'Carbon', 'DonDon',
'Kirby', 'Lou', 'Nollie', 'Chevy', 'Tito', 'Louie', 'Rupert',
'Rufus', 'Brudge', 'Shadoe', 'Colby', 'Angel', 'Brat', 'Tove',
'Aubie', 'Kota', 'Eve', 'Glenn', 'Shelby', 'Sephie', 'Bonaparte',
'Albert', 'Wishes', 'Rose', 'Theo', 'Rocco', 'Fido', 'Emma',
'Spencer', 'Lilli', 'Boston', 'Brandonald', 'Corey', 'Leonard',
'Chompsky', 'Beckham', 'Devón', 'Gert', 'Watson', 'Rubio', 'Keith',
'Dex', 'Carly', 'Ace', 'Tayzie', 'Grizzie', 'Fred', 'Gilbert',
'Zoe', 'Stewie', 'Calvin', 'Lilah', 'Spanky', 'Jameson', 'Piper',
'Atticus', 'Blu', 'Dietrich', 'Divine', 'Tripp', 'Cora', 'Huxley',
'Keurig', 'Bookstore', 'Linus', 'Abby', 'Shaggy', 'Shiloh',
'Gustav', 'Arlen', 'Percy', 'Lenox', 'Sugar', 'Harvey', 'Blanket',
'Geno', 'Stark', 'Beya', 'Kilo', 'Kayla', 'Maxaroni', 'Doug',
'Edmund', 'Aqua', 'Theodore', 'Chase', 'Rorie', 'Simba', 'Charles',
'Bayley', 'Axel', 'Storkson', 'Remy', 'Chadrick', 'Kellogg',
'Buckley', 'Livvie', 'Terry', 'Hermione', 'Ralpher', 'Aldrick',
'Rooney', 'Crystal', 'Ziva', 'Stefan', 'Pupcasso', 'Puff',
'Flurpson', 'Coleman', 'Enchilada', 'Raymond', 'Rueben',
'Cilantro', 'Karll', 'Sprout', 'Blitz', 'Bloop', 'Lillie',
'Ashleigh', 'Kreggory', 'Sarge', 'Luther', 'Ivar', 'Jangle',
'Schnitzel', 'Panda', 'Berkeley', 'Ralphé', 'Charleson', 'Clyde',
'Harnold', 'Sid', 'Pippa', 'Otis', 'Carper', 'Bowie',
'Alexanderson', 'Suki', 'Barclay', 'Skittle', 'Ebby', 'Flávio',
'Smokey', 'Link', 'Jennifur', 'Ozzy', 'Bluebert', 'Stephanus',
'Bubbles', 'Zeus', 'Bertson', 'Nico', 'Michelangelope', 'Siba',
'Calbert', 'Curtis', 'Travis', 'Thumas', 'Kanu', 'Lance', 'Opie',
'Kane', 'Olive', 'Chuckles', 'Staniel', 'Sora', 'Beemo', 'Gunner',
'Lacy', 'Tater', 'Olaf', 'Cecil', 'Vince', 'Karma', 'Billy',
'Walker', 'Rodney', 'Klevin', 'Malikai', 'Bobble', 'River',
'Jebberson', 'Remington', 'Farfle', 'Jiminus', 'Clarkus',
'Finnegus', 'Cupcake', 'Kathmandu', 'Ellie', 'Katie', 'Kara',
'Adele', 'Zara', 'Ambrose', 'Jimothy', 'Bode', 'Terrenth', 'Reese',
'Chesterson', 'Lucia', 'Bisquick', 'Ralphson', 'Socks', 'Rambo',
'Rudy', 'Fiji', 'Rilo', 'Bilbo', 'Coopson', 'Yoda', 'Millie',
'Chet', 'Crouton', 'Daniel', 'Kaia', 'Murphy', 'Dotsy', 'Eazy',
'Coops', 'Fillup', 'Miley', 'Charl', 'Reagan', 'Yukon', 'CeCe',
```

