In [4]:

```
# importing pandas! it is necessary to deal Dataframes!
import pandas as pd

# importing numpy! It does pretty good vectorized calculations!
import numpy as np

# importing Beautiful Soup 4! It is very useful for accessing contents within HTML
import bs4

# to download data from internet through an URL!
import requests

# API to download content from twitter
import tweepy


# to deal data in JSON format!
import json

#import %matplotlib to create visualizations!

import matplotlib.pyplot as plt
%matplotlib inline
```

Let us begin with the Gathering of the Data.

In [2]:

```
# reading the csv file! it is indeed very simple, since it is comma-separated!
twar=pd.read_csv('twitter-archive-enhanced.csv')

#visualizing the first 5 elements!
twar.head()
```

Out[2]:

	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
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter

ASSESSING the data: utilizando os métodos .info, .sample, .describe, .columns, .value_counts.

In [15]:

```
#vamos ver quais e quantas colunas nós temos!
print(twar.columns.tolist())
print('O numero de colunas eh igual a {0}'.format(len(twar.columns.tolist())))
```

```
['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timesta
mp', 'source', 'text', 'retweeted_status_id', 'retweeted_status_user_
id', 'retweeted_status_timestamp', 'expanded_urls', 'rating_numerato
r', 'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'pupp
o']
O numero de colunas eh igual a 17
```

Temos então 17 colunas com várias informações sobre o tweet de cada user_id e no total são 17 colunas.

In [8]:

```
# analisando o DF utilizando info!

twar.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
```

podemos ver várias colunas do tipo objeto e float.

In [18]:

```
# usando describe() para

twar.sample(5)
```

Out[18]:

	text	retweeted_status_id	retweeted_status_user_id	retweeted_status_timestamp	
	"Please, no parazzi" 11/10 ɔs://t.co/nJIX...	NaN	NaN	NaN	https://t
	r the last time, /E. DO. NOT. RATE. BULBASAU...	NaN	NaN	NaN	https://t
	ere we have a ell-established blockerspan...	NaN	NaN	NaN	https://t
	s is Akumi. It's is birthday. He received ...	NaN	NaN	NaN	https://t
	This is Beau. He's trying to eep his daddy fr...	NaN	NaN	NaN	https://t

Temos colunas com valores ausentes, como 'in_reply to_status_id' e 'in_reply_to_user_id' que devem ser eliminadas por não conter informações. Isso igualmente é válido para 'retweeted_status_user_id' e 'retweeted_status_timestamp' que praticamente somente apresentam valores que não são numéricos (np.NaN).

Cleaning of the Data: Fatores de Qualidade e Arrumação!

Quais são os problemas de qualidade e arrumação desse DataFrame (DF)?

Qualidade

1) timestamp do jeito que está escrito é um problema. vamos separa-la em month, day e year!

2) in_reply_to_status_id e in_reply_to_user_id não significam nada, podemos dar um drop nelas.

3) 'retweeted_status_user_id' e 'retweeted_status_timestamp' não significam nada, podemos dar um drop nelas.

Arrumação

1) a maneira que foi mostrada rating numerator e rating denominator para fazer concordância com DF posteriores, criaremos uma coluna denominada score_rate = rating_numerator/rating_denominator

Arrumando o erro de qualidade 1) com respeito à coluna timestamp!

In [20]:

```
# Fazendo uma cópia desse DF para poder manipulá-lo!

twar_cp= twar.copy()
```

Coding Codificando a mudança no DF para consertar a coluna timestamp

In [22]:

```
# criando código para colocar o DF em mes, dia  e ano!

list_month = []
list_day = []
list_year = []
for i in range(len(twar_cp['timestamp'])):
    list_month.append(twar_cp['timestamp'][i].split()[0].split('-')[2])
    list_day.append(twar_cp['timestamp'][i].split()[0].split('-')[1])
    list_year.append(twar_cp['timestamp'][i].split()[0].split('-')[0])
```

In [26]:

```
# agregando os valores guardados em uma lista ao DF!

twar_cp['month']=list_month
twar_cp['day']=list_day
twar_cp['year']=list_year
```

In [27]:

```
twar_cp['year'].head()
```

Out[27]:

```
0    2017
1    2017
2    2017
3    2017
4    2017
Name: year, dtype: object
```

In [28]:

```
# deletando a coluna timestamp
# não precisamos mais dela

twar_cp.drop(['timestamp'],axis=1,inplace=True)
```

testing

In [25]:

```
twar_cp
```

Out[25]:

	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/downloac
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/downloac
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/downloac
3	891689557279858688	NaN	NaN	2017-07-30	href="http://twitter.com/downloac

it seems to be fine!

Quality problem 2) Taking a drop in the unnecessary columns

In [30]:

```
twar_cp.drop(['in_reply_to_status_id'],axis=1,inplace=True)
```

In [31]:

```
twar_cp.drop(['in_reply_to_user_id'],axis=1,inplace=True)
```

Testing the modification

In [32]:

```
twar_cp.columns
```

Out[32]:

```
Index(['tweet_id', 'source', 'text', 'retweeted_status_id',
       'retweeted_status_user_id', 'retweeted_status_timestamp',
       'expanded_urls', 'rating_numerator', 'rating_denominator', 'na
me',
       'doggo', 'floofer', 'pupper', 'puppo', 'month', 'day', 'yea
r'],
      dtype='object')
```

Vemos que as colunas que foram excluídas realmente não estão mais presentes :)

Quality problem 3) Taking a drop in the unnecessary columns

In [33]:

```
twar_cp.drop(['retweeted_status_id'],axis=1,inplace=True)
twar_cp.drop(['retweeted_status_user_id'],axis=1,inplace=True)
```

In [34]:

```
twar_cp.drop(['retweeted_status_timestamp'],axis=1,inplace=True)
```

Testing the code

In [36]:

```
twar_cp.columns
```

Out[36]:

```
Index(['tweet_id', 'source', 'text', 'expanded_urls', 'rating_numerat
or',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'p
uppo',
       'month', 'day', 'year'],
      dtype='object')
```

Vemos que as colunas que foram excluídas realmente não estão mais presentes :)

Arrumação: Lidando com as colunas rating_numerator e rating_denominator

In [40]:

```
# criando a variável score
# devemos defini-la, creio eu, como o quociente de rating_numerator e
# rating_denominator

list_score=[]

#print(twar_cp['rating_numerator'])

for i in range(len(twar_cp['rating_numerator'])):
    list_score.append(float(twar_cp['rating_numerator'][i]/twar_cp['rating_denomina
```

```
/home/vagner/anaconda3/lib/python3.6/site-packages/ipykernel_launche
r.py:10: RuntimeWarning: divide by zero encountered in long_scalars
  # Remove the CWD from sys.path while we load stuff.
```

In [41]:

```
twar_cp['score']=list_score
```

In [42]:

```
# dropping the unnecessary columns!

twar_cp.drop(['rating_numerator'],axis=1, inplace=True)
twar_cp.drop(['rating_denominator'],axis=1,inplace=True)
```

Testing the code!

In [43]:

```
#limpeza básica do primeiro DF
twar_cp.head(2)
```

Out[43]:

	tweet_id	source	text	
0	892420643555336193	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rate
1	892177421306343426	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_rate

In [44]:

```
# usando rename para haver compatabilidade com outros DFs

twar_cp.rename(columns={'score':'rate_score'}, inplace=True)
```

In [46]:

```
twar_cp.sample(5)
```

Out[46]:

ce	text	expanded_urls	name	doggo	floofer	pupper	puppo
<a ıe" r...	This is Loki. He smiles like Elvis. Ain't noth...	https://twitter.com/dog_rates/status/826958653...	Loki	doggo	None	None	None
<a ıe" r...	Really guys? Again? I know this is a rare Alba...	https://twitter.com/dog_rates/status/703425003...	None	None	None	None	None
<a ıe" r...	Please only send in dogs. We only rate dogs, n...	https://twitter.com/dog_rates/status/809920764...	None	None	None	None	None
<a ıe" r...	This is Philbert. His toilet broke and he does...	https://twitter.com/dog_rates/status/767754930...	Philbert	None	None	None	None
<a ıe" r...	This is Louis. He's a river dancer. His friend...	https://twitter.com/dog_rates/status/679132435...	Louis	None	None	None	None

Parece que tudo está funcionando muito bem :) amazing!

