Wrangle and Analyze Data

by Eslam Saleh

Gather

Depending on the source of your data, and what format it's in, the steps in gathering data vary. High-level gathering process: obtaining data (downloading a file from the internet, scraping a web page, querying an API, etc.) and importing that data into your programming environment (e.g., Jupyter Notebook).

Assess

Assess data for:

Quality: issues with content. Low quality data is also known as dirty data. Tidiness: issues with structure that prevent easy analysis. Untidy data is also known as messy data. Tidy data requirements: Each variable forms a column. Each observation forms a row. Each type of observational unit forms a table.

Types of assessment:

Visual assessment: scrolling through the data in your preferred software application (Google Sheets, Excel, a text editor, etc.). Programmatic assessment: using code to view specific portions and summaries of the data (pandas' head, tail, and info methods, for example). Clean Types of cleaning:Manual (not recommended unless the issues are single occurrences)

Programmatic

The programmatic data cleaning process: Define: convert our assessments into defined cleaning tasks. These definitions also serve as an instruction list so others (or yourself in the future) can look at your work and reproduce it. Code: convert those definitions to code and run that code. Test: test your dataset, visually or with code, to make sure your cleaning operations worked.

Gathering Data

- gathering data from twitter-archive-enhanced.csv
- download programmatically from URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv (https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv)
- query the Twitter API for each tweet's JSON data using Python's Tweepy library

```
In [1]: import pandas as pd
        import numpy as np
        import tweepy
        import time
        import requests
        import json
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [2]: df=pd.read_csv('twitter-archive-enhanced.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [4]:
```
df.head()
```

Out[4]:

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

In [5]:
```
## download programmatically from URL: https://d17h27t6h515a5.cloudfront.net/topher/2017
url='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions
r=requests.get(url)
with open('image-predictions.tsv','wb') as f:
    f.write(r.content)
```

In [6]:
```
ip=pd.read_csv('image-predictions.tsv',sep='\t')
```

In [7]:
```
ip.head()
```

Out[7]:

	tweet_id	jpg_url	img_num	p1	p
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridgeback	0.
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pinscher	0.

In [8]:
```
### Tweepy
#https://realpython.com/twitter-bot-python-tweepy/
#http://docs.tweepy.org/en/latest/api.html
# Auth to Twitter
consumer_key='QxZvPDdszap2wuBVt5fFoqHVJ'
consumer_secret='aaqfQjCHz7j4hN9Ke1jWRpB15UVNcPHV3UzSHV3Qi91nebw189'
access_token='1330259629511675908-LkaV6O37C1eefxHQpsbmYqp8bEDOyA'
access_secret='jCrbLCUzzmouV5dtLfxRWgOw6v40uvScnEmnQrQxbcOC6'
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api= tweepy.API(auth_handler = auth,parser = tweepy.parsers.JSONParser(),wait_on_rate_li
```

In [9]:
```python
#http://docs.tweepy.org/en/latest/extended_tweets.html
Tweets=[]
error_tweets=[]


for tweet_id in df['tweet_id']:
    try:

        tweet = api.get_status(tweet_id, tweet_mode='extended')
        Tweets.append(tweet)
        print('ID : '+str(tweet_id)+'    Test:PASSED')
    except:
        error_tweets.append(tweet_id)

        print('ID :' + str(tweet_id)+'     Test:ERROR' )
print('------------------------------------------------------------------------------------------------------------
```

```
ID : 698178924120031232    Test:PASSED
ID : 697995514407682048    Test:PASSED
ID : 697990423684476929    Test:PASSED
ID : 697943111201378304    Test:PASSED
ID : 697881462549430272    Test:PASSED
ID : 697630435728322560    Test:PASSED
ID : 697616773278015490    Test:PASSED
ID : 697596423848730625    Test:PASSED
ID : 697575480820686848    Test:PASSED
ID : 697516214579523584    Test:PASSED
ID : 697482927769255936    Test:PASSED
ID : 697463031882764288    Test:PASSED
ID : 697270446429966336    Test:PASSED
ID :697259378236399616     Test:ERROR
ID : 697255105972801536    Test:PASSED
ID : 697242256848379904    Test:PASSED
ID : 696900204696625153    Test:PASSED
ID : 696894894812565505    Test:PASSED
ID : 696886256886657024    Test:PASSED
ID : 696877980375769088    Test:PASSED
```

In [10]:
```python
len(Tweets)
```

Out[10]:
```
2319
```

```
In [11]: error_tweets
```

```
Out[11]: [888202515573088257,
          873697596434513921,
          872668790621863937,
          872261713294495745,
          869988702071779329,
          866816280283807744,
          861769973181624320,
          856602993587888130,
          851953902622658560,
          845459076796616705,
          844704788403113984,
          842892208864923648,
          837366284874571778,
          837012587749474308,
          829374341691346946,
          827228250799742977,
          812747805718642688,
          812709060537683968,
          802247111496568832,
          779123168116150273,
          775096608509886464,
          771004394259247104,
          770743923962707968,
          759566828574212096,
          758041019896193024,
          754011816964026368,
          752701944171524096,
          746906459439529985,
          708479650088034305,
          707629649552134146,
          697259378236399616,
          680055455951884288,
          672267570918129665,
          670826280409919488,
          669353438988365824,
          667782464991965184,
          666104133288665088]
```

```
In [12]: len(error_tweets)
```

```
Out[12]: 37
```

In [13]:
```
error=[]
for tweet_id in error_tweets:
    try:

        tweet = api.get_status(tweet_id, tweet_mode='extended')
        Tweets.append(tweet)
        print('ID : '+str(tweet_id)+'    Test:PASSED')
    except:
        error.append(tweet_id)

        print('ID :' + str(tweet_id)+'     Test:ERROR' )
print('-----------------------------------------------------------------------------------
```