```
'Cuddles', 'Claude', 'Jessiga', 'Carter', 'Ole', 'Pherb',
'Blipson', 'Reptar', 'Trevith', 'Berb', 'Bob', 'Colin', 'Brian',
'Oliviér', 'Grady', 'Kobe', 'Freddery', 'Bodie', 'Dunkin', 'Wally',
'Tupawc', 'Amber', 'Edgar', 'Teddy', 'Kingsley', 'Brockly',
'Richie', 'Molly', 'Vinscent', 'Cedrick', 'Hazel', 'Lolo', 'Eriq',
'Phred', 'Oddie', 'Maxwell', 'Geoff', 'Covach', 'Durg', 'Fynn',
'Ricky', 'Herald', 'Lucky', 'Ferg', 'Trip', 'Clarence', 'Hamrick',
'Brad', 'Pubert', 'Frönq', 'Derby', 'Lizzie', 'Ember', 'Blakely',
'Opal', 'Marq', 'Kramer', 'Barry', 'Gordon', 'Baxter', 'Mona',
'Horace', 'Crimson', 'Birf', 'Hammond', 'Lorelei', 'Marty',
'Brooks', 'Petrick', 'Hubertson', 'Gerbald', 'Oreo', 'Bruiser',
'Perry', 'Bobby', 'Jeph', 'Obi', 'Tino', 'Kulet', 'Sweets', 'Lupe',
'Tiger', 'Jiminy', 'Griffin', 'Banjo', 'Brandy', 'Lulu', 'Darrel',
'Taco', 'Joey', 'Patrick', 'Kreg', 'Todo', 'Tess', 'Ulysses',
'Toffee', 'Apollo', 'Asher', 'Glacier', 'Chuck', 'Champ', 'Ozzie',
'Griswold', 'Cheesy', 'Moofasa', 'Hector', 'Goliath', 'Kawhi',
'Emmie', 'Penelope', 'Willie', 'Rinna', 'Mike', 'William',
'Dwight', 'Evy', 'Rascal', 'Linda', 'Tug', 'Tango', 'Grizz',
'Jerome', 'Crumpet', 'Jessifer', 'Izzy', 'Ralph', 'Sandy',
'Humphrey', 'Tassy', 'Juckson', 'Chuq', 'Tyrus', 'Karl',
'Godzilla', 'Vinnie', 'Kenneth', 'Herm', 'Bert', 'Striker',
'Donny', 'Pepper', 'Bernie', 'Buddah', 'Lenny', 'Arnold', 'Zuzu',
'Mollie', 'Laela', 'Tedders', 'Superpup', 'Rufio', 'Jeb', 'Rodman',
'Jonah', 'Chesney', 'Henry', 'Bobbay', 'Mitch', 'Kaiya', 'Acro',
'Aiden', 'Obie', 'Dot', 'Shnuggles', 'Kendall', 'Jeffri', 'Steve',
'Mac', 'Fletcher', 'Kenzie', 'Pumpkin', 'Schnozz', 'Gustaf',
'Cheryl', 'Ed', 'Leonidas', 'Norman', 'Caryl', 'Scott', 'Taz',
'Darby', 'Jackie', 'Jazz', 'Franq', 'Pippin', 'Rolf', 'Snickers',
'Ridley', 'Cal', 'Bradley', 'Bubba', 'Tuco', 'Patch', 'Mojo',
'Batdog', 'Dylan', 'Mark', 'JD', 'Alejandro', 'Scruffers', 'Pip',
'Julius', 'Tanner', 'Sparky', 'Anthony', 'Holly', 'Jett', 'Amy',
'Sage', 'Andy', 'Mason', 'Trigger', 'Antony', 'Creg', 'Traviss',
'Gin', 'Jeffrie', 'Danny', 'Ester', 'Pluto', 'Bloo', 'Edd',
'Willy', 'Herb', 'Damon', 'Peanut', 'Nigel', 'Butters', 'Sandra',
'Fabio', 'Randall', 'Liam', 'Tommy', 'Ben', 'Raphael', 'Julio',
'Andru', 'Kloey', 'Shawwn', 'Skye', 'Kollin', 'Ronduh', 'Billl',
'Saydee', 'Dug', 'Tessa', 'Sully', 'Kirk', 'Ralf', 'Clarq',
'Jaspers', 'Samsom', 'Harrison', 'Chaz', 'Jeremy', 'Jaycob',
'Lambeau', 'Ruffles', 'Amélie', 'Bobb', 'Banditt', 'Kevon',
'Winifred', 'Hanz', 'Churlie', 'Zeek', 'Timofy', 'Maks',
'Jomathan', 'Kallie', 'Marvin', 'Spark', 'Gòrdón', 'Jo', 'DayZ',
'Jareld', 'Torque', 'Ron', 'Skittles', 'Cleopatricia', 'Erik',
'Stu', 'Tedrick', 'Filup', 'Kial', 'Naphaniel', 'Dook', 'Hall',
'Philippe', 'Biden', 'Fwed', 'Genevieve', 'Joshwa', 'Bradlay',
'Clybe', 'Keet', 'Carll', 'Jockson', 'Josep', 'Lugan',
'Christoper'], dtype=object)
```

Issue : Assign 10 to rating_denominator column value

Define

In [9]:
```
tweet_archive_copy.rating_denominator.unique()
```

Out[9]:
```
array([ 10,   0,  15,  70,   7,  11, 150, 170,  20,  50,  90,  80,  40,
       130, 110,  16, 120,   2], dtype=int64)
```

Code

In [10]:
```
tweet_archive_copy.rating_denominator = tweet_archive_copy.rating_denominator = 10
```

Test

In [13]:
```
tweet_archive_copy.rating_denominator.unique()
```

Out[13]:
```
array([10], dtype=int64)
```

Issue

Define

Create function to remove lines to get text df_combine =

Code

In [12]:
```
for i in tweet_archive_copy.text[:]:
    if i.find('https') == -1:
        print(i)
        break
```

```
@NonWhiteHat @MayhewMayhem omg hello tanner you are a scary good boy 12/10 would pet
with extreme caution
```

In [13]:
```
def link_remover(x):
    pos_https = x.find('https')
    if pos_https == -1:
        x = x
    else:
        x = x[:pos_https - 1]

    return x

tweet_archive_copy.text = tweet_archive_copy.text.apply(lambda x: link_remover(x))
```

Test

In [14]:
```
for i in tweet_archive_copy.text[:5]:
    print(i)
```

