Udacity Project: Wrangling & Analyzing WeRateDogs Twitter Data

By: Cathy Moy

June 2019

Introduction

The goal of this project is to wrangle the WeRateDogs Twitter data to create insightful and meaningful analyses and visualizations. The Twitter archive contains very basic tweet information in JSON format. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. This project will showcase my data wrangling efforts with gathering, assessing, and cleaning the Twitter data for valuable analysis and visualization purposes.

In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import tweepy
from tweepy import OAuthHandler
import requests
import json
from timeit import default_timer as timer
import time
import os
```

Gather

1. Twitter Archive

The WeRateDogs Twitter archive, file on hand.

In [2]:
```
# Store and Read data
archives = pd.read_csv('twitter-archive-enhanced.csv')
```

In [3]:
```
archives.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 [4]:
```
archives.sort_values('timestamp')
archives.head()
```

Out[4]:

	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

In [5]:
```
archives
```

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://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte
				2017-07-	

In [6]:
```
archives['tweet_id'].astype(int)
archives
```

Out[6]:

	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://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte
				2017-07-	

2. Image Predictions

The tweet image predictions, i.e., what breed of dog (or other object, animal, etc.) is present in each tweet according to a neural network.

In [7]:
```
# Download URL programatically


url = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-pr
response = requests.get(url)

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

# Read TSV file
images = pd.read_csv('image-predictions.tsv', sep='\t' )
```

In [8]:
```
# Obtain more info
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
```

3. Each tweet's retweet count and favorite ("like") count at minimum

With use of the tweet IDs in the WeRateDogs Twitter archive.

In [9]:
```
# Query Twitter APIs for each JSON data using Python's Tweepy library

consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''

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

api = tweepy.API(auth_handler = auth,
                 parser = tweepy.parsers.JSONParser(),
                 wait_on_rate_limit = True,
                 wait_on_rate_limit_notify = True)
```

In [10]:
```
tweet_ids = archives.tweet_id.values
len(tweet_ids)
```

Out[10]:
```
2356
```

In [11]:
```
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 l
    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)
```

```
1: 892420643555336193
Fail
2: 892177421306343426
Fail
3: 891815181378084864
Fail
4: 891689557279858688
Fail
5: 891327558926688256
Fail
6: 891087950875897856
Fail
7: 890971913173991426
Fail
8: 890729181411237888
Fail
9: 890609185150312448
Fail
10: 890240255349198849
Fail
```

In [13]:
```
tweet_json = open('tweet_json.txt', 'r')
df_tweet = pd.DataFrame(columns=['tweet_id', 'retweets', 'favorites'])

for line in tweet_json:
    tweet = json.loads(line)
    df_tweet = df_tweet.append({'tweet_id': tweet['id'],
                                'retweets': tweet['retweet_count'],
                                'favorites': tweet['favorite_count']}, ignore_in
tweet_json.close()

df_tweet
```

Out[13]:

	tweet_id	retweets	favorites
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048
5	891087950875897856	3261	20562
6	890971913173991426	2158	12041
7	890729181411237888	16716	56848
8	890609185150312448	4429	28226
9	890240255349198849	7711	32467
10	890006608113172480	7624	31166
11	889880896479866881	5156	28268
12	889665388333682689	8538	38818
13	889638837579907072	4735	27672
14	889531135344209921	2321	15359
15	889278841981685760	5637	25652
16	888917238123831296	4709	29611
17	888804989199671297	4559	26080
18	888554962724278272	3732	20290
19	888078434458587136	3653	22201
20	887705289381826560	5609	30779
21	887517139158093824	12082	46959
22	887473957103951883	18781	69871
23	887343217045368832	10737	34222
24	887101392804085760	6167	31061
25	886983233522544640	8084	35859
26	886736880519319552	3443	12306
27	886680336477933568	4610	22798

	tweet_id	retweets	favorites
28	886366144734445568	3316	21524
29	886267009285017600	4	117
...	...	...	...
2324	666411507551481857	339	459
2325	666407126856765440	44	113
2326	666396247373291520	92	172
2327	666373753744588802	100	194
2328	666362758909284353	595	804
2329	666353288456101888	77	229
2330	666345417576210432	146	307
2331	666337882303524864	96	204
2332	666293911632134144	368	522
2333	666287406224695296	71	152
2334	666273097616637952	82	184
2335	666268910803644416	37	108
2336	666104133288665088	6871	14765
2337	666102155909144576	16	81
2338	666099513787052032	73	164
2339	666094000022159362	79	169
2340	666082916733198337	47	121
2341	666073100786774016	174	335
2342	666071193221509120	67	154
2343	666063827256086533	232	496
2344	666058600524156928	61	115
2345	666057090499244032	146	304
2346	666055525042405380	261	448
2347	666051853826850816	879	1253
2348	666050758794694657	60	136
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 [14]:
```
len(archives), len(images), len(df_tweet)
```