Agora lidando com o segundo DF que deve ser baixado programaticamente

Gathering!

In [48]:

```
# Downloading programatically using requests!
import io
url_imagepred='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_im

data_url = requests.get(url_imagepred).content

'''
!!! abaixo eu poderia eu poderia ter colocado sep='\t', mas eu esqueci inicialmente
Acabei trabalhando com uma maneira muito mais complicada de ler o arquivo e usar co
DF, mas deu certo! Pelo menos, fiquei orgulhoso de ainda ter dado certo! :)
'''

rawData = pd.read_csv(io.StringIO(data_url.decode('utf-8')))
```

Visualizando como os dados parecem estar organizados!

In [49]:

```
rawData['tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_dog\tp3\t
```

Out[49]:

```
0       666020888022790149\thttps://pbs.twimg.com/medi...
1       666029285002620928\thttps://pbs.twimg.com/medi...
2       666033412701032449\thttps://pbs.twimg.com/medi...
3       666044226329800704\thttps://pbs.twimg.com/medi...
4       666049248165822465\thttps://pbs.twimg.com/medi...
5       666050758794694657\thttps://pbs.twimg.com/medi...
6       666051853826850816\thttps://pbs.twimg.com/medi...
7       666055525042405380\thttps://pbs.twimg.com/medi...
8       666057090499244032\thttps://pbs.twimg.com/medi...
9       666058600524156928\thttps://pbs.twimg.com/medi...
10      666063827256086533\thttps://pbs.twimg.com/medi...
11      666071193221509120\thttps://pbs.twimg.com/medi...
12      666073100786774016\thttps://pbs.twimg.com/medi...
13      666082916733198337\thttps://pbs.twimg.com/medi...
14      666094000022159362\thttps://pbs.twimg.com/medi...
15      666099513787052032\thttps://pbs.twimg.com/medi...
16      666102155909144576\thttps://pbs.twimg.com/medi...
17      666104133288665088\thttps://pbs.twimg.com/medi...
18      666268910803644416\thttps://pbs.twimg.com/medi...
19      666273097616637952\thttps://pbs.twimg.com/medi...
20      666287406224695296\thttps://pbs.twimg.com/medi...
21      666293911632134144\thttps://pbs.twimg.com/medi...
22      666337882303524864\thttps://pbs.twimg.com/medi...
23      666345417576210432\thttps://pbs.twimg.com/medi...
24      666353288456101888\thttps://pbs.twimg.com/medi...
25      666362758909284353\thttps://pbs.twimg.com/medi...
26      666373753744588802\thttps://pbs.twimg.com/medi...
27      666396247373291520\thttps://pbs.twimg.com/medi...
28      666407126856765440\thttps://pbs.twimg.com/medi...
29      666411507551481857\thttps://pbs.twimg.com/medi...
                              ...
2045    886366144734445568\thttps://pbs.twimg.com/medi...
2046    886680336477933568\thttps://pbs.twimg.com/medi...
2047    886736880519319552\thttps://pbs.twimg.com/medi...
2048    886983233522544640\thttps://pbs.twimg.com/medi...
2049    887101392804085760\thttps://pbs.twimg.com/medi...
2050    887343217045368832\thttps://pbs.twimg.com/ext_...
2051    887473957103951883\thttps://pbs.twimg.com/medi...
2052    887517139158093824\thttps://pbs.twimg.com/ext_...
2053    887705289381826560\thttps://pbs.twimg.com/medi...
2054    888078434458587136\thttps://pbs.twimg.com/medi...
2055    888202515573088257\thttps://pbs.twimg.com/medi...
2056    888554962724278272\thttps://pbs.twimg.com/medi...
2057    888804989199671297\thttps://pbs.twimg.com/medi...
2058    889172381238312960\thttps://pbs.twimg.com/medi...
2059    889278841981685760\thttps://pbs.twimg.com/ext_...
2060    889531135344209921\thttps://pbs.twimg.com/medi...
2061    889638837579907072\thttps://pbs.twimg.com/medi...
2062    889665388333682689\thttps://pbs.twimg.com/medi...
```

```
2063    889880896479866881\thttps://pbs.twimg.com/medi...
2064    890006608113172480\thttps://pbs.twimg.com/medi...
2065    890240255349198849\thttps://pbs.twimg.com/medi...
2066    890609185150312448\thttps://pbs.twimg.com/medi...
2067    890729181411237888\thttps://pbs.twimg.com/medi...
2068    890971913173991426\thttps://pbs.twimg.com/medi...
2069    891087950875897856\thttps://pbs.twimg.com/medi...
2070    891327558926688256\thttps://pbs.twimg.com/medi...
2071    891689557279858688\thttps://pbs.twimg.com/medi...
2072    891815181378084864\thttps://pbs.twimg.com/medi...
2073    892177421306343426\thttps://pbs.twimg.com/medi...
2074    892420643555336193\thttps://pbs.twimg.com/medi...
Name: tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp
2_dog\tp3\tp3_conf\tp3_dog, Length: 2075, dtype: object
```

what a messy data! the \t separator is still there!

We need to deal this \t separator in the list by using a raw string

In [51]:

```
# remember to convert to a raw string
colnames_split = r"""tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\t
print(colnames_split.split('\\'))
print(len(colnames_split.split('\\')))
```

```
['tweet_id', 'tjpg_url', 'timg_num', 'tp1', 'tp1_conf', 'tp1_dog', 't
p2', 'tp2_conf', 'tp2_dog', 'tp3', 'tp3_conf', 'tp3_dog']
12
```

In [55]:

```
# let us take a look at what should be dataframe column names -> all mixed
all_mixed = 'tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_dog\t
print(("%r"%rawData[all_mixed][0]).split('\\'))
print(len(("%r"%rawData[all_mixed][0]).split('\\')))
```

```
["'666020888022790149", 'thttps://pbs.twimg.com/media/CT4udn0WwAA0aM
y.jpg', 't1', 'tWelsh_springer_spaniel', 't0.465074', 'tTrue', 'tcoll
ie', 't0.156665', 'tTrue', 'tShetland_sheepdog', 't0.0614285', "tTru
e'"]
12
```

In [57]:

```
colnames_split = r"""tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\t
colnam = colnames_split.split('\\t')

df_requests=pd.DataFrame({colnam[0]:[],colnam[1]:[],colnam[2]:[],colnam[3]:[],colna
#for i in range(len(colnames_split.split('\\'))):
len_to_iter = len(rawData['tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_
for i in range(len_to_iter):
    all_mixed = 'tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_d
    raw_splitted = ("%r"%rawData[all_mixed][i]).split('\\t')
#     for j in range(len(raw_splitted)):
#         print(i,j,len(raw_splitted))
#         df_requests.append(pd.DataFrame({colnam[j]:list(raw_splitted[j])},index=[i
    df_requests = df_requests.append(pd.Series([raw_splitted[k] for k in range(len(
#     print(df_requests)
```

Now let us see how does the tab-separated file imported in a non-common way looks!

In [59]:

```
# raw DataFrame worked!


df_requests.head(10)
```

Out[59]:

	tweet_id	jpg_url	img_num	
0	'666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_spr
1	'666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	'666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Germ
3	'666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesi
4	'666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniat
5	'666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_n
6	'666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	
7	'666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	
8	'666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	s
9	'666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	mini

In a hard way it has worked! hahahha

Assessing the data! Let us use functions like .sample(), .info(), .describe(),.value_counts()

Let us see the name of the dog races predicted make sense!