```
This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/1
0
This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she's
available for pats, snugs, boops, the whole bit. 13/10
This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall grass. You
never know when one may strike. 12/10
This is Darla. She commenced a snooze mid meal. 13/10 happens to the best of us
This is Franklin. He would like you to stop calling him "cute." He is a very fierce
shark and should be respected as such. 12/10 #BarkWeek
```

Issue: Extraneous columns

Define

Create a 'dog_type' column by combining 'doggo', 'floofer', 'pupper' and 'puppo'

Code

In [15]:
```
#creating a function

def dog_name(x):
    name = []
```

```
    if x['doggo'] == 'doggo':
        name.append('doggo')
    if x['floofer'] == 'floofer':
        name.append('floofer')
    if x['pupper'] == 'pupper':
        name.append('pupper')
    if x['puppo'] == 'puppo':
        name.append('puppo')

    if not name:
        return 'None'
    else:
        return ','.join(name)

tweet_archive_copy['dog_type'] = tweet_archive_copy.apply(lambda x:dog_name(x), axis
```

In [16]:
```
# Drop the extraneous columns
tweet_archive_copy.drop(['doggo', 'floofer', 'pupper', 'puppo'], axis = 1, inplace=T
```

Test

In [37]:
```
tweet_archive_copy.sample(3)
```

Out[37]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
750	779123168116150273	NaN	NaN	2016-09-23 01:00:13 +0000	href="http://twitter.co
1234	712717840512598017	NaN	NaN	2016-03-23 19:09:09 +0000	href="http://twitter.co
1891	674764817387900928	NaN	NaN	2015-12-10 01:37:23 +0000	href="http://twitter.co

In [19]:
```
tweet_archive_copy.dog_type.value_counts()
```

Out[19]:
```
None              1976
pupper             245
doggo               83
puppo               29
doggo,pupper        12
floofer              9
doggo,puppo          1
doggo,floofer        1
Name: dog_type, dtype: int64
```

Define

Change tweet_id to string

Code

In [17]:
```
tweet_archive_copy.tweet_id = tweet_archive_copy.tweet_id.astype('str')
```

In [35]:
```
tweet_archive_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2356 non-null   object
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   datetime64[ns, UTC]
 4   source                      2356 non-null   object
 5   text                        2356 non-null   object
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    datetime64[ns, UTC]
 9   expanded_urls               2297 non-null   object
 10  rating_numerator            2356 non-null   int64
 11  rating_denominator          2356 non-null   int64
 12  name                        2356 non-null   object
 13  dog_type                    2356 non-null   object
dtypes: datetime64[ns, UTC](2), float64(4), int64(2), object(6)
memory usage: 202.5+ KB
```

Image Prediction

Define

Capitalize the first letter in the values of p1, p2, p3 columns

Code

In [18]:
```
img_prediction_copy.p1 = img_prediction_copy.p1.str.capitalize()
img_prediction_copy.p2 = img_prediction_copy.p2.str.capitalize()
img_prediction_copy.p3 = img_prediction_copy.p3.str.capitalize()
```

Test

In [19]:
```
img_prediction_copy.head(3)
```

Out[19]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_she

Define

Replace _ and - with white space in column p1, p2 and p3 columns

Code

In [20]:
```
img_prediction_copy.p1 = img_prediction_copy.p1.str.replace('_', ' ')
img_prediction_copy.p2 = img_prediction_copy.p2.str.replace('_', ' ')
img_prediction_copy.p3 = img_prediction_copy.p3.str.replace('_', ' ')

img_prediction_copy.p1 = img_prediction_copy.p1.str.replace('-', ' ')
img_prediction_copy.p2 = img_prediction_copy.p2.str.replace('-', ' ')
img_prediction_copy.p3 = img_prediction_copy.p3.str.replace('-', ' ')
```

Test

In [23]:
```
img_prediction_copy.head(3)
```

Out[23]:

	tweet_id	jpg_url	img_num	p1	p1_
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh springer spaniel	0.465
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redbone	0.506
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German shepherd	0.596

Define

Change tweet_id to string

Code

In [21]:
```
img_prediction_copy.tweet_id = img_prediction_copy.tweet_id.astype('str')
```

Test

In [32]:
```
img_prediction_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   object 
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(1), object(5)
memory usage: 111.5+ KB
```

Twitter's API

Define

set tweet_ids to string

Code

In [22]:
```
tweetAPI_copy.id = tweetAPI_copy.id.astype('str')
```

Test

In [25]:
```
tweetAPI_copy.head()
```

Out[25]:

	id	retweet_count	favorite_count
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048

Define

Rename the column names

Code

In [23]:
```
tweetAPI_copy.rename(columns={'id':'tweet_id', 'retweet_count':'retweets', 'favorite
```

Test

In [24]:
```
tweetAPI_copy
```

Out[24]:

	tweet_id	retweets	likes
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048
...	...	...	...
2349	666049248165822465	41	111
2350	666044226329800704	147	311
2351	666033412701032449	47	128
2352	666029285002620928	48	132
2353	666020888022790149	532	2535

2354 rows × 3 columns

In [25]:
```
tweetAPI_copy.info()
```

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

In [45]:
```
tweetAPI_copy.describe()
```

Out[45]:

	retweets	likes
count	2354.000000	2354.000000
mean	3164.797366	8080.968564
std	5284.770364	11814.771334
min	0.000000	0.000000
25%	624.500000	1415.000000
50%	1473.500000	3603.500000
75%	3652.000000	10122.250000
max	79515.000000	132810.000000

Combine the three datasets

Define

Combine all datasets using merge

Code

In [26]:
```
df_combine = tweet_archive_copy.merge(img_prediction_copy, on='tweet_id', how='inner
df_combine = df_combine.merge(tweetAPI_copy, on='tweet_id', how='inner')
```

In [27]:
```
df_combine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2073 entries, 0 to 2072
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype              
---  ------                      --------------  -----              
 0   tweet_id                    2073 non-null   object             
 1   in_reply_to_status_id       23 non-null     float64            
 2   in_reply_to_user_id         23 non-null     float64            
 3   timestamp                   2073 non-null   datetime64[ns, UTC]
 4   source                      2073 non-null   object             
 5   text                        2073 non-null   object             
```

