We Rate Dogs Twitter Feed Analysis

1. Intro:

This project is all about gathering, assessing, cleaning, storing, analyzing and then visualizing the tweet history of the famous twitter account WeRatesDogs.

In [760]:
```
# At first, we start the project by installing all requested library.
import pandas as pd
import numpy as np
import requests
import tweepy
import json
import re
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
```

2. Gathering Data

In [761]:
```
#Getting data from an existing file and store as a dataframe named twitter_archiev
twitter_archive = pd.read_csv('twitter-archive-enhanced.csv')
```

In [762]:
```
#Using the Requests library to download a file from the internet, then write it do
url='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predi
response = requests.get(url)

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

In [763]:
```
# Read this newly created tsv file
image_predictions = pd.read_csv('image_predictions.tsv', sep='\t')
```

In [764]:
```
image_predictions.head(10)
```

Out[764]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_s
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_sh
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_rid
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pi
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_mounta
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	bo
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	shoppir
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	miniature_

In [765]:
```
# Personal API keys, secrets and tokens have been replaced with place holders
# consumer_key ='My Key Goes Here'
# consumer_secret = 'My Key Goes Here'
# access_token = 'My Key Goes Here'
# access_secret = 'My Key Goes Here'
```

In [766]:
```
# Variables created for tweepy query
# auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
# auth.set_access_token(access_token, access_secret)
# api = tweepy.API(auth, wait_on_rate_limit = True, wait_on_rate_limit_notify= Tru
```

In [767]:
```
# use For loop to add each tweet to a new line of a text file, write them all as
with open('tweet_json.txt', 'a', encoding = 'utf8') as f:
    for tweet_id in twitter_archive['tweet_id']:
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            json.dump(tweet._json, f)
            f.write('\n')
        except:
            continue
```

In [768]:
```
# use For loop to collect tweets into a list
tweets_data = []
tweet_file = open('tweet_json.txt', "r")

for line in tweet_file:
    try:
        tweet = json.loads(line)
        tweets_data.append(tweet)
    except:
        continue

tweet_file.close()
```

In [769]:
```
# create tweet_info DataFrame
tweet_info = pd.DataFrame()
```

In [770]:
```
# Add some variables to tweet_info DataFrame
tweet_info['id'] = list(map(lambda tweet: tweet['id'], tweets_data))
tweet_info['retweet_count'] = list(map(lambda tweet: tweet['retweet_count'], twee
tweet_info['farvorite_count'] = list(map(lambda tweet: tweet['favorite_count'], t
```

3. Accessing

We are going to access the twitter_archive, image_predictions and tweet_info DataFrame in term of their head(), tail(), info() and descriptive statistic categories. From that we will see a big picture of what these data set are like, what are they lacking and incorrect.

In [771]:
```
# first 15 rows of twitter_archive DataFrame
twitter_archive.head(15)
```

Out[771]:

	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 rel="nofollow">Twit
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com rel="nofollow">Twit
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com rel="nofollow">Twit
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.com rel="nofollow">Twit
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.com rel="nofollow">Twit
5	891087950875897856	NaN	NaN	2017-07-29 00:08:17 +0000	href="http://twitter.com rel="nofollow">Twit
6	890971913173991426	NaN	NaN	2017-07-28 16:27:12 +0000	href="http://twitter.com rel="nofollow">Twit
7	890729181411237888	NaN	NaN	2017-07-28 00:22:40 +0000	href="http://twitter.com rel="nofollow">Twit
8	890609185150312448	NaN	NaN	2017-07-27 16:25:51 +0000	href="http://twitter.com rel="nofollow">Twit
9	890240255349198849	NaN	NaN	2017-07-26 15:59:51 +0000	href="http://twitter.com rel="nofollow">Twit

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
10	890006608113172480	NaN	NaN	2017-07-26 00:31:25 +0000	href="http://twitter.com rel="nofollow">Twit
11	889880896479866881	NaN	NaN	2017-07-25 16:11:53 +0000	href="http://twitter.com rel="nofollow">Twit
12	889665388333682689	NaN	NaN	2017-07-25 01:55:32 +0000	href="http://twitter.com rel="nofollow">Twit
13	889638837579907072	NaN	NaN	2017-07-25 00:10:02 +0000	href="http://twitter.com rel="nofollow">Twit
14	889531135344209921	NaN	NaN	2017-07-24 17:02:04 +0000	href="http://twitter.com rel="nofollow">Twit

In [772]:
```
# last 15 rows of twitter_archive DataFrame
twitter_archive.tail(15)
```

Out[772]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
2341	666094000022159362	NaN	NaN	2015-11-16 03:22:39 +0000	href="http://twitter.cc rel="nofollow">T
2342	666082916733198337	NaN	NaN	2015-11-16 02:38:37 +0000	href="http://twitter.cc rel="nofollow">T
2343	666073100786774016	NaN	NaN	2015-11-16 01:59:36 +0000	href="http://twitter.cc rel="nofollow">T
2344	666071193221509120	NaN	NaN	2015-11-16 01:52:02 +0000	href="http://twitter.cc rel="nofollow">T
2345	666063827256086533	NaN	NaN	2015-11-16 01:22:45 +0000	href="http://twitter.cc rel="nofollow">T
2346	666058600524156928	NaN	NaN	2015-11-16 01:01:59 +0000	href="http://twitter.cc rel="nofollow">T
2347	666057090499244032	NaN	NaN	2015-11-16 00:55:59 +0000	href="http://twitter.cc rel="nofollow">T
2348	666055525042405380	NaN	NaN	2015-11-16 00:49:46 +0000	href="http://twitter.cc rel="nofollow">T

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
2349	666051853826850816	NaN	NaN	2015-11-16 00:35:11 +0000	href="http://twitter.cc rel="nofollow">T
2350	666050758794694657	NaN	NaN	2015-11-16 00:30:50 +0000	href="http://twitter.cc rel="nofollow">T
2351	666049248165822465	NaN	NaN	2015-11-16 00:24:50 +0000	href="http://twitter.cc rel="nofollow">T
2352	666044226329800704	NaN	NaN	2015-11-16 00:04:52 +0000	href="http://twitter.cc rel="nofollow">T
2353	666033412701032449	NaN	NaN	2015-11-15 23:21:54 +0000	href="http://twitter.cc rel="nofollow">T
2354	666029285002620928	NaN	NaN	2015-11-15 23:05:30 +0000	href="http://twitter.cc rel="nofollow">T
2355	666020888022790149	NaN	NaN	2015-11-15 22:32:08 +0000	href="http://twitter.cc rel="nofollow">T

In [773]:
```
twitter_archive.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 [774]:
```
twitter_archive.describe()
```