```
ID :888202515573088257    Test:ERROR
ID :873697596434513921    Test:ERROR
ID :872668790621863937    Test:ERROR
ID :872261713294495745    Test:ERROR
ID :869988702071779329    Test:ERROR
ID :866816280283807744    Test:ERROR
ID :861769973181624320    Test:ERROR
ID :856602993587888130    Test:ERROR
ID :851953902622658560    Test:ERROR
ID :845459076796616705    Test:ERROR
ID :844704788403113984    Test:ERROR
ID :842892208864923648    Test:ERROR
ID :837366284874571778    Test:ERROR
ID :837012587749474308    Test:ERROR
ID :829374341691346946    Test:ERROR
ID :827228250799742977    Test:ERROR
ID :812747805718642688    Test:ERROR
ID : 812709060537683968   Test:PASSED
ID :802247111496568832    Test:ERROR
ID :779123168116150273    Test:ERROR
ID :775096608509886464    Test:ERROR
ID :771004394259247104    Test:ERROR
ID :770743923962707968    Test:ERROR
ID :759566828574212096    Test:ERROR
ID : 758041019896193024   Test:PASSED
ID :754011816964026368    Test:ERROR
ID : 752701944171524096   Test:PASSED
ID : 746906459439529985   Test:PASSED
ID : 708479650088034305   Test:PASSED
ID : 707629649552134146   Test:PASSED
ID : 697259378236399616   Test:PASSED
ID :680055455951884288    Test:ERROR
ID : 672267570918129665   Test:PASSED
ID : 670826280409919488   Test:PASSED
ID : 669353438988365824   Test:PASSED
ID : 667782464991965184   Test:PASSED
ID : 666104133288665088   Test:PASSED
---------------------------------------------------------------------------------------
----------------------------
```

In [14]:
```
len(error)
```

Out[14]:
```
25
```

In [15]:
```
#https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/
# storing data as tweet_json.txt
with open('tweet_json.txt', 'w') as f:
    json.dump(Tweets, f)
```

In [16]:
```
#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_json.html
tweet_df=pd.read_json('tweet_json.txt')
tweet_df.info()
#tweet_df=tweet_df.T  #### Transpose
#tweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2331 entries, 0 to 2330
Data columns (total 32 columns):
contributors                     0 non-null float64
coordinates                      0 non-null float64
created_at                       2331 non-null datetime64[ns]
display_text_range               2331 non-null object
entities                         2331 non-null object
extended_entities                2059 non-null object
favorite_count                   2331 non-null int64
favorited                        2331 non-null bool
full_text                        2331 non-null object
geo                              0 non-null float64
id                               2331 non-null int64
id_str                           2331 non-null int64
in_reply_to_screen_name          77 non-null object
in_reply_to_status_id            77 non-null float64
in_reply_to_status_id_str        77 non-null float64
in_reply_to_user_id              77 non-null float64
in_reply_to_user_id_str          77 non-null float64
is_quote_status                  2331 non-null bool
lang                             2331 non-null object
place                            1 non-null object
possibly_sensitive               2197 non-null float64
possibly_sensitive_appealable    2197 non-null float64
quoted_status                    24 non-null object
quoted_status_id                 26 non-null float64
quoted_status_id_str             26 non-null float64
quoted_status_permalink          26 non-null object
retweet_count                    2331 non-null int64
retweeted                        2331 non-null bool
retweeted_status                 163 non-null object
source                           2331 non-null object
truncated                        2331 non-null bool
user                             2331 non-null object
dtypes: bool(4), datetime64[ns](1), float64(11), int64(4), object(12)
memory usage: 519.1+ KB
```

In [17]:
```
dfjson = pd.DataFrame(tweet_df, columns = ['id', 'favorite_count', 'retweet_count'])
dfjson.head()
```

Out[17]:

	id	favorite_count	retweet_count
0	892420643555336193	35390	7477
1	892177421306343426	30639	5548
2	891815181378084864	23033	3671
3	891689557279858688	38686	7649
4	891327558926688256	36969	8250

In [18]:
```
dfjson.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2331 entries, 0 to 2330
Data columns (total 3 columns):
id                 2331 non-null int64
favorite_count     2331 non-null int64
retweet_count      2331 non-null int64
dtypes: int64(3)
memory usage: 54.7 KB
```

Assessing Data for this Project

- Detect and document at least eight (8) quality issues and two (2) tidiness issues

In [19]:
```
df
```

Out[19]:

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

```
In [20]: ip
```

Out[20]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spa
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redb
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shep
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridgel
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pins
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_mountain_
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	box_t
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	c
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	shopping_
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	miniature_po
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg	1	golden_retri
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAlOoZ.jpg	1	Gordon_s
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg	1	Walker_ho
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg	1	
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg	1	bloodho
15	666099513787052032	https://pbs.twimg.com/media/CT51-JJUEAA6hV8.jpg	1	Lh
16	666102155909144576	https://pbs.twimg.com/media/CT54YGiWUAEZnoK.jpg	1	English_s
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1	
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1	desktop_comp
19	666273097616637952	https://pbs.twimg.com/media/CT8T1mtUwAA3aqm.jpg	1	Italian_greyho
20	666287406224695296	https://pbs.twimg.com/media/CT8g3BpUEAAuFjg.jpg	1	Maltese_
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1	three-toed_s
22	666337882303524864	https://pbs.twimg.com/media/CT9OwFIWEAMuRje.jpg	1	
23	666345417576210432	https://pbs.twimg.com/media/CT9Vn7PWoAA_ZCM.jpg	1	golden_retri
24	666353288456101888	https://pbs.twimg.com/media/CT9cx0tUEAAhNN_.jpg	1	malar
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1	guinea
26	666373753744588802	https://pbs.twimg.com/media/CT9vZEYWUAAlZ05.jpg	1	coated_wheaten_te
27	666396247373291520	https://pbs.twimg.com/media/CT-D2ZHWIAA3gK1.jpg	1	Chihua
28	666407126856765440	https://pbs.twimg.com/media/CT-NvwmW4AAugGZ.jpg	1	black-and-tan_coonho
29	666411507551481857	https://pbs.twimg.com/media/CT-RugiWIAELEaq.jpg	1	c
...	...	...	...	
2045	886366144734445568	https://pbs.twimg.com/media/DE0BTnQUwAApKEH.jpg	1	French_bul
2046	886680336477933568	https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg	1	conver
2047	886736880519319552	https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg	1	ku
2048	886983233522544640	https://pbs.twimg.com/media/DE8yicJW0AAAvBJ.jpg	2	Chihua
2049	887101392804085760	https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg	1	Samo
2050	887343217045368832	https://pbs.twimg.com/ext_tw_video_thumb/88734...	1	Mexican_hair
2051	887473957103951883	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	Pemb
2052	887517139158093824	https://pbs.twimg.com/ext_tw_video_thumb/88751...	1	limou
2053	887705289381826560	https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg	1	ba
2054	888078434458587136	https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg	1	French_bul

	tweet_id	jpg_url	img_num	
2055	888202515573088257	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	Pemb
2056	888554962724278272	https://pbs.twimg.com/media/DFTH_O-UQAACu20.jpg	3	Siberian_h
2057	888804989199671297	https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg	1	golden_retri
2058	888917238123831296	https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg	1	golden_retri
2059	889278841981685760	https://pbs.twimg.com/ext_tw_video_thumb/88927...	1	whi
2060	889531135344209921	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	1	golden_retri
2061	889638837579907072	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	1	French_bul
2062	889665388333682689	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	1	Pemb
2063	889880896479866881	https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg	1	French_bul
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	Sam
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	Pemb
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	Irish_te
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	Pomera
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	Appenz
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	Chesapeake_Bay_retri
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	ba
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	paper_t
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	Chihua
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	Chihua
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	ora

2075 rows × 12 columns

```
In [21]: dfjson
```

```
Out[21]:
                         id  favorite_count  retweet_count
   0     892420643555336193           35390           7477
   1     892177421306343426           30639           5548
   2     891815181378084864           23033           3671
   3     891689557279858688           38686           7649
   4     891327558926688256           36969           8250
   5     891087950875897856           18634           2759
   6     890971913173991426           10828           1792
   7     890729181411237888           59632          16726
   8     890609185150312448           25648           3815
   9     890240255349198849           29261           6489
  10     890006608113172480           28211           6500
  11     889880896479866881           25666           4415
  12     889665388333682689           44085           8858
  13     889638837579907072           24799           3970
  14     889531135344209921           13953           1998
  15     889278841981685760           23149           4719
  16     888917238123831296           26752           3977
  17     888804989199671297           23481           3747
  18     888554962724278272           18109           3066
  19     888078434458587136           20011           3074
  20     887705289381826560           27823           4785
  21     887517139158093824           42605          10437
  22     887473957103951883           63084          15931
  23     887343217045368832           30936           9332
  24     887101392804085760           28149           5284
  25     886983233522544640           31977           6778
  26     886736880519319552           10979           2825
  27     886680336477933568           20663           3967
  28     886366144734445568           19415           2805
  29     886267009285017600             110              4
 ...                    ...             ...            ...
2301     666268910803644416              94             32
2302     666102155909144576              69             11
2303     666099513787052032             140             57
2304     666094000022159362             153             66
2305     666082916733198337             101             41
2306     666073100786774016             285            141
2307     666071193221509120             135             52
2308     666063827256086533             437            191
2309     666058600524156928             104             51
2310     666057090499244032             263            120
```

	id	favorite_count	retweet_count
2311	666055525042405380	404	214
2312	666051853826850816	1099	752
2313	666050758794694657	122	51
2314	666049248165822465	96	40
2315	666044226329800704	265	124
2316	666033412701032449	109	39
2317	666029285002620928	119	41
2318	666020888022790149	2355	449
2319	812709060537683968	6586	1428
2320	758041019896193024	2648	364
2321	752701944171524096	0	2789
2322	746906459439529985	2841	289
2323	708479650088034305	2473	657
2324	707629649552134146	2495	838
2325	697259378236399616	3207	974
2326	672267570918129665	1399	570
2327	670826280409919488	5168	3771
2328	669353438988365824	589	241
2329	667782464991965184	386	227
2330	666104133288665088	13298	5814

2331 rows × 3 columns