In [61]:

```
df_requests['p1'].value_counts()
```

Out[61]:

```
golden_retriever             150
Labrador_retriever           100
Pembroke                      89
Chihuahua                     83
pug                           57
chow                          44
Samoyed                       43
toy_poodle                    39
Pomeranian                    38
malamute                      30
cocker_spaniel                30
French_bulldog                26
miniature_pinscher            23
Chesapeake_Bay_retriever      23
seat_belt                     22
Staffordshire_bullterrier     20
German_shepherd               20
Siberian_husky                20
Cardigan                      19
web_site                      19
Eskimo_dog                    18
Maltese_dog                   18
teddy                         18
beagle                        18
Shetland_sheepdog             18
Rottweiler                    17
Shih-Tzu                      17
Lakeland_terrier              17
Italian_greyhound             16
kuvasz                        16
                             ...
rain_barrel                    1
standard_schnauzer             1
toilet_seat                    1
sandbar                        1
ice_lolly                      1
bighorn                        1
pool_table                     1
EntleBucher                    1
jersey                         1
rapeseed                       1
sea_urchin                     1
coffee_mug                     1
convertible                    1
mud_turtle                     1
snowmobile                     1
walking_stick                  1
alp                            1
flamingo                       1
slug                           1
trombone                       1
```

```
tiger_shark                    1
shopping_basket                1
silky_terrier                  1
panpipe                        1
dining_table                   1
Egyptian_cat                   1
three-toed_sloth               1
cup                            1
stove                          1
china_cabinet                  1
Name: p1, Length: 378, dtype: int64
```

It turns out that there are names that does not make sense, such as web_site,limousine, cup, etc!

In [62]:

```
df_requests.sample(5)
```

Out[62]:

	tweet_id	jpg_url	img_num	
379	'673270968295534593	https://pbs.twimg.com/media/CVfwXuWWIAAqnoi.jpg	1	
901	'700002074055016451	https://pbs.twimg.com/media/CbboKP4WIAAw8xq.jpg	1	
547	'677331501395156992	https://pbs.twimg.com/media/CWZdaGxXAAAjGjb.jpg	1	
1586	'797971864723324932	https://pbs.twimg.com/media/CxL3IWeVEAAAIE2.jpg	1	America
1141	'729838605770891264	https://pbs.twimg.com/ext_tw_video_thumb/72983...	1	

We can notice two potential problems: 'tweet_id' column as a string, but should be a string, and True' and False' values for p3_dog while it should be the common boolean values True and False!

Erros de Qualidade!

Quality Issues

Em suma: nesse DF achamos três erros de qualidade que são:

1) na coluna p3_dog os valores True e False estão como True',True" e False';

2) na coluna tweet_id os valores devem ser inteiros, não strings!

3) tem raça classificada (coluna 'p1') como "website, limousine, fountain, revolver,military_uniform,seatbelt, etc".

Quality Issue 1: True', True" and False"

Coding to repair this!

In [63]:

```
# visualizing the problem

df_requests['p3_dog'][0]
```

Out[63]:

```
"True'"
```

In [64]:

```
# the for loop code to repair these typos!

list_p3_dog = []

for i in range(len(df_requests['p3_dog'])):
    if(df_requests['p3_dog'][i]=="True'" or df_requests['p3_dog'][i]=='True"'):
       list_p3_dog.append('True')
    elif(df_requests['p3_dog'][i]=="False'"):
       list_p3_dog.append('False')
    else:
         print(df_requests['p3_dog'][i])
```

In [65]:

```
df_requests['p3_dog_new']=list_p3_dog
```

Testing

In [66]:

```
df_requests.head(2)
```

Out[66]:

p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3
paniel	0.465074	True	collie	0.156665	True	Shetland_sheepdog
dbone	0.506826	True	miniature_pinscher	0.07419169999999999	True	Rhodesian_ridgeback

In [67]:

```
df_requests.drop(['p3_dog'],axis=1,inplace=True)
```

In [68]:

```
df_requests.head(3)
```

Out[68]:

p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	
ıniel	0.465074	True	collie	0.156665	True	Shetland_sheepdog	
one	0.506826	True	miniature_pinscher	0.07419169999999999	True	Rhodesian_ridgeback	
ıerd	0.596461	True	malinois	0.13858399999999998	True	bloodhound	0.

Let us rename p3_dog_new to have the old name p3_dog

In [80]:

```
df_requests.rename(columns={'p3_dog_new':'p3_dog'}, inplace=True)
```

In [81]:

```
df_requests.head(3)
```

Out[81]:

	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p
r_spaniel	0.465074	True	collie	0.156665	True	Shetland_sheepdc	
redbone	0.506826	True	miniature_pinscher	0.07419169999999999	True	Rhodesian_ridgebac	
shepherd	0.596461	True	malinois	0.13858399999999998	True	bloodhoun	

It seems OK!

Quality Problem 2: 'tweet_id' are strings

Coding

In [82]:

```
# testing! it must be an integer, not a string!

df_requests['tweet_id'][0][1:]
```

Out[82]:

```
'666020888022790149'
```

In [83]:

```
# converting these string values to int manuaally

list_id=[]

for i in range(len(df_requests['tweet_id'])):
    list_id.append(int(df_requests['tweet_id'][i][1:]))
```

In [84]:

```
list_id
```

Out[84]:

```
[666020888022790149,
 666029285002620928,
 666033412701032449,
 666044226329800704,
 666049248165822465,
 666050758794694657,
 666051853826850816,
 666055525042405380,
 666057090499244032,
 666058600524156928,
 666063827256086533,
 666071193221509120,
 666073100786774016,
 666082916733198337,
 666094000022159362,
 666099513787052032,
 666102155909144576,
```

In [85]:

```
df_requests['tweet_id_new'] = list_id
```

In [86]:

```
df_requests.head(3)
```

Out[86]:

	tweet_id	jpg_url	img_num	
0	'666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_spri
1	'666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	'666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Germa

In [87]:

```
# dropping the old column!

df_requests.drop('tweet_id',axis=1,inplace=True)
```

In [88]:

```
# renaming it

df_requests.rename({'tweet_id_new':'tweet_id'},axis=1,inplace=True)
```

Testing

In [94]:

```
df_requests.head(3)
```

Out[94]:

	jpg_url	img_num	p1	p1_conf
0	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074
1	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826
2	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.596461

In [90]:

```
type(df_requests['tweet_id'][0])
```

Out[90]:

```
numpy.int64
```

We can see that now it works fine! :)

Quality Problem 3) dogs classified as "website, limousine, fountain, revolver,military_uniform,seatbelt, etc"

coding

In [122]:

```
# making a function to do that!

def deleting_wrongp1(df,word):
    """
    inputs:
    outputs:
    """
    pos_list = df.index[df_requests['p1']==word].tolist()
    for i in range(len(pos_list)):
        df=df.drop(index = pos_list[i])
#         print(1)
    return df
#df_requests.index[df_requests['p1']=='web_site'].tolist()

df_requests = deleting_wrongp1(df_requests,'web_site')
df_requests = deleting_wrongp1(df_requests,'limousine')
df_requests = deleting_wrongp1(df_requests,'fountain')
df_requests = deleting_wrongp1(df_requests,'revolver')
df_requests = deleting_wrongp1(df_requests,'military_uniform')
df_requests = deleting_wrongp1(df_requests,'seatbelt')
df_requests = deleting_wrongp1(df_requests,'cup')
df_requests = deleting_wrongp1(df_requests,'coffee_mug')
```

