Project: Wrangling and Analyze Data

Data Gathering

In the cell below, gather **all** three pieces of data for this project and load them in the notebook. **Note:** the methods required to gather each data are different.

1. Directly download the WeRateDogs Twitter archive data (twitter_archive_enhanced.csv)

In [182…
```
import pandas as pd
import numpy as np
import requests

import re
import json
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime

pd.set_option('display.max_columns', 500)
pd.set_option('max_colwidth', 800)

np.random.seed(42)
```

In [183…
```
df_archive = pd.read_csv("twitter-archive-enhanced (1).csv")
df_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [184…
```
df_archive.shape
```

Out[184]: (2356, 17)

1. Use the Requests library to download the tweet image prediction (image_predictions.tsv)

In [185…
```
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-
response = requests.get(url)
response.status_code
```

Out[185]: 200

In [186…
```
with open("image_predictions.tsv", mode='wb') as file:
    file.write(response.content)
```

In [187…
```
df_image_predictions = pd.read_csv("image_predictions.tsv", sep='\t')
```

In [188…
```
df_image_predictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   tweet_id  2075 non-null   int64
 1   jpg_url   2075 non-null   object
 2   img_num   2075 non-null   int64
 3   p1        2075 non-null   object
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool
 6   p2        2075 non-null   object
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool
 9   p3        2075 non-null   object
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [189…
```
df_image_predictions.shape
```

Out[189]: (2075, 12)

1. Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

In [190…
```
#!pip install -U tweepy==4.0
```

In [191…
```
#import tweepy
#from tweepy import OAuthHandler

#consumer_key =
#consumer_secret =
#access_token =
#access_secret =
#auth = OAuthHandler(consumer_key, consumer_secret)
```

```
#auth.set_access_token(access_token, access_secret)
#api = tweepy.API(auth, wait_on_rate_limit=True)
```

In [192…
```
tweet_ids = df_archive.tweet_id.values
len(tweet_ids)
```

Out[192]:
```
2356
```

In [193…
```
#%%time

#import json

# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
#count = 0
#fails_dict = {}
# Save each tweet's returned JSON as a new line in a .txt file
#with open('tweet_json.txt', 'w') as outfile:
    # This loop will likely take 20-30 minutes to run because of Twitter's rate
#    for tweet_id in tweet_ids:
#        count += 1
#        print(str(count) + ": " + str(tweet_id))
#        try:
#            tweet = api.get_status(tweet_id, tweet_mode='extended')
#            json.dump(tweet._json, outfile)
#            print("Success")
#            outfile.write('\n')
#        except Exception as e:
#            print("Fail")
#            fails_dict[tweet_id] = e


#end = timer()
#print(end-start)
#print(fails_dict)
```

In [194…
```
json_df = pd.read_json("tweet-json2.txt", lines=True, encoding='utf-8')
json_df.columns
```

Out[194]:
```
Index(['created_at', 'id', 'id_str', 'full_text', 'truncated',
       'display_text_range', 'entities', 'extended_entities', 'source',
       'in_reply_to_status_id', 'in_reply_to_status_id_str',
       'in_reply_to_user_id', 'in_reply_to_user_id_str',
       'in_reply_to_screen_name', 'user', 'geo', 'coordinates', 'place',
       'contributors', 'is_quote_status', 'retweet_count', 'favorite_count',
       'favorited', 'retweeted', 'possibly_sensitive',
       'possibly_sensitive_appealable', 'lang', 'retweeted_status',
       'quoted_status_id', 'quoted_status_id_str', 'quoted_status'],
      dtype='object')
```

In [195…
```
json_df['tweet_id']=json_df['id']
```

In [196…
```
json_df = json_df[['tweet_id','favorite_count','retweet_count']]
```

In [197…
```
#json_df.columns = ['tweet_id', 'favorite_count', 'retweet_count']
```

In [198…
```
json_df.shape
```

Out[198]:
```
(2354, 3)
```

In [199…
```
complete_df = df_archive.merge(df_image_predictions, how='left', on='tweet_id')
```

In [200…
```
complete_df.columns
```

Out[200]:
```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'favorite_count', 'retweet_count'],
      dtype='object')
```

In [201…
```
complete_df.to_csv('complete_df.csv', index=False)
```

In [202…
```
df = pd.read_csv('complete_df.csv')
```

Assessing Data

In this section, detect and document at least **eight (8) quality issues and two (2) tidiness issue**. You must use **both** visual assessment programmatic assessement to assess the data.