```
In [22]: sum(df.duplicated())
Out[22]: 0
```

```
In [23]: sum(ip.duplicated())
Out[23]: 0
```

```
In [24]: sum(dfjson.duplicated())   ## Quality 1 duplicates in dfjson
Out[24]: 0
```

```
In [25]: sum(ip.jpg_url.duplicated()) ## Quality 2
Out[25]: 66
```

In [26]:
```
df.info() # quality 3 fix type in columns  in_reply_to_status_id,in_reply_to_user_id,ret
#Tidiness 1 doggo, floofer, pupper and puppo columns to be in one column
#Quality 4 source column has HTML
# Tidiness 2 name of column id in table dfjson need to change to tweet_id
# quality 5 rating_denominator should all to be 10
# quality 6 timestamp to datetime
# quality 7 null values
# quality 8 rating= rating_numerator/rating_denominator
# quality 9 types of tweet id should be object not int
```

```
<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 [27]:
```
ip.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 [28]:
```
dfjson.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2331 entries, 0 to 2330
Data columns (total 3 columns):
id                2331 non-null int64
favorite_count    2331 non-null int64
retweet_count     2331 non-null int64
dtypes: int64(3)
memory usage: 54.7 KB
```

Quality

- duplicates in dfjson (tweet_json)
- duplicates in ip.jpg_url (image)
- fix type in columns in_reply_to_status_id,in_reply_to_user_id,retweeted_status_id,retweeted_status_user_id
- source column has HTML
- rating_denominator should all to be 10
- timestamp to datetime
- null values
- rating= rating_numerator/rating_denominator
- types of tweet id should be object not int
- delect unneeded columns
- select two types of dogs at same stage
- p1 p2 p3 lower cases
- p1 p2 p3 space with _

Tidiness

- doggo, floofer, pupper and puppo columns to be in one column
- each observation forms a row, each type of observational unit forms a table
- All tables should be part of one dataset

```
In [29]: df_clean=df.copy()
```

```
In [30]: ip_clean=ip.copy()
```

```
In [31]: dfjson_clean=dfjson.copy()
```

define

- Remove duplicates in dfjson (tweet_json)

Code and Test

```
In [32]: #https://www.geeksforgeeks.org/python-pandas-dataframe-drop_duplicates/
         dfjson_clean.drop_duplicates(keep = False, inplace = True)
         sum(dfjson_clean.duplicated())

Out[32]: 0
```

define

- Remove duplicates in ip.jpg_url (image)

Code and Test

In [33]:
```
#https://www.geeksforgeeks.org/python-pandas-dataframe-drop_duplicates/
ip_clean.drop_duplicates(subset ="jpg_url",keep = False, inplace = True)
sum(ip_clean.jpg_url.duplicated())
```
Out[33]:
```
0
```

define

- fix type in columns in_reply_to_status_id,in_reply_to_user_id,retweeted_status_id,retweeted_status_user_id
- types of tweet id should be object not int

Code

In [34]:
```
df_clean['in_reply_to_status_id']=df_clean['in_reply_to_status_id'].astype('object')
```
In [35]:
```
df_clean['in_reply_to_user_id']=df_clean['in_reply_to_user_id'].astype('object')
```
In [36]:
```
df_clean['retweeted_status_id']=df_clean['retweeted_status_id'].astype('object')
```
In [37]:
```
df_clean['retweeted_status_user_id']=df_clean['retweeted_status_user_id'].astype('object
```
In [38]:
```
df_clean['tweet_id']=df_clean['tweet_id'].astype('object')
```

Test

In [39]:
```
df_clean.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null object
in_reply_to_status_id         78 non-null object
in_reply_to_user_id           78 non-null object
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null object
retweeted_status_user_id      181 non-null object
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: int64(2), object(15)
memory usage: 313.0+ KB
```

Code

In [40]:
```
ip_clean['tweet_id']=ip_clean['tweet_id'].astype('object')
```

Test

In [41]:
```
ip_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1943 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    1943 non-null object
jpg_url     1943 non-null object
img_num     1943 non-null int64
p1          1943 non-null object
p1_conf     1943 non-null float64
p1_dog      1943 non-null bool
p2          1943 non-null object
p2_conf     1943 non-null float64
p2_dog      1943 non-null bool
p3          1943 non-null object
p3_conf     1943 non-null float64
p3_dog      1943 non-null bool
dtypes: bool(3), float64(3), int64(1), object(5)
memory usage: 157.5+ KB
```

Code

In [42]:
```
dfjson_clean['id']=dfjson_clean['id'].astype('object')
```

Test

In [43]:
```
dfjson_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2331 entries, 0 to 2330
Data columns (total 3 columns):
id                2331 non-null object
favorite_count    2331 non-null int64
retweet_count     2331 non-null int64
dtypes: int64(2), object(1)
memory usage: 72.8+ KB
```

define

- source column has HTML

Code

In [44]:
```
#https://github.com/tkannab/Udacity-DAND-T2-P3-DW/blob/master/wrangle_act.ipynb
df_clean['source'] = df_clean['source'].str.extract('(<a href="https?)(:\/\/)(.+)(">)(.+
                                                                              expand = True)[4]
df_clean['source'] = df_clean['source'].astype('category')
```

Test

In [45]:
```
df_clean
```

Out[45]:

	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	This is Phinea mystical boy. O
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	Twitter for iPhone	This is Tilly. S checking pup c
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	Twitter for iPhone	This is Archie. He Norwegian P
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	Twitter for iPhone	This is D commenced a sn

define

- rating_denominator should all to be 10

Code

In [46]:
```
df_clean.loc[(df_clean.rating_denominator != 10), 'rating_denominator'] = 10
```

Test

In [47]:
```
sum(df_clean.rating_denominator.duplicated())
```

Out[47]:
```
2355
```

define

- timestamp to datetime

Code

In [48]:
```
df_clean['timestamp'] = pd.to_datetime(df_clean['timestamp'])
```

Test

In [49]:
```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null object
in_reply_to_status_id         78 non-null object
in_reply_to_user_id           78 non-null object
timestamp                     2356 non-null datetime64[ns]
source                        2356 non-null category
text                          2356 non-null object
retweeted_status_id           181 non-null object
retweeted_status_user_id      181 non-null object
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: category(1), datetime64[ns](1), int64(2), object(13)
memory usage: 297.1+ KB
```

define

- null values

Code

In [50]:
```
df_clean['rating']=df_clean['rating_numerator']/df_clean['rating_denominator']
```

Test