```
 6   retweeted_status_id          79 non-null     float64
 7   retweeted_status_user_id     79 non-null     float64
 8   retweeted_status_timestamp   79 non-null     datetime64[ns, UTC]
 9   expanded_urls                2073 non-null   object
 10  rating_numerator             2073 non-null   int64
 11  rating_denominator           2073 non-null   int64
 12  name                         2073 non-null   object
 13  dog_type                     2073 non-null   object
 14  jpg_url                      2073 non-null   object
 15  img_num                      2073 non-null   int64
 16  p1                           2073 non-null   object
 17  p1_conf                      2073 non-null   float64
 18  p1_dog                       2073 non-null   bool
 19  p2                           2073 non-null   object
 20  p2_conf                      2073 non-null   float64
 21  p2_dog                       2073 non-null   bool
 22  p3                           2073 non-null   object
 23  p3_conf                      2073 non-null   float64
 24  p3_dog                       2073 non-null   bool
 25  retweets                     2073 non-null   int64
 26  likes                        2073 non-null   int64
dtypes: bool(3), datetime64[ns, UTC](2), float64(7), int64(5), object(10)
memory usage: 411.0+ KB
```

In [48]:
```
df_combine.head()
```

Out[48]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56+00:00	href="http://twitter.co
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27+00:00	href="http://twitter.co
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03+00:00	href="http://twitter.co
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51+00:00	href="http://twitter.co
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24+00:00	href="http://twitter.co

In [28]:
```
# Calculate the value  of rating
df_combine['rating']  = df_combine.rating_numerator/df_combine.rating_denominator
```

In [50]:
```
df_combine.sample(2)
```

Out[50]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
924	731156023742988288	NaN	NaN	2016-05-13 16:15:54+00:00	href="http://twitte
1423	681891461017812993	NaN	NaN	2015-12-29 17:36:07+00:00	href="http://twitte

In [29]:
```
# Rename few columns
df_combine.rename(columns={'rating_numerator':'numerator', 'rating_denominator':'den