Note: pay attention to the following key points when you access the data.

- You only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.
- Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling. Therefore, the requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned. This unique rating system is a big part of the popularity of WeRateDogs.
- You do not need to gather the tweets beyond August 1st, 2017. You can, but note that you won't be able to gather the image predictions for these tweets since you don't have access to the algorithm used.

In [203…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 30 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
 17  jpg_url                     2075 non-null   object 
 18  img_num                     2075 non-null   float64
 19  p1                          2075 non-null   object 
 20  p1_conf                     2075 non-null   float64
 21  p1_dog                      2075 non-null   object 
 22  p2                          2075 non-null   object 
 23  p2_conf                     2075 non-null   float64
 24  p2_dog                      2075 non-null   object 
 25  p3                          2075 non-null   object 
 26  p3_conf                     2075 non-null   float64
 27  p3_dog                      2075 non-null   object 
 28  favorite_count              2354 non-null   float64
 29  retweet_count               2354 non-null   float64
dtypes: float64(10), int64(3), object(17)
memory usage: 552.3+ KB
```

In [204…

```
df.head()
```

Out[204]:

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

In [205…
```
df.rating_numerator.describe()
```

Out[205]:
```
count    2356.000000
mean       13.126486
std        45.876648
min         0.000000
25%        10.000000
50%        11.000000
75%        12.000000
max      1776.000000
Name: rating_numerator, dtype: float64
```

In [206…
```
df.columns
```

Out[206]:
```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestam
p',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'favorite_count', 'retweet_coun
t'],
      dtype='object')
```

In [207…
```
df.isnull()
```

Out[207]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweete
0	False	True	True	False	False	False	
1	False	True	True	False	False	False	
2	False	True	True	False	False	False	
3	False	True	True	False	False	False	
4	False	True	True	False	False	False	
...	...	...	...	...	...	...	
2351	False	True	True	False	False	False	
2352	False	True	True	False	False	False	
2353	False	True	True	False	False	False	
2354	False	True	True	False	False	False	
2355	False	True	True	False	False	False	

2356 rows × 30 columns

In [208…
```
df.iloc[:, 13:17].value_counts()
```

Out[208]:
```
doggo  floofer  pupper  puppo
None   None     None    None     1976
                pupper  None      245
doggo  None     None    None       83
None   None     None    puppo      29
doggo  None     pupper  None       12
None   floofer  None    None        9
doggo  None     None    puppo       1
       floofer  None    None        1
dtype: int64
```

In [209…
```
df[df.duplicated()]
```

Out[209]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_stat

In [210…
```
df.nunique()
```

Out[210]:

```
tweet_id                      2356
in_reply_to_status_id           77
in_reply_to_user_id             31
timestamp                     2356
source                           4
text                          2356
retweeted_status_id            181
retweeted_status_user_id        25
retweeted_status_timestamp     181
expanded_urls                 2218
rating_numerator                40
rating_denominator              18
name                           957
doggo                            2
floofer                          2
pupper                           2
puppo                            2
jpg_url                       2009
img_num                          4
p1                             378
p1_conf                       2006
p1_dog                           2
p2                             405
p2_conf                       2004
p2_dog                           2
p3                             408
p3_conf                       2006
p3_dog                           2
favorite_count                2007
retweet_count                 1724
dtype: int64
```

Quality issues

1. Names of dogs are not consistent with upper and lowercase letters
2. Names of dogs aren't consistent as far as underscore or spacing
3. melt column names to one new column name
4. type of dogs are in uppercase, change to lower case.
5. get rid of all retweets
6. change timestamp to datetime
7. some names are all just 'a', 'the', or 'none' and need to be changed to the unknown
8. change tweet_id to a str

Tidiness issues

1. change 3 columns of doggo floofer pupper into one column with its name type
2. combine numerator and denomenator into one column called rating

3. get rid of unncessary columns after getting rid of retweets

4. rearrange columns for a more readable look

Cleaning Data

In this section, clean **all** of the issues you documented while assessing.

Note: Make a copy of the original data before cleaning. Cleaning includes merging individual pieces of data according to the rules of tidy data. The result should be a high-quality and tidy master pandas DataFrame (or DataFrames, if appropriate).

In [211…
```
# Make copies of original pieces of data
df_clean = df.copy()
```

Quality

Quality Issue #1:

Define:

Remove invalid names from the names column

Code

In [212…
```
df_clean['name'] = df_clean.name.str.extract('\\b([A-Z]\\S*)\\b')
df_clean.dropna(subset=['name'], inplace=True)
df_clean = df_clean[df_clean.name != 'None']
```

Test

In [213…
```
df_clean['name'].value_counts().head(10)
```

Out[213]:
```
Charlie    12
Lucy       11
Cooper     11
Oliver     11
Lola       10
Penny      10
Tucker     10
Bo          9
Winston     9
Sadie       8
Name: name, dtype: int64
```

Quality Issue #2:

Define

Remove underscore from dog names so that it is has consistent spacing & p1, p2, p3 names should all be lowercase

Code

In [214…
```
df_clean.p1 = df_clean.p1.str.replace('_', ' ')
df_clean.p2 = df_clean.p2.str.replace('_', ' ')
df_clean.p3 = df_clean.p3.str.replace('_', ' ')