Out[774]:

	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 [775]:
```
# first 15 rows of image_predictions DataFrame
image_predictions.head(15)
```

Out[775]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_s
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ri
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_mount
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	b
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	shopp
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	miniature
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg	1	golden_
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAIOoZ.jpg	1	Gordo
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg	1	Walke
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg	1	
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg	1	bloo

In [776]:
```
# last 15 rows of image_predictions DataFrame
image_predictions.tail(15)
```

Out[776]:

	tweet_id	jpg_url	img_num	
2060	885931135344209921	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	1	go
2061	886938837579907072	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	1	F
2062	886965388333682689	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	1	
2063	886980896479866881	https://pbs.twimg.com/media/DFI99B1WsAITKsg.jpg	1	F
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMIiS.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	

In [777]:
```
image_predictions.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 [778]:
```
image_predictions.describe()
```

Out[778]:

	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 [779]:
```
# Rename the id columns to "tweet_id" in order to match with another Dataframe, th
tweet_info.rename(index=str, columns={"id":"tweet_id"}, inplace=True)
tweet_info.head(5)
```

Out[779]:

	tweet_id	retweet_count	farvorite_count
0	892420643555336193	8328	38075
1	892177421306343426	6151	32683
2	891815181378084864	4072	24595
3	891689557279858688	8472	41435
4	891327558926688256	9166	39610

In [780]:
```
tweet_info.describe()
```

Out[780]:

	tweet_id	retweet_count	farvorite_count
count	1.783000e+03	1783.000000	1783.000000
mean	7.644104e+17	3596.747056	9872.496915
std	6.367276e+16	5366.963916	13331.072948
min	6.772289e+17	0.000000	0.000000
25%	7.039145e+17	1000.000000	2594.000000
50%	7.580996e+17	1994.000000	5111.000000
75%	8.182835e+17	4140.000000	12548.000000
max	8.924206e+17	83770.000000	164485.000000

4. Issue Gathered

Quality Issue:

*** twitter_archive dataset**

Issue #1: Inappropriate columns type. There are some including:tweet_id, source, timestamp... columns should be corrected to its type.

Issue #2: Most of the rating should use 10 as a denominator. We should correct the one that doesn't.

Issue #3: 'name' columns have many cells which are None, missing or incorrect value.

Issue #4: The only data we want should have original (no retweets) ratings and have images as well.

Issue #5: In the 'text' column, there are a few extra characters after '&'

Issue #6: Some collumns should be renamed to be more details and easy to read. For example: timestamp, source, text...

Issue #7: The 'source' column is diffcult to read as It contains tags and links

*** image_predictions and tweet_info dataset**

Issue #8: Duplicated tweet_id cell noticed.

Tidiness

Issue #9: We can demostrate Dog "stage" information by collecting from four columns: doggo, floofer, pupper, puppo.

Issue #10: We can join image_predictions and tweet_info to twitter_archive table for easy read and collecting information.

5. Cleaning

In this step, we will start fixing our Quality and Tidiness issues which is stated in the last step. At first, we have to create copies of Original DataFrames. It is better to be safe here.

In [781]:
```
twitter_archive_backup = twitter_archive.copy()
image_predictions_backup = image_predictions.copy()
tweet_info_backup = tweet_info.copy()
```

Define

By joining all DataFrame at one, it is easier to us to manage and solve all of the issues

Code

In [782]:
```
df_all = pd.merge(twitter_archive, image_predictions, how = 'left', on=['tweet_id
df_all = pd.merge(df_all, tweet_info, how ='left', on=['tweet_id'])
df_all.to_csv('df_all.csv', encoding='utf-8')
#This will solve Issue #10 stated above
```

Test

In [783]:
```
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 0 to 2355