```
In [51]: df_clean['rating'].value_counts()
```

```
Out[51]: 1.2      558
         1.1      464
         1.0      461
         1.3      351
         0.9      158
         0.8      102
         0.7       55
         1.4       54
         0.5       37
         0.6       32
         0.3       19
         0.4       17
         0.1        9
         0.2        9
         7.5        2
         1.5        2
         42.0       2
         0.0        2
         96.0       1
         5.0        1
         8.0        1
         4.5        1
         6.0        1
         2.0        1
         20.4       1
         8.4        1
         14.3       1
         2.7        1
         14.4       1
         1.7        1
         8.8        1
         2.6        1
         12.1       1
         4.4        1
         16.5       1
         9.9        1
         18.2       1
         2.4        1
         66.6       1
         177.6      1
         Name: rating, dtype: int64
```

define

- p1 p2 p3 lower cases
- p1 p2 p3 space with _

Code and Test

In [52]:
```
#https://www.geeksforgeeks.org/apply-uppercase-to-a-column-in-pandas-dataframe/
ip_clean['p1']=ip_clean['p1'].str.title()
ip_clean['p2']=ip_clean['p2'].str.title()
ip_clean['p3']=ip_clean['p3'].str.title()
ip_clean
```

Out[52]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_Springer_Sp
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Red
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_She
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_Ridge
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniature_Pin
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese_Mountain
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg	1	Box_
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg	1	(
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg	1	Shopping
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg	1	Miniature_P
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg	1	Golden_Ret
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAlOoZ.jpg	1	Gordon_
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg	1	Walker_H
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg	1	
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg	1	Bloodh
15	666099513787052032	https://pbs.twimg.com/media/CT51-JJUEAA6hV8.jpg	1	L
16	666102155909144576	https://pbs.twimg.com/media/CT54YGiWUAEZnoK.jpg	1	English_
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg	1	
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg	1	Desktop_Com
19	666273097616637952	https://pbs.twimg.com/media/CT8T1mtUwAA3aqm.jpg	1	Italian_Greyh
20	666287406224695296	https://pbs.twimg.com/media/CT8g3BpUEAAuFjg.jpg	1	Maltese
21	666293911632134144	https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg	1	Three-Toed_
22	666337882303524864	https://pbs.twimg.com/media/CT9OwFIWEAMuRje.jpg	1	
23	666345417576210432	https://pbs.twimg.com/media/CT9Vn7PWoAA_ZCM.jpg	1	Golden_Ret
24	666353288456101888	https://pbs.twimg.com/media/CT9cx0tUEAAhNN_.jpg	1	Mala
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1	Guinea
26	666373753744588802	https://pbs.twimg.com/media/CT9vZEYWUAAlZ05.jpg	1	Coated_Wheaten_T
27	666396247373291520	https://pbs.twimg.com/media/CT-D2ZHWIAA3gK1.jpg	1	Chihu
28	666407126856765440	https://pbs.twimg.com/media/CT-NvwmW4AAugGZ.jpg	1	Black-And-Tan_Coonh
29	666411507551481857	https://pbs.twimg.com/media/CT-RugiWIAELEaq.jpg	1	
...	...	...	...	
2043	885984800019947520	https://pbs.twimg.com/media/DEumeWWV0AA-Z61.jpg	1	Blenheim_Sp
2044	886258384151887873	https://pbs.twimg.com/media/DEyfTG4UMAE4aE9.jpg	1	
2045	886366144734445568	https://pbs.twimg.com/media/DE0BTnQUwAApKEH.jpg	1	French_Bu
2046	886680336477933568	https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg	1	Conve
2047	886736880519319552	https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg	1	Ku
2048	886983233522544640	https://pbs.twimg.com/media/DE8yicJW0AAAvBJ.jpg	2	Chihu

	tweet_id	jpg_url	img_num	
2049	887101392804085760	https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg	1	Sam
2050	887343217045368832	https://pbs.twimg.com/ext_tw_video_thumb/88734...	1	Mexican_Ha
2052	887517139158093824	https://pbs.twimg.com/ext_tw_video_thumb/88751...	1	Limo
2053	887705289381826560	https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg	1	B
2054	888078434458587136	https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg	1	French_Bu
2056	888554962724278272	https://pbs.twimg.com/media/DFTH_O-UQAACu20.jpg	3	Siberian_H
2057	888804989199671297	https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg	1	Golden_Ret
2058	888917238123831296	https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg	1	Golden_Ret
2059	889278841981685760	https://pbs.twimg.com/ext_tw_video_thumb/88927...	1	Wh
2060	889531135344209921	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	1	Golden_Ret
2061	889638837579907072	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	1	French_Bu
2062	889665388333682689	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	1	Peml
2063	889880896479866881	https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg	1	French_Bu
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	Sam
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	Peml
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	Irish_T
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	Pomera
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	Appen
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	Chesapeake_Bay_Ret
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	B
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	Paper_
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	Chihu
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	Chihu
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	Or

1943 rows × 12 columns

Code and Test

In [53]:
```
ip_clean['p1']=ip_clean['p1'].str.replace('_', ' ')
ip_clean['p2']=ip_clean['p2'].str.replace('_', ' ')
ip_clean['p3']=ip_clean['p3'].str.replace('_', ' ')
ip_clean
#in some column dont use _ use - need to change to space
ip_clean['p1']=ip_clean['p1'].str.replace('-', ' ')
ip_clean['p2']=ip_clean['p2'].str.replace('-', ' ')
ip_clean['p3']=ip_clean['p3'].str.replace('-', ' ')
ip_clean
```

Out[53]:

	tweet_id	jpg_url	img_num	p1	p1_
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh Springer Spaniel	0.46
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	Redbone	0.50
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German Shepherd	0.59
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian Ridgeback	0.40
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	Miniature Pinscher	0.56
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg	1	Bernese Mountain Dog	0.65

define

Tidiness

- doggo, floofer, pupper and puppo columns to be in one column

Code

In [54]:
```
#https://stackoverflow.com/questions/33098383/merge-multiple-column-values-into-one-colu
#https://stackoverflow.com/questions/24619145/rename-none-value-in-pandas
cols = ['doggo', 'floofer', 'pupper', 'puppo']
df_clean['doggo'].replace('None', np.nan, inplace=True)
df_clean['floofer'].replace('None', np.nan, inplace=True)
df_clean['puppo'].replace('None', np.nan, inplace=True)
df_clean['pupper'].replace('None', np.nan, inplace=True)
df_clean["Stage"] = df_clean[cols].apply(lambda x: ','.join(x.dropna()), axis=1)
#df_clean['Stage']= df_clean[df_clean.columns[13:17]].apply(lambda x: ','.join(x.dropna(
#df_clean['Stage'] = df_clean[df_clean.columns[13:-2]].apply(lambda x != None: '|'.join(
```

Test

```
In [55]: df_clean
```

Out[55]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56	Twitter for iPhone	This is Phinea mystical boy. O
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. S checking pup (
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He Norwegian P
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51	Twitter for iPhone	This is D commenced a sn
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. I like you to :

