In [1]:
```
import requests
import os
import pandas as pd
import numpy as np
import tweepy
import json
```

Gather ¶

Loading twitter-archive-enhanced.csv onto a dataframe: archive

In [2]:
```
archive = pd.read_csv('twitter-archive-enhanced.csv')
archive.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	<a href="http r...
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	<a href="http r...
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	<a href="http r...
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	<a href="http r...
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	<a href="http r...

Downloading image_predictions.tsv from Udacity servers and loading on dataframe : image

In [3]:
```
#Define the url from which we need to download image_predictions.tsv
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image
-predictions/image-predictions.tsv'
response = requests.get(url)
response
```

Out[3]: <Response [200]>

The repsonse above means that the request has succeeded

In [4]:
```
response.content[0:100]
```

Out[4]:
```
b'tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_dog\tp3\tp
3_conf\tp3_dog\n666020888022790149\tht'
```

Output above starts with 'b' so content is binary

In [5]:
```
#Download the file programatically to image_predictions.tsv
with open ('image_predictions.tsv', mode = 'wb') as file:
    file.write(response.content)
```

In [6]:
```
image = pd.read_csv('image_predictions.tsv', sep='\t')
image.head()
```

Out[6]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	W
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Ge
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rh
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	mi

Downloading tweepy API data into file 'tweet_json.txt'. Loading into dataframe: tweet_df

In [7]:
```
#Entering keys from twitter and saving to variables
consumer_key = 'YOUR CONSUMER KEY'
consumer_secret = 'YOUR CONSUMER SECRET'

access_token = 'YOUR ACCESS TOKEN'
access_secret = 'YOUR ACCESS SECRET'
```

In [8]:
```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
```

In [9]:
```
failed = []
with open('tweet_json.txt', 'w') as file:
    for id in archive['tweet_id']:
        try:
            tweet = api.get_status(id)
            file.write(json.dumps(tweet._json))
            file.write('\n')
        except:
            failed.append(id)
            continue
file.close()
```

In [10]:
```
for i in range(len(failed)):
    print(archive[archive['tweet_id'] == failed[i]]['text'])
```

In [11]:
```
try:
    api.get_status(754011816964026368)
except Exception as e:
    print('Exception: {}'.format(e))
```

We can ignore the first 14 ids since they are all Retweets. The last one is not a retweet but the API cannot get any info for that one.

In [12]:
```
archive[archive['tweet_id'] == 754011816964026368]
```

Out[12]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
932	754011816964026368	NaN	NaN	2016-07-15 17:56:40 +0000	<a href="h r...

In [13]:
```
archive['expanded_urls'][932]
```

Out[13]:
```
'https://twitter.com/dog_rates/status/754011816964026368/photo/1,https://twit
ter.com/dog_rates/status/754011816964026368/photo/1'
```

Attempted to use the expanded_urls to reach the page but they all show that the page doesn't exist. Delete this row.

In [14]:
```
with open('tweet_json.txt') as file:
    status = []
    for line in file:
        status.append(json.loads(line))
```

In [15]:
```
tweet_dict = {}
tweet_dict['tweet_id'] = []
tweet_dict['retweet_count'] = []
tweet_dict['favorite_count'] = []
tweet_dict['expanded_url'] = []
index = []
for i in range(len(status)):
    try:
        tweet_dict['tweet_id'].append(status[i]['id_str'])
        tweet_dict['retweet_count'].append(status[i]['retweet_count'])
        tweet_dict['favorite_count'].append(status[i]['favorite_count'])
        tweet_dict['expanded_url'].append(status[i]['entities']['media'][0]['e
xpanded_url'])
    except:
        tweet_dict['expanded_url'].append('None')
```

In [16]:
```
tweet_df = pd.DataFrame(tweet_dict)
tweet_df = tweet_df[['tweet_id','retweet_count','favorite_count','expanded_ur
l']]
tweet_df.head()
```

Out[16]:

	tweet_id	retweet_count	favorite_count	ex
0	892420643555336193	8411	38326	https://twitter.com/dog_rates/status/8
1	892177421306343426	6199	32846	None
2	891815181378084864	4101	24743	None
3	891689557279858688	8538	41667	https://twitter.com/dog_rates/status/8
4	891327558926688256	9265	39839	None

Assess

In [17]:
```
archive.head()
```

Out[17]:

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

In [19]:
```
image.head()
```

Out[19]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	W
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	Ge
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rh
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	mi

In [20]:
```
tweet_df.head()
```

Out[20]:

	tweet_id	retweet_count	favorite_count	ex
0	892420643555336193	8411	38326	https://twitter.com/dog_rates/status/8
1	892177421306343426	6199	32846	None
2	891815181378084864	4101	24743	None
3	891689557279858688	8538	41667	https://twitter.com/dog_rates/status/8
4	891327558926688256	9265	39839	None

In [21]:
```
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 [22]:
```
archive.isnull().sum()
```