df_clean.p1 = df_clean.p1.str.lower()
df_clean.p2 = df_clean.p2.str.lower()
df_clean.p3 = df_clean.p3.str.lower()
```

Test

In [215…
```
df_clean[['p1', 'p2', 'p3']]
```

Out[215]:

	p1	p2	p3
0	orange	bagel	banana
1	chihuahua	pekinese	papillon
2	chihuahua	malamute	kelpie
3	paper towel	labrador retriever	spatula
4	basset	english springer	german short-haired pointer
...	...	...	...
2315	german shepherd	beagle	bloodhound
2317	ibizan hound	pembroke	west highland white terrier
2318	dalmatian	labrador retriever	great pyrenees
2319	curly-coated retriever	giant schnauzer	labrador retriever
2325	toy terrier	papillon	chihuahua

1502 rows × 3 columns

Quality Issue #3:

Define

Remove HTML from rows in source column and replace with shortened phrase

Code

In [216…
```
# remove HTML from text from source column and replace with phrase
df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitter.co
```

```
df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitter.co
df_clean['source'] = df_clean['source'].str.replace('<a href="https://about.twi
df_clean['source'] = df_clean['source'].str.replace('<a href="http://vine.co" r
```

```
/var/folders/0b/1vbj_50x46n7w0s6p7_jvpf80000gn/T/ipykernel_43191/1631779530.p
y:2: FutureWarning: The default value of regex will change from True to False
in a future version.
  df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitte
r.com/download/iphone" rel="nofollow">Twitter for iPhone</a>', 'Twitter for iP
hone')
/var/folders/0b/1vbj_50x46n7w0s6p7_jvpf80000gn/T/ipykernel_43191/1631779530.p
y:3: FutureWarning: The default value of regex will change from True to False
in a future version.
  df_clean['source'] = df_clean['source'].str.replace('<a href="http://twitte
r.com" rel="nofollow">Twitter Web Client</a>', 'Twitter Web Client')
/var/folders/0b/1vbj_50x46n7w0s6p7_jvpf80000gn/T/ipykernel_43191/1631779530.p
y:4: FutureWarning: The default value of regex will change from True to False
in a future version.
  df_clean['source'] = df_clean['source'].str.replace('<a href="https://about.
twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>', 'TweetDeck')
/var/folders/0b/1vbj_50x46n7w0s6p7_jvpf80000gn/T/ipykernel_43191/1631779530.p
y:5: FutureWarning: The default value of regex will change from True to False
in a future version.
  df_clean['source'] = df_clean['source'].str.replace('<a href="http://vine.c
o" rel="nofollow">Vine - Make a Scene</a>', 'Vine - Make a Scene')
```

Test

In [217…
```
df_clean.source.value_counts()
```

Out[217]:
```
Twitter for iPhone     1436
Vine - Make a Scene      41
Twitter Web Client       16
TweetDeck                 9
Name: source, dtype: int64
```

Quality Issue 4

Define

drop all rows with duplicate addresses fro jpg_url and drop img_num column

Code

In [218…
```
df_clean = df_clean.drop_duplicates(subset='jpg_url')
df_clean = df_clean.drop(['img_num'], axis=1)
```

Test

In [219…
```
len(df_clean[df_clean.jpg_url.duplicated()])
```

Out[219]:
```
0
```

In [220…
```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1351 entries, 0 to 2325
Data columns (total 29 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    1351 non-null   int64  
 1   in_reply_to_status_id       1 non-null      float64
 2   in_reply_to_user_id         1 non-null      float64
 3   timestamp                   1351 non-null   object 
 4   source                      1351 non-null   object 
 5   text                        1351 non-null   object 
 6   retweeted_status_id         48 non-null     float64
 7   retweeted_status_user_id    48 non-null     float64
 8   retweeted_status_timestamp  48 non-null     object 
 9   expanded_urls               1351 non-null   object 
 10  rating_numerator            1351 non-null   int64  
 11  rating_denominator          1351 non-null   int64  
 12  name                        1351 non-null   object 
 13  doggo                       1351 non-null   object 
 14  floofer                     1351 non-null   object 
 15  pupper                      1351 non-null   object 
 16  puppo                       1351 non-null   object 
 17  jpg_url                     1350 non-null   object 
 18  p1                          1350 non-null   object 
 19  p1_conf                     1350 non-null   float64
 20  p1_dog                      1350 non-null   object 
 21  p2                          1350 non-null   object 
 22  p2_conf                     1350 non-null   float64
 23  p2_dog                      1350 non-null   object 
 24  p3                          1350 non-null   object 
 25  p3_conf                     1350 non-null   float64
 26  p3_dog                      1350 non-null   object 
 27  favorite_count              1350 non-null   float64
 28  retweet_count               1350 non-null   float64
dtypes: float64(9), int64(3), object(17)
memory usage: 316.6+ KB
```

Quality Issue #5:

Define

Remove all rows that have values in the retweet columns

Code

In [221…