# Drop unnecessary columns
df_combine.drop(['retweeted_status_id', 'retweeted_status_user_id', 'retweeted_statu

# Avoid collapsing of columns
pd.set_option('display.max_columns', None)
```

Test

In [30]:
```
df_combine.head(2)
```

Out[30]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56+00:00	href="http://twitter.co
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27+00:00	href="http://twitter.co

Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".

In [31]:
```
df_combine.to_csv('data/twitter_archive_master.csv', index=False)
```

Analyzing and Visualizing Data

In this section, analyze and visualize your wrangled data. You must produce at least **three (3) insights and one (1) visualization.**

In [32]:
```
data = pd.read_csv('data/twitter_archive_master.csv')
data.head()
```

Out[32]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56+00:00	href="http://twitter.co
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27+00:00	href="http://twitter.co
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03+00:00	href="http://twitter.co
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51+00:00	href="http://twitter.co
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24+00:00	href="http://twitter.co

In [70]:
```
data.loc[data.rating > 2]
```

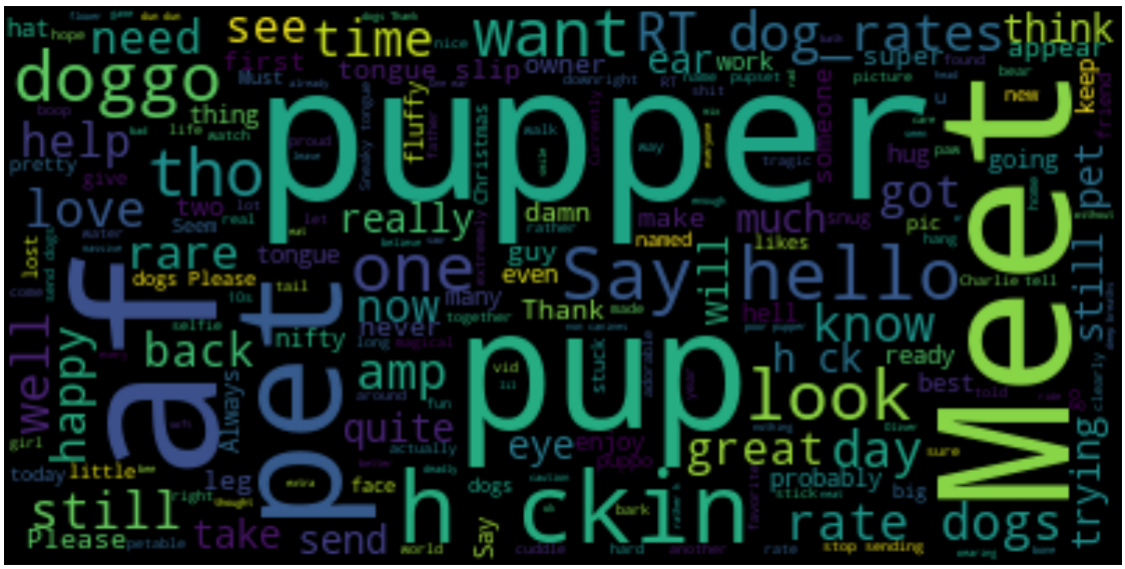
Out[70]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
345	820690176645140481	NaN	NaN	2017-01-15 17:52:40+00:00	href="http://twi
415	810984652412424192	NaN	NaN	2016-12-19 23:06:23+00:00	href="http://twi
558	786709082849828864	NaN	NaN	2016-10-13 23:23:56+00:00	href="http://twi
614	778027034220126208	NaN	NaN	2016-09-20 00:24:34+00:00	href="http://twi

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
734	758467244762497024	NaN	NaN	2016-07-28 01:00:57+00:00	href="http://twi
802	749981277374128128	NaN	NaN	2016-07-04 15:00:45+00:00	href="https://abou
924	731156023742988288	NaN	NaN	2016-05-13 16:15:54+00:00	href="http://twi
1001	716439118184652801	NaN	NaN	2016-04-03 01:36:11+00:00	href="http://twi
1022	713900603437621249	NaN	NaN	2016-03-27 01:29:02+00:00	href="http://twi
1047	710658690886586372	NaN	NaN	2016-03-18 02:46:49+00:00	href="http://twi
1065	709198395643068416	NaN	NaN	2016-03-14 02:04:08+00:00	href="http://twi
1131	704054845121142784	NaN	NaN	2016-02-28 21:25:30+00:00	href="http://twi
1207	697463031882764288	NaN	NaN	2016-02-10 16:51:59+00:00	href="http://twi
1379	684225744407494656	6.842229e+17	4.196984e+09	2016-01-05 04:11:44+00:00	href="http://twi
1380	684222868335505415	NaN	NaN	2016-01-05 04:00:18+00:00	href="http://twi
1451	680494726643068929	NaN	NaN	2015-12-25 21:06:00+00:00	href="http://twi
1512	677716515794329600	NaN	NaN	2015-12-18 05:06:23+00:00	href="http://twi
1571	675853064436391936	NaN	NaN	2015-12-13 01:41:41+00:00	href="http://twi
1795	670842764863651840	NaN	NaN	2015-11-29 05:52:33+00:00	href="http://twi

In [33]:
```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [57]:
```
#Word Cloud of texts
text = ' '.join(review for review in df_combine.text)
# Create stopword list
stopwords = set(STOPWORDS)
stopwords.update(['puppy', 'Bruno', 'dog', 'good', 'boy', 'cool'])
#print(stopwords)

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color='black').generate(text)

# display the generate image
# the matplotlib way
plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [58]:
```
mask_dog = np.array(Image.open('data/kisspng-dog-paw-puppy-cat-clip-art-paw-prints-5
mask_dog
```

Out[58]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
```

```
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[254, 254, 254],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [254, 254, 254],
        [254, 254, 254],
        [254, 254, 254]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

In [59]:

```
# Generate a word cloud image
wordcloud_ = WordCloud(stopwords=stopwords, background_color='black',
    max_words=1000, mask=mask_dog, contour_width=3, contour_color='blue')
wordcloud_.generate(text)
wordcloud_.to_file('data/dog.png')

# show
plt.figure(figsize=([10, 10]))
plt.imshow(wordcloud_, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Visualization

Programmatic accessing

In [34]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2073 entries, 0 to 2072
Data columns (total 25 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   tweet_id               2073 non-null   int64  
 1   in_reply_to_status_id  23 non-null     float64
 2   in_reply_to_user_id    23 non-null     float64
 3   timestamp              2073 non-null   object 
 4   source                 2073 non-null   object 
 5   text                   2073 non-null   object 
 6   expanded_urls          2073 non-null   object 
 7   numerator              2073 non-null   int64  
 8   denominator            2073 non-null   int64  
 9   name                   2073 non-null   object 
 10  dog_type               2073 non-null   object 
 11  jpg_url                2073 non-null   object 
 12  img_num                2073 non-null   int64  
 13  p1                     2073 non-null   object 
 14  p1_conf                2073 non-null   float64
 15  p1_dog                 2073 non-null   bool   
```

```
 16  p2                    2073 non-null   object
 17  p2_conf               2073 non-null   float64
 18  p2_dog                2073 non-null   bool
 19  p3                    2073 non-null   object
 20  p3_conf               2073 non-null   float64
 21  p3_dog                2073 non-null   bool
 22  retweets              2073 non-null   int64
 23  likes                 2073 non-null   int64
 24  rating                2073 non-null   float64
dtypes: bool(3), float64(6), int64(6), object(10)
memory usage: 362.5+ KB
```

In [35]:

```
# Convert columns to their  appropriate typs
# Set timestamp to index

data.tweet_id = data.tweet_id.astype('object')
data.timestamp = pd.to_datetime(data.timestamp)
data.dog_type = data.dog_type.astype('category')

data.set_index('timestamp', inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2073 entries, 2017-08-01 16:23:56+00:00 to 2015-11-15 22:32:08+00:00
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   tweet_id              2073 non-null   object
 1   in_reply_to_status_id  23 non-null    float64
 2   in_reply_to_user_id    23 non-null    float64
 3   source                2073 non-null   object
 4   text                  2073 non-null   object
 5   expanded_urls         2073 non-null   object
 6   numerator             2073 non-null   int64
 7   denominator           2073 non-null   int64
 8   name                  2073 non-null   object
 9   dog_type              2073 non-null   category
 10  jpg_url               2073 non-null   object
 11  img_num               2073 non-null   int64
 12  p1                    2073 non-null   object
 13  p1_conf               2073 non-null   float64
 14  p1_dog                2073 non-null   bool
 15  p2                    2073 non-null   object
 16  p2_conf               2073 non-null   float64
 17  p2_dog                2073 non-null   bool
 18  p3                    2073 non-null   object
 19  p3_conf               2073 non-null   float64
 20  p3_dog                2073 non-null   bool
 21  retweets              2073 non-null   int64
 22  likes                 2073 non-null   int64
 23  rating                2073 non-null   float64
dtypes: bool(3), category(1), float64(6), int64(5), object(9)
memory usage: 348.6+ KB
```

In [36]:

```
data.describe()
```

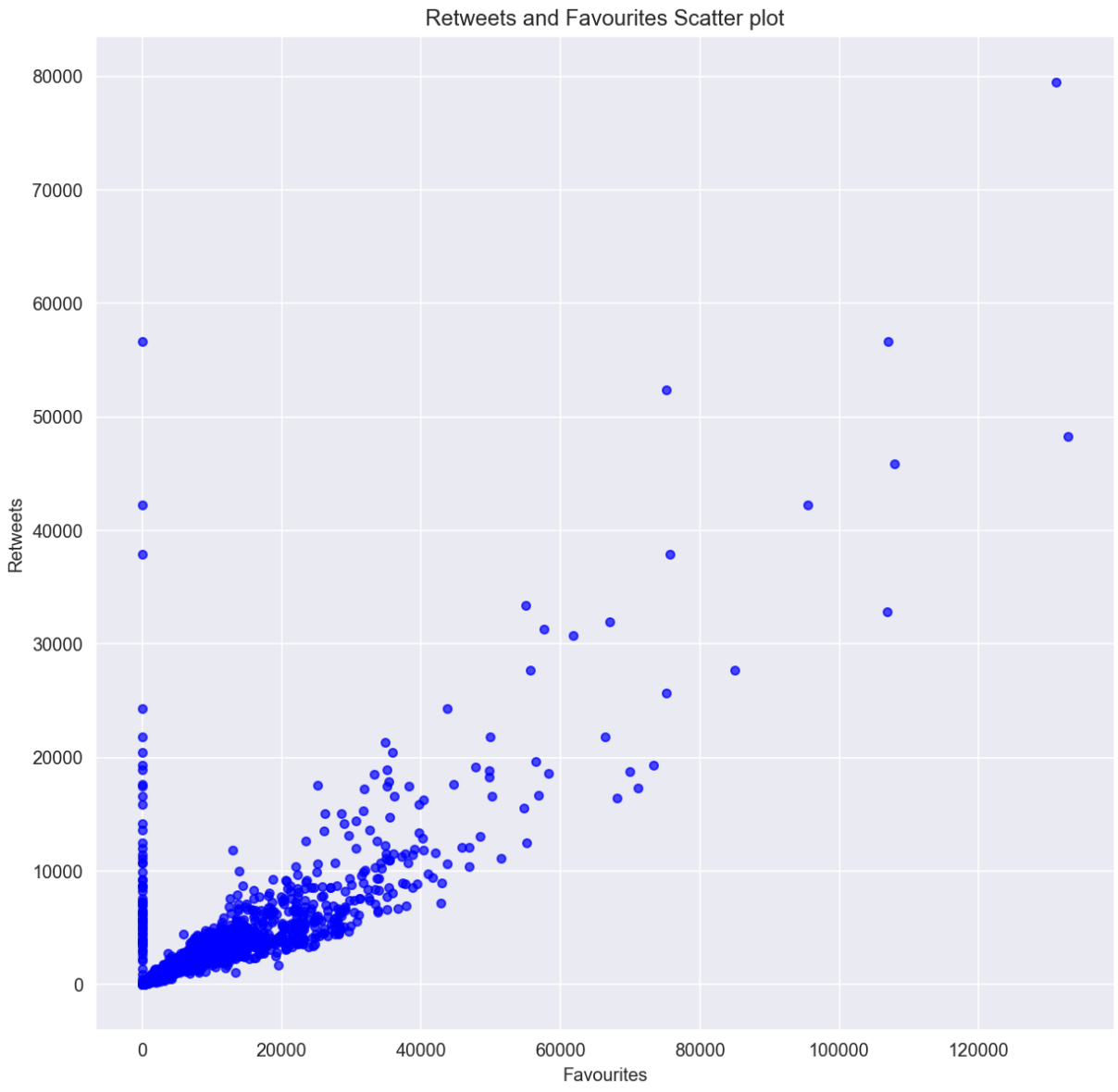
Out[36]:

	in_reply_to_status_id	in_reply_to_user_id	numerator	denominator	img_num	p1_conf
count	2.300000e+01	2.300000e+01	2073.000000	2073.000000	2073.000000	2073.000000
mean	6.978112e+17	4.196984e+09	12.265798	10.511819	1.203570	0.594532
std	4.359384e+16	0.000000e+00	40.699924	7.180517	0.561856	0.271234

	in_reply_to_status_id	in_reply_to_user_id	numerator	denominator	img_num	p1_conf
min	6.671522e+17	4.196984e+09	0.000000	2.000000	1.000000	0.044333
25%	6.732411e+17	4.196984e+09	10.000000	10.000000	1.000000	0.364095
50%	6.757073e+17	4.196984e+09	11.000000	10.000000	1.000000	0.588230
75%	7.031489e+17	4.196984e+09	12.000000	10.000000	1.000000	0.843911
max	8.558181e+17	4.196984e+09	1776.000000	170.000000	4.000000	1.000000

In [37]:
```
data.plot(kind='scatter', x='likes', y='retweets', alpha=0.7, figsize=(10,10), color