Out[22]:
```
tweet_id                      0
in_reply_to_status_id      2278
in_reply_to_user_id        2278
timestamp                     0
source                        0
text                          0
retweeted_status_id        2175
retweeted_status_user_id   2175
retweeted_status_timestamp 2175
expanded_urls                59
rating_numerator              0
rating_denominator            0
name                          0
doggo                         0
floofer                       0
pupper                        0
puppo                         0
dtype: int64
```

In [23]:
```
image.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 [24]:
```
tweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2341 entries, 0 to 2340
Data columns (total 4 columns):
tweet_id          2341 non-null object
retweet_count     2341 non-null int64
favorite_count    2341 non-null int64
expanded_url      2341 non-null object
dtypes: int64(2), object(2)
memory usage: 73.2+ KB
```

In [25]:
```
archive['source'].value_counts()
```

Out[25]:
```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhon
e</a>     2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDe
ck</a>      11
Name: source, dtype: int64
```

We could clean up the 'source' column. Instead of showing full url we could just record the name e.g. Twitter for iphone

In [26]:
```
archive['rating_denominator'].value_counts()
```

Out[26]:
```
10     2333
11        3
50        3
80        2
20        2
2         1
16        1
40        1
70        1
15        1
90        1
110       1
120       1
130       1
150       1
170       1
7         1
0         1
Name: rating_denominator, dtype: int64
```

In [27]:
```
archive['rating_numerator'].value_counts()
```

Out[27]:
```
12      558
11      464
10      461
13      351
9       158
8       102
7        55
14       54
5        37
6        32
3        19
4        17
1         9
2         9
420       2
0         2
15        2
75        2
80        1
20        1
24        1
26        1
44        1
50        1
60        1
165       1
84        1
88        1
144       1
182       1
143       1
666       1
960       1
1776      1
17        1
27        1
45        1
99        1
121       1
204       1
Name: rating_numerator, dtype: int64
```

In [28]:
```
archive[archive['rating_numerator'] == 0]
```

Out[28]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
315	835152434251116546	NaN	NaN	2017-02-24 15:40:31 +0000	<a href=" r...
1016	746906459439529985	7.468859e+17	4.196984e+09	2016-06-26 03:22:31 +0000	<a href=" r...

In [29]:
```
archive[archive['expanded_urls'].isnull()].head()
```

Out[29]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
30	886267009285017600	8.862664e+17	2.281182e+09	2017-07-15 16:51:35 +0000	<a href="h r...
55	881633300179243008	8.816070e+17	4.738443e+07	2017-07-02 21:58:53 +0000	<a href="h r...
64	879674319642796034	8.795538e+17	3.105441e+09	2017-06-27 12:14:36 +0000	<a href="h r...
113	870726314365509632	8.707262e+17	1.648776e+07	2017-06-02 19:38:25 +0000	<a href="h r...
148	863427515083354112	8.634256e+17	7.759620e+07	2017-05-13 16:15:35 +0000	<a href="h r...

In [30]:
```
#Retweets
archive[archive['retweeted_status_id'].isnull() == False].head()
```

Out[30]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
19	888202515573088257	NaN	NaN	2017-07-21 01:02:36 +0000	<a href="ht r...
32	886054160059072513	NaN	NaN	2017-07-15 02:45:48 +0000	<a href="ht r...
36	885311592912609280	NaN	NaN	2017-07-13 01:35:06 +0000	<a href="ht r...
68	879130579576475649	NaN	NaN	2017-06-26 00:13:58 +0000	<a href="ht r...