```
# drop all null values in the retweet rows
df_clean = df_clean[df_clean['retweeted_status_id'].isnull()]
```

Test

In [222…

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1303 entries, 0 to 2325
Data columns (total 29 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    1303 non-null   int64  
 1   in_reply_to_status_id       1 non-null      float64
 2   in_reply_to_user_id         1 non-null      float64
 3   timestamp                   1303 non-null   object 
 4   source                      1303 non-null   object 
 5   text                        1303 non-null   object 
 6   retweeted_status_id         0 non-null      float64
 7   retweeted_status_user_id    0 non-null      float64
 8   retweeted_status_timestamp  0 non-null      object 
 9   expanded_urls               1303 non-null   object 
 10  rating_numerator            1303 non-null   int64  
 11  rating_denominator          1303 non-null   int64  
 12  name                        1303 non-null   object 
 13  doggo                       1303 non-null   object 
 14  floofer                     1303 non-null   object 
 15  pupper                      1303 non-null   object 
 16  puppo                       1303 non-null   object 
 17  jpg_url                     1303 non-null   object 
 18  p1                          1303 non-null   object 
 19  p1_conf                     1303 non-null   float64
 20  p1_dog                      1303 non-null   object 
 21  p2                          1303 non-null   object 
 22  p2_conf                     1303 non-null   float64
 23  p2_dog                      1303 non-null   object 
 24  p3                          1303 non-null   object 
 25  p3_conf                     1303 non-null   float64
 26  p3_dog                      1303 non-null   object 
 27  favorite_count              1303 non-null   float64
 28  retweet_count               1303 non-null   float64
dtypes: float64(9), int64(3), object(17)
memory usage: 305.4+ KB
```

Quality Issue #6:

Define

convert data type of tweet_id to a string amd timestamp into a date and change numerator and denomenator to category

Code

In [223…

```
# convert tweet_id into a string
df_clean.tweet_id = df_clean.tweet_id.astype(str)
#convert timestamp into a datetime
df_clean.timestamp = pd.to_datetime(df_clean.timestamp)
df_clean.timestamp = df_clean.timestamp.dt.date
# convert numerator and denomenator to category
df_clean.rating_numerator = df_clean.rating_numerator.astype(str)
```

Test

In [224…
```
df_clean.timestamp.info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 1303 entries, 0 to 2325
Series name: timestamp
Non-Null Count  Dtype
--------------  -----
1303 non-null   object
dtypes: object(1)
memory usage: 20.4+ KB
```

Quality Issue #7:

Define

Only keep rows with entries that have the p1 p2 p3 dogs that are True and get rid of all false dog names

Code

In [225…
```
# keep rows that are set to true
df_clean = df_clean[((df_clean['p1_dog'] == True) &
                     (df_clean['p2_dog'] == True) &
                     (df_clean['p3_dog'] == True))]
len(df_clean)
```

Out[225]: 826

Test

In [226…
```
# verify
df_clean[((df_clean['p1_dog'] == False) &
                     (df_clean['p2_dog'] == False) &
                     (df_clean['p3_dog'] == False))]
```

Out[226]:

tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_stat

Quality Issue #8:

Define

extract dog breed from prediction data and create it's own column. Drop all other breed prediction columns

Code

In [227…
```
# create empty lists to save our choice for each row in the dataset
breed = []
```

```
confidence = []

# function that iterates through prediction columns to find the best prediction
def breed_confidence(row):
    if row.p1_dog == True:
        breed.append(row.p1)
        confidence.append(row.p1_conf)
    elif row.p2_dog == True:
        breed.append(row.p2)
        confidence.append(row.p2_conf)
    elif row.p3_dog == True:
        breed.append(row.p3)
        confidence.append(row.p3_conf)
    else:
        breed.append('Unpredicted')
        confidence.append(0)

# call function using pandas apply by columns
df_clean.apply(breed_confidence, axis=1)

# add lists created to master dataframe
df_clean['breed'] = breed
df_clean['confidence'] = confidence
```

In [228…

```
# drop all of the uninterested columns
df_clean.drop(columns=['p1','p2','p3','p1_dog','p2_dog','p3_dog','p1_conf', 'p2
```

Test

In [229…

```
df_clean[['breed', 'confidence']].head()
```

Out[229]:

	breed	confidence
1	chihuahua	0.323581
2	chihuahua	0.716012
4	basset	0.555712
8	irish terrier	0.487574
9	pembroke	0.511319

Tidiness

Tidy Issue #1:

Define

Get rid of the retweet columns now that the retweeted rows are gone

Code

In [230…
```
df_clean = df_clean.drop(['in_reply_to_status_id', 'in_reply_to_user_id','retwe
```

Test

In [231…
```
df_clean.columns
```

Out[231]:
```
Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls',
       'rating_numerator', 'rating_denominator', 'name', 'doggo', 'floofer',
       'pupper', 'puppo', 'jpg_url', 'favorite_count', 'retweet_count',
       'breed', 'confidence'],
      dtype='object')
```

Tidy Issue #2

Define

Combine all 4 columns of "doggo floofer pupper and puppo" into one column with its name type.

Code

In [232…
```
df_clean.doggo.unique(), df_clean.floofer.unique(), df_clean.pupper.unique(), c
```

Out[232]:
```
(array(['None', 'doggo'], dtype=object),
 array(['None', 'floofer'], dtype=object),
 array(['None', 'pupper'], dtype=object),
 array(['None', 'puppo'], dtype=object))
```

In [233…
```
df_clean.doggo.replace('None', '', inplace=True)
df_clean.floofer.replace('None', '', inplace=True)
df_clean.pupper.replace('None', '', inplace=True)
df_clean.puppo.replace('None', '', inplace=True)

df_clean['dog_type'] = df_clean.doggo + df_clean.floofer + df_clean.pupper + df


df_clean.loc[df_clean.dog_type == 'doggopupper', 'dog_type'] = 'doggo,pupper'
df_clean.loc[df_clean.dog_type == 'doggopuppo', 'dog_type'] = 'doggo,puppo'
df_clean.loc[df_clean.dog_type == 'doggofloofer', 'dog_type'] = 'doggo,floofer'
```

In [234…
```
df_clean['dog_type'].value_counts()
```

Out[234]:
```
                718
pupper           68
doggo            20
puppo            13
floofer           4
doggo,pupper      3
Name: dog_type, dtype: int64
```

Tidy Issue #3

Define

Arrange columns to look cleaner

Code

In [235…
```
df_clean.columns
```

Out[235]:
```
Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls',
       'rating_numerator', 'rating_denominator', 'name', 'doggo', 'floofer',
       'pupper', 'puppo', 'jpg_url', 'favorite_count', 'retweet_count',
       'breed', 'confidence', 'dog_type'],
      dtype='object')
```

In [236…
```
df_clean = df_clean[['tweet_id', 'timestamp', 'source', 'name', 'breed', 'confi
            'rating_denominator', 'text','favorite_count', 'retweet_count','jp
```

Test

In [237…
```
df_clean.columns
```

Out[237]:
```
Index(['tweet_id', 'timestamp', 'source', 'name', 'breed', 'confidence',
       'dog_type', 'rating_numerator', 'rating_denominator', 'text',
       'favorite_count', 'retweet_count', 'jpg_url', 'expanded_urls'],
      dtype='object')
```

Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".