```
In [56]: #checking after replace None  to NAN and combining
         df_clean[(df_clean.doggo=="doggo") & (df_clean.pupper=="pupper")][['tweet_id','Stage']]
```

Out[56]:

	tweet_id	Stage
460	817777686764523521	doggo,pupper
531	808106460588765185	doggo,pupper
565	802265048156610565	doggo,pupper
575	801115127852503040	doggo,pupper
705	785639753186217984	doggo,pupper
733	781308096455073793	doggo,pupper
778	775898661951791106	doggo,pupper
822	770093767776997377	doggo,pupper
889	759793422261743616	doggo,pupper
956	751583847268179968	doggo,pupper
1063	741067306818797568	doggo,pupper
1113	733109485275860992	doggo,pupper

```
In [57]: df_clean = df_clean.drop(['doggo','floofer','puppo','pupper'],axis=1)
In [58]: df_clean
```

Out[58]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56	Twitter for iPhone	This is Phinea mystical boy. O
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. S checking pup (
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He Norwegian F
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51	Twitter for iPhone	This is D commenced a sni
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. I like you to s

```
In [59]: df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 15 columns):
tweet_id                      2356 non-null object
in_reply_to_status_id         78 non-null object
in_reply_to_user_id           78 non-null object
timestamp                     2356 non-null datetime64[ns]
source                        2356 non-null category
text                          2356 non-null object
retweeted_status_id           181 non-null object
retweeted_status_user_id      181 non-null object
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
rating                        2356 non-null float64
Stage                         2356 non-null object
dtypes: category(1), datetime64[ns](1), float64(1), int64(2), object(10)
memory usage: 260.3+ KB
```

define

- name of column id in table dfjson need to change to tweet_id

Code

```
In [60]: dfjson_clean=dfjson_clean.rename(columns={'id': 'tweet_id'})
```

Test

```
In [61]: dfjson_clean
```

```
Out[61]:
                    tweet_id  favorite_count  retweet_count
   0  892420643555336193           35390           7477
   1  892177421306343426           30639           5548
   2  891815181378084864           23033           3671
   3  891689557279858688           38686           7649
   4  891327558926688256           36969           8250
   5  891087950875897856           18634           2759
   6  890971913173991426           10828           1792
   7  890729181411237888           59632          16726
   8  890609185150312448           25648           3815
   9  890240255349198849           29261           6489
  10  890006608113172480           28211           6500
  11  889880896479866881           25666           4415
  12  889665388333682689           44085           8858
  13  889638837579907072           24799           3970
  14  889531135344209921           13953           1998
  15  889278841981685760           23149           4719
  16  888917238123831296           26752           3977
  17  888804989199671297           23481           3747
  18  888554962724278272           18109           3066
  19  888078434458587136           20011           3074
  20  887705289381826560           27823           4785
  21  887517139158093824           42605          10437
  22  887473957103951883           63084          15931
  23  887343217045368832           30936           9332
  24  887101392804085760           28149           5284
  25  886983233522544640           31977           6778
  26  886736880519319552           10979           2825
  27  886680336477933568           20663           3967
  28  886366144734445568           19415           2805
  29  886267009285017600             110              4
 ...                 ...             ...            ...
2301  666268910803644416              94             32
2302  666102155909144576              69             11
2303  666099513787052032             140             57
2304  666094000022159362             153             66
2305  666082916733198337             101             41
2306  666073100786774016             285            141
2307  666071193221509120             135             52
2308  666063827256086533             437            191
2309  666058600524156928             104             51
2310  666057090499244032             263            120
```

	tweet_id	favorite_count	retweet_count
2311	666055525042405380	404	214
2312	666051853826850816	1099	752
2313	666050758794694657	122	51
2314	666049248165822465	96	40
2315	666044226329800704	265	124
2316	666033412701032449	109	39
2317	666029285002620928	119	41
2318	666020888022790149	2355	449
2319	812709060537683968	6586	1428
2320	758041019896193024	2648	364
2321	752701944171524096	0	2789
2322	746906459439529985	2841	289
2323	708479650088034305	2473	657
2324	707629649552134146	2495	838
2325	697259378236399616	3207	974
2326	672267570918129665	1399	570
2327	670826280409919488	5168	3771
2328	669353438988365824	589	241
2329	667782464991965184	386	227
2330	666104133288665088	13298	5814

2331 rows × 3 columns

define

- merge all dataframes in one

Code

In [66]:
```
merged_df = pd.merge(df_clean, ip_clean, left_on='tweet_id', right_on='tweet_id', how='l
all_dfclean=pd.merge(merged_df, dfjson_clean, left_on='tweet_id', right_on='tweet_id', h
all_dfclean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 0 to 2355
Data columns (total 28 columns):
tweet_id                      2356 non-null object
in_reply_to_status_id         78 non-null object
in_reply_to_user_id           78 non-null object
timestamp                     2356 non-null datetime64[ns]
source                        2356 non-null category
text                          2356 non-null object
retweeted_status_id           181 non-null object
retweeted_status_user_id      181 non-null object
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
rating                        2356 non-null float64
Stage                         2356 non-null object
jpg_url                       1943 non-null object
img_num                       1943 non-null float64
p1                            1943 non-null object
p1_conf                       1943 non-null float64
p1_dog                        1943 non-null object
p2                            1943 non-null object
p2_conf                       1943 non-null float64
p2_dog                        1943 non-null object
p3                            1943 non-null object
p3_conf                       1943 non-null float64
p3_dog                        1943 non-null object
favorite_count                2331 non-null float64
retweet_count                 2331 non-null float64
dtypes: category(1), datetime64[ns](1), float64(7), int64(2), object(17)
memory usage: 517.9+ KB
```

```
In [68]: all_dfclean = all_dfclean[all_dfclean.retweeted_status_id.isnull()]
         #df_1_clean = df_1_clean[df_1_clean.retweeted_status_user_id.isnull()]
         #df_1_clean = df_1_clean[df_1_clean.retweeted_status_timestamp.isnull()]
         all_dfclean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2175 entries, 0 to 2355
Data columns (total 28 columns):
tweet_id                      2175 non-null object
in_reply_to_status_id         78 non-null object
in_reply_to_user_id           78 non-null object
timestamp                     2175 non-null datetime64[ns]
source                        2175 non-null category
text                          2175 non-null object
retweeted_status_id           0 non-null object
retweeted_status_user_id      0 non-null object
retweeted_status_timestamp    0 non-null object
expanded_urls                 2117 non-null object
rating_numerator              2175 non-null int64
rating_denominator            2175 non-null int64
name                          2175 non-null object
rating                        2175 non-null float64
Stage                         2175 non-null object
jpg_url                       1928 non-null object
img_num                       1928 non-null float64
p1                            1928 non-null object
p1_conf                       1928 non-null float64
p1_dog                        1928 non-null object
p2                            1928 non-null object
p2_conf                       1928 non-null float64
p2_dog                        1928 non-null object
p3                            1928 non-null object
p3_conf                       1928 non-null float64
p3_dog                        1928 non-null object
favorite_count                2168 non-null float64
retweet_count                 2168 non-null float64
dtypes: category(1), datetime64[ns](1), float64(7), int64(2), object(17)
memory usage: 478.1+ KB
```

define

- delect unneeded columns

Code