plt.xlabel('Favourites')
plt.ylabel('Retweets')
plt.title('Retweets and Favourites Scatter plot')
```

Out[37]:
```
Text(0.5, 1.0, 'Retweets and Favourites Scatter plot')
```

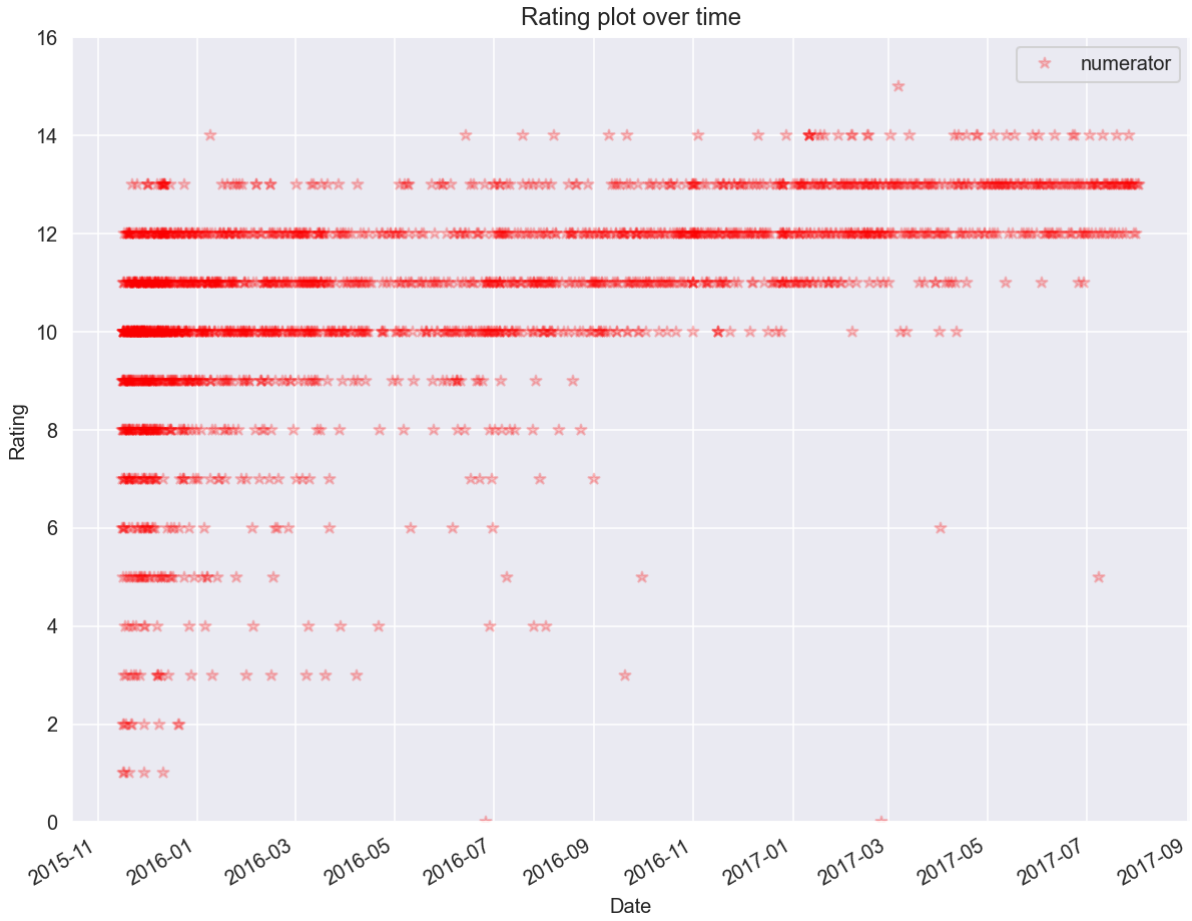


As likes increases , retweets increase. Hence; from the plot, they're linearly correlated

In [38]:
```
data.plot(y='numerator', ylim=[0,16], alpha=0.2, style='*', figsize=(10,8), color='r
```