Out[14]:
```
(2356, 2075, 2354)
```

In [15]:
```
# Copy dataframes for cleaning to keep original
archives_clean = archives.copy()
image_clean = images.copy()
df_tweet_clean = df_tweet.copy()
```

Assess

In [16]:
```
archives_clean
```

Out[16]:

	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://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte
				2017-07-	

In [17]:
```
archives_clean.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 [18]:
```
archives_clean.describe()
```

Out[18]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweeted_statu
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	1.8
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	1.2
std	6.856705e+16	7.582492e+16	1.252797e+17	6.236928e+16	9.5
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	7.8
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	4.1
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	4.1
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	4.1
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	7.8

In [19]:
```
image_clean
```

Out[19]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_s
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Ger
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhode
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	mini
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	mi
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg	1	g
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAlOoZ.jpg	1	
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg	1	
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg	1	
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg	1	
15	666099513787052032	https://pbs.twimg.com/media/CT51-JJUEAA6hV8.jpg	1	
16	666102155909144576	https://pbs.twimg.com/media/CT54YGiWUAEZnoK.jpg	1	
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1	
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1	des
19	666273097616637952	https://pbs.twimg.com/media/CT8T1mtUwAA3aqm.jpg	1	Ital
20	666287406224695296	https://pbs.twimg.com/media/CT8g3BpUEAAuFjg.jpg	1	
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1	th
22	666337882303524864	https://pbs.twimg.com/media/CT9OwFIWEAMuRje.jpg	1	
23	666345417576210432	https://pbs.twimg.com/media/CT9Vn7PWoAA_ZCM.jpg	1	g
24	666353288456101888	https://pbs.twimg.com/media/CT9cx0tUEAAhNN_.jpg	1	
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1	
26	666373753744588802	https://pbs.twimg.com/media/CT9vZEYWUAAlZ05.jpg	1	coated_v
27	666396247373291520	https://pbs.twimg.com/media/CT-D2ZHWIAA3gK1.jpg	1	
28	666407126856765440	https://pbs.twimg.com/media/CT-NvwmW4AAugGZ.jpg	1	black-and-t
29	666411507551481857	https://pbs.twimg.com/media/CT-RugiWIAELEaq.jpg	1	
...	...	...	...	
2045	886366144734445568	https://pbs.twimg.com/media/DE0BTnQUwAApKEH.jpg	1	l
2046	886680336477933568	https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg	1	
2047	886736880519319552	https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg	1	

	tweet_id	jpg_url	img_num	
2048	886983233522544640	https://pbs.twimg.com/media/DE8yicJW0AAAvBJ.jpg	2	
2049	887101392804085760	https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg	1	
2050	887343217045368832	https://pbs.twimg.com/ext_tw_video_thumb/88734...	1	Me
2051	887473957103951883	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	
2052	887517139158093824	https://pbs.twimg.com/ext_tw_video_thumb/88751...	1	
2053	887705289381826560	https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg	1	
2054	888078434458587136	https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg	1	H
2055	888202515573088257	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	
2056	888554962724278272	https://pbs.twimg.com/media/DFTH_O-UQAACu20.jpg	3	S
2057	888804989199671297	https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg	1	g
2058	888917238123831296	https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg	1	g
2059	889278841981685760	https://pbs.twimg.com/ext_tw_video_thumb/88927...	1	
2060	889531135344209921	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	1	g
2061	889638837579907072	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	1	H
2062	889665388333682689	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	1	
2063	889880896479866881	https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg	1	H
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	Chesapeake
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	

2075 rows × 12 columns

In [20]:
```
image_clean.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 [21]:
```
image_clean.describe()
```

Out[21]:

	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 [22]:
```
image_clean.p1.value_counts()
image_clean.p2.value_counts()
image_clean.p3.value_counts()
```

Out[22]:
```
Labrador_retriever               79
Chihuahua                        58
golden_retriever                 48
Eskimo_dog                       38
kelpie                           35
kuvasz                           34
Staffordshire_bullterrier        32
chow                             32
beagle                           31
cocker_spaniel                   31
Pomeranian                       29
Pekinese                         29
toy_poodle                       29
Great_Pyrenees                   27
Chesapeake_Bay_retriever         27
Pembroke                         27
French_bulldog                   26
malamute                         26
American_Staffordshire_terrier   24
Cardigan                         23
pug                              23
basenji                          21
toy_terrier                      20
bull_mastiff                     20
Siberian_husky                   19
Shetland_sheepdog                17
Boston_bull                      17
Lakeland_terrier                 16
boxer                            16
doormat                          16
                                 ..
neck_brace                        1
cloak                             1
pickup                            1
barbell                           1
broccoli                          1
beach_wagon                       1
swimming_trunks                   1
grocery_store                     1
padlock                           1
rain_barrel                       1
grand_piano                       1
orangutan                         1
African_chameleon                 1
wing                              1
panpipe                           1
coffeepot                         1
sea_cucumber                      1
chickadee                         1
bib                               1
loupe                             1
eel                               1
rotisserie                        1
```

```
vacuum                          1
lampshade                       1
toyshop                         1
mountain_tent                   1
theater_curtain                 1
bannister                       1
bow_tie                         1
bullfrog                        1
Name: p3, Length: 408, dtype: int64
```

In [23]:
```
df_tweet_clean
```

Out[23]:

	tweet_id	retweets	favorites
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048
5	891087950875897856	3261	20562
6	890971913173991426	2158	12041
7	890729181411237888	16716	56848
8	890609185150312448	4429	28226
9	890240255349198849	7711	32467
10	890006608113172480	7624	31166
11	889880896479866881	5156	28268
12	889665388333682689	8538	38818
13	889638837579907072	4735	27672
14	889531135344209921	2321	15359
15	889278841981685760	5637	25652
16	888917238123831296	4709	29611
17	888804989199671297	4559	26080
18	888554962724278272	3732	20290
19	888078434458587136	3653	22201
20	887705289381826560	5609	30779
21	887517139158093824	12082	46959
22	887473957103951883	18781	69871
23	887343217045368832	10737	34222
24	887101392804085760	6167	31061
25	886983233522544640	8084	35859
26	886736880519319552	3443	12306
27	886680336477933568	4610	22798
28	886366144734445568	3316	21524
29	886267009285017600	4	117
...	...	...	...
2324	666411507551481857	339	459
2325	666407126856765440	44	113
2326	666396247373291520	92	172
2327	666373753744588802	100	194

	tweet_id	retweets	favorites
2328	666362758909284353	595	804
2329	666353288456101888	77	229
2330	666345417576210432	146	307
2331	666337882303524864	96	204
2332	666293911632134144	368	522
2333	666287406224695296	71	152
2334	666273097616637952	82	184
2335	666268910803644416	37	108
2336	666104133288665088	6871	14765
2337	666102155909144576	16	81
2338	666099513787052032	73	164
2339	666094000022159362	79	169
2340	666082916733198337	47	121
2341	666073100786774016	174	335
2342	666071193221509120	67	154
2343	666063827256086533	232	496
2344	666058600524156928	61	115
2345	666057090499244032	146	304
2346	666055525042405380	261	448
2347	666051853826850816	879	1253
2348	666050758794694657	60	136
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 [24]:
```
df_tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
tweet_id     2354 non-null object
retweets     2354 non-null object
favorites    2354 non-null object
dtypes: object(3)
memory usage: 55.2+ KB
```

In [25]:
```
df_tweet_clean.describe()
```

Out[25]:

	tweet_id	retweets	favorites
count	2354	2354	2354
unique	2354	1724	2007
top	667495797102141441	3652	0
freq	1	5	179

Quality

Issues with content: missing data, inaccurate data, invalid data, inconsistent data

Data quality requirements:

1. **Completeness:** do we have all of the records that we should? Do we have missing records or not? Are there specific rows, columns, or cells missing?
2. **Validity:** we have the records, but they're not valid, i.e., they don't conform to a defined schema. A schema is a defined set of rules for data. These rules can be real-world constraints (e.g. negative height is impossible) and table-specific constraints (e.g. unique key constraints in tables).
3. **Accuracy:** inaccurate data is wrong data that is valid. It adheres to the defined schema, but it is still incorrect. Example: a patient's weight that is 5 lbs too heavy because the scale was faulty.
4. **Consistency:** inconsistent data is both valid and accurate, but there are multiple correct ways of referring to the same thing. Consistency, i.e., a standard format, in columns that represent the same data across tables and/or within tables is desired.

ARCHIVES TABLE:

1. Remove the 181 retweets
2. Replace 'None' with 'NaN' for all dog stages
3. tweet_id is integer type, change to string
4. Exclude unnecessary columns for analysis

IMAGE TABLE:

1. Remove '_' between words for p1, p2, and p3
2. Remove entries where p1_dog, p2_dog, and p3_dog are all 'False'
3. Inconsistent capitialization for p1, p2, p3

DF TWEET TABLE:

1. All columns should be changed into an integer type

Tidiness

Tidy Data requirements:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Issues with structure: messy data

1. Combine doggo, floofer, pupper and puppo columns in *archives* dataframe into one: 'stage' column
2. Merge all three dataframes with tweet_id column

Cleaning

Quality Issues

Define

Cleaning #1:

Remove the 181 retweets in *archives* table

In [26]:
```
archives_clean
```

Out[26]:

	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://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte

Code

In [27]:
```
# Find the retweets
archives_clean[archives_clean['retweeted_status_id'].notnull() == True]
```

Out[27]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
19	888202515573088257	NaN	NaN	2017-07-21 01:02:36 +0000	href="http://twitte
32	886054160059072513	NaN	NaN	2017-07-15 02:45:48 +0000	href="http://twitte
36	885311592912609280	NaN	NaN	2017-07-13 01:35:06 +0000	href="http://twitte
68	879130579576475649	NaN	NaN	2017-06-26 00:13:58 +0000	href="http://twitte
				2017-06-	

In [28]:
```
# Remove these retweet values from the dataframe using the drop() function
archives_clean.drop(archives_clean[archives_clean['retweeted_status_id'].notnull
```

Test

In [29]:
```
archives_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2175 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2175 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2175 non-null object
source                        2175 non-null object
text                          2175 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                 2117 non-null object
rating_numerator              2175 non-null int64
rating_denominator            2175 non-null int64
name                          2175 non-null object
doggo                         2175 non-null object
floofer                       2175 non-null object
pupper                        2175 non-null object
puppo                         2175 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 305.9+ KB
```

Define

Cleaning #2:

Replace 'None' with 'NaN' for all dog stages: doggo, floofer, pupper, and puppo in *archives* dataframe

In [30]:
```
archives_clean
```

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 +0000	href="http://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte

Code

In [31]:
```
archives_clean['doggo'] = np.where(archives_clean['doggo'] == 'None' , np.nan, ar
archives_clean['floofer'] = np.where(archives_clean['floofer'] == 'None' , np.nan
archives_clean['pupper'] = np.where(archives_clean['pupper'] == 'None' , np.nan,
archives_clean['puppo'] = np.where(archives_clean['puppo'] == 'None' , np.nan, ar
```

Test

In [32]:
```
archives_clean
```

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 +0000	href="http://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte
				2017-07-	

Define

Cleaning #3:

Change tweet_id from integer to string type in *archives* dataframe

In [33]:
```
archives_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2175 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                    2175 non-null int64
in_reply_to_status_id       78 non-null float64
in_reply_to_user_id         78 non-null float64
timestamp                   2175 non-null object
source                      2175 non-null object
text                        2175 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               2117 non-null object
rating_numerator            2175 non-null int64
rating_denominator          2175 non-null int64
name                        2175 non-null object
doggo                       87 non-null object
floofer                     10 non-null object
pupper                      234 non-null object
puppo                       25 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 305.9+ KB
```

Code

In [34]:
```
archives_clean.tweet_id = archives_clean.tweet_id.astype(str)
```

Test

In [35]:
```
type(archives_clean['tweet_id'].iloc[0])
```

Out[35]:
```
str
```

Define

Cleaning #4:

Exclude unnecessary columns for analysis in *archives* dataframe

In [36]:
```
archives_clean
```

Out[36]:

	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://twitte
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitte
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitte
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitte
				2017-07-	

Code

In [37]:
```
archives_clean.drop(columns=['retweeted_status_id','retweeted_status_user_id', '
```

Test

In [38]:
```
archives_clean
```

Out[38]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	This is Phine mystical boy. C
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	This is Tilly. checking pup
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	This is Archie. H Norwegian
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	This is D commenced a sr
				2017-07-	

In [39]:
```
archives_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2175 entries, 0 to 2355
Data columns (total 12 columns):
tweet_id                 2175 non-null object
in_reply_to_status_id    78 non-null float64
in_reply_to_user_id      78 non-null float64
timestamp                2175 non-null object
text                     2175 non-null object
rating_numerator         2175 non-null int64
rating_denominator       2175 non-null int64
name                     2175 non-null object
doggo                    87 non-null object
floofer                  10 non-null object
pupper                   234 non-null object
puppo                    25 non-null object
dtypes: float64(2), int64(2), object(8)
memory usage: 220.9+ KB
```

Define

Cleaning #5:

Remove '_' between words for p1, p2, and p3 in *image* dataframe

In [40]:
```
image_clean
```

Out[40]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_s
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Ger
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhode
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	mini
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	mi
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg	1	g
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAlOoZ.jpg	1	
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg	1	
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg	1	
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg	1	
15	666099513787052032	https://pbs.twimg.com/media/CT51-JJUEAA6hV8.jpg	1	
16	666102155909144576	https://pbs.twimg.com/media/CT54YGiWUAEZnoK.jpg	1	
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1	
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1	des
19	666273097616637952	https://pbs.twimg.com/media/CT8T1mtUwAA3aqm.jpg	1	Ital
20	666287406224695296	https://pbs.twimg.com/media/CT8g3BpUEAAuFjg.jpg	1	
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1	th
22	666337882303524864	https://pbs.twimg.com/media/CT9OwFIWEAMuRje.jpg	1	
23	666345417576210432	https://pbs.twimg.com/media/CT9Vn7PWoAA_ZCM.jpg	1	g
24	666353288456101888	https://pbs.twimg.com/media/CT9cx0tUEAAhNN_.jpg	1	
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1	
26	666373753744588802	https://pbs.twimg.com/media/CT9vZEYWUAAlZ05.jpg	1	coated_v
27	666396247373291520	https://pbs.twimg.com/media/CT-D2ZHWIAA3gK1.jpg	1	
28	666407126856765440	https://pbs.twimg.com/media/CT-NvwmW4AAugGZ.jpg	1	black-and-t
29	666411507551481857	https://pbs.twimg.com/media/CT-RugiWIAELEaq.jpg	1	
...	...	...	...	
2045	886366144734445568	https://pbs.twimg.com/media/DE0BTnQUwAApKEH.jpg	1	l
2046	886680336477933568	https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg	1	
2047	886736880519319552	https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg	1	

	tweet_id	jpg_url	img_num	
2048	886983233522544640	https://pbs.twimg.com/media/DE8yicJW0AAAvBJ.jpg	2	
2049	887101392804085760	https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg	1	
2050	887343217045368832	https://pbs.twimg.com/ext_tw_video_thumb/88734...	1	Me
2051	887473957103951883	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	
2052	887517139158093824	https://pbs.twimg.com/ext_tw_video_thumb/88751...	1	
2053	887705289381826560	https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg	1	
2054	888078434458587136	https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg	1	l
2055	888202515573088257	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	
2056	888554962724278272	https://pbs.twimg.com/media/DFTH_O-UQAACu20.jpg	3	S
2057	888804989199671297	https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg	1	g
2058	888917238123831296	https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg	1	g
2059	889278841981685760	https://pbs.twimg.com/ext_tw_video_thumb/88927...	1	
2060	889531135344209921	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	1	g
2061	889638837579907072	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	1	l
2062	889665388333682689	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	1	
2063	889880896479866881	https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg	1	l
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	Chesapeake
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	

2075 rows × 12 columns

Code

In [41]:
```
image_clean['p1'] = image_clean['p1'].str.replace('_', ' ')
image_clean['p2'] = image_clean['p2'].str.replace('_', ' ')
image_clean['p3'] = image_clean['p3'].str.replace('_', ' ')
```

Test

In [42]:
```
image_clean.head()
```

Out[42]:

	tweet_id	jpg_url	img_num	p1	p1_
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh springer spaniel	0.46
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.50
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German shepherd	0.59
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian ridgeback	0.40
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature pinscher	0.56

Define

Cleaning #6:

Remove entries where p1_dog, p2_dog, and p3_dog are all 'False' in *image* dataframe

Code

In [45]:
```
image_clean_df = image_clean.query('p1_dog == True or p2_dog == True or p3_dog ==
```

Test

In [46]:
```
image_clean_df= image_clean.query('p1_dog == False & p2_dog == False & p3_dog ==
image_clean_df
```

Out[46]:

tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog

Define

Cleaning #7:

Inconsistent capitialization for p1, p2, p3 in *images* dataframe

In [47]:
```
image_clean
```

Out[47]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	We sprin spa
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redb
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Gerr sheph
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodes ridgeb
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniat pinsc
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bern moun

Code

In [48]:
```
image_clean['p1'] = image_clean['p1'].str.title()
image_clean['p2'] = image_clean['p2'].str.title()
image_clean['p3'] = image_clean['p3'].str.title()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

Test

In [49]:
```
image_clean.head()
```

Out[49]:

	tweet_id	jpg_url	img_num	p1	p1_
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh Springer Spaniel	0.46
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redbone	0.50
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German Shepherd	0.59
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian Ridgeback	0.40
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniature Pinscher	0.56

Define

Cleaning #8:

All columns should be changed into an integer type in *df_tweet* dataframe

Code

In [50]:
```
df_tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
tweet_id     2354 non-null object
retweets     2354 non-null object
favorites    2354 non-null object
dtypes: object(3)
memory usage: 55.2+ KB
```

In [51]:
```
df_tweet_clean = df_tweet_clean.astype(str).astype(int)
```

Test

In [52]:
```
df_tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
tweet_id     2354 non-null int64
retweets     2354 non-null int64
favorites    2354 non-null int64
dtypes: int64(3)
memory usage: 55.2 KB
```

Tidiness Issues

Define

Tidiness #1:

Combine doggo, floofer, pupper and puppo columns in *archives* dataframe into one: 'stage' column

Code

In [53]:
```
archives_clean['stage'] = archives[['doggo', 'floofer','pupper','puppo']].apply(

archives_clean['stage'].replace("NoneNoneNoneNone","None ", inplace=True)
archives_clean['stage'].replace("doggoNoneNoneNone","doggo", inplace=True)
archives_clean['stage'].replace("NoneflooferNoneNone","floofer", inplace=True)
archives_clean['stage'].replace("NoneNonepupperNone","pupper", inplace=True)
archives_clean['stage'].replace("NoneNoneNonepuppo","puppo", inplace=True)
```

Test

In [54]:
```
archives_clean
```

Out[54]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	This is Phine mystical boy. C
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	This is Tilly. checking pup
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	This is Archie. H Norwegian
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	This is D commenced a sr
				2017-07-	

In [55]:
```
archives_clean['stage']
```

Out[55]:
```
0        None
1        None
2        None
3        None
4        None
5        None
6        None
7        None
8        None
9       doggo
10       None
11       None
12      puppo
13       None
14      puppo
15       None
16       None
17       None
18       None
20       None
21       None
22       None
23       None
24       None
25       None
26       None
27       None
28       None
29     pupper
30       None
        ...
2326     None
2327     None
2328     None
2329     None
2330     None
2331     None
2332     None
2333     None
2334     None
2335     None
2336     None
2337     None
2338     None
2339     None
2340     None
2341     None
2342     None
2343     None
2344     None
2345     None
2346     None
2347     None
2348     None
2349     None
```

```
2350     None
2351     None
2352     None
2353     None
2354     None
2355     None
Name: stage, Length: 2175, dtype: object
```

Define

Tidiness #2:

Merge all three dataframes with tweet_id column

Code

In [56]:
```
archives_clean['tweet_id'] = archives_clean['tweet_id'].astype(str).astype(int)
image_clean['tweet_id'] = image_clean['tweet_id'].astype(str).astype(int)
df_tweet_clean['tweet_id'] = df_tweet_clean['tweet_id'].astype(str).astype(int)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [57]:
```
df_merge_temp = pd.merge(image_clean, df_tweet_clean,on='tweet_id', how = 'inner
df_merge_temp
```

Out[57]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	We Sprin Spa
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redb
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Gerr Sheph
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodes Ridgeb
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniat Pinsc
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bern Moun [

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1750 entries, 0 to 1749
Data columns (total 14 columns):
tweet_id     1750 non-null int64
jpg_url      1750 non-null object
img_num      1750 non-null int64
p1           1750 non-null object
p1_conf      1750 non-null float64
p1_dog       1750 non-null bool
p2           1750 non-null object
p2_conf      1750 non-null float64
p2_dog       1750 non-null bool
p3           1750 non-null object
p3_conf      1750 non-null float64
p3_dog       1750 non-null bool
retweets     1750 non-null int64
favorites    1750 non-null int64
dtypes: bool(3), float64(3), int64(4), object(4)
memory usage: 169.2+ KB
```

In [59]:
```
df_merge_master = pd.merge(df_merge_temp, archives_clean, on='tweet_id', how = '
df_merge_master
```

Out[59]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	We Sprin Spa
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redb
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Gerr Sheph
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodes Ridgeb
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniat Pinsc

Test

In [60]:
```
df_merge_master
```

Out[60]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	We Sprin Spa
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redb
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Gerr Sheph
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodes Ridgeb
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniat Pinsc

Storing, Analyzing, and Visualizing Data for this Project

In [61]:
```
# Store to a master file
df_merge_master.to_csv('twitter_archive_master.csv', index=False)
```

Analyze and Visualize Wrangled Data

In [62]:
```
# Read wrangled and updated master datafram
df_master= pd.read_csv('twitter_archive_master.csv')
```

In [63]:
```
# Look at dataframe detailed information
df_master.head()
```

Out[63]:

	tweet_id	jpg_url	img_num	p1	p1_
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh Springer Spaniel	0.46
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redbone	0.50
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German Shepherd	0.59
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian Ridgeback	0.40
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniature Pinscher	0.56

5 rows × 26 columns

In [64]:
```
# Find information about new dataframe
df_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1686 entries, 0 to 1685
Data columns (total 26 columns):
tweet_id                 1686 non-null int64
jpg_url                  1686 non-null object
img_num                  1686 non-null int64
p1                       1686 non-null object
p1_conf                  1686 non-null float64
p1_dog                   1686 non-null bool
p2                       1686 non-null object
p2_conf                  1686 non-null float64
p2_dog                   1686 non-null bool
p3                       1686 non-null object
p3_conf                  1686 non-null float64
p3_dog                   1686 non-null bool
retweets                 1686 non-null int64
favorites                1686 non-null int64
in_reply_to_status_id    20 non-null float64
in_reply_to_user_id      20 non-null float64
timestamp                1686 non-null object
text                     1686 non-null object
rating_numerator         1686 non-null int64
rating_denominator       1686 non-null int64
name                     1686 non-null object
doggo                    64 non-null object
floofer                  8 non-null object
pupper                   176 non-null object
puppo                    22 non-null object
stage                    1686 non-null object
dtypes: bool(3), float64(5), int64(6), object(12)
memory usage: 308.0+ KB
```

In [65]:
```
# Find information about new dataframe
df_master.describe()
```

Out[65]:

	tweet_id	img_num	p1_conf	p2_conf	p3_conf	retweets	fav
count	1.686000e+03	1686.000000	1686.000000	1686.000000	1.686000e+03	1686.000000	1686.0
mean	7.392710e+17	1.215896	0.605150	0.137147	6.135847e-02	2817.118031	9233.1
std	6.800607e+16	0.576191	0.266416	0.101252	5.188051e-02	4807.199770	12583.2
min	6.660209e+17	1.000000	0.044333	0.000010	2.160900e-07	16.000000	81.0
25%	6.773835e+17	1.000000	0.378746	0.054938	1.598680e-02	649.250000	2163.2
50%	7.132943e+17	1.000000	0.606701	0.121432	4.995100e-02	1437.500000	4440.0
75%	7.931770e+17	1.000000	0.851578	0.198429	9.468462e-02	3247.750000	11656.7
max	8.921774e+17	4.000000	0.999984	0.467678	2.734190e-01	79515.000000	132810.0

Insights

1. Average Rating Overall

In [66]:
```
# Calculate the overall average rating with the means for numerator and denominat
(df_master['rating_numerator'].mean())/(df_master['rating_denominator'].mean())
```

Out[66]: 1.0891841675581331

2. Confidence Percentages that p1, p2, p3 are in fact dogs

In [67]:
```
# Find means of confidences for p1, p2, p3
df_master.p1_conf.mean(), df_master.p2_conf.mean(), df_master.p3_conf.mean()
```

Out[67]: (0.60515001992882556, 0.13714714857544485, 0.061358469266672606)

In [69]:
```
# Find the mean of the means
(0.60515001992882556 + 0.13714714857544485 + 0.061358469266672606)/3
```

Out[69]: 0.26788521259031434

3. Most favorited Dog

In [74]:
```
# From describe function, find more information about the most favorited dog unde
df_master[df_master['favorites']== 132810]
```

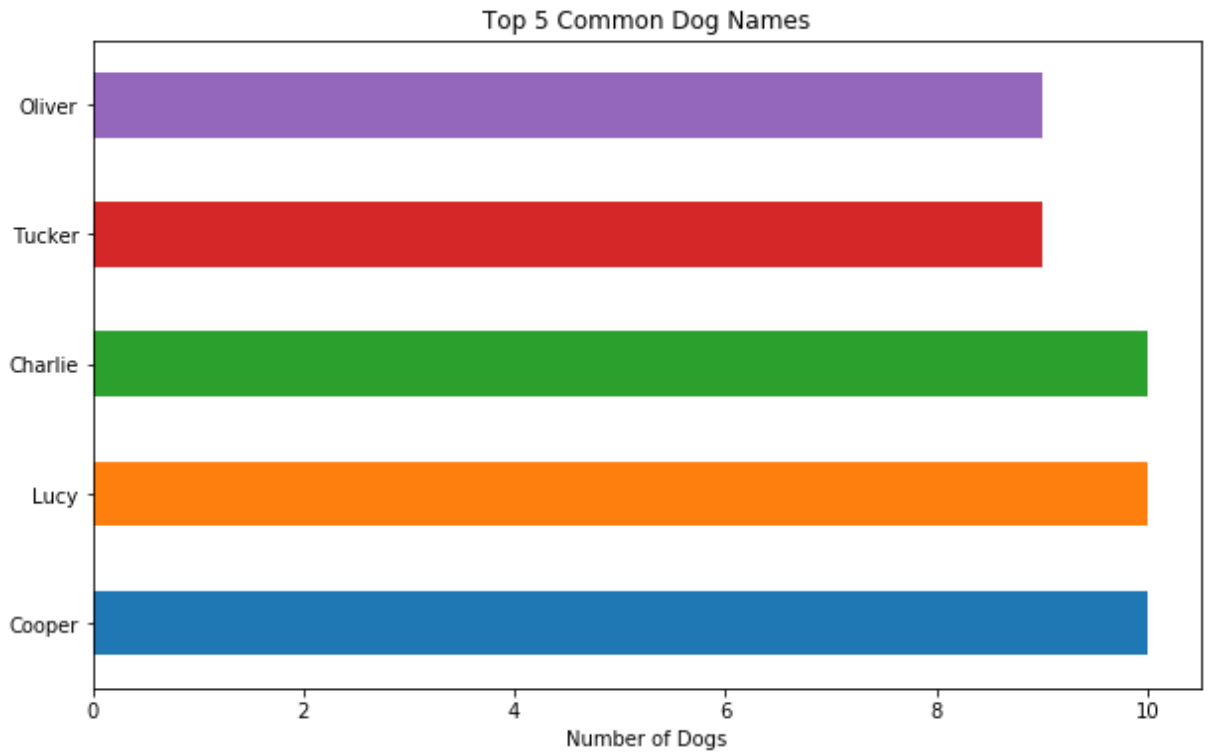
Out[74]:

	tweet_id	jpg_url	img_num	p1	p1_
1407	822872901745569793	https://pbs.twimg.com/media/C2tugXLXgAArJO4.jpg	1	Lakeland Terrier	0.19

1 rows × 26 columns

Visualization

In [72]:
```
df_master.name.value_counts()[2:7].plot('barh', figsize=(10,6), title='Top 5 Comm
plt.savefig('common_dog_names')
```



According to the bar plot above, the top five common dog names are Cooper, Lucy, Charlie, Tucker, and Oliver.

References:

1. https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/ (https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/)
2. https://stackoverflow.com/questions/47612822/how-to-create-pandas-dataframe-from-twitter-search-api (https://stackoverflow.com/questions/47612822/how-to-create-pandas-dataframe-from-twitter-search-api)
3. https://www.guru99.com/python-json.html (https://www.guru99.com/python-json.html)
4. http://pandas.pydata.org/pandas-docs/version/0.16/merging.html (http://pandas.pydata.org/pandas-docs/version/0.16/merging.html)