In [238…
```
df_clean.to_csv('twitter_archive_master.csv')
```

Analyzing and Visualizing Data

In this section, analyze and visualize your wrangled data. You must produce at least **three (3) insights and one (1) visualization.**

In [239…
```
df_final = pd.read_csv('twitter_archive_master.csv')
```

In [240…
```
df_final.head(2)
```

Out[240]:

	Unnamed: 0	tweet_id	timestamp	source	name	breed	confidence	dog_ty
0	1	892177421306343426	2017-08-01	Twitter for iPhone	Tilly	chihuahua	0.323581	Na
1	2	891815181378084864	2017-07-31	Twitter for iPhone	Archie	chihuahua	0.716012	Na

Insights:

1. Pupper was the most used word to describe the dog
2. Cooper is the top name in this dataset
3. The most common rating givent was a 12/10
4. The dop 5 breeds with the highest confidence are Chow, Dalmation, Pug, French Bulldog, & Saint Bernard
5. Retweets and Favorites are highly correlated

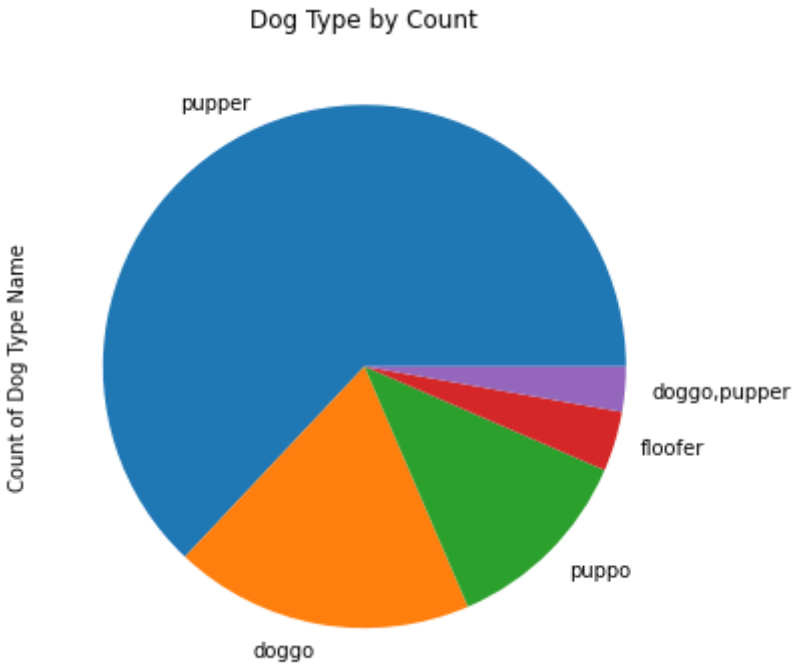
Visualization

In [241…

```
df_final.dog_type.value_counts().plot(kind='pie', xlabel='Dog Type Names', ylab
```

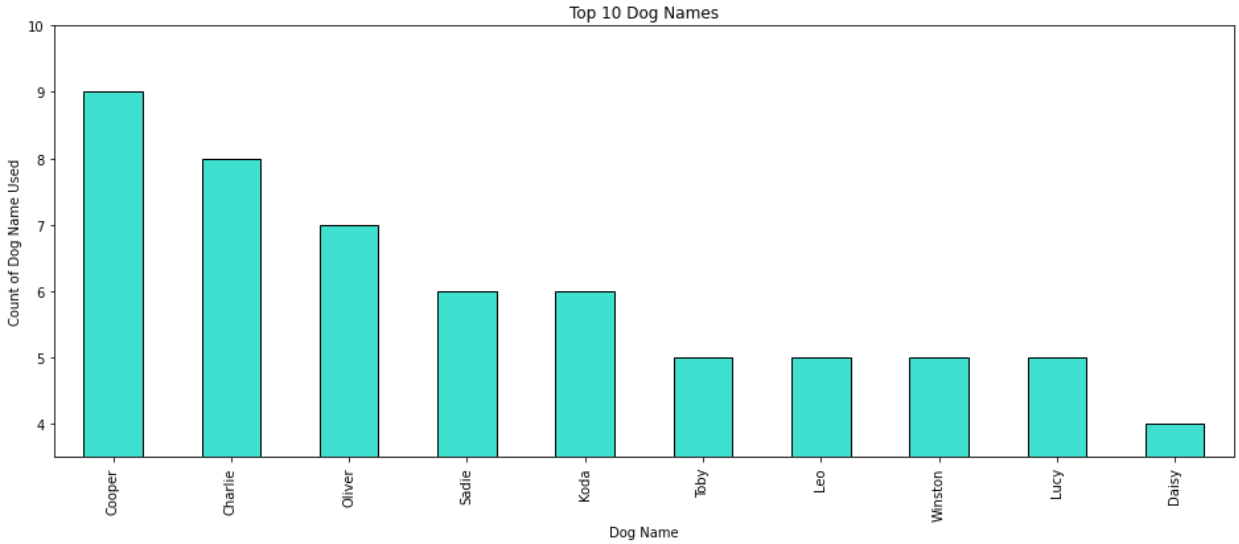
Out[241]:

```
<AxesSubplot:title={'center':'Dog Type by Count'}, ylabel='Count of Dog Type N
ame'>
```