Data columns (total 30 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
jpg_url                       2075 non-null object
img_num                       2075 non-null float64
p1                            2075 non-null object
p1_conf                       2075 non-null float64
p1_dog                        2075 non-null object
p2                            2075 non-null object
p2_conf                       2075 non-null float64
p2_dog                        2075 non-null object
p3                            2075 non-null object
p3_conf                       2075 non-null float64
p3_dog                        2075 non-null object
retweet_count                 1783 non-null float64
farvorite_count               1783 non-null float64
dtypes: float64(10), int64(3), object(17)
memory usage: 570.6+ KB
```

Define

By following order, we delete the retweets, duplicated tweet_id and tweets with no image

Code

In [784]:
```
df_all = df_all[pd.isnull(df_all.retweeted_status_id)]
df_all = df_all.drop_duplicates()
df_all = df_all.dropna(subset=['jpg_url'])
#This will solve Issue #8 stated above
```

In [785]:
```
# Now remove abundance columns we don't need any more
df_all = df_all.drop(columns=['in_reply_to_status_id', 'in_reply_to_user_id','retu
# Issue #4 is solved
```

Test

In [786]:
```
list(df_all)
```

Out[786]:
```
['tweet_id',
 'timestamp',
 'source',
 'text',
 'expanded_urls',
 'rating_numerator',
 'rating_denominator',
 'name',
 'doggo',
 'floofer',
 'pupper',
 'puppo',
 'jpg_url',
 'img_num',
 'p1',
 'p1_conf',
 'p1_dog',
 'p2',
 'p2_conf',
```

Define

Blending 4 different dog stages ('doggo', 'floofer', 'pupper', 'puppo') into one column helps us to easily access and manage the content.

Code

We are extracting dog stages variable from the 'text' columns. Some dogs may have more than one stage, however, I don't think it is an issue. I think that the dogs just get an additional stage when they get older. So, we can take the first stage that listed in the column.

In [787]:
```
df_all['dog_stages'] = df_all['text'].str.extract('(puppo|pupper|floofer|doggo)',
```

In [788]:
```
selected_columns = ['doggo', 'floofer', 'pupper', 'puppo']
df_all = df_all.drop(selected_columns, axis=1)
#Solved Issue #9
```

Test

In [789]:
```
df_all.dog_stages.value_counts()
```

Out[789]:
```
pupper     223
doggo       72
puppo       28
floofer      3
Name: dog_stages, dtype: int64
```

Define

Most of the rating should use 10 as a denominator. We should correct the one that doesn't.

Code

In [790]:
```
# Take a look at the denominators where they are incorrected.
incorrect_denominator = twitter_archive[
    twitter_archive.rating_denominator != 10]
incorrect_denominator = incorrect_denominator[['tweet_id', 'text',
    'rating_numerator', 'rating_denominator']]
incorrect_denominator.sample(5)
```

Out[790]:

	tweet_id	text	rating_numerator	rating_denominator
1254	710658690886586372	Here's a brigade of puppers. All look very prepared for whatever happens next. 80/80 https://t.co/0eb7R1Om12	80	80
1120	731156023742988288	Say hello to this unbelievably well behaved squad of doggos. 204/170 would try to pet all at once https://t.co/yGQI3He3xv	204	170
1228	713900603437621249	Happy Saturday here's 9 puppers on a bench. 99/90 good work everybody https://t.co/mpvaVxKmc1	99	90
902	758467244762497024	Why does this never happen at my front door... 165/150 https://t.co/HmwrdfEfUE	165	150
1165	722974582966214656	Happy 4/20 from the squad! 13/10 for all https://t.co/eV1diwds8a	4	20

In [791]:
```
# We count how many denominator are incorrected.
len(incorrect_denominator)
```

Out[791]:
```
23
```

In [792]:
```
# Change the value to its correct number.
df_all.replace(df_all['rating_denominator'], 10, inplace = True)
```

Test

In [793]:
```
twitter_archive.rating_denominator.value_counts().head(10)
# Fixed Issue 2
```

Out[793]:
```
10    2333
11    3
50    3
80    2
20    2
2     1
16    1
40    1
70    1
15    1
Name: rating_denominator, dtype: int64
```

Define

After researching the 'text' column, I noticed the pattern: This is [name] .. Meet [name] .. Say hello to [name] .. Here we have [name] .. .. named [name] ..

Code

In [794]:
```
#By looping through all the text, we can easily extract the dog's name from the tw