73	878404777348136964	NaN	NaN	2017-06-24 00:09:53 +0000	<a href="ht r...

In [31]:
```
#replies
archive[archive['in_reply_to_status_id'].isnull() == False].head()
```

Out[31]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
30	886267009285017600	8.862664e+17	2.281182e+09	2017-07-15 16:51:35 +0000	<a href="h r...
55	881633300179243008	8.816070e+17	4.738443e+07	2017-07-02 21:58:53 +0000	<a href="h r...
64	879674319642796034	8.795538e+17	3.105441e+09	2017-06-27 12:14:36 +0000	<a href="h r...
113	870726314365509632	8.707262e+17	1.648776e+07	2017-06-02 19:38:25 +0000	<a href="h r...
148	863427515083354112	8.634256e+17	7.759620e+07	2017-05-13 16:15:35 +0000	<a href="h r...

In [32]:
```
#not dogs?
p1 = image['p1_dog'] == False
p2 = image['p2_dog'] == False
p3 = image['p3_dog'] == False
image[p1 & p2 & p3].head()
```

Out[32]:

	tweet_id	jpg_url	img_num
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1

In [33]:
```
(archive['doggo'] != 'None').sum()
```

Out[33]: 97

In [34]:
```
(archive['floofer'] != 'None').sum()
```

Out[34]: 10

In [35]:
```
(archive['pupper'] != 'None').sum()
```

Out[35]: 257

In [36]:
```
(archive['puppo'] != 'None').sum()
```

Out[36]: 30

In [37]:
```
#every tweet_id is unique in archive table
archive[archive['tweet_id'].duplicated()]
```

Out[37]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted

Check if retweets from archive are repeated in image table

In [38]:
```
image[image['tweet_id'] ==      888202515573088257]
```

Out[38]:

	tweet_id	jpg_url	img_num
2055	888202515573088257	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2

In [39]:
```
#every tweet_id is unique in image table
image[image['tweet_id'].duplicated()]
```

Out[39]:

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

In [40]:
```
#These 2 rows have 0 numerator rating...wierd?
archive[archive['rating_numerator'] == 0]
```

Out[40]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
315	835152434251116546	NaN	NaN	2017-02-24 15:40:31 +0000	<a href=" r...
1016	746906459439529985	7.468859e+17	4.196984e+09	2016-06-26 03:22:31 +0000	<a href=" r...

In [41]:
```
#dont delete
archive['expanded_urls'][315]
```

Out[41]:
```
'https://twitter.com/dog_rates/status/835152434251116546/photo/1,https://twit
ter.com/dog_rates/status/835152434251116546/photo/1,https://twitter.com/dog_r
ates/status/835152434251116546/photo/1'
```

In [42]:
```
archive['expanded_urls'][1016]
```

Out[42]:
```
'https://twitter.com/dog_rates/status/746906459439529985/photo/1'
```

Delete this. This tweet has no dog in it.

In [43]:
```
#This row has 0 denominator...what??
archive[archive['rating_denominator'] == 0]
```

Out[43]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
313	835246439529840640	8.352460e+17	26259576.0	2017-02-24 21:54:03 +0000	<a href="h r...

In [44]:
```
#doesnt exist in image either. DELETE THIS
image[image['tweet_id'] == 835246439529840640]
```

Out[44]:

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

In [45]:
```
archive[archive['doggo'] != 'None'].head()
```

Out[45]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
9	890240255349198849	NaN	NaN	2017-07-26 15:59:51 +0000	<a href="h r...
43	884162670584377345	NaN	NaN	2017-07-09 21:29:42 +0000	<a href="h r...
99	872967104147763200	NaN	NaN	2017-06-09 00:02:31 +0000	<a href="h r...
108	871515927908634625	NaN	NaN	2017-06-04 23:56:03 +0000	<a href="h r...
110	871102520638267392	NaN	NaN	2017-06-03 20:33:19 +0000	<a href="h r...

In [46]:
```
doggo = archive['doggo'] != 'None'
floofer = archive['floofer'] != 'None'
pupper = archive['pupper'] != 'None'
puppo = archive['puppo'] != 'None'