Testing

In [123]:

```
df_requests.p1.value_counts()
```

Out[123]:

```
golden_retriever             150
Labrador_retriever           100
Pembroke                      89
Chihuahua                     83
pug                           57
chow                          44
Samoyed                       43
toy_poodle                    39
Pomeranian                    38
malamute                      30
cocker_spaniel                30
French_bulldog                26
Chesapeake_Bay_retriever      23
miniature_pinscher            23
seat_belt                     22
Staffordshire_bullterrier     20
German_shepherd               20
Siberian_husky                20
Cardigan                      19
Shetland_sheepdog             18
Maltese_dog                   18
Eskimo_dog                    18
beagle                        18
teddy                         18
Rottweiler                    17
Shih-Tzu                      17
Lakeland_terrier              17
kuvasz                        16
Italian_greyhound             16
Great_Pyrenees                14
                            ...
bearskin                       1
bee_eater                      1
rain_barrel                    1
alp                            1
snowmobile                     1
standard_schnauzer             1
sandbar                        1
ice_lolly                      1
bighorn                        1
pool_table                     1
EntleBucher                    1
jersey                         1
rapeseed                       1
sea_urchin                     1
convertible                    1
mud_turtle                     1
toilet_seat                    1
stove                          1
walking_stick                  1
flamingo                       1
slug                           1
```

```
trombone                        1
tiger_shark                     1
shopping_basket                 1
silky_terrier                   1
panpipe                         1
dining_table                    1
Egyptian_cat                    1
three-toed_sloth                1
china_cabinet                   1
Name: p1, Length: 371, dtype: int64
```

The values are no longer present in the column p1! :)

Analisando agora o terceiro DataFrame! Baixando os dados com tweepy!

Gathering

In [12]:

```
# dealing tweepy API!!!
# Essa é a parte mais interessante do projeto!


import tweepy
"""
As chaves abaixo são específicas de cada usuário desenvolvedor no Twitter e
não devem ser colocadas no envio do projeto!
"""
consumer_key = '???'
consumer_secret = '???'
access_token = '???'
access_secret = '???'

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

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

In [13]:

```
api
```

Out[13]:

```
<tweepy.api.API at 0x7fb707c02668>
```

Taking a look at how the JSON file looks!

In [14]:

```
twar_id = list(twar.tweet_id)
#print((api.get_status(twar_id[0],tweet_mode='extended')).text)
print(api.get_status(twar_id[0],tweet_mode='extended')._json)
```


```
{'created_at': 'Tue Aug 01 16:23:56 +0000 2017', 'id': 89242064355533
6193, 'id_str': '892420643555336193', 'full_text': "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",) 'truncated': Fa
lse, 'display_text_range': [0, 85], 'entities': {'hashtags': [], 'sym
bols': [], 'user_mentions': [], 'urls': [], 'media': [{'id': 89242063
9486877696, 'id_str': '892420639486877696', 'indices': [86, 109], 'me
dia_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg', 'media_ur
l_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg', 'url':
 'https://t.co/MgUWQ76dJU', 'display_url': 'pic.twitter.com/MgUWQ76dJ
U', 'expanded_url': 'https://twitter.com/dog_rates/status/89242064355
5336193/photo/1', 'type': 'photo', 'sizes': {'thumb': {'w': 150, 'h':
 150, 'resize': 'crop'}, 'medium': {'w': 540, 'h': 528, 'resize': 'fi
t'}, 'small': {'w': 540, 'h': 528, 'resize': 'fit'}, 'large': {'w': 5
40, 'h': 528, 'resize': 'fit'}}}]}, 'extended_entities': {'media':
 [{'id': 892420639486877696, 'id_str': '892420639486877696', 'indice
s': [86, 109], 'media_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIA
UK.jpg', 'media_url_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAI
AUK.jpg', 'url': 'https://t.co/MgUWQ76dJU', 'display_url': 'pic.twitt
er.com/MgUWQ76dJU', 'expanded_url': 'https://twitter.com/dog_rates/st
atus/892420643555336193/photo/1', 'type': 'photo', 'sizes': {'thumb':
 {'w': 150, 'h': 150, 'resize': 'crop'}, 'medium': {'w': 540, 'h': 52
8, 'resize': 'fit'}, 'small': {'w': 540, 'h': 528, 'resize': 'fit'},
 'large': {'w': 540, 'h': 528, 'resize': 'fit'}}}]}, 'source': '<a hr
ef="http://twitter.com/download/iphone" rel="nofollow">Twitter for iP
hone</a>', 'in_reply_to_status_id': None, 'in_reply_to_status_id_st
r': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': Non
e, 'in_reply_to_screen_name': None, 'user': {'id': 4196983835, 'id_st
r': '4196983835', 'name': 'WeRateDogs™', 'screen_name': 'dog_rates',
 'location': 'DM YOUR DOGS', 'description': 'Your Only Source For Pro
fessional Dog Ratings    IG, FB, Snapchat ⇨ WeRateDogs partnerships@
weratedogs.com', 'url': 'https://t.co/N7sNNHAEXS', 'entities': {'ur
l': {'urls': [{'url': 'https://t.co/N7sNNHAEXS', 'expanded_url': 'htt
p://weratedogs.com', 'display_url': 'weratedogs.com', 'indices': [0,
 23]}]}, 'description': {'urls': []}}, 'protected': False, 'followers
_count': 7624742, 'friends_count': 12, 'listed_count': 5759, 'created
_at': 'Sun Nov 15 21:41:29 +0000 2015', 'favourites_count': 140897,
 'utc_offset': None, 'time_zone': None, 'geo_enabled': True, 'verifie
d': True, 'statuses_count': 9576, 'lang': 'en', 'contributors_enable
d': False, 'is_translator': False, 'is_translation_enabled': False,
 'profile_background_color': '000000', 'profile_background_image_ur
l': 'http://abs.twimg.com/images/themes/theme1/bg.png', 'profile_back
ground_image_url_https': 'https://abs.twimg.com/images/themes/theme1/
bg.png', 'profile_background_tile': False, 'profile_image_url': 'htt
p://pbs.twimg.com/profile_images/1080268745619189760/CyqCf_dA_normal.
jpg', 'profile_image_url_https': 'https://pbs.twimg.com/profile_image
s/1080268745619189760/CyqCf_dA_normal.jpg', 'profile_banner_url': 'ht
tps://pbs.twimg.com/profile_banners/4196983835/1544368760', 'profile_
link_color': 'F5ABB5', 'profile_sidebar_border_color': '000000', 'pro
file_sidebar_fill_color': '000000', 'profile_text_color': '000000',
 'profile_use_background_image': False, 'has_extended_profile': Fals
e, 'default_profile': False, 'default_profile_image': False, 'followi
```

```
ng': True, 'follow_request_sent': False, 'notifications': False, 'tra
nslator_type': 'none'}, 'geo': None, 'coordinates': None, 'place': No
ne, 'contributors': None, 'is_quote_status': False, 'retweet_count':
 8319, 'favorite_count': 38032, 'favorited': False, 'retweeted': Fals
e, 'possibly_sensitive': False, 'possibly_sensitive_appealable': Fals
e, 'lang': 'en'}
```

In [23]:

```
# dealing the First DataFrame (csv), the 'twar'

# it takes a while to run! Therefore, it is not recommended to run it everytime!

i_want_json_change = False

if i_want_json_change:
    type(twar.tweet_id)
    twar_id = list(twar.tweet_id)

    data = dict()

    for i in twar_id:
        try:
            data[i]= api.get_status(i,tweet_mode='extended')._json
        except:
            continue

    with open('tweet_son.txt','w') as f:
        json.dump(data,f)
```

In [24]:

```
# Use timeit.default_timer instead of timeit.timeit. The former provides the best
# clock available on your platform and version of Python automatically:


from timeit import default_timer as timer

start = timer()
# ...
end = timer()
print(end - start) # Time in seconds, e.g. 5.38091952400282
```