dog_names = []

for text in df_all['text']:
    # Sentences starts with 'This is ' and the fisrt letter of the name is upperca
    if text.startswith('This is ') and re.match(r'[A-Z].*', text.split()[2]):
        dog_names.append(text.split()[2].strip(',').strip('.'))
    # Sentences start with 'Meet ' and the fisrt letter of the name is uppercase
    elif text.startswith('Meet ') and re.match(r'[A-Z].*', text.split()[1]):
        dog_names.append(text.split()[1].strip(',').strip('.'))
    # Sentences start with 'Say hello to ' and the fisrt letter of the name is upp
    elif text.startswith('Say hello to ') and re.match(r'[A-Z].*', text.split()[3]
        dog_names.append(text.split()[3].strip(',').strip('.'))
    # Sentences start with 'Here we have ' and the fisrt letter of the name is upp
    elif text.startswith('Here we have ') and re.match(r'[A-Z].*', text.split()[3]
        dog_names.append(text.split()[3].strip(',').strip('.'))
    # Sentences contain 'named' and the fisrt letter of the name is uppercase
    elif 'named' in text and re.match(r'[A-Z].*', text.split()[text.split().index(
        dog_names.append(text.split()[text.split().index('named') + 1].strip(',')
    # Otherwise..
    else:
        dog_names.append('NaN')

# Save the result in a column 'dog_name' and append to the table
df_all['dog_names'] = dog_names
```

In [795]:
```
df_all['dog_names']
```

Out[795]:
```
0       Phineas
1         Tilly
2        Archie
3         Darla
4      Franklin
5           NaN
6           Jax
7           NaN
8          Zoey
9        Cassie
10         Koda
11        Bruno
12          NaN
13          Ted
14       Stuart
15       Oliver
16          Jim
17         Zeke
18      Ralphus
20       Gerald
21      Jeffrey
22          NaN
23       Canela
24          NaN
25          NaN
26         Maya
27       Mingus
28        Derek
29       Roscoe
31      Waffles
         ...
2326        NaN
2327        NaN
2328        NaN
2329        NaN
2330        NaN
2331        NaN
2332        NaN
2333        NaN
2334        NaN
2335        NaN
2336        NaN
2337        NaN
2338        NaN
2339        NaN
2340        NaN
2341        NaN
2342        NaN
2343        NaN
2344        NaN
2345        NaN
2346        NaN
2347        NaN
2348        NaN
```