archive[doggo | floofer | pupper | puppo].head()
```

Out[46]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
9	890240255349198849	NaN	NaN	2017-07-26 15:59:51 +0000	<a href="ht r...
12	889665388333682689	NaN	NaN	2017-07-25 01:55:32 +0000	<a href="ht r...
14	889531135344209921	NaN	NaN	2017-07-24 17:02:04 +0000	<a href="ht r...
29	886366144734445568	NaN	NaN	2017-07-15 23:25:31 +0000	<a href="ht r...
43	884162670584377345	NaN	NaN	2017-07-09 21:29:42 +0000	<a href="ht r...

In [47]:
```
archive.query('rating_numerator == 0')
```

Out[47]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
315	835152434251116546	NaN	NaN	2017-02-24 15:40:31 +0000	<a href=" r...
1016	746906459439529985	7.468859e+17	4.196984e+09	2016-06-26 03:22:31 +0000	<a href=" r...

In [48]:
```
archive.query('rating_denominator == 0')
```

Out[48]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
313	835246439529840640	8.352460e+17	26259576.0	2017-02-24 21:54:03 +0000	<a href="h r...

Neither of these ratings are about dogs, delete all rows where rating_numerator or rating_denominator = 0

Quality

`archive` table

- Change tweet_id datatype to string ✓
- Delete 'retweeted_status_id' and 'retweeted_status_user_id' and all retweet rows. ✓
- Delete rows that have null in 'expanded_urls' . These rows have no images. ✓
- Convert timestamp to datetime format. ✓
- Data in 'in_reply_to_status_id' and 'in_reply_to_user_id' saved as float. Should have been saved as string. ✓
- Delete row 1016, tweet_id = 746906459439529985. This row has no dog in it. ✓
- Delete row 932, tweet_id = 754011816964026368 from the archive table. ✓
- Clean up source. ✓
- Increase column width to be able to read complete text or url. ✓
- Delete rows with rating_numerator = 0 or rating_denominator = 0. ✓

`image` table

- Find rows that are not dogs and delete them ✓

Tidiness

- Add retweets_count and favorite_count to the archive table by merging archive and tweet_df ✓
- Combine 'floofer', 'pupper', 'doggo', 'puppo' into 1 column, Age. ✓
- Duplicate 'expanded_urls' column in tweet_df table made to check if we could fill any nulls in archive table. Delete this column later. ✓

Clean

In [49]:
```
archive_clean = archive.copy()
image_clean = image.copy()
tweet_df_clean = tweet_df.copy()
```

`archive`: Increase column width

Define

Increase column width to read full text and url

Code

In [50]:
```
#Use -1 to set column width to fit the content
pd.set_option('display.max_colwidth', -1)
```

Test

In [51]:
```
archive.head()
```

Out[51]:

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

`archive`: Clean up Source column

Define

Source column has repeated URLS. Replace urls with only the text

Code

In [52]:
```
archive_clean['source'] = archive_clean['source'].str.replace('<a href="htt
p://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>', 'Twit
ter for iPhone')
archive_clean['source'] = archive_clean['source'].str.replace('<a href="htt
p://vine.co" rel="nofollow">Vine - Make a Scene</a>', 'Vine - Make a Scene')
archive_clean['source'] = archive_clean['source'].str.replace('<a href="htt
p://twitter.com" rel="nofollow">Twitter Web Client</a>', 'Twitter Web Client')
archive_clean['source'] = archive_clean['source'].str.replace('<a href="http
s://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>', 'Twee
tDeck')
```

Test

In [53]:
```
archive_clean['source'].value_counts()
```

Out[53]:
```
Twitter for iPhone     2221
Vine - Make a Scene    91
Twitter Web Client     33
TweetDeck              11
Name: source, dtype: int64
```

`archive`: Delete 0 rating rows

Define

Delete rows with rating_denominator = 0 or rating_numerator = 0

Code

In [54]:
```
#Find all rows where rating_numerator = 0 or rating_denominator = 0
archive_clean[(archive_clean['rating_numerator'] == 0) | (archive_clean['ratin
g_denominator'] == 0)]
```

Out[54]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	sourc
313	835246439529840640	8.352460e+17	2.625958e+07	2017-02-24 21:54:03 +0000	Twitte for iPhon
315	835152434251116546	NaN	NaN	2017-02-24 15:40:31 +0000	Twitte for iPhon
1016	746906459439529985	7.468859e+17	4.196984e+09	2016-06-26 03:22:31 +0000	Twitte for iPhon

In [55]:
```
#Get index of all the above rows and delete them.
zero_rate_index = archive_clean[(archive_clean['rating_numerator'] == 0) | (ar
chive_clean['rating_denominator'] == 0)].index
archive_clean.drop(archive_clean.index[[zero_rate_index]], inplace = True)
```

Test

In [56]:
```
archive_clean[(archive_clean['rating_numerator'] == 0) | (archive_clean['ratin
g_denominator'] == 0)]
```

Out[56]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted

archive: Wrong type on tweet_id column

Define

Change tweet_id column to string type

Code