```
4.680702113546431e-05
```

In [124]:

```
df_from_json = []

filename='tweet_son.txt'

with open(filename) as json_file:
    data = json.load(json_file)
    for key,value in data.items():
        df_from_json.append({'id':value['id'],'created_at':value['created_at'],'ful
```

In [161]:

```
#Converting list to DataFrame
df_pd_json = pd.DataFrame(df_from_json)
display(df_pd_json)
```

	created_at	favorite_count	full_text	id	retweet_count
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068
3	Sun Jul 30 15:58:51 +0000	41381	This is Darla. She commenced a snooze mid meal	891689557279858688	8462

Let us explore a little bit the data

In [162]:

```
# making a straightforward copy of the DF
df_pd_json_cp = df_pd_json.copy()
```

In [163]:

```
fulltext_to_split = df_pd_json_cp['full_text'][1000]

# Testing Code to Extract Information

# dog's name
print(fulltext_to_split.split('.')[0].split()[-1])

#dog's gender
print(fulltext_to_split.split('.')[1].split()[0])

#dog's rate
print(fulltext_to_split.split('.')[-2].split()[0])
```

```
any
Even
0/10
```

ASSESSING the data: using functions like .describe, .value_counts, .info, .sample

In [164]:

```
df_pd_json_cp.sample(5)
```

Out[164]:

	created_at	favorite_count	full_text	id	retweet_count
1702	Fri Dec 25 00:00:11 +0000 2015	4087	This pupper is patiently waiting to scare the ...	680176173301628928	1677
1870	Thu Dec 10 03:11:43 +0000 2015	833	Say hello to Maggie. She's a Western Septic Do...	674788554665512960	218
2123	Thu Nov 26 22:16:09 +0000 2015	338	This is Raphael. He is a Baskerville Conquista...	670003130994700288	96
141	Sat May 13 19:11:30 +0000 2017	0	RT @dog_rates: Say hello to Quinn. She's quite...	863471782782697472	2479
1275	Fri Mar 11 02:36:57 +0000 2016	2806	This is Cooper. He basks in the glory of rebel...	708119489313951744	1041

We have columns with tons of information! We can notice that we must work on the 'created_at' column and get important information of the column 'full_text'

In [168]:

```
df_pd_json_cp['favorite_count'].describe()
```

Out[168]:

```
count      2339.000000
mean       7964.002565
std       12333.049372
min           0.000000
25%        1372.500000
50%        3467.000000
75%        9743.000000
max      164255.000000
Name: favorite_count, dtype: float64
```

We can see that 'favorite_count' has plausible min, max, quantiles values. Many interesting graphs can be obtained from it!

In [165]:

```
df_pd_json_cp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2339 entries, 0 to 2338
Data columns (total 5 columns):
created_at        2339 non-null object
favorite_count    2339 non-null int64
full_text         2339 non-null object
id                2339 non-null int64
retweet_count     2339 non-null int64
dtypes: int64(3), object(2)
memory usage: 91.4+ KB
```

In the above we can see the info about each column and their types!

Quality and Tidyness issues to be addressed!

Podemos ver os seguintes problemas no DF acima:

Qualidade (Quality)

1) 'full_text' que deve ser desmembrado em genero, nome e em rate_score (somente nessa parte são três problemas de qualidade);

Estruturais (Tidyness)

1) novamente, não há sentido na timestamp que foi apresentada, ele teve que ser reestrurada convenientemente!

Quality Problem 1: Desmembrando as informações da coluna 'full_text' e extraindo outras colunas a partir dela!

Q1.1) Taking the name of the dog from the column 'full_text'

In [131]:

```
list_name =[]
all_to_iter = len(df_pd_json_cp['full_text'])
for i in range(all_to_iter):

    fulltext_to_split = df_pd_json_cp['full_text'][i]
    try:
        name = fulltext_to_split.split('.')[0].split()[-1]
    except:
        name = np.NaN
    list_name.append({name})
```

In [133]:

```
# taking a look at an ordinary dog!

df_pd_json_cp['full_text'][2335]
```

Out[133]:

```
'This is a purebred Piers Morgan. Loves to Netflix and chill. Always
looks like he forgot to unplug the iron. 6/10 https://t.co/DWnyCjf2m
x' (https://t.co/DWnyCjf2mx')
```

In [134]:

```
#creating the column names!

df_name =pd.DataFrame(list_name,columns=['Name'])
display(df_name)
```

	Name
0	Phineas
1	Tilly
2	Archie
3	Darla
4	Franklin
5	coast
6	Jax
7	boy
8	Zoey
9	Cassie
10	Koda
11	Bruno
12	her
13	Ted
14	Stuart
15	Oliver
16	Jim
17	Zeke
18	Ralphus
19	Gerald
20	Jeffrey
21	Wiener
22	Canela
23	today
24	This
25	Maya
26	Mingus
27	Derek
28	Roscoe
29	caution
...	...
2309	dog

	Name
2310	bumblegruff
2311	goodness
2312	jacket
2313	here
2314	Islands
2315	him
2316	Parthenon
2317	dog
2318	Episcopalian
2319	https://t
2320	computer
2321	breed
2322	my
2323	while
2324	mix
2325	sunblockerspaniel
2326	(lol)
2327	Rhododendron
2328	see
2329	poker
2330	my
2331	mix
2332	dog
2333	retriever
2334	vulpix
2335	Morgan
2336	pup
2337	terrier
2338	Setter

2339 rows × 1 columns

In [136]:

```
result1 = pd.concat([df_pd_json_cp,df_name],axis=1,sort=False)
```

Testing

In [137]:

```
result1
```

Out[137]:

	created_at	favorite_count	full_text	id	retweet_count	Nam
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phinea
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Till
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archi

We can see that name is ok now! :)

Q1.2) Now we can take a look at the gender of the dog!

coding

In [138]:

```
gender_name =[]
all_to_iter = len(df_pd_json_cp['full_text'])
for i in range(all_to_iter):
#    print(i)
    fulltext_to_split = df_pd_json_cp['full_text'][i]
    try:
        name = fulltext_to_split.split('.')[1].split()[0]
        if(name =='He' or name =='he' or name =="He's" or name =="he's"or name =='H
            gender_name.append({'male'})
        elif(name =='She' or name =='she'or name =="She's" or name =="she's"or name
            gender_name.append({'female'})
        else:
            gender_name.append({np.NaN})
    except:
        name = np.NaN
        gender_name.append({name})

#print(fulltext_to_split.split('.')[1].split()[0])
```

In [139]:

```
df_gender = pd.DataFrame(gender_name,columns=['gender'])
#display(df_gender)
```

In [155]:

```
result2 = pd.concat([result1,df_gender],axis=1,sort=False)
```

Testing

In [158]:

```
result2.head(3)
```

Out[158]:

	created_at	favorite_count	full_text	id	retweet_count	Name	gender
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phineas	male
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Tilly	female
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archie	male

it seems that now gender is ok!

Q1.3) Now we can take some information from the Rate given in the full_text

Coding

In [143]:

```
rate_score =[]
all_to_iter = len(df_pd_json_cp['full_text'])
for i in range(all_to_iter):
#    print(i)
    fulltext_to_split = df_pd_json_cp['full_text'][i]
    try:
        score1 = fulltext_to_split.split('.')[-2].split()[0].split("/")[0]
        score2 = fulltext_to_split.split('.')[-2].split()[0].split("/")[1]
        try:
            score1=int(score1)
            score2=int(score2)
            score=float(score1/score2)
        except:
            score=np.NaN
        rate_score.append({score})
    except:
        score = np.NaN
        rate_score.append({score})
```

In [144]:

```
# a creating the DataFrame column
df_rate_score = pd.DataFrame(rate_score,columns=['rate_score'])
display(df_rate_score)
```

	rate_score
0	1.3
1	1.3
2	1.2
3	1.3
4	1.2
5	1.3
6	NaN
7	1.3
8	1.3
9	1.4
10	1.3
11	1.3
12	1.3
13	1.2
14	1.3
15	1.3
16	1.2
17	1.3
18	1.3
19	1.2
20	1.3
21	NaN
22	1.3
23	1.3
24	1.2
25	1.3
26	NaN
27	NaN
28	1.2
29	NaN
...	...
2309	0.2

	rate_score
2310	0.7
2311	0.9
2312	1.1
2313	NaN
2314	0.8
2315	NaN
2316	0.9
2317	0.3
2318	0.9
2319	NaN
2320	1.0
2321	0.1
2322	1.1
2323	0.8
2324	0.9
2325	0.6
2326	1.0
2327	0.9
2328	1.0
2329	0.8
2330	0.9
2331	1.0
2332	0.2
2333	1.0
2334	0.5
2335	0.6
2336	0.9
2337	0.7
2338	0.8

2339 rows × 1 columns

In [145]:

```
# concatenating with the old result
result3 = pd.concat([result2,df_rate_score],axis=1,sort=False)
display(result3)
```

	created_at	favorite_count	full_text	id	retweet_count	Nam
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phinea
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Till
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archi
3	Sun Jul 30 15:58:51	41381	This is Darla. She commenced a snooze mid	891689557279858688	8462	Darl

In [322]:

```
result3.drop(['full_text'],axis=1,inplace=True)
```

Testing!

In [146]:

```
result3.head(5)
```

Out[146]:

	created_at	favorite_count	full_text	id	retweet_count	Name	gende
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phineas	mal
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Tilly	femal
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archie	mal
3	Sun Jul 30 15:58:51 +0000 2017	41381	This is Darla. She commenced a snooze mid meal...	891689557279858688	8462	Darla	femal
4	Sat Jul 29 16:00:24 +0000 2017	39563	This is Franklin. He would like you to stop ca...	891327558926688256	9161	Franklin	mal

We have successfully created the column rate_score!

Structural Problem: 'created_at' column

Now let us deal the column 'created_at' to make it in weekday, month and year!

In [147]:

```
list_weekday = []
list_month = []
list_day = []
list_year = []

for i in range(len(result3['created_at'])):
    splitted = result3['created_at'][i].split()
#    print(splitted)
    weekday = splitted[0]
    month = splitted[1]
    day = splitted[2]
    year = splitted[-1]

    # appending!
    list_weekday.append({weekday})
    list_month.append({month})
    list_day.append({day})
    list_year.append({year})
```

In [148]:

```
# converting to DataFrames!
df_weekday=pd.DataFrame(list_weekday,columns=['weekday'])
df_month=pd.DataFrame(list_month,columns=['month'])
df_day=pd.DataFrame(list_day,columns=['day'])
df_year=pd.DataFrame(list_year,columns=['year'])
```

In [149]:

```
# concatenating the results to the old DataFrame

result4 = pd.concat([result3,df_weekday],axis=1,sort=False)
result5 = pd.concat([result4,df_month],axis=1,sort=False)
result6 = pd.concat([result5,df_day],axis=1,sort=False)
result7 = pd.concat([result6,df_year],axis=1,sort=False)
```

In [150]:

```
result7
```

Out[150]:

	created_at	favorite_count	full_text	id	retweet_count	Nam
0	Tue Aug 01 16:23:56 +0000 2017	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phinea
1	Tue Aug 01 00:17:27 +0000 2017	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Till
2	Mon Jul 31 00:18:03 +0000 2017	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archi

In [ ]:

```
```

In [151]:

```
# let us remove the 'created_at' column!

result7.drop(['created_at'],axis=1,inplace=True)
```

Testing the code!

In [152]:

```
result7
```

Out[152]:

	favorite_count	full_text	id	retweet_count	Name	gender
0	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phineas	male
1	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Tilly	female
2	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archie	male
3	41381	This is Darla. She commenced a snooze mid meal...	891689557279858688	8462	Darla	female
4	39563	This is Franklin. He would like you to stop ca...	891327558926688256	9161	Franklin	male
5	19869	Here we have a majestic	891087950875897856	3049	coast	NaN

Quality Problem: 'Name' column must be changed to 'name'

In [153]:

```
# para haver compatibilidade com outros DF's
result7.rename(columns={'Name':'name'}, inplace=True)
```

In [154]:

```
result7
```

Out[154]:

	favorite_count	full_text	id	retweet_count	name	gender
0	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phineas	male
1	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Tilly	female
2	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archie	male
3	41381	This is Darla. She commenced a snooze mid meal...	891689557279858688	8462	Darla	female
4	39563	This is Franklin. He would like you to stop ca...	891327558926688256	9161	Franklin	male
5	19869	Here we have a majestic	891087950875897856	3049	coast	NaN

In [ ]:

```

```

Looking back again to the DataFrames we have been working on so far!

In [169]:

```
#the first DF

twar_cp.head(2)
```

Out[169]:

	tweet_id	source	text	
0	892420643555336193	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rate
1	892177421306343426	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_rate

In [616]:

```
# the second DF
df_requests.head(3)
```

Out[616]:

	jpg_url	img_num	p1	p1_conf
0	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074
1	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826
2	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.596461

In [171]:

```
# the third DF:

result7.head(5)
```

Out[171]:

	favorite_count	full_text	id	retweet_count	name	gender	rate_scor
0	38032	This is Phineas. He's a mystical boy. Only eve...	892420643555336193	8319	Phineas	male	1.
1	32653	This is Tilly. She's just checking pup on you....	892177421306343426	6143	Tilly	female	1.
2	24571	This is Archie. He is a rare Norwegian Pouncin...	891815181378084864	4068	Archie	male	1.
3	41381	This is Darla. She commenced a snooze mid meal...	891689557279858688	8462	Darla	female	1.
4	39563	This is Franklin. He would like you to stop ca...	891327558926688256	9161	Franklin	male	1.

In [ ]:

```

```

In [ ]:

```

```

Vamos fazer cópias desses DF's

In [173]:

```
firsttwar=twar_cp.copy()
```

In [174]:

```
secondreq = df_requests.copy()
```

In [175]:

```
thirdjson = result7.copy()
```

Criando os Master's DataFrames

Merge's pertinentes, para criar os DF's masters, seria fazer um merge do primeiro e segundo DF e do terceiro e segundo DF!

In [176]:

```
merge_1_2 = firsttwar.merge(secondreq,on='tweet_id',how='inner')
```

In [177]:

```
merge_1_2.head(2)
```

Out[177]:

	tweet_id	source	text	
0	892420643555336193	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rate
1	892177421306343426	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_rate

2 rows × 24 columns

In [179]:

```
# vamos renomear o nome de thirdjson de 'id' para 'tweet_id'
#thirdjson['id']
thirdjson.rename({'id':'tweet_id'},axis=1,inplace=True)
```

In [180]:

```
merge_2_3 = secondreq.merge(thirdjson,on='tweet_id',how='inner')
```

In [181]:

```
merge_2_3.head(2)
```

Out[181]:

	jpg_url	img_num	p1	p1_conf
0	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074
1	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826

2 rows × 22 columns

In [182]:

```
# visualizando as colunas do primeiro DF merged!