```
In [69]: all_dfclean = all_dfclean.drop(['in_reply_to_status_id','in_reply_to_user_id','retweeted
```

Test

In [70]:
```
all_dfclean.isnull().sum()
```

Out[70]:
```
tweet_id                0
timestamp               0
source                  0
text                    0
rating_numerator        0
rating_denominator      0
name                    0
rating                  0
Stage                   0
jpg_url               247
img_num               247
p1                    247
p1_conf               247
p1_dog                247
p2                    247
p2_conf               247
p2_dog                247
p3                    247
p3_conf               247
p3_dog                247
favorite_count          7
retweet_count           7
dtype: int64
```

In [71]:
```
all_dfclean.dropna(inplace=True)
```

In [72]:
```
all_dfclean.isnull().sum()
```

Out[72]:
```
tweet_id              0
timestamp             0
source                0
text                  0
rating_numerator      0
rating_denominator    0
name                  0
rating                0
Stage                 0
jpg_url               0
img_num               0
p1                    0
p1_conf               0
p1_dog                0
p2                    0
p2_conf               0
p2_dog                0
p3                    0
p3_conf               0
p3_dog                0
favorite_count        0
retweet_count         0
dtype: int64
```

In [73]:
```
all_dfclean
```
Out[73]:

	tweet_id	timestamp	source	text	rating_numerator	rating_denomin
0	892420643555336193	2017-08-01 16:23:56	Twitter for iPhone	This is Phineas. He's a mystical boy. Only eve...	13	
1	892177421306343426	2017-08-01 00:17:27	Twitter for iPhone	This is Tilly. She's just checking pup on you....	13	
2	891815181378084864	2017-07-31 00:18:03	Twitter for iPhone	This is Archie. He is a rare Norwegian Pouncin...	12	
3	891689557279858688	2017-07-30 15:58:51	Twitter for iPhone	This is Darla. She commenced a snooze mid meal...	13	
4	891327558926688256	2017-07-29 16:00:24	Twitter for iPhone	This is Franklin. He would like you to stop ca...	12	

Storing cleaned Data

In [74]:
```
all_dfclean.to_csv('twitter_archive_master.csv')
```

In [75]:
```

df = pd.read_csv('twitter_archive_master.csv')
df.drop(df.columns[0], axis=1)
```

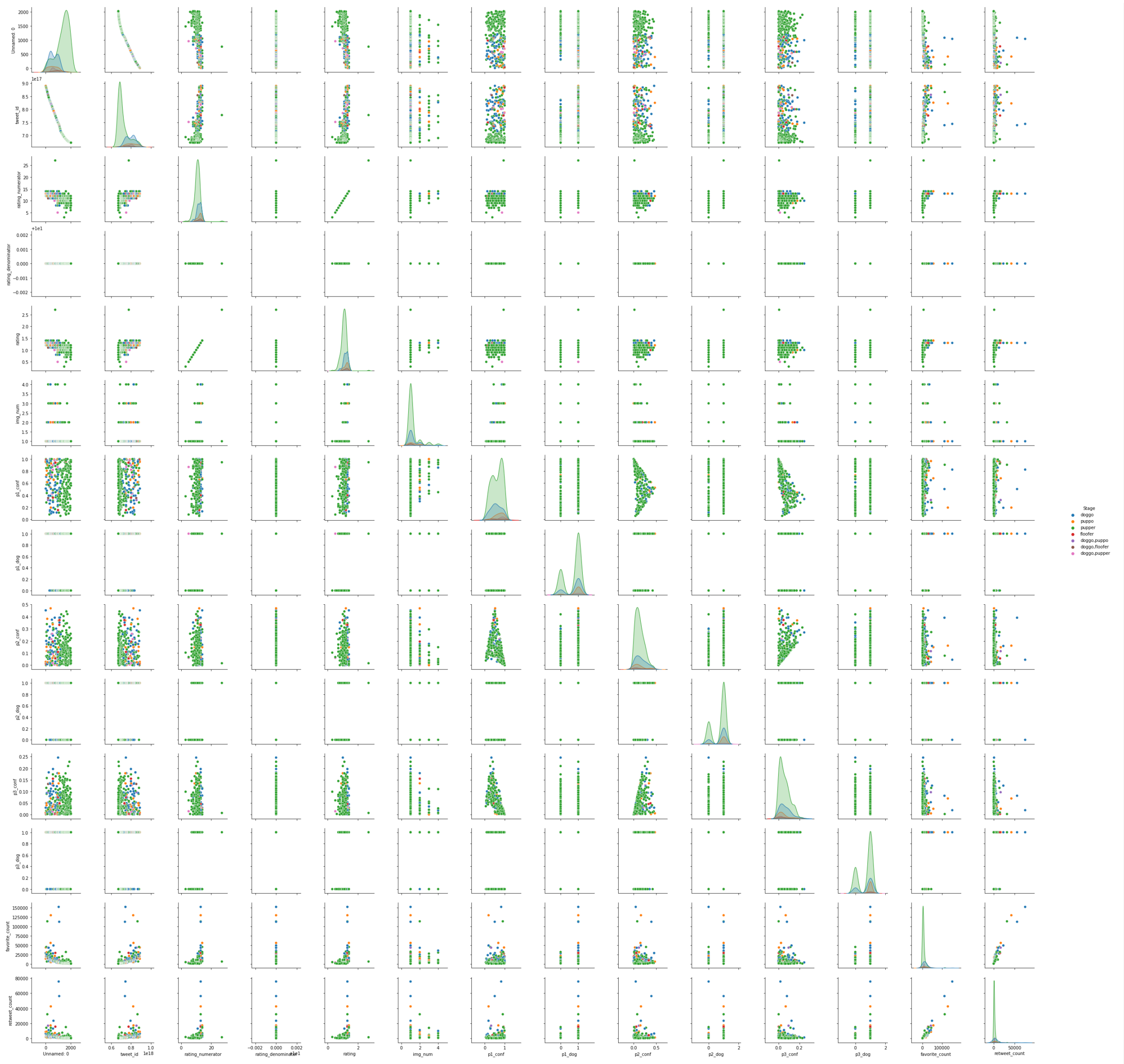
		00:49:46	iPhone	armored polar bear ...			
1915	666051853826850816	2015-11-16 00:35:11	Twitter for iPhone	This is an odd dog. Hard on the outside but lo...	2	10	an
1916	666050758794694657	2015-11-16 00:30:50	Twitter for iPhone	This is a truly beautiful English Wilson Staff...	10	10	a
1917	666049248165822465	2015-11-16 00:24:50	Twitter for iPhone	Here we have a 1949 1st generation vulpix. Enj...	5	10	None
1918	666044226329800704	2015-11-16 00:04:52	Twitter for iPhone	This is a purebred Piers Morgan. Loves to Netf...	6	10	a
1919	666033412701032449	2015-11-15 23:21:54	Twitter for iPhone	Here is a very happy pup. Big fan of well-main...	9	10	a

In [76]:
```
df.describe()
```
Out[76]:

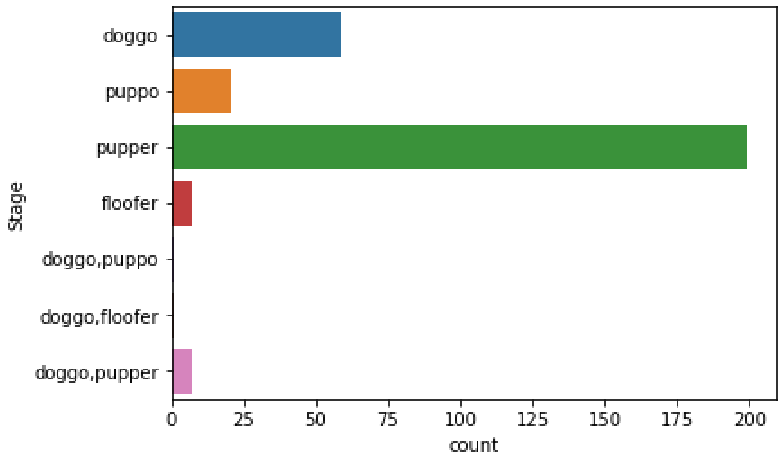
	Unnamed: 0	tweet_id	rating_numerator	rating_denominator	rating	img_num	p1_c
count	1922.000000	1.922000e+03	1922.000000	1922.0	1922.000000	1922.000000	1922.000
mean	1261.070760	7.348195e+17	12.293965	10.0	1.229396	1.201873	0.593
std	681.552866	6.764813e+16	42.267651	0.0	4.226765	0.558719	0.273
min	0.000000	6.660209e+17	0.000000	10.0	0.000000	1.000000	0.044
25%	703.250000	6.755322e+17	10.000000	10.0	1.000000	1.000000	0.359
50%	1308.500000	7.071784e+17	11.000000	10.0	1.100000	1.000000	0.587
75%	1854.750000	7.859140e+17	12.000000	10.0	1.200000	1.000000	0.848
max	2355.000000	8.924206e+17	1776.000000	10.0	177.600000	4.000000	1.000