```
In [57]: archive_clean['tweet_id'].dtypes
```

```
Out[57]: dtype('int64')
```

```
In [58]: archive_clean['tweet_id'] = archive_clean['tweet_id'].apply(str)
```

Test

```
In [59]: archive_clean['tweet_id'].dtypes
```

```
Out[59]: dtype('O')
```

`image`: Wrong type on tweet_id column

Define

Change tweet_id column to string type

Code

```
In [60]: image_clean['tweet_id'].dtypes
```

```
Out[60]: dtype('int64')
```

```
In [61]: image_clean['tweet_id'] = image_clean['tweet_id'].apply(str)
```

Test

```
In [62]: image_clean['tweet_id'].dtypes
```

```
Out[62]: dtype('O')
```

`archive`: Delete 'retweeted_status_id' and 'retweeted_status_user_id' and all retweet rows.

Define

We only want original tweets. So we should delete all retweet rows. This includes any rows where 'retweeted_status_id' or 'retweeted_status_user_id' are present. Then delete both the columns and 'retweeted_status_timestamp' as well since they aren't relevant.

Code

In [63]:
```
#Find rows that are retweets and drop them
retweet_status = archive_clean['retweeted_status_id'].isnull() == False
retweet_user = archive_clean['retweeted_status_user_id'].isnull() == False
drop_index = archive_clean[retweet_status | retweet_user].index
archive_clean.drop(archive_clean.index[[drop_index]], inplace = True)
```

In [64]:
```
#Drop the columns that are affiliated with retweets.
archive_clean.drop(['retweeted_status_id','retweeted_status_user_id', 'retweet
ed_status_timestamp'], axis = 1, inplace=True)
archive_clean.reset_index(drop = True, inplace = True)
```

Test

In [65]:
```
archive_clean.head()
```

Out[65]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	Twitter for iPhone
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	Twitter for iPhone
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	Twitter for iPhone
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	Twitter for iPhone
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	Twitter for iPhone

`archive`: Convert timestamp column to datetime format

Define

Convert the timestamp column to datetime format

Code

```
In [66]: archive_clean['timestamp'] = pd.to_datetime(archive_clean['timestamp'])
```

Test

```
In [67]: #timestamp datatype is datetime64 now
         archive_clean.dtypes
```