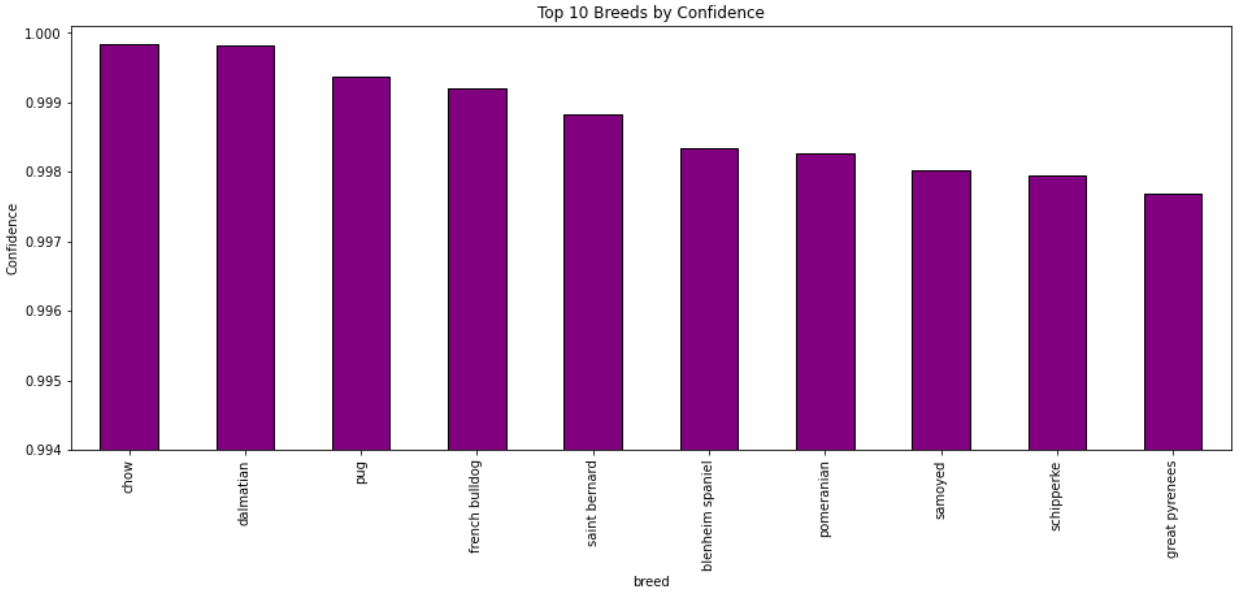
In [242…
```
df_top_10 = df_final.name.value_counts()
df_top_10 =df_top_10.head(10)
df_top_10.plot(kind='bar', title='Top 10 Dog Names', ylim=(3.5,10), xlabel='Dog
```

Out[242]:
```
<AxesSubplot:title={'center':'Top 10 Dog Names'}, xlabel='Dog Name', ylabel='C
ount of Dog Name Used'>
```



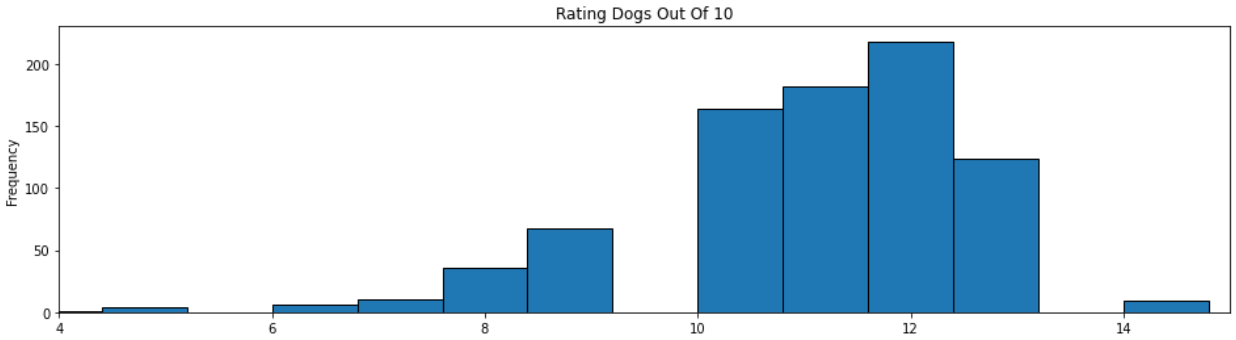
In [243…
```
df_final.groupby('breed')['confidence'].max().sort_values(ascending=False).head
```