```
In [77]: #https://seaborn.pydata.org/generated/seaborn.countplot.html
         sns.pairplot(df, hue='Stage');
```

```
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\HP\miniconda3\lib\site-packages\seaborn\distributions.py:305: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

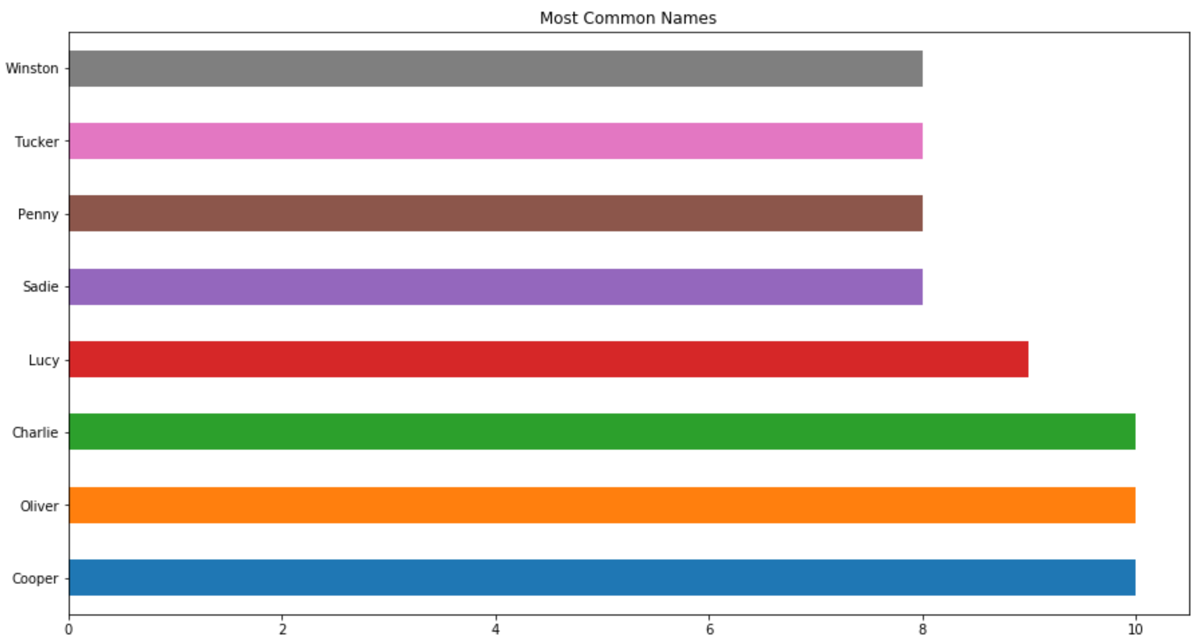


```
In [78]: #https://stackoverflow.com/questions/48043365/how-to-improve-this-seaborn-countplot
         g=sns.countplot(y="Stage", data=df);
         g.set_yticklabels(g.get_yticklabels(),rotation=0);
```

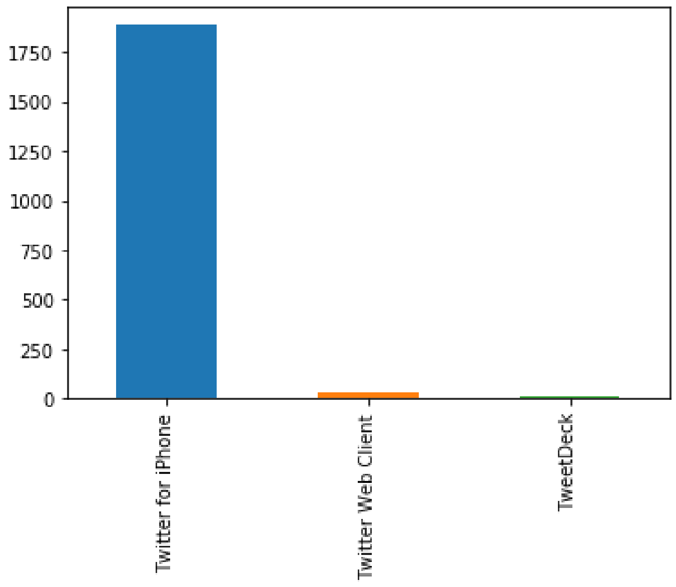


```
In [79]: df.name.value_counts()[2:10].plot.barh( figsize=(15,8), title='Most Common Names')
```

```
Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0x189914a6780>
```

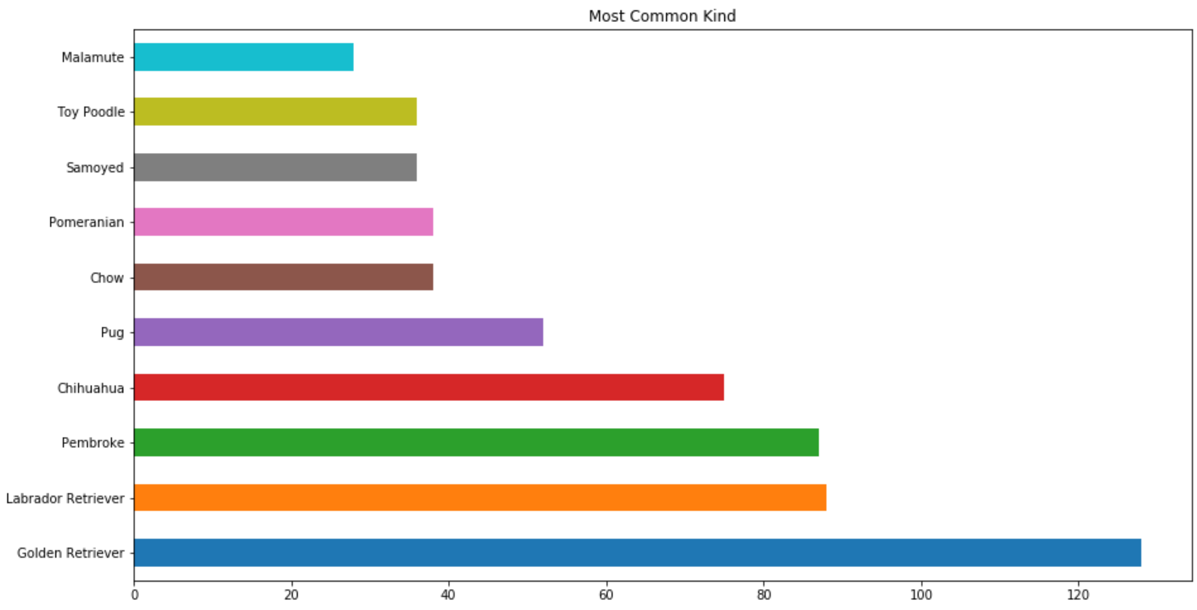


```
In [80]: ax = df['source'].value_counts().plot(kind='bar')
```