```
Out[67]: tweet_id                 object
         in_reply_to_status_id    float64
         in_reply_to_user_id      float64
         timestamp                datetime64[ns]
         source                   object
         text                     object
         expanded_urls            object
         rating_numerator         int64
         rating_denominator       int64
         name                     object
         doggo                    object
         floofer                  object
         pupper                   object
         puppo                    object
         dtype: object
```

archive: Delete rows that have null in 'expanded_urls' . These rows have no images. First check if we can find any expanded_url data of same tweet_id from API dataframe, tweet_df

Define

Find rows with null in 'expanded_urls'. Delete them

Code

In [68]:
```
#Find all rows tweets for which we dont have a url
archive_clean[archive_clean['expanded_urls'].isnull()].head()
```

Out[68]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source
29	886267009285017600	8.862664e+17	2.281182e+09	2017-07-15 16:51:35	Twitter for iPhone
52	881633300179243008	8.816070e+17	4.738443e+07	2017-07-02 21:58:53	Twitter for iPhone
61	879674319642796034	8.795538e+17	3.105441e+09	2017-06-27 12:14:36	Twitter for iPhone
101	870726314365509632	8.707262e+17	1.648776e+07	2017-06-02 19:38:25	Twitter for iPhone
130	863427515083354112	8.634256e+17	7.759620e+07	2017-05-13 16:15:35	Twitter for iPhone

In [69]:
```
null_expanded_url_id = archive_clean[archive_clean['expanded_urls'].isnull()][
'tweet_id']
missing_urls = {}
for id in null_expanded_url_id:
    missing_urls[id] = []
    missing_urls[id].append(tweet_df_clean[tweet_df_clean['tweet_id'] == id][
'expanded_url'])
```

In [70]:
```
missing_urls
```

Out[70]:
```
{'667070482143944705': [2283    None
  Name: expanded_url, dtype: object], '668967877119254528': [2174    None
  Name: expanded_url, dtype: object], '669684865554620416': [2134    None
  Name: expanded_url, dtype: object], '671550332464455680': [2023    None
  Name: expanded_url, dtype: object], '673716320723169284': [1925    None
  Name: expanded_url, dtype: object], '674330906434379776': [1899    None
  Name: expanded_url, dtype: object], '674606911342424069': [1890    None
  Name: expanded_url, dtype: object], '674742531037511680': [1880    None
  Name: expanded_url, dtype: object], '675849018447167488': [1829    None
  Name: expanded_url, dtype: object], '676590572941893632': [1804    None
  Name: expanded_url, dtype: object], '678023323247357953': [1759    None
  Name: expanded_url, dtype: object], '681340665377193984': [1674    None
  Name: expanded_url, dtype: object], '682808988178739200': [1648    None
  Name: expanded_url, dtype: object], '684969860808454144': [1603    None
  Name: expanded_url, dtype: object], '685681090388975616': [1590    None
  Name: expanded_url, dtype: object], '686035780142297088': [1583    None
  Name: expanded_url, dtype: object], '690607260360429569': [1508    None
  Name: expanded_url, dtype: object], '692423280028966913': [1482    None
  Name: expanded_url, dtype: object], '693582294167244802': [1464    None
  Name: expanded_url, dtype: object], '693644216740769793': [1459    None
  Name: expanded_url, dtype: object], '696490539101908992': [1431    None
  Name: expanded_url, dtype: object], '696518437233913856': [1430    None
  Name: expanded_url, dtype: object], '704491224099647488': [1330    None
  Name: expanded_url, dtype: object], '707983188426153984': [1280    None
  Name: expanded_url, dtype: object], '738891149612572673': [1065    None
  Name: expanded_url, dtype: object], '747651430853525504': [990    None
  Name: expanded_url, dtype: object], '750381685133418496': [952    None
  Name: expanded_url, dtype: object], '763956972077010945': [843    None
  Name: expanded_url, dtype: object], '785515384317313025': [696    None
  Name: expanded_url, dtype: object], '786051337297522688': [690    None
  Name: expanded_url, dtype: object], '797165961484890113': [600    None
  Name: expanded_url, dtype: object], '811647686436880384': [503    None
  Name: expanded_url, dtype: object], '813130366689148928': [488    None
  Name: expanded_url, dtype: object], '823333489516937216': [399    None
  Name: expanded_url, dtype: object], '826598799820865537': [377    None
  Name: expanded_url, dtype: object], '828361771580813312': [366    None
  Name: expanded_url, dtype: object], '831926988323639298': [337    None
  Name: expanded_url, dtype: object], '838085839343206401': [283    None
  Name: expanded_url, dtype: object], '838150277551247360': [282    None
  Name: expanded_url, dtype: object], '840698636975636481': [266    None
  Name: expanded_url, dtype: object], '847617282490613760': [228    None
  Name: expanded_url, dtype: object], '848213670039564288': [222    None
  Name: expanded_url, dtype: object], '850333567704068097': [212    None
  Name: expanded_url, dtype: object], '855860136149123072': [183    None
  Name: expanded_url, dtype: object], '855862651834028034': [182    None
  Name: expanded_url, dtype: object], '856288084350160898': [180    None
  Name: expanded_url, dtype: object], '857214891891077121': [173    None
  Name: expanded_url, dtype: object], '863427515083354112': [143    None
  Name: expanded_url, dtype: object], '870726314365509632': [110    None
  Name: expanded_url, dtype: object], '879674319642796034': [63    None
  Name: expanded_url, dtype: object], '881633300179243008': [54    None
  Name: expanded_url, dtype: object], '886267009285017600': [29    None
  Name: expanded_url, dtype: object]}
```

Seems like these rows don't have any expanded url data even from the API. Would be a good idea to delete them from both 'archive' table.

In [71]:
```
drop_index = archive_clean[archive_clean['expanded_urls'].isnull()].index
archive_clean.drop(archive_clean.index[[drop_index]], inplace = True)
archive_clean.reset_index(drop = True, inplace = True)
```

Test

In [72]:
```
archive_clean[archive_clean['expanded_urls'].isnull()]
```

Out[72]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	expanded

`tweet_df`: Delete expanded_urls column

Define

Delete expanded_urls column

Code

In [73]:
```
tweet_df_clean.drop(['expanded_url'], axis = 1, inplace = True)
```

Test

In [74]:
```
tweet_df_clean.head()
```

Out[74]:

	tweet_id	retweet_count	favorite_count
0	892420643555336193	8411	38326
1	892177421306343426	6199	32846
2	891815181378084864	4101	24743
3	891689557279858688	8538	41667
4	891327558926688256	9265	39839

`archive`: Change 'in_reply_to_status_id' and 'in_reply_to_user_id' type from float to string

Define

A lot of replies include new relevant dog posts so I am no deleting these replies and columns. Change the column data type from float to string.

Code

```
In [75]: archive_clean['in_reply_to_status_id'].dtype
```

```
Out[75]: dtype('float64')
```

```
In [76]: archive_clean['in_reply_to_user_id'].dtype
```

```
Out[76]: dtype('float64')
```

```
In [77]: archive_clean['in_reply_to_status_id'] = archive_clean['in_reply_to_status_id'
         ].apply(str)
         archive_clean['in_reply_to_user_id'] = archive_clean['in_reply_to_user_id'].ap
         ply(str)
```

Test

```
In [78]: archive_clean['in_reply_to_status_id'].dtype
```

```
Out[78]: dtype('O')
```

```
In [79]: archive_clean['in_reply_to_user_id'].dtype
```

```
Out[79]: dtype('O')
```

`archive`: Delete tweets that had no dogs

Define

Delete row 1016, tweet_id = 746906459439529985. This row has no dog in it.

Code

In [80]:
```
drop_index = archive_clean[archive_clean['tweet_id'] == 746906459439529985].in
dex
archive_clean.drop(archive_clean.index[[drop_index]], inplace = True)
archive_clean.reset_index(drop = True, inplace = True)
```

Test

In [81]:
```
archive_clean[archive_clean['tweet_id'] == 746906459439529985]
```

Out[81]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	expandec

Define

Delete row 932, tweet_id = 754011816964026368 from the archive table

Code

In [82]:
```
drop_index = archive_clean[archive_clean['tweet_id'] == 754011816964026368].in
dex
archive_clean.drop(archive_clean.index[[drop_index]], inplace = True)
archive_clean.reset_index(drop = True, inplace = True)
```

Test

In [83]:
```
archive_clean[archive_clean['tweet_id'] == 754011816964026368]
```

Out[83]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	expandec

`images`: Find rows which are not dogs and delete them

Define

Find rows in image table that are not predicted to be dogs and delete them.

Code

There still might be rows which aren't dogs but assuming that the the 3 predictions are correct we can presume that if p1_dog, p2_dog & p3_dog are all False, that tweet does not have a dog in it

In [84]:
```
image_clean[(image_clean['p1_dog'] == False) & (image_clean['p2_dog'] == False
) & (image_clean['p3_dog'] == False)].head()
```

Out[84]:

	tweet_id	jpg_url	img_num
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1
25	666362758909284353	https://pbs.twimg.com/media/CT9IXGsUcAAyUFt.jpg	1

In [85]:
```
len(image_clean[(image_clean['p1_dog'] == False) & (image_clean['p2_dog'] == F
alse) & (image_clean['p3_dog'] == False)])
```

Out[85]: 324

We should delete these rows for sure. Some might be wrong predictions and we might end up deleting dog tweets. But I feel that would be a a minimal few rows from the total

We need to drop these rows from both 'archive' table and the 'image' table

In [86]:
```
drop_image_ids = image_clean[(image_clean['p1_dog'] == False)
                             & (image_clean['p2_dog'] == False) & (image_clean
['p3_dog'] == False)]['tweet_id']
drop_archive_index = archive_clean[archive_clean['tweet_id'].isin(drop_image_i
ds)].index
drop_image_index = image_clean[(image_clean['p1_dog'] == False) &
                               (image_clean['p2_dog'] == False) & (image_clean
['p3_dog'] == False)].index
```

In [87]:
```
image_clean.drop(image_clean.index[[drop_image_index]], inplace = True)
archive_clean.drop(archive_clean.index[[drop_archive_index]], inplace = True)

archive_clean.reset_index(drop = True, inplace = True)
image_clean.reset_index(drop = True, inplace = True)
```

Test

In [88]:
```
image_clean[(image_clean['p1_dog'] == False) & (image_clean['p2_dog'] == False
) & (image_clean['p3_dog'] == False)]
```

Out[88]:

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

archive: Add retweets_count and favorite_count to the archive table by merging archive and tweet_df

Define

Merge archive and tweet_df

Code

In [89]:
```
archive_clean = archive_clean.merge(right = tweet_df_clean, on = 'tweet_id', h
ow = 'inner')
```

Test

In [90]:
```
archive_clean.head()
```

Out[90]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source
0	892177421306343426	nan	nan	2017-08-01 00:17:27	Twitter for iPhone
1	891815181378084864	nan	nan	2017-07-31 00:18:03	Twitter for iPhone
2	891689557279858688	nan	nan	2017-07-30 15:58:51	Twitter for iPhone
3	891327558926688256	nan	nan	2017-07-29 16:00:24	Twitter for iPhone
4	891087950875897856	nan	nan	2017-07-29 00:08:17	Twitter for iPhone

`archive`: Age has 4 different columns

Define

Merge "doggo","floofer","pupper","puppo" columns into one column "Age" since they are all identifiers of age.

Code

In [91]:
```
archive_clean.shape
```

Out[91]: (1813, 16)

In [92]:
```
#dataframe of all tweets without dog age
no_age = archive_clean[(archive_clean['doggo'] == 'None') &
                       (archive_clean['floofer'] == 'None') & (archive_clean[
'pupper'] == 'None')
                       & (archive_clean['puppo'] == 'None')].copy()
```

In [93]:
```
melt = pd.melt(archive_clean, id_vars = ['tweet_id','in_reply_to_status_id','i
n_reply_to_user_id','timestamp','source','text','expanded_urls',
                        'rating_numerator','rating_denominator','name','retwee
t_count','favorite_count'],
       value_vars = ['doggo','floofer','pupper','puppo'], var_name = 'Age')
```

In [94]:
```
melt.head()
```

Out[94]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892177421306343426	nan	nan	2017-08-01 00:17:27	Twitter for iPhone	
1	891815181378084864	nan	nan	2017-07-31 00:18:03	Twitter for iPhone	
2	891689557279858688	nan	nan	2017-07-30 15:58:51	Twitter for iPhone	
3	891327558926688256	nan	nan	2017-07-29 16:00:24	Twitter for iPhone	
4	891087950875897856	nan	nan	2017-07-29 00:08:17	Twitter for iPhone	

In [95]:
```
drop_melt_index = melt[melt['value'] == 'None'].index
melt.drop(melt.index[[drop_melt_index]],inplace=True)
melt.reset_index(drop = True, inplace = True)
```

In [96]:
```
duplicate = melt[melt['tweet_id'].duplicated()]
len(duplicate)
```

Out[96]: 13

There are 13 rows with more than one dog age in the tweet. Since this is a very small part of the dataset (13 rows is < 1% of the dataset), we can consider these as outliers of the "one dog per photo" norm and omit them.

In [97]:
```
melt.drop(melt.index[[duplicate.index]], inplace=True)
melt.reset_index(drop = True, inplace = True)
melt[melt['tweet_id'].duplicated()]
melt.drop('value', axis = 1, inplace = True)
```

In [98]:
```
no_age.drop(['doggo','floofer','pupper','puppo'], axis = 1, inplace = True)
no_age["Age"] = 'None'
```

In [99]:
```
no_age.head()
```

Out[99]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892177421306343426	nan	nan	2017-08-01 00:17:27	Twitter for iPhone	
1	891815181378084864	nan	nan	2017-07-31 00:18:03	Twitter for iPhone	
2	891689557279858688	nan	nan	2017-07-30 15:58:51	Twitter for iPhone	
3	891327558926688256	nan	nan	2017-07-29 16:00:24	Twitter for iPhone	
4	891087950875897856	nan	nan	2017-07-29 00:08:17	Twitter for iPhone	

In [100]:
```
melt.head()
```

Out[100]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source
0	890240255349198849	nan	nan	2017-07-26 15:59:51	Twitter for iPhone
1	884162670584377345	nan	nan	2017-07-09 21:29:42	Twitter for iPhone
2	872967104147763200	nan	nan	2017-06-09 00:02:31	Twitter for iPhone
3	871515927908634625	nan	nan	2017-06-04 23:56:03	Twitter for iPhone
4	871102520638267392	nan	nan	2017-06-03 20:33:19	Twitter for iPhone

In [101]:
```
archive_clean = no_age.append(melt)
```

Test

In [102]:
```
archive_clean.head()
```

Out[102]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892177421306343426	nan	nan	2017-08-01 00:17:27	Twitter for iPhone	
1	891815181378084864	nan	nan	2017-07-31 00:18:03	Twitter for iPhone	
2	891689557279858688	nan	nan	2017-07-30 15:58:51	Twitter for iPhone	
3	891327558926688256	nan	nan	2017-07-29 16:00:24	Twitter for iPhone	
4	891087950875897856	nan	nan	2017-07-29 00:08:17	Twitter for iPhone	

In [103]:
```
archive_clean.shape
```

Out[103]: (1813, 13)

In [104]:
```
len(archive_clean['tweet_id'].unique())
```

Out[104]: 1813

We have retained all the original number of unique tweets

Store Datasets

In [105]:
```
archive_clean.to_csv('twitter_archive_master.csv')
```

In [106]:
```
image_clean.to_csv('image_master.csv')
```

I am keeping the image predictions in a separate dataset since I feel it is a separate observation as per the rules of tidy data. The "twitter_archive_master" dataset has all the information related to the tweets from twitter, while the "image_predictions" dataset contains all neural net results that was obtained on the images.