Out[243]:
```
<AxesSubplot:title={'center':'Top 10 Breeds by Confidence'}, xlabel='breed', y
label='Confidence'>
```



In [244…
```
df_final.rating_numerator.plot(kind='hist',figsize=(16,4), bins=60, xlim=(4,15)
```

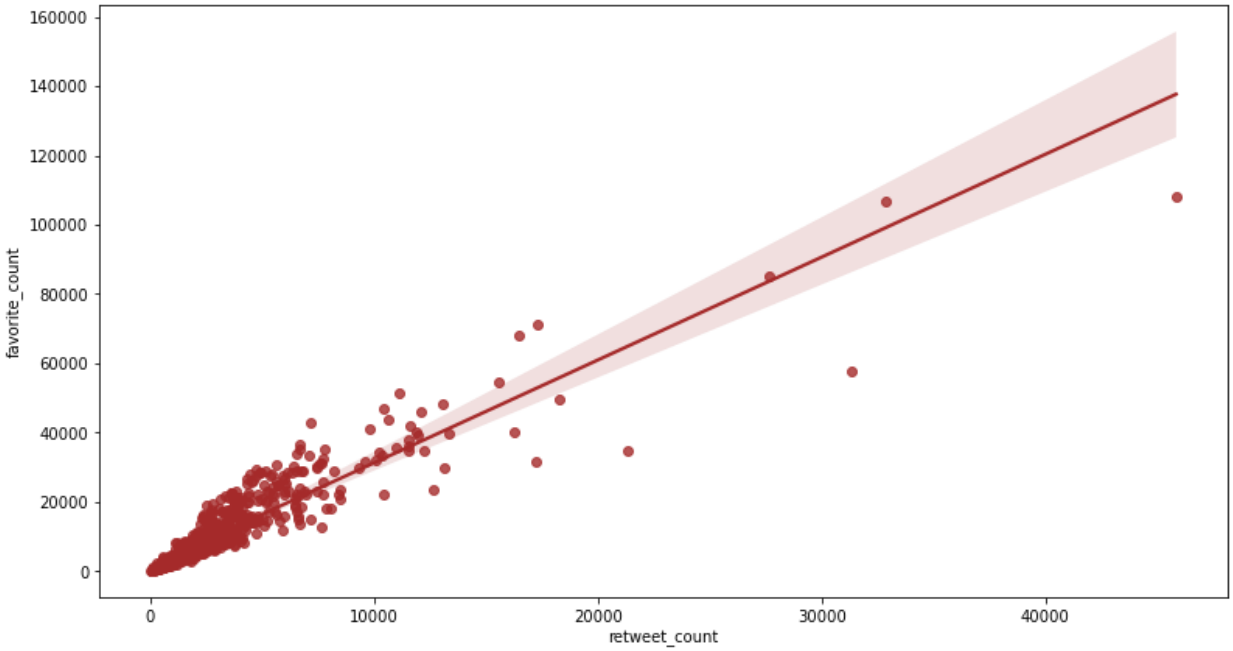
Out[244]:
```
<AxesSubplot:title={'center':'Rating Dogs Out Of 10'}, ylabel='Frequency'>
```



In [245…
```
df_scatter = df_final[['retweet_count', 'favorite_count']].corr()
```

In [246…
```
plt.figure(figsize=(13,7))
ax = sns.regplot(x='retweet_count',
            y='favorite_count',
            data=df_final, color='brown')
plt.title
```

Out[246]:
```
<function matplotlib.pyplot.title(label, fontdict=None, loc=None, pad=None, *,
y=None, **kwargs)>
```



Reference websites

- Seaborn
- Changing Axis Labels
- Pandas
- Assign multiple values to multiple columns
- Regex
- Using 'Where'

In [ ]: