WeRateDogs Twitter Data Wrangling

Table of Contents

Gathering Data

In [1]:

```
#import packages
import requests
import pandas as pd
import numpy as np
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer
import re
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#download file image-predictions.tsv from url
url='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
r=requests.get(url)
open('image-predictions.tsv', 'wb').write(r.content)
```

Out[2]:

```
335079
```

In [3]:

```
#read twitter-archive-enhanced.csv file
df_enhanced=pd.read_csv('twitter-archive-enhanced.csv')

#read image-predictions.tsv file
df_image=pd.read_csv('image-predictions.tsv',sep='\t')
```

- Below Twitter API code is copied from given file twitter-api.py due to no access right on twitter developer account.

In [4]:

```
# Query Twitter API for each tweet in the Twitter archive and save JSON in a text file
# These are hidden to comply with Twitter's API terms and conditions
consumer_key = 'HIDDEN'
consumer_secret = 'HIDDEN'
access_token = 'HIDDEN'
access_secret = 'HIDDEN'

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

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

# NOTE TO STUDENT WITH MOBILE VERIFICATION ISSUES:
# df_1 is a DataFrame with the twitter_archive_enhanced.csv file. You may have to
# change line 17 to match the name of your DataFrame with twitter_archive_enhanced.csv
# 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 = df_enhanced.tweet_id.values
len(tweet_ids)

# 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 limit
    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
11: 890006608113172480
Fail
12: 889880896479866881
Fail
13: 889665388333682689
Fail
14: 889638837579907072
Fail
15: 889531135344209921
Fail
16: 889278841981685760
Fail
17: 888917238123831296
Fail
18: 888804989199671297
Fail
19: 888554962724278272
Fail
20: 888202515573088257
Fail
21: 888078434458587136
Fail
22: 887705289381826560
Fail
23: 887517139158093824
Fail
24: 887473957103951883
Fail
25: 887343217045368832
Fail
26: 887101392804085760
Fail
27: 886983233522544640
Fail
28: 886736880519319552
Fail
29: 886680336477933568
Fail
30: 886366144734445568
Fail
31: 886267009285017600
```

```
Fail
32: 886258384151887873
Fail
33: 886054160059072513
Fail
34: 885984800019947520
Fail
35: 885528943205470208
Fail
36: 885518971528720385
Fail
37: 885311592912609280
Fail
38: 885167619883638784
Fail
39: 884925521741709313
```

```
---------------------------------------------------------------------------
-
KeyboardInterrupt                         Traceback (most recent call las
t)
<ipython-input-4-7d81073f978e> in <module>
     31         print(str(count) + ": " + str(tweet_id))
     32         try:
---> 33             tweet = api.get_status(tweet_id, tweet_mode='extended'
)
     34             print("Success")
     35             json.dump(tweet._json, outfile)

~\anaconda3\lib\site-packages\tweepy\binder.py in _call(*args, **kwargs)
    248                 return method
    249             else:
--> 250                 return method.execute()
    251         finally:
    252             method.session.close()

~\anaconda3\lib\site-packages\tweepy\binder.py in execute(self)
    187                                                 timeout=self.api.t
imeout,
    188                                                 auth=auth,
--> 189                                                 proxies=self.api.p
roxy)
    190                 except Exception as e:
    191                     six.reraise(TweepError, TweepError('Failed to
 send request: %s' % e), sys.exc_info()[2])

~\anaconda3\lib\site-packages\requests\sessions.py in request(self, metho
d, url, params, data, headers, cookies, files, auth, timeout, allow_redire
cts, proxies, hooks, stream, verify, cert, json)
    531         }
    532         send_kwargs.update(settings)
--> 533         resp = self.send(prep, **send_kwargs)
    534
    535         return resp

~\anaconda3\lib\site-packages\requests\sessions.py in send(self, request,
 **kwargs)
    644
    645         # Send the request
--> 646         r = adapter.send(request, **kwargs)
    647
    648         # Total elapsed time of the request (approximately)

~\anaconda3\lib\site-packages\requests\adapters.py in send(self, request,
 stream, timeout, verify, cert, proxies)
    447                     decode_content=False,
    448                     retries=self.max_retries,
--> 449                     timeout=timeout
    450                 )
    451

~\anaconda3\lib\site-packages\urllib3\connectionpool.py in urlopen(self, m
ethod, url, body, headers, retries, redirect, assert_same_host, timeout, p
ool_timeout, release_conn, chunked, body_pos, **response_kw)
    670                 body=body,
    671                 headers=headers,
--> 672                 chunked=chunked,
    673             )
```

```
    674

~\anaconda3\lib\site-packages\urllib3\connectionpool.py in _make_request(s
elf, conn, method, url, timeout, chunked, **httplib_request_kw)
    419                     # Python 3 (including for exceptions like Syst
emExit).
    420                     # Otherwise it looks like a bug in the code.
--> 421                     six.raise_from(e, None)
    422         except (SocketTimeout, BaseSSLError, SocketError) as e:
    423             self._raise_timeout(err=e, url=url, timeout_value=read
_timeout)

~\anaconda3\lib\site-packages\urllib3\packages\six.py in raise_from(value,
from_value)

~\anaconda3\lib\site-packages\urllib3\connectionpool.py in _make_request(s
elf, conn, method, url, timeout, chunked, **httplib_request_kw)
    414                 # Python 3
    415                 try:
--> 416                     httplib_response = conn.getresponse()
    417                 except BaseException as e:
    418                     # Remove the TypeError from the exception chai
n in

~\anaconda3\lib\http\client.py in getresponse(self)
   1342         try:
   1343             try:
-> 1344                 response.begin()
   1345             except ConnectionError:
   1346                 self.close()

~\anaconda3\lib\http\client.py in begin(self)
    304         # read until we get a non-100 response
    305         while True:
--> 306             version, status, reason = self._read_status()
    307             if status != CONTINUE:
    308                 break

~\anaconda3\lib\http\client.py in _read_status(self)
    265
    266     def _read_status(self):
--> 267         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    268         if len(line) > _MAXLINE:
    269             raise LineTooLong("status line")

~\anaconda3\lib\socket.py in readinto(self, b)
    587         while True:
    588             try:
--> 589                 return self._sock.recv_into(b)
    590             except timeout:
    591                 self._timeout_occurred = True

~\anaconda3\lib\site-packages\urllib3\contrib\pyopenssl.py in recv_into(se
lf, *args, **kwargs)
    323                     raise
    324         except OpenSSL.SSL.WantReadError:
--> 325             if not util.wait_for_read(self.socket, self.socket.get
timeout()):
    326                 raise timeout("The read operation timed out")
    327             else:
```

```
~\anaconda3\lib\site-packages\urllib3\util\wait.py in wait_for_read(sock,
 timeout)
    144     Returns True if the socket is readable, or False if the timeou
t expired.
    145     """
--> 146     return wait_for_socket(sock, read=True, timeout=timeout)
    147
    148

~\anaconda3\lib\site-packages\urllib3\util\wait.py in select_wait_for_sock
et(sock, read, write, timeout)
     84     # thing.)
     85     fn = partial(select.select, rcheck, wcheck, wcheck)
---> 86     rready, wready, xready = _retry_on_intr(fn, timeout)
     87     return bool(rready or wready or xready)
     88

~\anaconda3\lib\site-packages\urllib3\util\wait.py in _retry_on_intr(fn, t
imeout)
     41     # Modern Python, that retries syscalls by default
     42     def _retry_on_intr(fn, timeout):
---> 43         return fn(timeout)
     44
     45

KeyboardInterrupt:
```

- tweet-json.txt: given file with the resulting data from twitter-api.py.

In [5]:

```
#read given file tweet-json.txt line by line into a pandas DataFrame with tweet ID, ret
weet count, and favorite count.
with open('tweet-json.txt') as file:
    json_list = []
    for line in file:
        json_list.append(json.loads(line))

api_list = []
for tweet in json_list:
    tweet_id = tweet['id']
    retweet_count = tweet['retweet_count']
    favorite_count = tweet['favorite_count']
    api_list.append({'tweet_id': tweet_id,
                     'retweet_count': retweet_count,
                     'favorite_count': favorite_count})
df_api = pd.DataFrame(api_list, columns = ['tweet_id', 'retweet_count', 'favorite_coun
t'])
df_api = df_api.sort_values('tweet_id').reset_index(drop=True)
```

Assessing Data

In [6]:

```
#view records of table df_enhanced
pd.set_option('max_colwidth',None)
df_enhanced
```

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://t rel="nofo
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://t rel="nofo
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://t rel="nofo
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://t rel="nofo
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://t rel="nofo
...	...	...	...	...	
2351	666049248165822465	NaN	NaN	2015-11-16 00:24:50 +0000	href="http://t rel="nofo
2352	666044226329800704	NaN	NaN	2015-11-16 00:04:52 +0000	href="http://t rel="nofo
2353	666033412701032449	NaN	NaN	2015-11-15 23:21:54 +0000	href="http://t rel="nofo
2354	666029285002620928	NaN	NaN	2015-11-15 23:05:30 +0000	href="http://t rel="nofo

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
2355	666020888022790149	NaN	NaN	2015-11-15 22:32:08 +0000	href="http://t rel="nofo

2356 rows × 17 columns

In [7]:

```
df_enhanced.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: 313.0+ KB
```

In [8]:

```
#records with reply or retweet
df_enhanced[(df_enhanced.retweeted_status_id.notnull()) | (df_enhanced.in_reply_to_stat
us_id.notnull())]
```

Out[8]:

	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://t rel="nofo
30	886267009285017600	8.862664e+17	2.281182e+09	2017-07-15 16:51:35 +0000	href="http://t rel="nofo
32	886054160059072513	NaN	NaN	2017-07-15 02:45:48 +0000	href="http://t rel="nofo
36	885311592912609280	NaN	NaN	2017-07-13 01:35:06 +0000	href="http://t rel="nofo
55	881633300179243008	8.816070e+17	4.738443e+07	2017-07-02 21:58:53 +0000	href="http://t rel="nofo
...	...	...	...	...	
2169	669353438988365824	6.678065e+17	4.196984e+09	2015-11-25 03:14:30 +0000	href="http://t rel="nofo
2189	668967877119254528	6.689207e+17	2.143566e+07	2015-11-24 01:42:25 +0000	href="http://t rel="nofo
2259	667550904950915073	NaN	NaN	2015-11-20 03:51:52 +0000	rel="nofol
2260	667550882905632768	NaN	NaN	2015-11-20 03:51:47 +0000	rel="nofol
2298	667070482143944705	6.670655e+17	4.196984e+09	2015-11-18 20:02:51 +0000	href="http://t rel="nofo

259 rows × 17 columns

In [9]:

```
df_enhanced.describe()
```

Out[9]:

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

In [10]:

```
#check the records for low rating_denominator
df_enhanced[df_enhanced.rating_denominator<10][['text','rating_numerator','rating_denom
inator']]
```

Out[10]:

	text	rating_numerator	rating_denominator
313	@jonnysun @Lin_Manuel ok jomny I know you're excited but 960/00 isn't a valid rating, 13/10 is tho	960	0
516	Meet Sam. She smiles 24/7 & secretly aspires to be a reindeer. \nKeep Sam smiling by clicking and sharing this link:\nhttps://t.co/98tB8y7y7t https://t.co/LouL5vdvxx	24	7
2335	This is an Albanian 3 1/2 legged Episcopalian. Loves well-polished hardwood flooring. Penis on the collar. 9/10 https://t.co/d9NcXFKwLv	1	2

In [11]:

```
#some records with wrong rating_numerator
df_enhanced[df_enhanced.rating_numerator==75][['text','rating_numerator']]
```

Out[11]:

	text	rating_numerator
340	RT @dog_rates: This is Logan, the Chow who lived. He solemnly swears he's up to lots of good. H*ckin magical af 9.75/10 https://t.co/yBO5wu…	75
695	This is Logan, the Chow who lived. He solemnly swears he's up to lots of good. H*ckin magical af 9.75/10 https://t.co/yBO5wuqaPS	75

In [12]:

```
#missing value and invalid name in name column
df_enhanced.name.value_counts()
```

Out[12]:

```
None          745
a              55
Charlie        12
Oliver         11
Cooper         11
             ...
officially      1
Marvin          1
Pherb           1
Spark           1
Coleman         1
Name: name, Length: 957, dtype: int64
```

In [13]:

```
#name with lower case
df_enhanced[df_enhanced.name.str.islower()].name.value_counts()
```

Out[13]:

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

In [14]:

```
#None value instead of NaN, and only one category besides None
df_enhanced.doggo.value_counts()
```

Out[14]:

```
None     2259
doggo      97
Name: doggo, dtype: int64
```

In [15]:

```
#None value instead of NaN, and only one category besides None
df_enhanced.floofer.value_counts()
```

Out[15]:

```
None       2346
floofer      10
Name: floofer, dtype: int64
```

In [16]:

```
#None value instead of NaN, and only one category besides None
df_enhanced.pupper.value_counts()
```

Out[16]:

```
None      2099
pupper     257
Name: pupper, dtype: int64
```

In [17]:

```
#None value instead of NaN, and only one category besides None
df_enhanced.puppo.value_counts()
```

Out[17]:

```
None     2326
puppo      30
Name: puppo, dtype: int64
```

In [18]:

```
#multiple dog stages in records
df_enhanced[(df_enhanced['doggo']=='doggo') & ((df_enhanced['floofer']!='None') | (df_e
nhanced['pupper']!='None') | (df_enhanced['puppo']!='None'))]
```

Out[18]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
191	855851453814013952	NaN	NaN	2017-04-22 18:31:02 +0000	href="http://t rel="nofo
200	854010172552949760	NaN	NaN	2017-04-17 16:34:26 +0000	href="http://t rel="nofo
460	817777686764523521	NaN	NaN	2017-01-07 16:59:28 +0000	href="http://t rel="nofo
531	808106460588765185	NaN	NaN	2016-12-12 00:29:28 +0000	href="http://t rel="nofo
565	802265048156610565	7.331095e+17	4.196984e+09	2016-11-25 21:37:47 +0000	href="http://t rel="nofo
575	801115127852503040	NaN	NaN	2016-11-22 17:28:25 +0000	href="http://t rel="nofo
705	785639753186217984	NaN	NaN	2016-10-11 00:34:48 +0000	href="http://t rel="nofo
733	781308096455073793	NaN	NaN	2016-09-29 01:42:20 +0000	rel="nofollow
778	775898661951791106	NaN	NaN	2016-09-14 03:27:11 +0000	href="http://t rel="nofo
822	770093767776997377	NaN	NaN	2016-08-29 03:00:36 +0000	href="http://t rel="nofo

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
889	759793422261743616	NaN	NaN	2016-07-31 16:50:42 +0000	href="http://t rel="nofo
956	751583847268179968	NaN	NaN	2016-07-09 01:08:47 +0000	href="http://t rel="nofo
1063	741067306818797568	NaN	NaN	2016-06-10 00:39:48 +0000	href="http://t rel="nofo
1113	733109485275860992	NaN	NaN	2016-05-19 01:38:16 +0000	href="http://t rel="nofo

In [19]:

```
#view records of table df_image
df_image
```

Out[19]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	C
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhc
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	n
...	...	...	...	
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]:

```
df_image.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: 152.1+ KB
```

In [21]:

```
df_image.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]:

```
df_image.p1.value_counts()
```

Out[22]:

```
golden_retriever       150
Labrador_retriever     100
Pembroke                89
Chihuahua               83
pug                     57
                      ...
lion                     1
soccer_ball              1
clumber                  1
black-footed_ferret      1
American_black_bear      1
Name: p1, Length: 378, dtype: int64
```

In [23]:

```
df_image.p2.value_counts()
```

Out[23]:

```
Labrador_retriever    104
golden_retriever       92
Cardigan               73
Chihuahua              44
Pomeranian             42
                     ...
sliding_door            1
Kerry_blue_terrier      1
tarantula               1
streetcar               1
torch                   1
Name: p2, Length: 405, dtype: int64
```

In [24]:

```
df_image.p3.value_counts()
```

Out[24]:

```
Labrador_retriever    79
Chihuahua             58
golden_retriever      48
Eskimo_dog            38
kelpie                35
                      ..
go-kart                1
Kerry_blue_terrier     1
whiptail               1
grey_fox               1
jeep                   1
Name: p3, Length: 408, dtype: int64
```

In [25]:

```
#view records of table df_image
df_api
```

Out[25]:

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

2354 rows × 3 columns

In [26]:

```
df_api.info()
```

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

In [27]:

```
df_api.describe()
```

Out[27]:

	tweet_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

Quality

df_enhanced table

- Erroneous datatypes (timestamp columns)
- Retweets and replies included (redundant records)
- Drop unuseful columns: in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp
- Incorrect rating_denominator
  - tweet_id:666287406224695296(2->10)
- Incorrect rating_numerator
  - tweet_id:666287406224695296(1->9)
  - Float number was recognized as integer
- There are 745 None values under name column, and the names with lowercase are invalid names.
- doggo, floofer, pupper, puppo columns contain 'None' value which is not counted as null
- There are records with more than one stages (doggo with one of the floofer, pupper, puppo columns)
  - tweet_id:855851453814013952
  - 854010172552949760
  - 817777686764523521
  - 808106460588765185
  - 802265048156610565
  - 801115127852503040
  - 785639753186217984
  - 781308096455073793
  - 759793422261743616
  - 751583847268179968
  - 741067306818797568
  - 733109485275860992
  - 775898661951791106
  - 770093767776997377
- Calculate rating with values of rating_numerator divided by rating_deniminator

df_image table

- Sometimes lowercase and sometimes uppercase for breed names in p1, p2, p3 columns

Tidiness

- doggo, floofer, pupper, puppo columns should be combined in one column with category data type in the `df_enhanced` table
- Merge datasets to one

Cleaning Data

In [28]:

```
#make copy of tables for cleaning
df_enhanced_clean=df_enhanced.copy()
df_image_clean=df_image.copy()
df_api_clean=df_api.copy()
```

Quality

- **Erroneous datatypes (timestamp column)**

Define

- Change datatype to datetime for timestamp column

Code

In [29]:

```
#change data type to datetime
df_enhanced_clean.timestamp = pd.to_datetime(df_enhanced_clean.timestamp)
```

Test

In [30]:

```
#confirm data type changed to datetime
df_enhanced_clean.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   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    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: datetime64[ns, UTC](1), float64(4), int64(3), object(9)
memory usage: 313.0+ KB
```

- **Retweets and replies included (redundant records)**

Define

- Remove the records which columns "in_reply_to_status_id" or "retweeted_status_id" is not null

Code

In [31]:

```
#filter out records which columns "in_reply_to_status_id" or "retweeted_status_id" is n
ot null
df_enhanced_clean = df_enhanced_clean[~((df_enhanced_clean['in_reply_to_status_id'].not
null()) | (df_enhanced_clean['retweeted_status_id'].notnull()))]
```

Test

In [32]:

```
#confirm records removed
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2097 non-null   int64
 1   in_reply_to_status_id       0 non-null      float64
 2   in_reply_to_user_id         0 non-null      float64
 3   timestamp                   2097 non-null   datetime64[ns, UTC]
 4   source                      2097 non-null   object
 5   text                        2097 non-null   object
 6   retweeted_status_id         0 non-null      float64
 7   retweeted_status_user_id    0 non-null      float64
 8   retweeted_status_timestamp  0 non-null      object
 9   expanded_urls               2094 non-null   object
 10  rating_numerator            2097 non-null   int64
 11  rating_denominator          2097 non-null   int64
 12  name                        2097 non-null   object
 13  doggo                       2097 non-null   object
 14  floofer                     2097 non-null   object
 15  pupper                      2097 non-null   object
 16  puppo                       2097 non-null   object
dtypes: datetime64[ns, UTC](1), float64(4), int64(3), object(9)
memory usage: 294.9+ KB
```

In [33]:

```
#no records for retweet or reply
df_enhanced_clean[(df_enhanced_clean['in_reply_to_status_id'].notnull()) | (df_enhanced
_clean['retweeted_status_id'].notnull())]
```

Out[33]:

tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_sta

- **Drop unuseful columns: in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp**

Define

- Drop columns which will not be used: in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp

Code

In [34]:

```
#drop the columns which will not be used
list_col=['in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retwe
eted_status_user_id', 'retweeted_status_timestamp']
df_enhanced_clean.drop(list_col,axis=1,inplace=True)
```

Test

In [35]:

```
#confirm columns dropped
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2097 non-null   int64
 1   timestamp           2097 non-null   datetime64[ns, UTC]
 2   source              2097 non-null   object
 3   text                2097 non-null   object
 4   expanded_urls       2094 non-null   object
 5   rating_numerator    2097 non-null   int64
 6   rating_denominator  2097 non-null   int64
 7   name                2097 non-null   object
 8   doggo               2097 non-null   object
 9   floofer             2097 non-null   object
 10  pupper              2097 non-null   object
 11  puppo               2097 non-null   object
dtypes: datetime64[ns, UTC](1), int64(3), object(8)
memory usage: 213.0+ KB
```

- **Incorrect rating_denominator**
  - tweet_id:666287406224695296(2->10)

Define

- replace 2 with 10 for rating denominator for tweet_id:666287406224695296

Code

In [36]:

```
#replace with 10 for the incorrect rating denominator
df_enhanced_clean.loc[df_enhanced_clean['tweet_id'] == 666287406224695296, 'rating_deno
minator'] = 10
```

Test

In [37]:

```
#confirm rating denominator corrected
df_enhanced_clean[(df_enhanced_clean['tweet_id']==666287406224695296)]
```

Out[37]:

	tweet_id	timestamp	source	
2335	666287406224695296	2015-11-16 16:11:11+00:00	Twitter for iPhone	This is an Alba legged Epi Loves wel hardwoo Penis on the c https://t.co/d9N

- **Incorrect rating_numerator**
  - tweet_id:666287406224695296(1->9)
  - Float number was recognized as integer.

Define

- replace 1 with 9 for rating numerator for tweet_id:666287406224695296
- extract float number from text column and store in column rating_numerator
- change datatype to float

Code

In [38]:

```
#replace with 9 for the incorrect rating numerator
df_enhanced_clean.loc[df_enhanced_clean['tweet_id'] == 666287406224695296, 'rating_nume
rator'] = 9
```

In [39]:

```
#extract float number and store in column rating_numerator
for row in df_enhanced_clean.itertuples():
    match=re.findall("\d+\.\d+/", row.text)
    if match:
        df_enhanced_clean.loc[row.Index,'rating_numerator']=match[0][:-1]
```

In [40]:

```
#change data type for column rating_numerator to float
df_enhanced_clean.rating_numerator=df_enhanced_clean.rating_numerator.astype('float')
```

Test

In [41]:

```
#confirm rating numerator corrected
df_enhanced_clean[(df_enhanced_clean['tweet_id']==666287406224695296)]
```

Out[41]:

	tweet_id	timestamp	source	
2335	666287406224695296	2015-11-16 16:11:11+00:00	Twitter for iPhone	This is an Alba legged Epi Loves wel hardwoo Penis on the c https://t.co/d9N

In [42]:

```
#confirm float number stored in column
df_enhanced_clean.rating_numerator.value_counts()
```

Out[42]:

```
12.00      486
10.00      436
11.00      413
13.00      287
9.00       154
8.00        98
7.00        52
14.00       38
5.00        33
6.00        32
3.00        19
4.00        16
2.00         9
1.00         4
13.50        1
0.00         1
24.00        1
84.00        1
420.00       1
1776.00      1
80.00        1
60.00        1
44.00        1
144.00       1
88.00        1
11.26        1
11.27        1
121.00       1
9.75         1
99.00        1
204.00       1
45.00        1
165.00       1
50.00        1
Name: rating_numerator, dtype: int64
```

In [43]:

```
#confirm column data type
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2097 non-null   int64              
 1   timestamp           2097 non-null   datetime64[ns, UTC]
 2   source              2097 non-null   object             
 3   text                2097 non-null   object             
 4   expanded_urls       2094 non-null   object             
 5   rating_numerator    2097 non-null   float64            
 6   rating_denominator  2097 non-null   int64              
 7   name                2097 non-null   object             
 8   doggo               2097 non-null   object             
 9   floofer             2097 non-null   object             
 10  pupper              2097 non-null   object             
 11  puppo               2097 non-null   object             
dtypes: datetime64[ns, UTC](1), float64(1), int64(2), object(8)
memory usage: 293.0+ KB
```

- **There are 745 None values under name column, and the names with lowercase are invalid names.**

Define

- Change None values and names with lowercase to null.
- Extract name from text column by using keywords.

Code

In [44]:

```
#replace with null values for name under lowercase
df_enhanced_clean.name.replace(df_enhanced_clean[df_enhanced_clean.name.str.islower()].
name,np.nan,inplace=True)

#replace with null values for name under None value
df_enhanced_clean.name.replace('None',np.nan,inplace=True)
```

In [45]:

```
# extract name from text column by using keywords
df_enhanced_clean['name'] = df_enhanced_clean.text.str.extract(r'(?:his is|Meet|Say hel
lo to|Here is|named|name is|Here we have)\s([A-Z][^\s.,]*)',expand=True)
```

Test

In [46]:

```
#confirm the null values under name column
df_enhanced_clean[df_enhanced_clean.name.notnull()].name.count()
```

Out[46]:

```
1424
```

In [47]:

```
#confirm name under lowercase not exists
df_enhanced_clean.name.unique()
```

Out[47]:

```
array(['Phineas', 'Tilly', 'Archie', 'Darla', 'Franklin', nan, 'Jax',
       'Zoey', 'Cassie', 'Koda', 'Bruno', 'Ted', 'Stuart', 'Oliver',
       'Jim', 'Zeke', 'Ralphus', 'Gerald', 'Jeffrey', 'Canela', 'Maya',
       'Mingus', 'Derek', 'Roscoe', 'Waffles', 'Jimbo', 'Maisey',
       'Howard', 'Earl', 'Lola', 'Kevin', 'Yogi', 'Noah', 'Bella',
       'Grizzwald', 'Rusty', 'Gus', 'Stanley', 'Alfy', 'Koko', 'Rey',
       'Gary', 'Elliot', 'Louis', 'Jesse', 'Romeo', 'Bailey', 'Duddles',
       'Jack', 'Steven', 'Beau', 'Snoopy', 'Shadow', 'Emmy', 'Aja',
       'Penny', 'Dante', 'Nelly', 'Ginger', 'Benedict', 'Venti', 'Goose',
       'Nugget', 'Cash', 'Jed', 'Sebastian', 'Sierra', 'Monkey', 'Harry',
       'Kody', 'Lassie', 'Rover', 'Napolean', 'Boomer', 'Cody', 'Rumble',
       'Clifford', 'Dewey', 'Scout', 'Gizmo', 'Walter', 'Cooper',
       'Harold', 'Shikha', 'Lili', 'Jamesy', 'Coco', 'Sammy', 'Meatball',
       'Paisley', 'Albus', 'Neptune', 'Belle', 'Quinn', 'Zooey', 'Dave',
       'Jersey', 'Hobbes', 'Burt', 'Lorenzo', 'Carl', 'Jordy', 'Milky',
       'Trooper', 'Sophie', 'Wyatt', 'Rosie', 'Thor', 'Oscar', 'Callie',
       'Cermet', 'Marlee', 'Arya', 'Einstein', 'Alice', 'Rumpole',
       'Benny', 'Aspen', 'Jarod', 'Wiggles', 'General', 'Sailor', 'Iggy',
       'Snoop', 'Kyle', 'Leo', 'Riley', 'Noosh', 'Odin', 'Jerry',
       'Georgie', 'Rontu', 'Cannon', 'Furzey', 'Daisy', 'Tuck', 'Barney',
       'Vixen', 'Jarvis', 'Mimosa', 'Pickles', 'Brady', 'Luna', 'Charlie',
       'Margo', 'Sadie', 'Hank', 'Tycho', 'Indie', 'Winnie', 'George',
       'Bentley', 'Max', 'Dawn', 'Maddie', 'Monty', 'Sojourner',
       'Winston', 'Odie', 'Arlo', 'Vincent', 'Lucy', 'Clark', 'Mookie',
       'Meera', 'Ava', 'Eli', 'Ash', 'Tucker', 'Tobi', 'Chester',
       'Wilson', 'Sunshine', 'Lipton', 'Bronte', 'Poppy', 'Gidget',
       'Rhino', 'Willow', 'Orion', 'Eevee', 'Smiley', 'Miguel', 'Emanuel',
       'Kuyu', 'Blue', 'Dutch', 'Pete', 'Scooter', 'Reggie', 'Lilly',
       'Samson', 'Mia', 'Astrid', 'Malcolm', 'Dexter', 'Alfie', 'Fiona',
       'Mutt', 'Bear', 'Doobert', 'Beebop', 'Alexander', 'Sailer',
       'Brutus', 'Kona', 'Boots', 'Ralphie', 'Loki', 'Cupid', 'Pawnd',
       'Pilot', 'Ike', 'Mo', 'Toby', 'Sweet', 'Pablo', 'Nala', 'Crawford',
       'Gabe', 'Jimison', 'Duchess', 'Harlso', 'Sundance', 'Luca',
       'Flash', 'Sunny', 'Howie', 'Jazzy', 'Anna', 'Finn', 'Bo', 'Wafer',
       'Tom', 'Florence', 'Autumn', 'Buddy', 'Dido', 'Eugene', 'Ken',
       'Strudel', 'Tebow', 'Chloe', 'Timber', 'Binky', 'Moose', 'Dudley',
       'Comet', 'Akumi', 'Titan', 'Olivia', 'Alf', 'Oshie', 'Chubbs',
       'Sky', 'Atlas', 'Eleanor', 'Layla', 'Rocky', 'Baron', 'Tyr',
       'Bauer', 'Swagger', 'Brandi', 'Mary', 'Moe', 'Halo', 'Augie',
       'Craig', 'Sam', 'Hunter', 'Pavlov', 'Phil', 'Kyro', 'Wallace',
       'Ito', 'Seamus', 'Burke', 'Ollie', 'Stephan', 'Lennon', 'Major',
       'Duke', 'Sansa', 'Shooter', 'Django', 'Diogi', 'Sonny', 'Marley',
       'Severus', 'Ronnie', 'Milo', 'Bones', 'Mauve', 'Chef', 'Doc',
       'Peaches', 'Sobe', 'Longfellow', 'Mister', 'Iroh', 'Pancake',
       'Snicku', 'Ruby', 'Brody', 'Mack', 'Nimbus', 'Laika', 'Maximus',
       'Dobby', 'Moreton', 'Juno', 'Maude', 'Lily', 'Newt', 'Benji',
       'Nida', 'Robin', 'Monster', 'BeBe', 'Remus', 'Levi', 'Mabel',
       'Misty', 'Betty', 'Mosby', 'Maggie', 'Bruce', 'Happy', 'Ralphy',
       'Brownie', 'Rizzy', 'Stella', 'Butter', 'Frank', 'Tonks',
       'Lincoln', 'Rory', 'Logan', 'Dale', 'Rizzo', 'Arnie', 'Mattie',
       'Pinot', 'Dallas', 'Hero', 'Frankie', 'Stormy', 'Reginald',
       'Balto', 'Mairi', 'Loomis', 'Godi', 'Cali', 'Deacon', 'Timmy',
       'Sampson', 'Chipson', 'Combo', 'Oakley', 'Dash', 'Hercules', 'Jay',
       'Mya', 'Strider', 'Wesley', 'Solomon', 'Huck', "O'Malley",
       'Anakin', 'Finley', 'Sprinkles', 'Heinrich', 'Shakespeare',
       'Chelsea', 'Bungalo', 'Chip', 'Grey', 'Roosevelt', 'Willem',
       'Davey', 'Dakota', 'Fizz', 'Dixie', 'Al', 'Jackson', 'Carbon',
       'Klein', 'DonDon', 'Kirby', 'Lou', 'Chevy', 'Tito', 'Philbert',
       'Louie', 'Rupert', 'Rufus', 'Brudge', 'Shadoe', 'Angel', 'Brat',
```