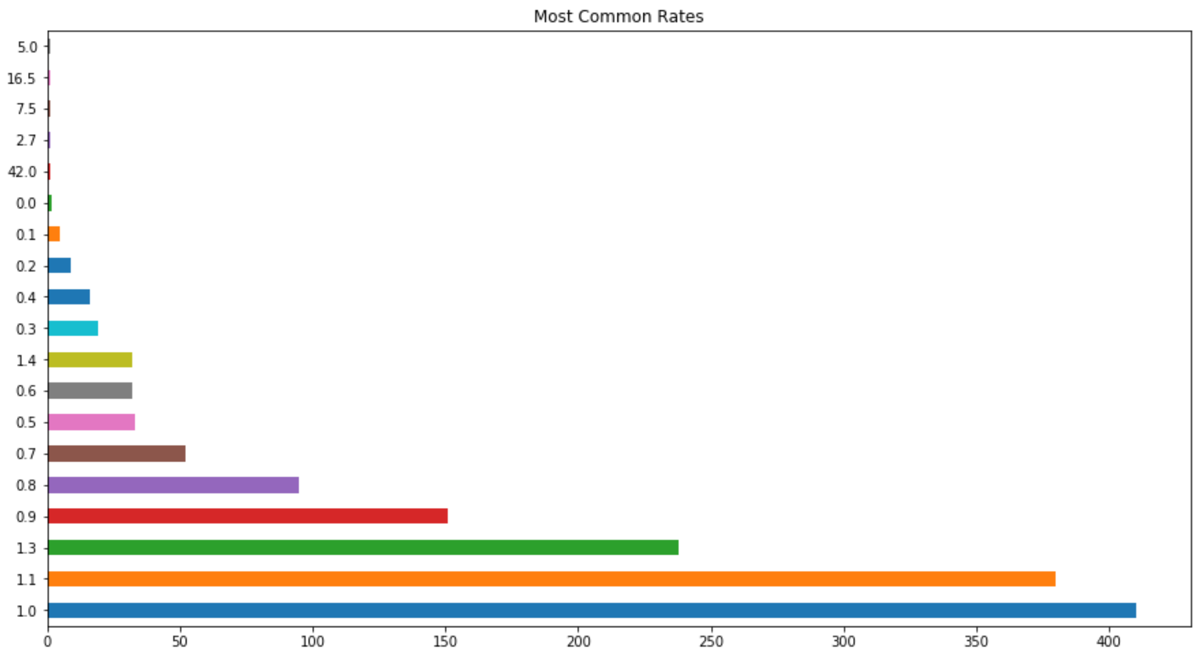
```
In [81]: df.p1.value_counts()[0:10].plot.barh( figsize=(15,8), title='Most Common Kind')
```

```
Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x1898f91c8d0>
```



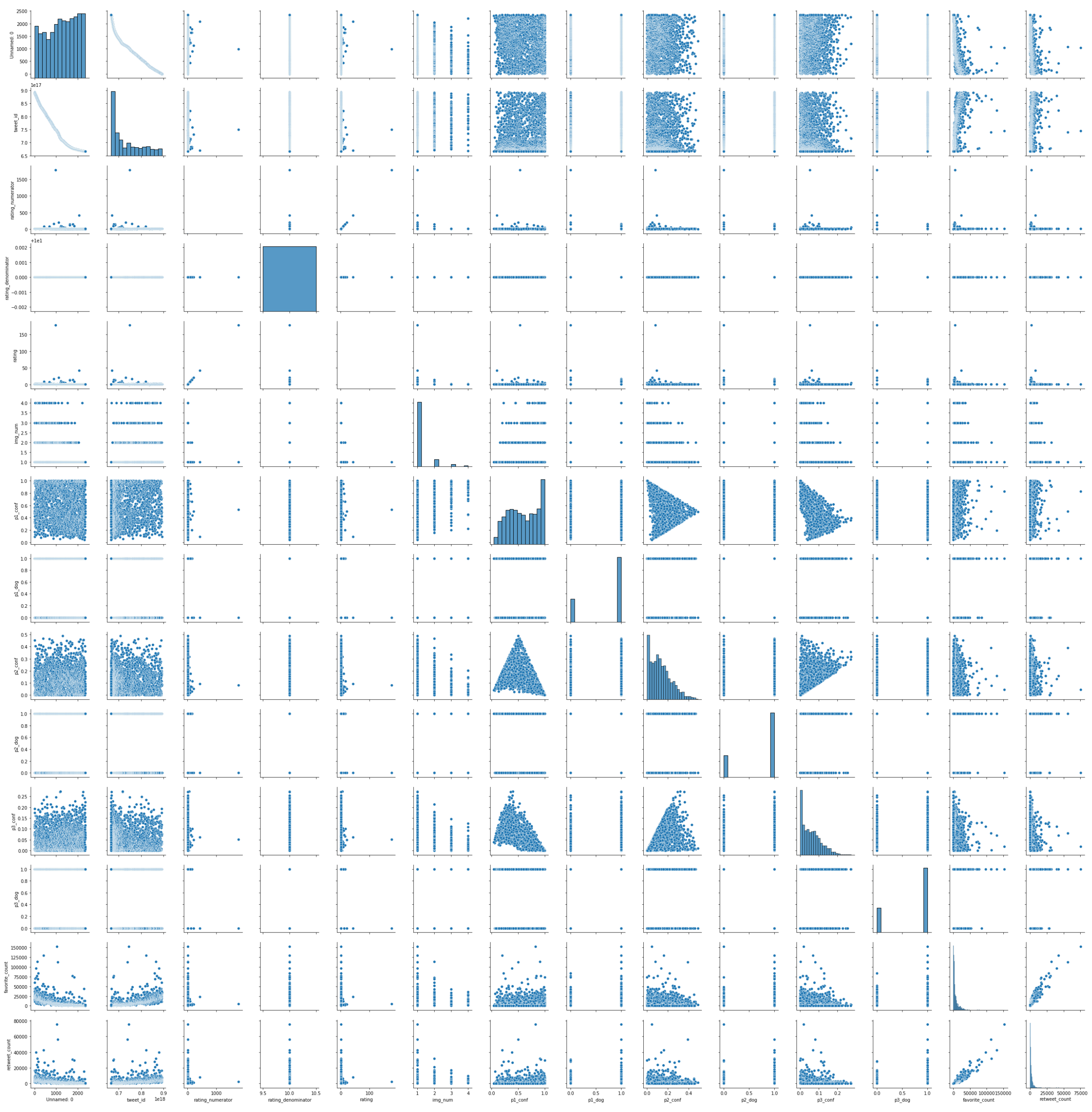
```
In [82]: df.rating.value_counts()[1:5].plot.barh( figsize=(15,8), title='Most Common Rates')
```

```
Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x18993168550>
```



In [83]:
```
sns.pairplot(df);
```

```
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
<string>:6: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for com
patibility.
```



In [ ]:

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```