```
2349    NaN
2350    NaN
2351    NaN
2352    NaN
2353    NaN
2354    NaN
2355    NaN
Name: dog_names, Length: 1994, dtype: object
```

In [796]:
```
#Now we can delete the 'name' column as we dont need it any more
df_all= df_all.drop(columns = 'name', axis =1)
#This will solve Issue #3 stated above
```

Test

In [797]:
```
df_all
```

Out[797]:

	tweet_id	timestamp	source	
0	892420643555336193	2017-08-01 16:23:56 +0000	Twitter for iPhone	This is Phineas. He's boy. Only ever app hole of a dc https://t.co/MgU
1	892177421306343426	2017-08-01 00:17:27 +0000	Twitter for iPhone	This is Tilly. She's jus pup on you. Hopes y ok. If not, she's availab snugs, boops, the 13/10 https://t.co/0
2	891815181378084864	2017-07-31 00:18:03 +0000	Twitter for iPhone	This is Archie. H Norwegian Pounc Lives in the tall grass. know when one may st https://t.co/wl
3	891689557279858688	2017-07-30	<a href="http://twitter.com/download/iphone"	This is Darla. She con snooze mid meal. 13/1

Define

We can convert each column to its appropriate type by the following commands.

Code

In [798]:
```
df_all['tweet_id'] = df_all['tweet_id'].astype(object)
df_all['timestamp'] = pd.to_datetime(df_all.timestamp)
df_all['source'] = df_all['source'].astype('category')
df_all['dog_stages'] = df_all['dog_stages'].astype('category')
df_all['rating_numerator'] = df_all['rating_numerator'].astype(float)
df_all['rating_denominator'] = df_all['rating_denominator'].astype(float)
# Issue #1 solved
```

Test

In [799]:
```
df_all.dtypes
```

Out[799]:
```
tweet_id              object
timestamp             datetime64[ns]
source                category
text                  object
expanded_urls         object
rating_numerator      float64
rating_denominator    float64
jpg_url               object
img_num               float64
p1                    object
p1_conf               float64
p1_dog                object
p2                    object
p2_conf               float64
p2_dog                object
p3                    object
p3_conf               float64
p3_dog                object
retweet_count         float64
```

Define

To make 'source' column easier to read, we need to remove the embedded url

Code

In [800]:
```
df_all['source'] = df_all['source'].str.replace('<a href="http://twitter.com/down
df_all['source'] = df_all['source'].str.replace('<a href="http://vine.co" rel="no
df_all['source'] = df_all['source'].str.replace('<a href="http://twitter.com" rel
df_all['source'] = df_all['source'].str.replace('<a href="https://about.twitter.c
#This will solve Issue #7 stated above
```

Test

In [801]:
```
df_all.head()
```

Out[801]:

	tweet_id	timestamp	source	text	
0	892420643555336193	2017-08-01 16:23:56	Twitter for iPhone	This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 https://t.co/MgUWQ76dJU	
1	892177421306343426	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she's available for pats, snugs, boops, the whole bit. 13/10 https://t.co/0Xxu71qeIV	
2	891815181378084864	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall grass. You never know when one may strike. 12/10 https://t.co/wUnZnhtVJB	
3	891689557279858688	2017-07-30 15:58:51	Twitter for iPhone	This is Darla. She commenced a snooze mid meal. 13/10 happens to the best of us https://t.co/tD36da7qLQ	
4	891327558926688256	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. He would like you to stop calling him "cute." He is a very fierce shark and should be respected as such. 12/10 #BarkWeek https://t.co/AtUZn91f7f	https://twitter.com/dog_rates/status

5 rows × 22 columns

Define

We can rename the columns to make them more more specific and detailed

Code

In [802]:
```
df_all = df_all.rename(columns = {'timestamp': 'tweet_date', 'source': 'tweet_sou
#This will solve Issue #6 stated above
```

Test

In [803]:
```
df_all.head()
```

Out[803]:

	tweet_id	tweet_date	tweet_source	tweet_content	
0	892420643555336193	2017-08-01 16:23:56	Twitter for iPhone	This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 https://t.co/MgUWQ76dJU	
1	892177421306343426	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she's available for pats, snugs, boops, the whole bit. 13/10 https://t.co/0Xxu71qeIV	
2	891815181378084864	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall grass. You never know when one may strike. 12/10 https://t.co/wUnZnhtVJB	
3	891689557279858688	2017-07-30 15:58:51	Twitter for iPhone	This is Darla. She commenced a snooze mid meal. 13/10 happens to the best of us https://t.co/tD36da7qLQ	
4	891327558926688256	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. He would like you to stop calling him "cute." He is a very fierce shark and should be respected as such. 12/10 #BarkWeek https://t.co/AtUZn91f7f	https://twitter.com/dog_rates

5 rows × 22 columns

Define

In this section, we are removing extra characters in the 'tweet_content' column

Code

In [804]:
```
# View rows which contain '&amp;' instead of '&' in 'tweet_content' column
df_all[df_all.tweet_content.str.contains('&amp;')]
```

Out[804]:

	tweet_id	tweet_date	tweet_source	tweet_content	
262	842765311967449089	2017-03-17 15:51:22	Twitter for iPhone	Meet Indie. She's not a fan of baths but she's definitely a fan of hide & seek. 12/10 click the link to help Indie\n\nhttps://t.co/fvGkIuAIFK https://t.co/kiCFtmJd7I	
320	834458053273591808	2017-02-22 17:41:18	Twitter for iPhone	Meet Chester (bottom) & Harold (top). They are different dogs not only in appearance, but in personality as well. Both 12/10 symbiotic af https://t.co/8ZOZS2FSJe	
461	817536400337801217	2017-01-07 01:00:41	Twitter for iPhone	Say hello to Eugene & Patti Melt. No matter how dysfunctional they get, they will never top their owners. Both 12/10 would pet at same time	https://twitter.cc

In [805]:
```
df_all.tweet_content.replace("&amp;","&")
# Fixed Issue #5
```

Out[805]:
```
0       This is Phineas. He's a mystical boy. Only ever appears in the hole
of a donut. 13/10 https://t.co/MgUWQ76dJU (https://t.co/MgUWQ76dJU)

1       This is Tilly. She's just checking pup on you. Hopes you're doing o
k. If not, she's available for pats, snugs, boops, the whole bit. 13/10 http
s://t.co/0Xxu71qeIV (https://t.co/0Xxu71qeIV)
2       This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the
 tall grass. You never know when one may strike. 12/10 https://t.co/wUnZnhtV
JB (https://t.co/wUnZnhtVJB)
3       This is Darla. She commenced a snooze mid meal. 13/10 happens to the
 best of us https://t.co/tD36da7qLQ (https://t.co/tD36da7qLQ)

4       This is Franklin. He would like you to stop calling him "cute." He i
s a very fierce shark and should be respected as such. 12/10 #BarkWeek http
s://t.co/AtUZn91f7f (https://t.co/AtUZn91f7f)
5       Here we have a majestic great white breaching off South Africa's coa
st. Absolutely h*ckin breathtaking. 13/10 (IG: tucker_marlo) #BarkWeek http
s://t.co/kQ04fDDRmh (https://t.co/kQ04fDDRmh)
6       Meet Jax. He enjoys ice cream so much he gets nervous around it. 13/
```

Test

In [806]:
```
df_all.head(5)
```

Out[806]:

	tweet_id	tweet_date	tweet_source	tweet_content	
0	892420643555336193	2017-08-01 16:23:56	Twitter for iPhone	This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 https://t.co/MgUWQ76dJU	
1	892177421306343426	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she's available for pats, snugs, boops, the whole bit. 13/10 https://t.co/0Xxu71qeIV	
2	891815181378084864	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall grass. You never know when one may strike. 12/10 https://t.co/wUnZnhtVJB	
3	891689557279858688	2017-07-30 15:58:51	Twitter for iPhone	This is Darla. She commenced a snooze mid meal. 13/10 happens to the best of us https://t.co/tD36da7qLQ	
4	891327558926688256	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. He would like you to stop calling him "cute." He is a very fierce shark and should be respected as such. 12/10 #BarkWeek https://t.co/AtUZn91f7f	https://twitter.com/dog_rates

5 rows × 22 columns

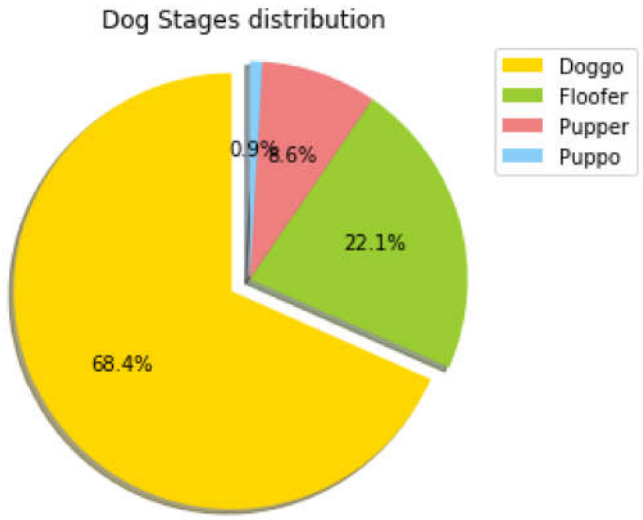
In [807]:
```
# Store the clean DataFrame in a CSV file
df_all.to_csv('twitter_archive_master.csv', index=False, encoding = 'utf-8')
```

6. Analyzing and visualizing data

In [808]:
```
df_all = pd.read_csv('twitter_archive_master.csv')
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1994 entries, 0 to 1993
Data columns (total 22 columns):
tweet_id                    1994 non-null int64
tweet_date                  1994 non-null object
tweet_source                1994 non-null object
tweet_content               1994 non-null object
tweet_url                   1994 non-null object
rating_numerator            1994 non-null float64
rating_denominator          1994 non-null float64
tweet_picture_predicted     1994 non-null object
img_num                     1994 non-null float64
p1                          1994 non-null object
p1_conf                     1994 non-null float64
p1_dog                      1994 non-null bool
p2                          1994 non-null object
p2_conf                     1994 non-null float64
p2_dog                      1994 non-null bool
p3                          1994 non-null object
p3_conf                     1994 non-null float64
p3_dog                      1994 non-null bool
retweet_count               1452 non-null float64
farvorite_count             1452 non-null float64
dog_stages                  326 non-null object
dog_names                   1369 non-null object
dtypes: bool(3), float64(8), int64(1), object(10)
memory usage: 301.9+ KB
```

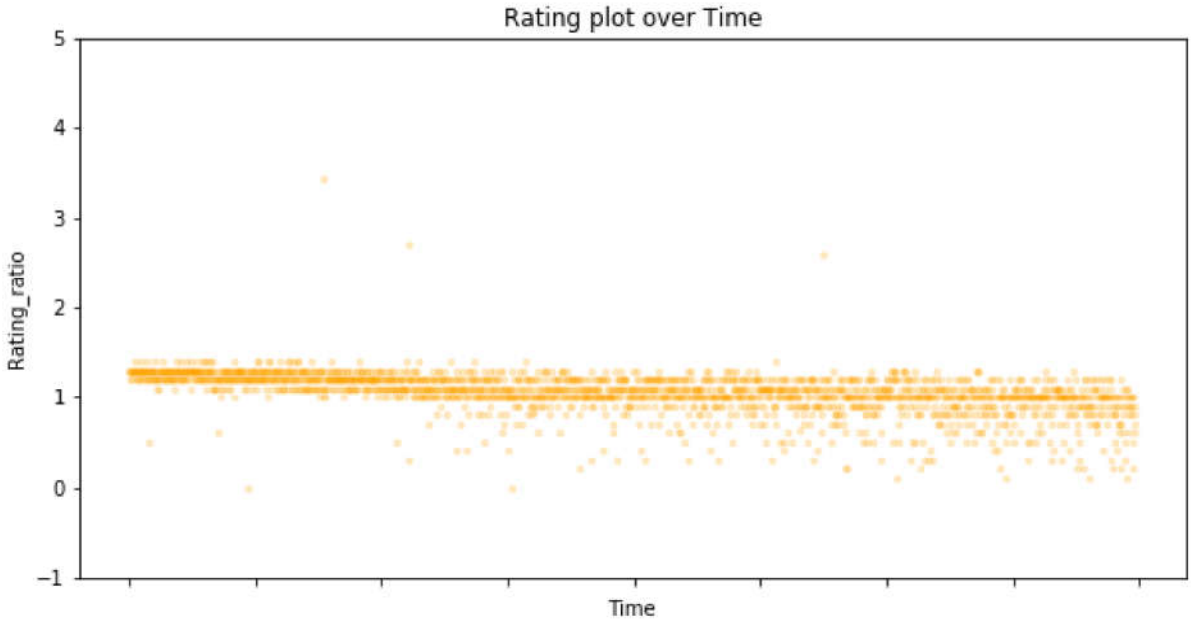
In [809]:
```
# Creating the dog_stages pie chart
labels = ['Doggo', 'Floofer', 'Pupper', 'Puppo']
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0)  # only "explode" the 1st slice (i.e. 'Doggo')
plt.pie(df_all['dog_stages'].value_counts(),explode = explode, autopct='%1.1f%%',
plt.title('Dog Stages distribution')
plt.legend(labels)
plt.tight_layout()

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')
plt.show()
```



One can not deny that 'doggo' is the most popular dog stage among the survey. It accounted for nearly 70% of all stage which is two times higher that of the rest's combination.

In [810]:
```
#Set timestamp as an index
df_all.set_index('tweet_date', inplace = True)
#calculation the rationg_ratio then plot it
df_all['rating_ratio'] = df_all['rating_numerator']/df_all['rating_denominator']
df_all['rating_ratio'].plot(figsize=(10, 5),style = '.', alpha = .2, color = 'ora
plt.title('Rating Ratio over Time')
plt.xlabel('Time')
plt.ylabel('Rating_ratio')
#Set our range of rating ratio to easier read
plt.ylim(-1,5)
```

Out[810]: Text(0,0.5,'Rating_ratio')



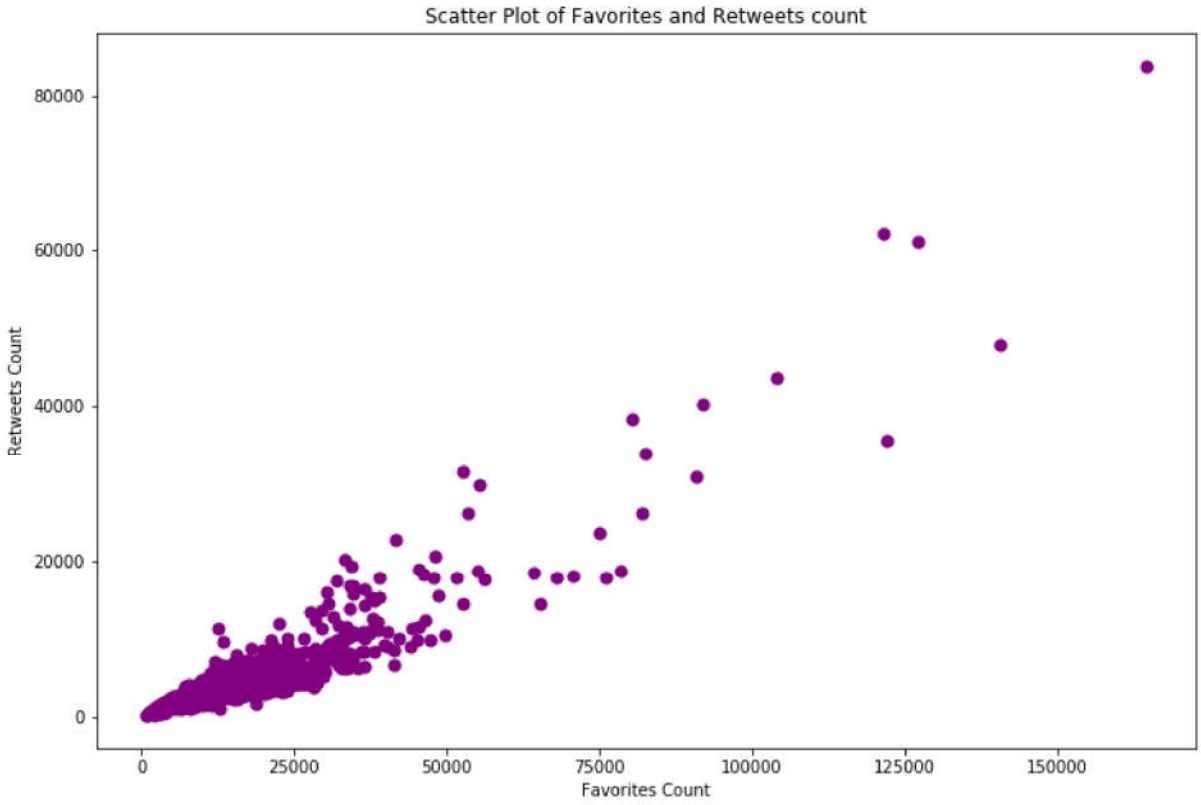
In [814]:
```
df_all[df_all['rating_ratio'] >=1]['rating_ratio'].describe()
```

Out[814]:
```
count    1579.000000
mean        1.286972
std         4.562160
min         1.000000
25%         1.000000
50%         1.100000
75%         1.200000
max       177.600000
Name: rating_ratio, dtype: float64
```

More than 75% of the data has more than 12/10 as rating More than 50% of the data has more than 11/10 as rating Rating is trending down over time. More ever, the rating range seems more diversity than that of the past

In [812]:
```
# Scatter plot of favorites and retweets count.
df_all.plot(kind = 'scatter', x = 'farvorite_count', y = 'retweet_count', s=50, f
plt.xlabel('Favorites Count')
plt.ylabel('Retweets Count')
plt.title('Scatter Plot of Favorites and Retweets count')
```

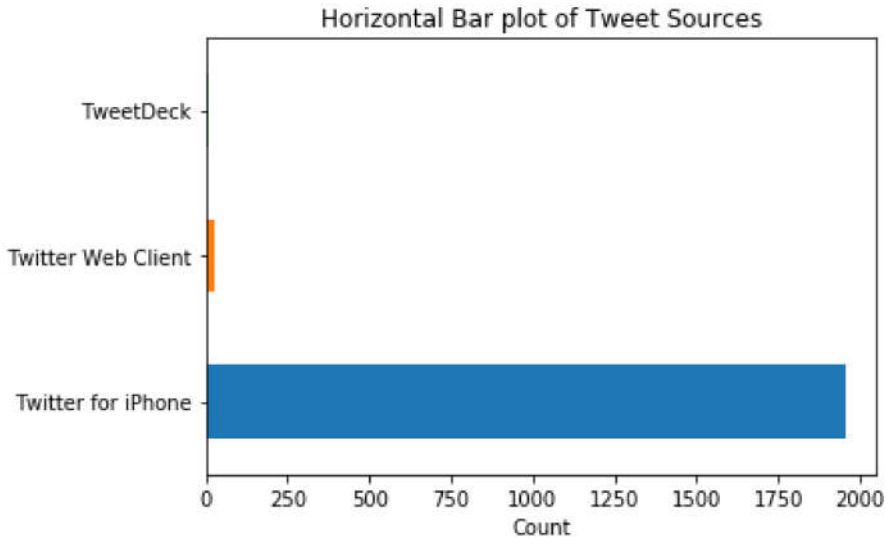
Out[812]:
```
Text(0.5,1,'Scatter Plot of Favorites and Retweets count')
```



Not supprisingly, a correlation map illustrates that if the number of the retweet is high then the number of farvorite will be high as well.

In [813]:
```
df_all['tweet_source'].value_counts().plot(kind = 'barh')
plt.xlabel('Count')
plt.title('Horizontal Bar plot of Tweet Sources')
```

Out[813]:
```
Text(0.5,1,'Horizontal Bar plot of Tweet Sources')
```



The number of tweets submitted from Iphone is ridiculously higher than that of all other source, indicating that people are tent to tweet by their Iphone instead of other means when using Twitter.

Conclusion:

The Twitter account WeRateDog is a good source to collect information regarding user's beloved dogs' stages, rating, favorite and retweets counted over time. However, we need to put effort in cleaning data retrive from Twitter API as the raw data is having a lot of problems making it difficult to read and research. If you want a specific row of data to analyze, my adivise is to validate its own and related columns data before using them for visualization