```
'Tove', 'Gromit', 'Aubie', 'Kota', 'Leela', 'Glenn', 'Shelby',
'Sephie', 'Bonaparte', 'Albert', 'Wishes', 'Rose', 'Theo', 'Rocco',
'Fido', 'Emma', 'Spencer', 'Lilli', 'Boston', 'Brandonald',
'Corey', 'Leonard', 'Beckham', 'Devón', 'Gert', 'Watson', 'Keith',
'Dex', 'Ace', 'Tayzie', 'Grizzie', 'Fred', 'Gilbert', 'Meyer',
'Zoe', 'Stewie', 'Calvin', 'Lilah', 'Spanky', 'Jameson', 'Piper',
'Atticus', 'Blu', 'Dietrich', 'Divine', 'Tripp', 'Cora', 'Huxley',
'Keurig', 'Bookstore', 'Linus', 'Abby', 'Shiloh', 'Gustav',
'Arlen', 'Percy', 'Lenox', 'Sugar', 'Harvey', 'Blanket', 'Geno',
'Stark', 'Beya', 'Kilo', 'Kayla', 'Maxaroni', 'Bell', 'Doug',
'Edmund', 'Aqua', 'Theodore', 'Baloo', 'Bretagne', 'Chase',
'Nollie', 'Rorie', 'Simba', 'Charles', 'Bayley', 'Axel',
'Storkson', 'Remy', 'Chadrick', 'Kellogg', 'Buckley', 'Livvie',
'Terry', 'Hermione', 'Ralpher', 'Aldrick', 'Larry', 'Rooney',
'Crystal', 'Ziva', 'Stefan', 'Pupcasso', 'Puff', 'Flurpson',
'Coleman', 'Enchilada', 'Raymond', 'Rueben', 'Cilantro', 'Karll',
'Sprout', 'Blitz', 'Bloop', 'Colby', 'Lillie', 'Fred-Rick',
'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',
'Stubert', 'Kane', 'Olive', 'Chuckles', 'Staniel', 'Sora', 'Beemo',
'Gunner', 'Lacy', 'Tater', 'Olaf', 'Cecil', 'Vince', 'Karma',
'Billy', 'Walker', 'Rodney', 'Klevin', 'Malikai', 'Bobble',
'River', 'Jebberson', 'Remington', 'Farfle', 'Jiminus', 'Harper',
'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-E', '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', 'Herschel', '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',
'Tyrone', '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', 'Thea', 'Ulysses', 'Toffee', 'Apollo',
'Carly', 'Asher', 'Glacier', 'Chuck', 'Champ', 'Ozzie', 'Griswold',
'Cheesy', 'Moofasa', 'Hector', 'Goliath', 'Kawhi', 'Emmie',
'Penelope', 'Willie', 'Rinna', 'Sabertooth', 'Mike', 'William',
'Dwight', 'Evy', 'Hurley', 'Rubio', 'Chompsky', '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', 'Wylie', 'Arnold',
'Zuzu', 'Mollie', 'Laela', 'Tedders', 'Superpup', 'Rufio', 'Jeb',
'Rodman', 'Jonah', 'Chesney', 'Kenny', 'Henry', 'Bobbay', 'Mitch',
'Kaiya', 'Acro', 'Aiden', 'Obie', 'Dot', 'Shnuggles', 'Kendall',
'Kip', 'Jeffri', 'Steve', 'Eve', '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',
'Jacob', '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', 'Paull', 'Willy',
'Spork', 'Herb', 'Damon', 'Peanut', 'Nigel', 'Cherokee', 'Butters',
'Hemry', 'Sandra', 'Fabio', 'Randall', 'Liam', 'Tommy', 'Ben',
'Raphael', 'Julio', 'Andru', 'Alphred', 'Kloey', 'Shawwn', 'Skye',
'Kollin', 'Alfredo', 'Ronduh', 'Billl', 'Saydee', 'Dug', 'Sully',
'Kirk', 'Ralf', 'Clarq', 'Jaspers', 'Samsom', 'Pancho', 'Terrance',
'Harrison', 'Chaz', 'Jeremy', 'Jaycob', 'Leroi', 'Lambeau',
'Ruffles', 'Amélie', 'Bobb', 'Banditt', 'Kevon', 'Winifred',
'Hanz', 'Berta', 'Churlie', 'Zeek', 'Timofy', 'Maks', 'Jomathan',
'Kallie', 'Marvin', 'Spark', 'Gòrdón', 'Chuk', 'Jo', 'DayZ',
'Guss', 'Jareld', 'Torque', 'Ron', 'Skittles', 'Alfonso',
'Cleopatricia', 'Erik', 'Stu', 'Tedrick', 'Shaggy', 'Filup',
'Kial', 'Klint', 'Naphaniel', 'Big', 'Dook', 'Tickles', 'Hall',
'Philippe', 'Kohl', 'Biden', 'Fwed', 'Genevieve', 'Joshwa',
'Timison', 'Daryl', 'Bradlay', 'Pipsy', 'Clybe', 'Keet', 'Carll',
'Pepe', 'Jockson', 'Octaviath', 'Josep', 'Lugan', 'Johm',
'Christoper'], dtype=object)
```

- **doggo, floofer, pupper, puppo columns contain 'None' value which is not counted as null**

Define

- Replace None with NaN for columns doggo, floofer, pupper, puppo

Code

In [48]:

```
#replace none to NaN
df_enhanced_clean.doggo.replace('None',np.nan,inplace=True)
df_enhanced_clean.floofer.replace('None',np.nan,inplace=True)
df_enhanced_clean.pupper.replace('None',np.nan,inplace=True)
df_enhanced_clean.puppo.replace('None',np.nan,inplace=True)
```

Test

In [49]:

```
#confirm the null values exist
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2097 non-null   int64              
 1   timestamp           2097 non-null   datetime64[ns, UTC]
 2   source              2097 non-null   object             
 3   text                2097 non-null   object             
 4   expanded_urls       2094 non-null   object             
 5   rating_numerator    2097 non-null   float64            
 6   rating_denominator  2097 non-null   int64              
 7   name                1424 non-null   object             
 8   doggo               83 non-null     object             
 9   floofer             10 non-null     object             
 10  pupper              230 non-null    object             
 11  puppo               24 non-null     object             
dtypes: datetime64[ns, UTC](1), float64(1), int64(2), object(8)
memory usage: 293.0+ KB
```

In [50]:

```
#confirm no none values
print(df_enhanced_clean.doggo.value_counts())
print(df_enhanced_clean.floofer.value_counts())
print(df_enhanced_clean.pupper.value_counts())
print(df_enhanced_clean.puppo.value_counts())
```

```
doggo    83
Name: doggo, dtype: int64
floofer    10
Name: floofer, dtype: int64
pupper    230
Name: pupper, dtype: int64
puppo    24
Name: puppo, dtype: int64
```

- **There are records with more than one stages (doggo with one of the floofer, pupper, puppo columns)**
  - tweet_id:855851453814013952
  - 854010172552949760
  - 817777686764523521
  - 808106460588765185
  - 802265048156610565
  - 801115127852503040
  - 785639753186217984
  - 781308096455073793
  - 759793422261743616
  - 751583847268179968
  - 741067306818797568
  - 733109485275860992
  - 775898661951791106
  - 770093767776997377

Define

- Records with tweet_id 775898661951791106 and 770093767776997377 were removed due to records of retweet or reply.
- Remove records of ratings for two dogs with tweet_id: 808106460588765185, 781308096455073793, 759793422261743616, 741067306818797568, 733109485275860992
- Remove records not related to dog with tweet_id: 785639753186217984
- Correct the dog stage for the wrong classification with tweet_id(doggo->nan): 855851453814013952, 854010172552949760, 817777686764523521, 801115127852503040
- Correct the dog stage for the wrong classification with tweet_id(pupper->nan): 751583847268179968

Code

In [51]:

```
#series for the tweet ids which needs to be removed
id_remove=pd.Series([808106460588765185, 781308096455073793, 759793422261743616, 741067
306818797568, 733109485275860992,785639753186217984])
#remove records with above tweet ids
df_enhanced_clean=df_enhanced_clean[~df_enhanced_clean.tweet_id.isin(id_remove) ]

#series for the tweet ids which needs to be corrected
id_change=pd.Series([855851453814013952, 854010172552949760, 817777686764523521, 801115
127852503040])
#correct the dog stage for the records with above tweet ids
df_enhanced_clean.doggo.replace(df_enhanced_clean[df_enhanced_clean.tweet_id.isin(id_ch
ange)].doggo,np.nan,inplace=True)

#correct the dog stage for the records with tweet id 751583847268179968
df_enhanced_clean.loc[df_enhanced_clean.tweet_id==751583847268179968,'pupper']=np.nan
```

Test

In [52]:

```
#confirm the df info after modification
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2091 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2091 non-null   int64
 1   timestamp           2091 non-null   datetime64[ns, UTC]
 2   source              2091 non-null   object
 3   text                2091 non-null   object
 4   expanded_urls       2088 non-null   object
 5   rating_numerator    2091 non-null   float64
 6   rating_denominator  2091 non-null   int64
 7   name                1421 non-null   object
 8   doggo               73 non-null     object
 9   floofer             10 non-null     object
 10  pupper              223 non-null    object
 11  puppo               24 non-null     object
dtypes: datetime64[ns, UTC](1), float64(1), int64(2), object(8)
memory usage: 292.4+ KB
```

In [53]:

```
#confirm no multiple dog stages records
df_enhanced_clean[(df_enhanced_clean['doggo']=='doggo') & ((df_enhanced_clean['floofer'
].notnull()) | (df_enhanced_clean['pupper'].notnull()) | (df_enhanced_clean['puppo'].no
tnull()))]
```

Out[53]:

tweet_id	timestamp	source	text	expanded_urls	rating_numerator	rating_denominator	n

- **Calculate rating with values of rating_numerator divided by rating_denominator**

Define

- Create new column rating: rating_numerator/rating_denominator

Code

In [54]:

```
#calculate rating column by rating_numerator/rating_denominator
df_enhanced_clean['rating']=df_enhanced_clean['rating_numerator']/df_enhanced_clean['ra
ting_denominator']
```

Test

In [55]:

```
#confirm new column created
df_enhanced_clean.head()
```

Out[55]:

	tweet_id	timestamp	source	
0	892420643555336193	2017-08-01 16:23:56+00:00	Twitter for iPhone	This is Phineas mystical boy. O appears in the h donu https://t.co/MgUW(
1	892177421306343426	2017-08-01 00:17:27+00:00	Twitter for iPhone	This is Tilly. Sl checking pup Hopes you're doi not, she's avai pats, snugs, bo whole b https://t.co/0Xx(
2	891815181378084864	2017-07-31 00:18:03+00:00	Twitter for iPhone	This is Archie. rare Norwegian P Corgo. Lives ir grass. You nev when one ma https://t.co/wUnz
3	891689557279858688	2017-07-30 15:58:51+00:00	Twitter for iPhone	This is Da commenced a mid meal. 13/10 h to the be https://t.co/tD36
4	891327558926688256	2017-07-29 16:00:24+00:00	Twitter for iPhone	This is Franklin. H like you to stop him "cute." He i fierce shark and be respected a 12/10 #Ba https://t.co/AtU

In [56]:

```
#confirm new column info
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2091 entries, 0 to 2355
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2091 non-null   int64              
 1   timestamp           2091 non-null   datetime64[ns, UTC]
 2   source              2091 non-null   object             
 3   text                2091 non-null   object             
 4   expanded_urls       2088 non-null   object             
 5   rating_numerator    2091 non-null   float64            
 6   rating_denominator  2091 non-null   int64              
 7   name                1421 non-null   object             
 8   doggo               73 non-null     object             
 9   floofer             10 non-null     object             
 10  pupper              223 non-null    object             
 11  puppo               24 non-null     object             
 12  rating              2091 non-null   float64            
dtypes: datetime64[ns, UTC](1), float64(2), int64(2), object(8)
memory usage: 308.7+ KB
```

- **Sometimes lowercase and sometimes uppercase for breed names in p1, p2, p3 columns**

Define

- Capitalize first letter for p1,p2,p3 columns

Code

In [57]:

```
#change first letter to capital letter
df_image_clean.p1=df_image_clean.p1.str.capitalize()
df_image_clean.p2=df_image_clean.p2.str.capitalize()
df_image_clean.p3=df_image_clean.p3.str.capitalize()
```

Test

In [58]:

```
#confirm values updated
df_image_clean
```

Out[58]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	C
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rho
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	M
...	...	...	...	
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

Tidiness

- **doggo, floofer, pupper, puppo columns should be combined in one column with category data type in the `df_enhanced` table**

Define

- Join the four dog stage columns doggo,floofer,pupper,puppo into one
- Drop the four dog stage columns after combined
- Change datatype to category

Code

In [59]:

```
df_enhanced_clean[['doggo', 'floofer', 'pupper', 'puppo']].apply(lambda x: ','.join(x.dropna().astype('category')),axis=1)
```

Out[59]:

```
0
1
2
3
4
      ..
2351
2352
2353
2354
2355
Length: 2091, dtype: object
```

In [60]:

```
#create column dog_stage to combine columns doggo,floofer,pupper,puppo, and set datatype to category
df_enhanced_clean['dog_stage'] = df_enhanced_clean[['doggo', 'floofer', 'pupper', 'puppo']].apply(
    lambda x: ','.join(x.dropna().astype('category')),axis=1)

#replace with null values
df_enhanced_clean.replace('',np.nan,inplace=True)

#drop the four columns after combined
df_enhanced_clean.drop(['doggo','floofer','pupper','puppo'],axis=1,inplace=True)
```

Test

In [61]:

```
#check columns combined
df_enhanced_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2091 entries, 0 to 2355
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2091 non-null   int64
 1   timestamp           2091 non-null   datetime64[ns, UTC]
 2   source              2091 non-null   object
 3   text                2091 non-null   object
 4   expanded_urls       2088 non-null   object
 5   rating_numerator    2091 non-null   float64
 6   rating_denominator  2091 non-null   int64
 7   name                1421 non-null   object
 8   rating              2091 non-null   float64
 9   dog_stage           330 non-null    object
dtypes: datetime64[ns, UTC](1), float64(2), int64(2), object(5)
memory usage: 259.7+ KB
```

In [62]:

```
df_enhanced_clean.dog_stage.unique()
```

Out[62]:

```
array([nan, 'doggo', 'puppo', 'pupper', 'floofer'], dtype=object)
```

- **Merge datasets to one**

Define

- Merge three datasets to one on tweet_id

Code

In [63]:

```
#merge datasets into one on tweet_id
from functools import reduce
df_clean=reduce(lambda x,y: pd.merge(x,y, on='tweet_id'), [df_enhanced_clean, df_image_
clean, df_api_clean])
```

Test

In [64]:

```
#confirm newly created dataframe
df_clean
```

Out[64]:

	tweet_id	timestamp	source	
0	892420643555336193	2017-08-01 16:23:56+00:00	Twitter for iPhone	This is Phir mystical bo appears in t c https://t.co/Mgl
1	892177421306343426	2017-08-01 00:17:27+00:00	Twitter for iPhone	This is Till checking Hopes you're not, she's a pats, snugs who https://t.co/(
2	891815181378084864	2017-07-31 00:18:03+00:00	Twitter for iPhone	This is Arc rare Norwegia Corgo. Live grass. You when one https://t.co/w
3	891689557279858688	2017-07-30 15:58:51+00:00	Twitter for iPhone	This is commence mid meal. 13/ to th https://t.co/tl
4	891327558926688256	2017-07-29 16:00:24+00:00	Twitter for iPhone	This is Frankli like you to him "cute." fierce shark be respect 12/10 https://t.co/
...	...	...	...	
1961	666049248165822465	2015-11-16 00:24:50+00:00	Twitter for iPhone	Here we have generation vu sweat tea anc Cannot be p https://t.co/4E
1962	666044226329800704	2015-11-16 00:04:52+00:00	Twitter for iPhone	This is a pur Morgan. Lov and chill. A like he forg th https://t.co/D
1963	666033412701032449	2015-11-15 23:21:54+00:00	Twitter for iPhone	Here is a very Big fan of well decks. Just tongue. https://t.co/y
1964	666029285002620928	2015-11-15 23:05:30+00:00	Twitter for iPhone	This is a we Mitsubishi te about lea dogs here. walk th https://t.co/r7

	tweet_id	timestamp	source	
1965	666020888022790149	2015-11-15 22:32:08+00:00	Twitter for iPhone	Here we have Irish Setter. Vietnam (? relaxing o https://t.co/

1966 rows × 23 columns

In [65]:

```
#confirm newly created dataframe info
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1966 entries, 0 to 1965
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1966 non-null   int64
 1   timestamp           1966 non-null   datetime64[ns, UTC]
 2   source              1966 non-null   object
 3   text                1966 non-null   object
 4   expanded_urls       1966 non-null   object
 5   rating_numerator    1966 non-null   float64
 6   rating_denominator  1966 non-null   int64
 7   name                1379 non-null   object
 8   rating              1966 non-null   float64
 9   dog_stage           298 non-null    object
 10  jpg_url             1966 non-null   object
 11  img_num             1966 non-null   int64
 12  p1                  1966 non-null   object
 13  p1_conf             1966 non-null   float64
 14  p1_dog              1966 non-null   bool
 15  p2                  1966 non-null   object
 16  p2_conf             1966 non-null   float64
 17  p2_dog              1966 non-null   bool
 18  p3                  1966 non-null   object
 19  p3_conf             1966 non-null   float64
 20  p3_dog              1966 non-null   bool
 21  retweet_count       1966 non-null   int64
 22  favorite_count      1966 non-null   int64
dtypes: bool(3), datetime64[ns, UTC](1), float64(5), int64(5), object(9)
memory usage: 328.3+ KB
```

In [66]:

```
df_clean.describe()
```

Out[66]:

	tweet_id	rating_numerator	rating_denominator	rating	img_num	p1
count	1.966000e+03	1966.000000	1966.000000	1966.000000	1966.000000	1966.0
mean	7.359668e+17	12.184527	10.482706	1.165290	1.201933	0.5
std	6.758285e+16	41.659400	6.859288	4.093021	0.559367	0.2
min	6.660209e+17	0.000000	7.000000	0.000000	1.000000	0.0
25%	6.758475e+17	10.000000	10.000000	1.000000	1.000000	0.3
50%	7.087246e+17	11.000000	10.000000	1.100000	1.000000	0.5
75%	7.881228e+17	12.000000	10.000000	1.200000	1.000000	0.8
max	8.924206e+17	1776.000000	170.000000	177.600000	4.000000	1.0

Storing, Analyzing, and Visualizing Data

- Store the clean DataFrame(s) in a CSV file with the main one named twitter_archive_master.csv.
- Analyze and visualize your wrangled data in your wrangle_act.ipynb Jupyter Notebook.
- At least three (3) insights and one (1) visualization must be produced.

In [67]:

```
#save clean dataframe to csv file
df_clean.to_csv('twitter_archive_master.csv', index=False)
```

In this section I will answer the following questions:

- The highest rated dog
- The dog with highest retweet counts
- The dog with highest favorite counts
- What are most popular 10 dogs' names?
- What is the most common dog stage? Is there any difference on rating, retweet counts, favorite counts between dog stages?
- Is there any impact on retweet and favorite counts based on ratings?
- What is the percentage the algorithm can predict a dog breed?

The highest rated dog

In [68]:

```
#the record of highest rated dog
highest_rating=df_clean['rating'].max()
df_clean[df_clean['rating']==highest_rating]
```

Out[68]:

	tweet_id	timestamp	source	
719	749981277374128128	2016-07-04 15:00:45+00:00	TweetDeck	This i https:/

1 rows × 23 columns

Answer: The highest rated dog rated at rating of 177.6, whose name is Atticus.

The dog with highest retweet counts

In [69]:

```
#the record of highest retweeted dog
highest_retweet=df_clean['retweet_count'].max()
df_clean[df_clean['retweet_count']==highest_retweet]
```

Out[69]:

	tweet_id	timestamp	source	
766	744234799360020481	2016-06-18 18:26:18+00:00	Twitter for iPhone	Here's a doggo you can stand 13/10 enlighten by Tina https://t.co/7wE

1 rows × 23 columns

Answer: The highest retweeted dog has been retweeted for 79,515 times.

The dog with highest favorite counts

In [70]:

```
#the record with highest favorite counts
highest_favorite=df_clean['favorite_count'].max()
df_clean[df_clean['favorite_count']==highest_favorite]
```

Out[70]:

	tweet_id	timestamp	source	
306	822872901745569793	2017-01-21 18:26:02+00:00	Twitter for iPhone	Here's a supportive participatin #Women today https://t.co/nTz3

1 rows × 23 columns

Answer: There are favorite counts of 132,810 times for the dog with highest favorite counts.

What are most popular 10 dogs' names?

In [71]:

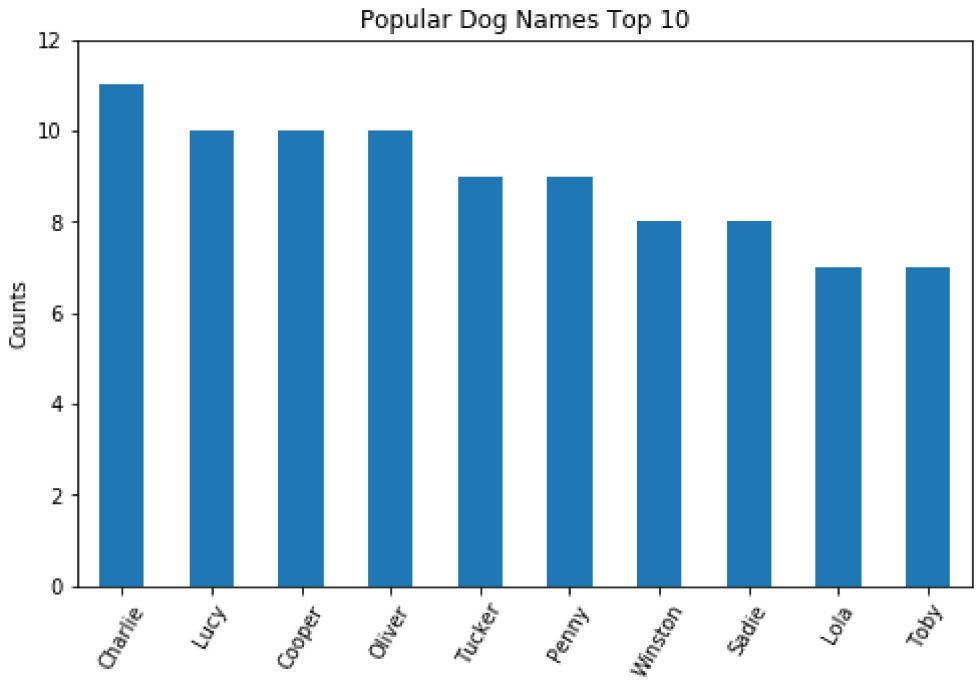
```
#most popular 10 names
pop_name_10=df_clean.name.value_counts()[:10]
pop_name_10
```

Out[71]:

```
Charlie    11
Lucy       10
Cooper     10
Oliver     10
Tucker      9
Penny       9
Winston     8
Sadie       8
Lola        7
Toby        7
Name: name, dtype: int64
```

In [72]:

```
#bar plot for the top 10 names
pop_name_10.plot(kind='bar',figsize=(8,5));
plt.ylim([0,12]);
plt.title('Popular Dog Names Top 10');
plt.ylabel('Counts');
plt.xticks(rotation=60);
```



Answer: The most popular names top 10 are: Charlie, Cooper, Lucy, Oliver, Penny, Tucker, Winston, Sadie, Lola and Toby.

What is the most common dog stage? Is there any difference on rating, retweet counts, favorite counts between dog stages?

In [73]:

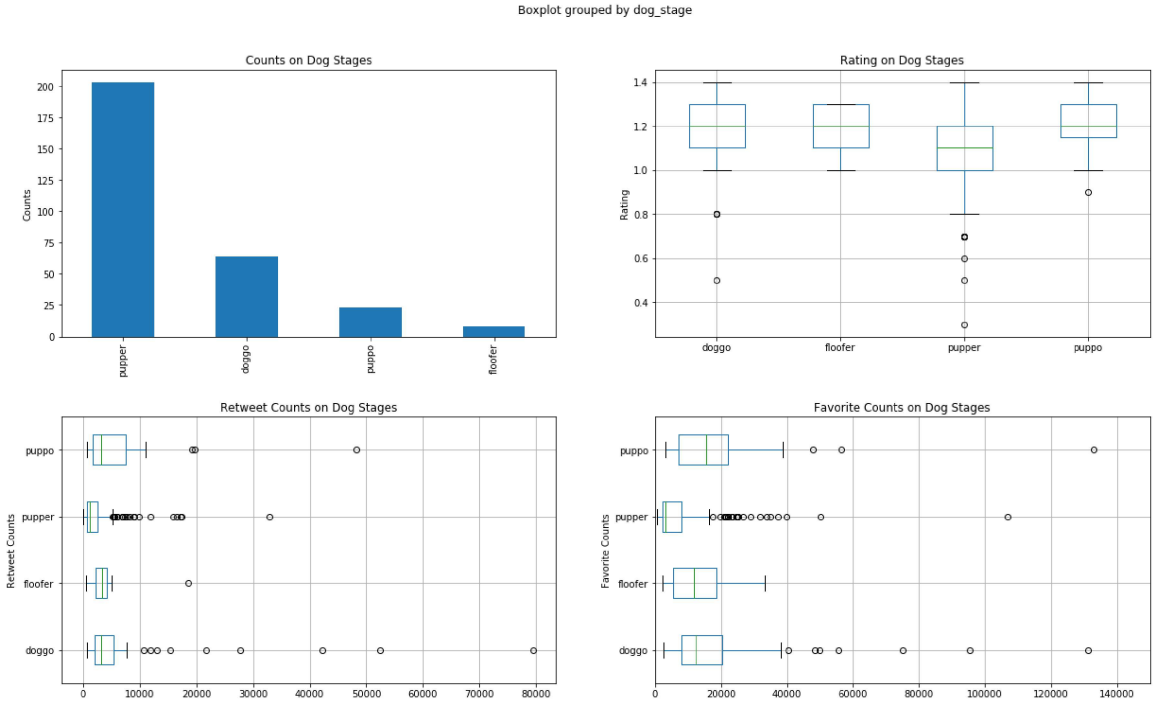
```
#adjust figure size and space between plots
fig = plt.figure(figsize=(20, 12));
plt.subplots_adjust(wspace=0.3,hspace=0.3);

#plot bar chart of number of dog stages
ax1 = fig.add_subplot(2, 2, 1);
df_clean.dog_stage.value_counts().plot(ax=ax1,kind='bar');
plt.title('Counts on Dog Stages');
plt.ylabel('Counts');

#plot bar chart of rating on dog stages
ax2 = fig.add_subplot(2, 2, 2);
df_clean.boxplot("rating", by="dog_stage",ax=ax2);
plt.title('Rating on Dog Stages');
plt.ylabel('Rating');
plt.xlabel('');

#plot bar chart of retweet counts on dog stages
ax3 = fig.add_subplot(2, 2, 3)
df_clean.boxplot("retweet_count", by="dog_stage",ax=ax3,vert=False);
plt.title('Retweet Counts on Dog Stages');
plt.ylabel('Retweet Counts');
plt.xlabel('');

#plot bar chart of favorite counts on dog stages
ax4 = fig.add_subplot(2, 2, 4);
df_clean.boxplot("favorite_count", by="dog_stage",ax=ax4,vert=False);
plt.xlim([0,150000]);
plt.title('Favorite Counts on Dog Stages');
plt.ylabel('Favorite Counts');
plt.xlabel('');
```



Answer:

- Pupper is the most common dog stage. However, for rating, retweet counts, and favorite counts, pupper is the lowest among all of the four dog stages.
- The median of rating, retweet counts, and favorite counts for other three dog stages are almost the same.
- For retweet and favorite counts, there are outliers for doggo with higher counts.

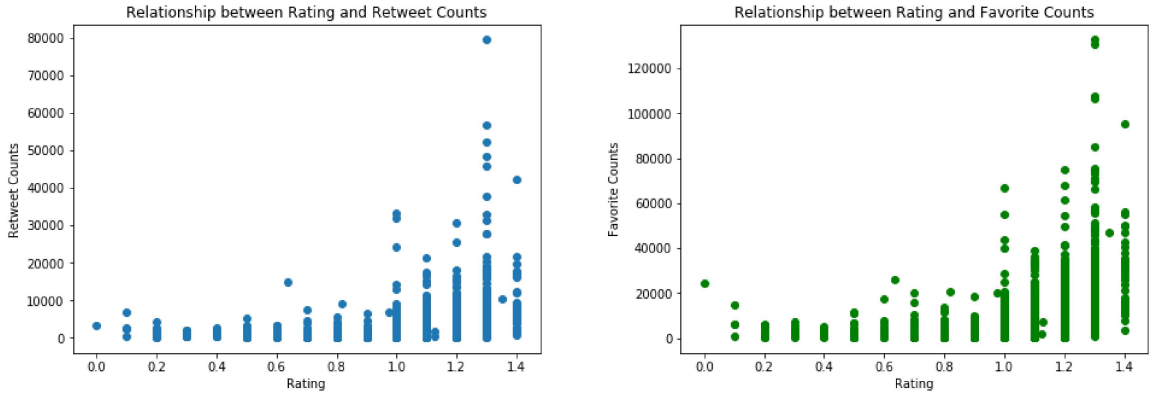
Is there any impact on retweet and favorite counts based on ratings?

In [74]:

```
#remove outliers: filter out the rating less than 2
df_rating_2=df_clean[df_clean.rating<=2]
#adjust figure size and space between plots
fig = plt.figure(figsize=(16, 5))
plt.subplots_adjust(wspace=0.3)

#plot scatter chart to view the relationship between rating and retweet counts
ax1 = fig.add_subplot(1, 2, 1)
ax1.scatter(df_rating_2.rating,df_rating_2.retweet_count);
plt.title('Relationship between Rating and Retweet Counts');
plt.xlabel('Rating');
plt.ylabel('Retweet Counts');


#plot scatter chart to view the relationship between rating and favorite counts
ax2 = fig.add_subplot(1, 2, 2)
ax2.scatter(df_rating_2.rating,df_rating_2.favorite_count,color='g');
plt.title('Relationship between Rating and Favorite Counts');
plt.xlabel('Rating');
plt.ylabel('Favorite Counts');
```



Answer: Both plots show that higher the rating is, the number of retweet or favorite is higher, which implies that rating has impact on the retweet and favorite counts.

What is the percentage the algorithm can predict a dog breed?

In [75]:

```
#predict a dog breed for the first trial
p1_percent=df_clean.p1_dog.mean()
p1_percent
```

Out[75]:

```
0.742115971515768
```

In [76]:

```
#predict a dog breed before the second trial
p2_percent=((df_clean.p1_dog==True) | (df_clean.p2_dog==True)).mean()
p2_percent
```

Out[76]:

```
0.814852492370295
```

In [77]:

```
#predict a dog breed before the third trial
p3_percent=((df_clean.p1_dog==True) | (df_clean.p2_dog==True) | (df_clean.p3_dog==True
)).mean()
p3_percent
```

Out[77]:

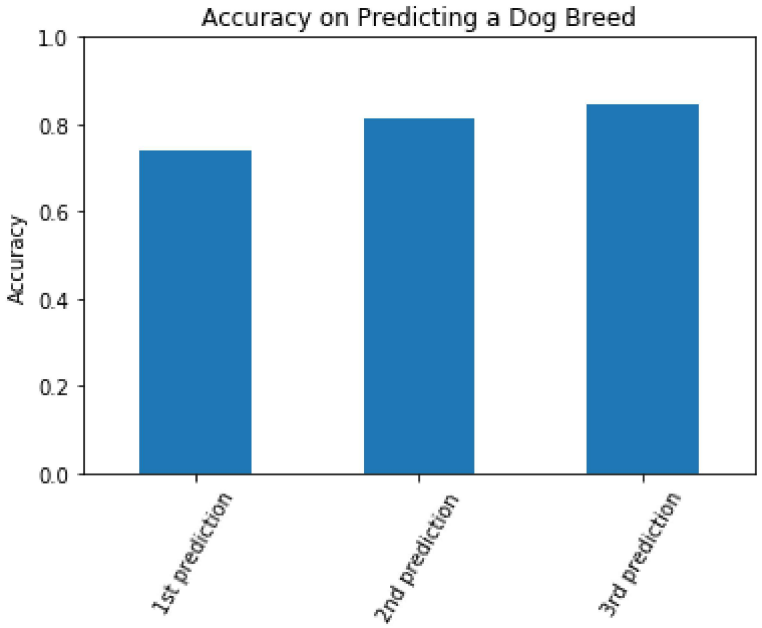
```
0.8453713123092573
```

In [78]:

```
#plot bar chart on accuracy on prediction of dog breed
pd.Series([p1_percent,p2_percent,p3_percent]).plot(kind='bar',figsize=(6,4));
plt.xticks([0, 1, 2],['1st prediction','2nd prediction','3rd prediction'],rotation=60);
plt.title('Accuracy on Predicting a Dog Breed');
plt.ylabel('Accuracy');
plt.ylim([0,1])
```

Out[78]:

```
(0, 1)
```



Answer:

- The accuracy of first prediction on successful predicting a dog breed is about 75%.
- The accuracy of second prediction is increased by about 7%, while the increase for accuracy of the third prediction is getting lower at 3%.
- The accuracy of the algorithm for predicting a dog breed is around 85% total.