```
plt.xlabel('Date')
plt.ylabel('Rating')
plt.title('Rating plot over time')
```

Out[38]:
```
Text(0.5, 1.0, 'Rating plot over time')
```



Numerator rating 12 is the most common

In [ ]:
```
from textblob import TextBlob #an api to analyze text
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
```

In [ ]:
```
data.text[10]
```

In [ ]:
```
TextBlob(data.text[10]).sentiment
```

In [ ]:
```
text = ' '.join(txt for txt in df.text)
```

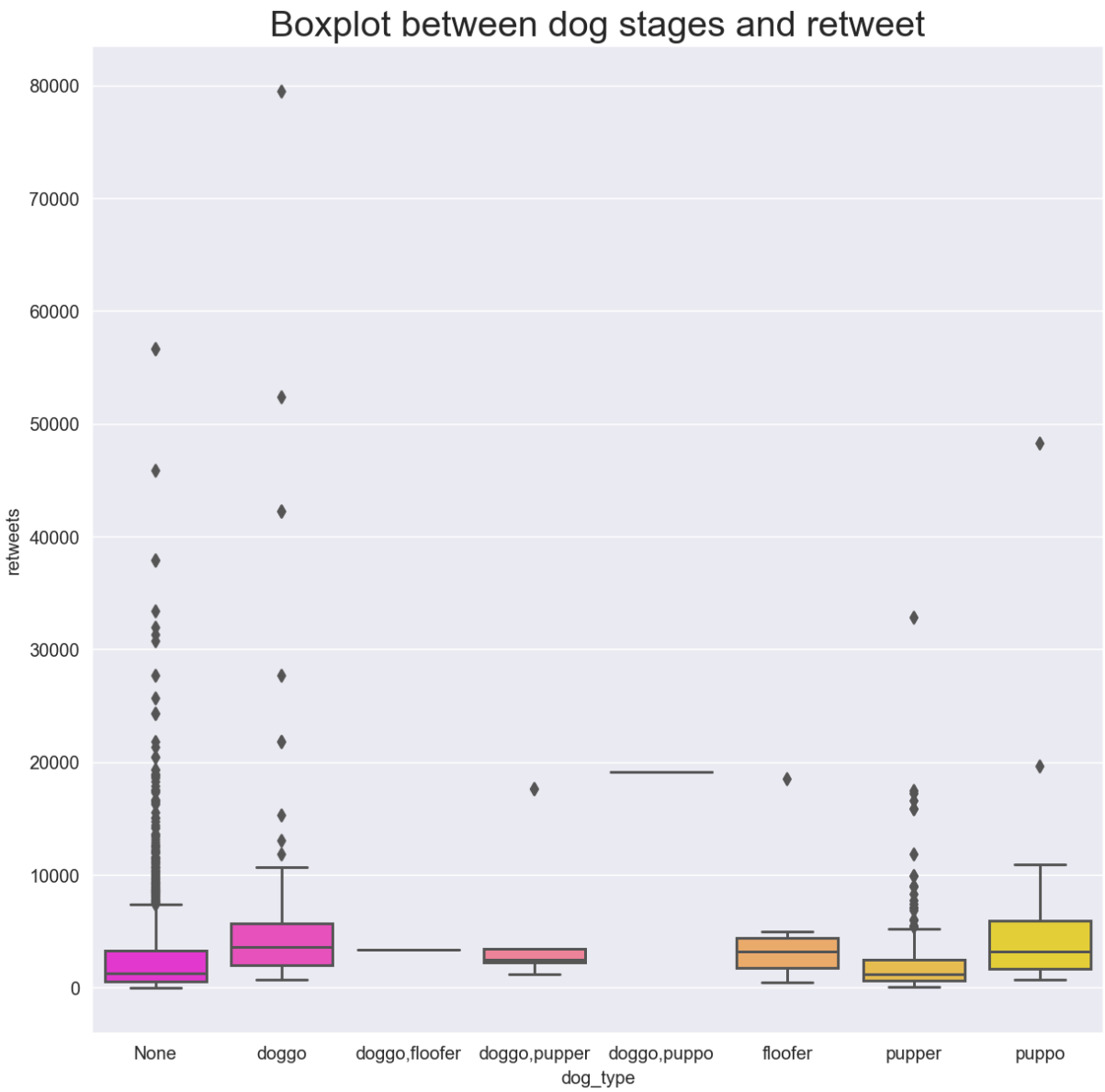
In [ ]:
```
bloblist_tags = list()
df_tags_str = data.text
for row in df_tags_str:
    blob = TextBlob(row)
    bloblist_tags.append(row, blob.sentiment.polarity, blob.sentiment.subjectivity)
    df_polarity_tags = pd.DataFrame(bloblist_tags, columns=['sentence', 'sentiment',

def f_tags(df_polarity_tags):
    if df_polarity_tags['sentiment'] > 0:
```

```
        val = 'Positive'
    elif df_polarity_tags['sentiment'] == 0:
        val = 'Neutral'
    else:
        val = 'Negative'
    return val

df_polarity_tags['sentiment'] = df_polarity_tags.apply(f_tags, axis=1)

plt.figure(figsize=(10, 10))
sns.set_style('whitegrid')
```

Most of texts are positive and secondly neutral and less than 50% in comparison to positive words are negative in polarity

In [39]:
```
plt.figure(figsize=(10, 10))
sn = sns.boxplot(x='dog_type', y='retweets', data = data, palette='spring')
sn.axes.set_title('Boxplot between dog stages and retweet', fontsize=20)
```

Out[39]:
```
Text(0.5, 1.0, 'Boxplot between dog stages and retweet')
```



This shows that most dogs are categorized as Puppo but highest retweets are about doggo category.

Insights:

1. As likes increases , retweets increase. Hence; from the plot, they're linearly correlated
2. Numerator rating 12 is the most common
3. This shows that most dogs are categorized as Puppo but highest retweets are about doggo category.