merge_1_2.columns
```

Out[182]:

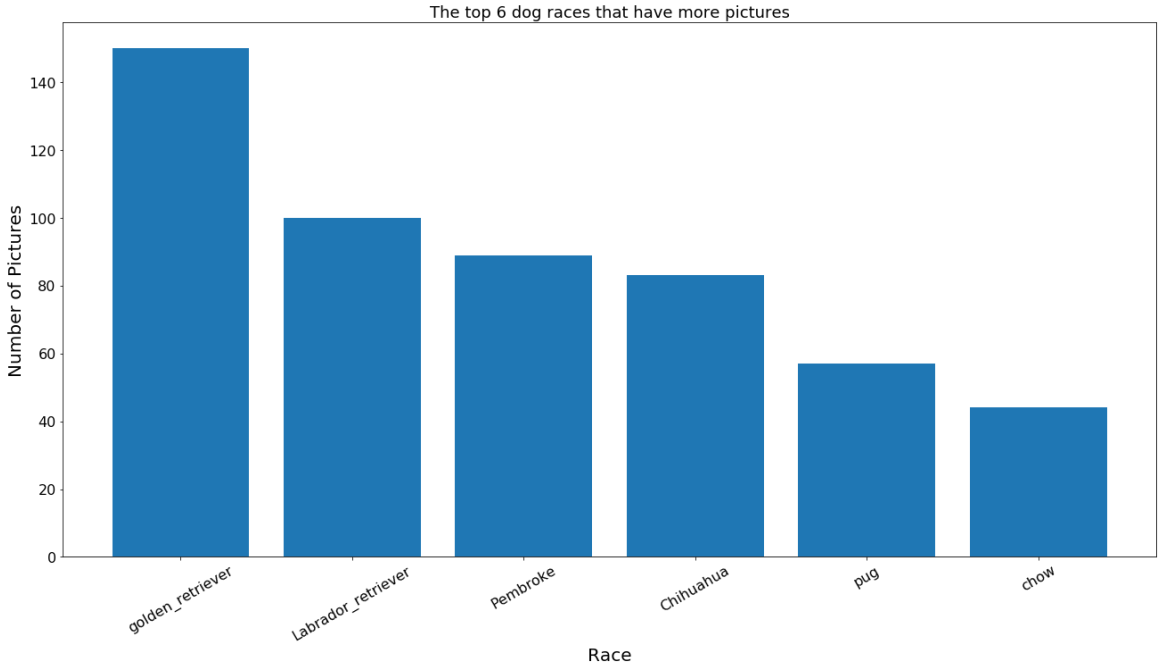
```
Index(['tweet_id', 'source', 'text', 'expanded_urls', 'name', 'dogg
o',
       'floofer', 'pupper', 'puppo', 'month', 'day', 'year', 'rate_sc
ore',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_con
f',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
      dtype='object')
```

Exploring the Master DataFrames! :)

fazendo um histograma das 6 primeiras raças que tem mais fotos!

In [184]:

```
race1=merge_1_2['p1'].value_counts().index.tolist()[0]
race2=merge_1_2['p1'].value_counts().index.tolist()[1]
race3=merge_1_2['p1'].value_counts().index.tolist()[2]
race4=merge_1_2['p1'].value_counts().index.tolist()[3]
race5=merge_1_2['p1'].value_counts().index.tolist()[4]
race6=merge_1_2['p1'].value_counts().index.tolist()[5]


top1count=merge_1_2['p1'].value_counts()[0]
top2count=merge_1_2['p1'].value_counts()[1]
top3count=merge_1_2['p1'].value_counts()[2]
top4count=merge_1_2['p1'].value_counts()[3]
top5count=merge_1_2['p1'].value_counts()[4]
top6count=merge_1_2['p1'].value_counts()[5]

#month_hist_array = np.array([month_jan,month_feb,month_mar,month_apr,month_may,mon
month_hist_list = [top1count,top2count,top3count,top4count,top5count,top6count]

x_values = [1,2,3,4,5,6]

plt.figure(figsize=(22,11))

plt.bar(x_values,height=month_hist_list)
plt.xticks(x_values, [race1,race2,race3,race4,race5,race6]) # no need to add .5 any
plt.title('The top 6 dog races that have more pictures',size=18)
plt.xlabel('Race',size=20)
plt.ylabel('Number of Pictures',size=20)
plt.xticks(rotation=30,size=16)
plt.yticks(size=16)

plt.savefig('top6races.eps',dpi=500)

plt.show()
```



In [ ]:

```
```

In [185]:

```
merge_2_3.columns
```

Out[185]:

```
Index(['jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_con
f',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'tweet_id', 'favorite_cou
nt',
       'full_text', 'retweet_count', 'name', 'gender', 'rate_score',
'weekday',
       'month', 'day', 'year'],
      dtype='object')
```

In [186]:

```
merge_2_3.groupby(['gender'])['favorite_count'].sum()
```

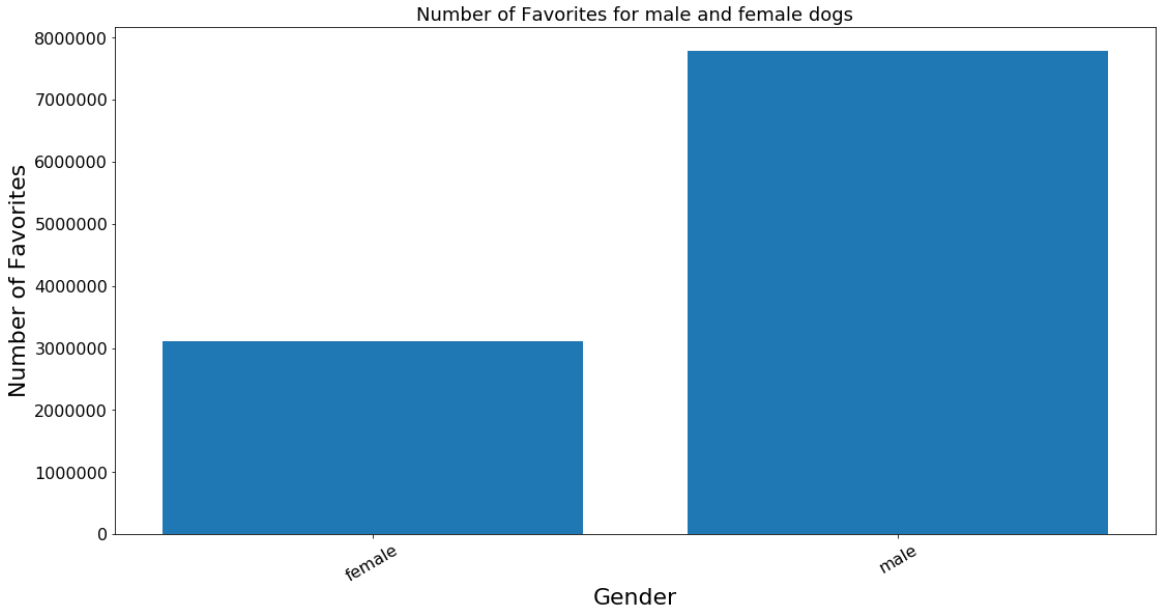
Out[186]:

```
gender
female    3117458
male      7783837
Name: favorite_count, dtype: int64
```

fazer grafico acima sobre generos e favorite count

In [187]:

```
first_sum = merge_2_3.groupby(['gender'])['favorite_count'].sum()[0]
second_sum = merge_2_3.groupby(['gender'])['favorite_count'].sum()[1]

first_sum_index = merge_2_3.groupby(['gender'])['favorite_count'].sum().index.tolis
second_sum_index = merge_2_3.groupby(['gender'])['favorite_count'].sum().index.toli

month_hist_list = [first_sum,second_sum]

x_values = [1,2]
plt.figure(figsize=(18,9))
plt.bar(x_values,height=month_hist_list)
plt.xticks(x_values, [first_sum_index,second_sum_index]) # no need to add .5 anymor
plt.title('Number of Favorites for male and female dogs',size=18)
plt.xlabel('Gender',size=22)
plt.ylabel('Number of Favorites',size=22)
plt.xticks(rotation=30,size=16)
plt.yticks(size=16)

plt.savefig('favorites_gender.eps',dpi=500)


plt.show()
```

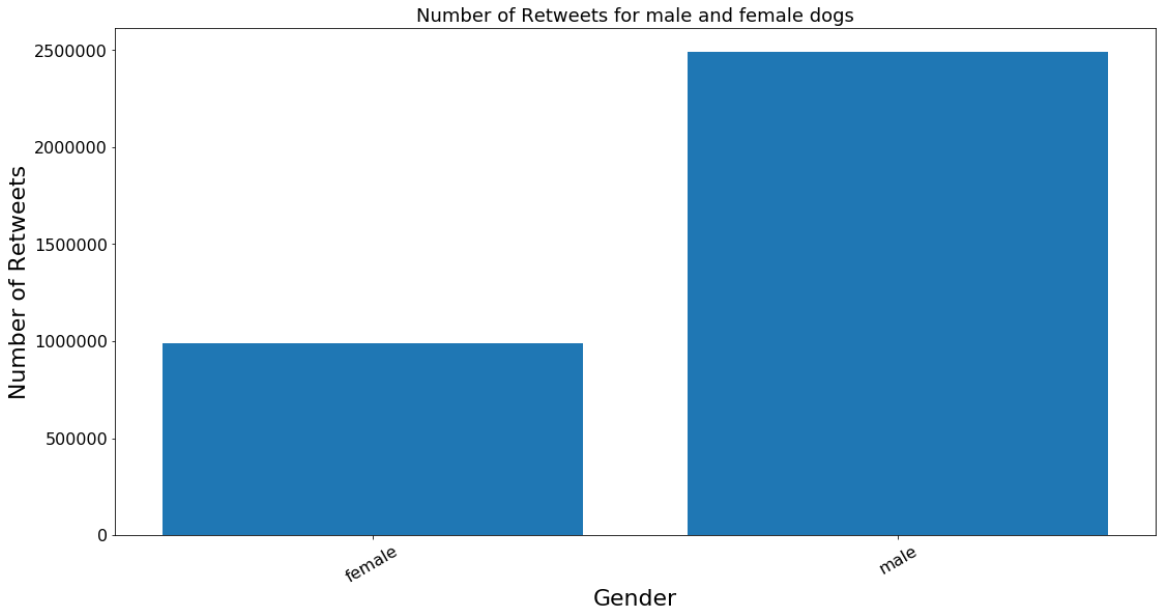


In [ ]:

fazer grafico acima sobre generos e retweet_count

In [188]:

```
first_sum = merge_2_3.groupby(['gender'])['retweet_count'].sum()[0]
second_sum = merge_2_3.groupby(['gender'])['retweet_count'].sum()[1]

first_sum_index = merge_2_3.groupby(['gender'])['retweet_count'].sum().index.tolist
second_sum_index = merge_2_3.groupby(['gender'])['retweet_count'].sum().index.tolis

month_hist_list = [first_sum,second_sum]

x_values = [1,2]
plt.figure(figsize=(18,9))
plt.bar(x_values,height=month_hist_list)
plt.xticks(x_values, [first_sum_index,second_sum_index]) # no need to add .5 anymor
plt.title('Number of Retweets for male and female dogs',size=18)
plt.xlabel('Gender',size=22)
plt.ylabel('Number of Retweets',size=22)
plt.xticks(rotation=30,size=16)
plt.yticks(size=16)

plt.savefig('retweets_gender.eps',dpi=500)


plt.show()
```



In [ ]:

In [ ]:

```
```

In [ ]:

```
```

In [189]:

```
merge_2_3.groupby(['gender'])['retweet_count'].sum()
```

Out[189]:

```
gender
female     987188
male      2490413
Name: retweet_count, dtype: int64
```

In [190]:

```
# fazer grafico acima sobre generos e retweet_count
```

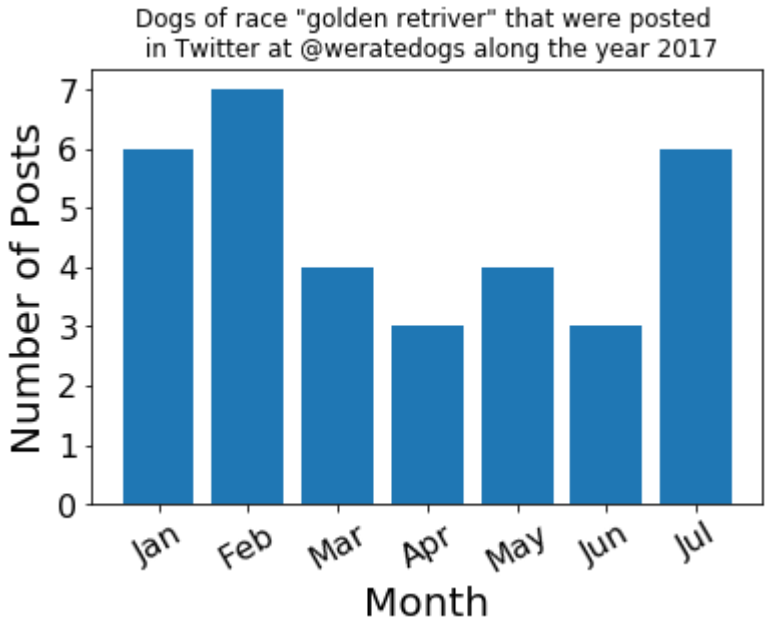
In [191]:

```
# última visualizacao
# vendo quantos cães da raça 'golden_retriever' foram postados nos
# anos de 2015 e 2017 em função dos meses! :)
ano = '2015'
raca = 'golden_retriever'

#merge_2_3[merge_2_3['year']==ano and merge_2_3['p1']==raca]
print(merge_2_3[(merge_2_3['year']==ano) & (merge_2_3['p1']==raca)]['month'].value_
ano = '2017'
print(merge_2_3[(merge_2_3['year']==ano) & (merge_2_3['p1']==raca)]['month'].value_

# Logo, os resultados para o ano de 2017 sao os mais ricos!
# Faremos uma visualização do ano 2017 na forma de histograma!!
```

```
Nov    13
Dec    13
Name: month, dtype: int64
Feb    7
Jan    6
Jul    6
May    4
Mar    4
Apr    3
Jun    3
Name: month, dtype: int64
```

In [192]:

```
raca = 'golden_retriever'

ano = '2017'
month_iter = merge_2_3[(merge_2_3['year']==ano) & (merge_2_3['p1']==raca)]['month']
month_jan = month_iter[1]
month_feb = month_iter[0]
month_mar = month_iter[4]
month_apr = month_iter[-1]
month_may = month_iter[3]
month_jun = month_iter[-2]
month_jul = month_iter[2]

#month_hist_array = np.array([month_jan,month_feb,month_mar,month_apr,month_may,mon
month_hist_list = [month_jan,month_feb,month_mar,month_apr,month_may,month_jun,mont

x_values = [1,2,3,4,5,6,7]

plt.bar(x_values,height=month_hist_list)
plt.xticks(x_values, ['Jan','Feb','Mar','Apr','May','Jun','Jul']) # no need to add
plt.title('Dogs of race "golden retriver" that were posted \n in Twitter at @werate
plt.xlabel('Month',size=20)
plt.ylabel('Number of Posts',size=20)
plt.xticks(rotation=30,size=16)
plt.yticks(size=16)
plt.savefig('golden.retriever_2017_months.eps',dpi=200)

plt.show()
```



In [ ]:

Exporting the master DataFrames!

In [193]:

```
# exporting the first master dataframe
merge_1_2.to_csv('twitter_archive_master1.csv',encoding='utf-8',index=False)
```

In [194]:

```
# exporting the second master dataframe
merge_2_3.to_csv('twitter_archive_master2.csv',encoding='utf-8',index=False)
```

In [ ]:

```

```

UFA!!!!! BAITA PROJETO! MUITO DIVERTIDO!!!

In [ ]:

```

```