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 [215…
```
# Import necessary libraries
# *NOTE*
import pandas as pd
import numpy as np
import requests
import os
import tweepy
import json
import seaborn as sns
import matplotlib.pyplot as plt
from tweepy import OAuthHandler
from timeit import default_timer as timer
%matplotlib inline
```

In [216…
```
# Create dataframe from 'twitter-archive-enhanced.csv'
df_archive = pd.read_csv('twitter-archive-enhanced.csv')
```

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

In [217…
```
# Download 'image-predictions.tsv' and create a dataframe called 'df_image'
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_ima
r = requests.get(url, allow_redirects=True)
df_image = pd.read_csv('image-predictions.tsv', sep = '\t')


# https://www.tutorialspoint.com/downloading-files-from-web-using-python
# Used this tutorial as a reference/to supplement course materials
```

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

> NOTE: The code below does not need to run, it was provided by Udacity instructors for those who do not want to create a Twitter/X account

In [218…

```
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer

# Query Twitter API for each tweet in the Twitter archive and save JSON in a
# These are hidden to comply with Twitter's API terms and conditions
consumer_key = 'HIDDEN'
consumer_secret = 'HIDDEN'
access_token = 'HIDDEN'
access_secret = 'HIDDEN'

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

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


# !!! NOTE TO REVIEWER: this student had mobile verification issues so the f
# Twitter API code was sent to this student from a Udacity instructor !!!
# Tweet IDs for which to gather additional data via Twitter's API
tweet_ids = df_1.tweet_id.values
len(tweet_ids)

# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
count = 0
fails_dict = {}
start = timer()
# 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 r
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Success")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except tweepy.TweepError as e:
            print("Fail")
            fails_dict[tweet_id] = e
            pass
end = timer()
print(end - start)
print(fails_dict)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[218], line 22
     16 api = tweepy.API(auth, wait_on_rate_limit=True)
     19 # !!! NOTE TO REVIEWER: this student had mobile verification issues
so the following
     20 # Twitter API code was sent to this student from a Udacity instructo
r !!!
     21 # Tweet IDs for which to gather additional data via Twitter's API
---> 22 tweet_ids = df_1.tweet_id.values
     23 len(tweet_ids)
     25 # Query Twitter's API for JSON data for each tweet ID in the Twitter
archive

NameError: name 'df_1' is not defined
```

In [219…

```
# Create list to store extracted data
json_data = []

# Loop through the JSON data from 'tweet-json.txt' and extract the three nec
with open('tweet-json.txt', 'r') as file:
    for line in file:
        tweet_data = json.loads(line)
        tweet_id = tweet_data['id']
        retweet_count = tweet_data['retweet_count']
        favorite_count = tweet_data['favorite_count']
        json_data.append({'id':tweet_id, 'retweet_count':retweet_count, 'fav

# Create the dataframe 'df_json' that holds the extracted data
df_json = pd.DataFrame(json_data)

# 'https://www.youtube.com/watch?v=SNv7E-cXCu0'
# I used this video to supplement the course materials about how to parse 'j
```

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.

Assessing df_archive

In [220…
```
df_archive.shape
```

Out[220]:
```
(2356, 17)
```

In [221…
```
df_archive.head()
```

Out[221]:

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

In [222…

```
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                        1611 non-null   object 
 13  doggo                       97 non-null     object 
 14  floofer                     10 non-null     object 
 15  pupper                      257 non-null    object 
 16  puppo                       30 non-null     object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [223…
```
df_archive.describe()
```

Out[223]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweete
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	
std	6.856705e+16	7.582492e+17	1.252797e+17	6.236928e+16	
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	

In [224…
```
# See how many duplicates exist
df_archive.duplicated().sum()
```

Out[224]:
```
0
```

In [225…
```
# Get a sense of how many null values exist
df_archive.isnull().sum()
```

Out[225]:
```
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                           745
doggo                         2259
floofer                       2346
pupper                        2099
puppo                         2326
dtype: int64
```

In [226…
```
# Create a list of names to view to see if any problems exist
names = df_archive.name.value_counts(dropna=False)
pd.set_option('display.max_rows', None)
print(names)
```

```
name
NaN              745
a                 55
Charlie           12
Cooper            11
Lucy              11
Oliver            11
Tucker            10
Penny             10
Lola              10
Winston            9
Bo                 9
Sadie              8
the                8
Daisy              7
Buddy              7
Toby               7
an                 7
Bailey             7
Leo                6
Oscar              6
Stanley            6
Rusty              6
Bella              6
Dave               6
Jack               6
Koda               6
Scout              6
Jax                6
```

Milo	6
Louis	5
Phil	5
Sammy	5
Gus	5
Chester	5
Alfie	5
Sunny	5
Bentley	5
very	5
Oakley	5
Finn	5
George	5
Larry	5
Clarence	4
Loki	4
Walter	4
Reggie	4
Hank	4
Scooter	4
Duke	4
quite	4
Reginald	4
Maggie	4
Luna	4
Maddie	4
Dexter	4
Jerry	4
Carl	4
Ruby	4
Riley	4
Clark	4
Chip	4
Brody	4
just	4
Sophie	4
Sampson	4
Boomer	4
Winnie	4
Gary	4
Archie	4
Jeffrey	4
Gerald	4
one	4
Cassie	4
Maximus	4
Moose	4
Bear	4
Bruce	4
Beau	4
Derek	4
Shadow	4
Rory	3

```
Waffles             3
Earl                3
Nala                3
Louie               3
Colby               3
Otis                3
Doug                3
Rosie               3
Wyatt               3
Lily                3
Coco                3
Kyle                3
Olive               3
Peaches             3
Vincent             3
Reese               3
Jimothy             3
Frankie             3
Ted                 3
Arnie               3
Klevin              3
Max                 3
Zoey                3
Ellie               3
Zeke                3
Wilson              3
Calvin              3
Sebastian           3
Wallace             3
Lorenzo             3
Malcolm             3
Mia                 3
Samson              3
Paisley             3
Steven              3
Gizmo               3
Benji               2
Philbert            2
Django              2
Sansa               2
Tyr                 2
Moe                 2
Misty               2
Herm                2
Cali                2
Kenneth             2
Chelsea             2
Happy               2
Ollie               2
Leela               2
Sam                 2
Hunter              2
Seamus              2
```

Tyrone	2
Atlas	2
Rocky	2
Oshie	2
Olivia	2
Davey	2
Timison	2
Paull	2
Titan	2
Levi	2
Layla	2
Stubert	2
Rizzy	2
Baloo	2
Betty	2
Herschel	2
Butter	2
Mister	2
Lenny	2
Doc	2
Juno	2
Anakin	2
Bernie	2
Marley	2
Phineas	2
Nollie	2
Meyer	2
Blitz	2
Raymond	2
Crystal	2
Phred	2
Terry	2
Axel	2
Charles	2
Herald	2
getting	2
Theodore	2
Maxaroni	2
Kilo	2
Brad	2
actually	2
Sugar	2
Percy	2
Opal	2
Shaggy	2
Wally	2
Oliviér	2
mad	2
Kreggory	2
Remington	2
Bisquick	2
Chuckles	2
Thumas	2

```
Curtis          2
Calbert         2
Chet            2
Bubbles         2
Coops           2
Ozzy            2
Smokey          2
Flávio          2
CeCe            2
Pippa           2
Panda           2
Bob             2
Sarge           2
Abby            2
Keurig          2
Baxter          2
Atticus         2
Dakota          2
Gromit          2
Roosevelt       2
Fizz            2
Penelope        2
Finley          2
Solomon         2
Hurley          2
Bell            2
Dash            2
Linda           2
Chipson         2
Harper          2
Kenny           2
Sandy           2
Lincoln         2
Frank           2
Jackson         2
Kirby           2
Lou             2
Jeph            2
Piper           2
Hammond         2
Fred            2
Carly           2
Keith           2
Rubio           2
Watson          2
Chompsky        2
Rufus           2
Jiminy          2
Griffin         2
Rocco           2
Albert          2
Patrick         2
Kreg            2
```

Eve	2
Luca	2
Lennon	2
Cupcake	2
Emmy	2
Ken	2
Pipsy	2
Nelly	2
Odie	2
Ava	2
Ash	2
Terrance	2
not	2
Romeo	2
Stephan	2
Jesse	2
Elliot	2
Logan	2
Moreton	2
Klein	2
Kyro	2
Fiona	2
Yogi	2
Indie	2
Benedict	2
Lilly	2
Churlie	2
Hobbes	2
Trooper	2
Belle	2
Quinn	2
Neptune	2
Albus	2
Alice	2
Aspen	2
Jamesy	2
Bungalo	2
Harold	2
Astrid	2
Gidget	2
Cody	2
Dawn	2
Sierra	2
Cash	2
Pickles	2
Kevin	2
Eli	2
Gabe	2
Balto	2
Pablo	2
Franklin	2
Jimison	2
Hercules	2

```
Canela          2
Mattie          2
Dotsy           1
Marvin          1
Lizzie          1
Kallie          1
Ember           1
Frönq           1
Blakely         1
Jockson         1
Ambrose         1
Jomathan        1
Marq            1
Kramer          1
Maks            1
Derby           1
Pubert          1
Spark           1
Ferg            1
Ricky           1
Carll           1
Ron             1
Torque          1
Lucky           1
Terrenth        1
Trip            1
Gordon          1
Jareld          1
Bode            1
DayZ            1
Jo              1
Hamrick         1
Gòrdón          1
Barry           1
Horace          1
Timofy          1
Kulet           1
Perry           1
Bobby           1
Winifred        1
Obi             1
Kevon           1
Tino            1
Sweets          1
Bruiser         1
Lupe            1
Tiger           1
Lugan           1
Kathmandu       1
Banditt         1
Banjo           1
Katie           1
Oreo            1
```

Mona	1
Adele	1
Fynn	1
Zara	1
Crimson	1
Birf	1
Josep	1
Zeek	1
Lorelei	1
Gerbald	1
Marty	1
Brooks	1
Petrick	1
Hubertson	1
Hanz	1
Kara	1
Skittles	1
Chesterson	1
Biden	1
Naphaniel	1
Ole	1
Pherb	1
Blipson	1
Genevieve	1
Reptar	1
Trevith	1
Joshwa	1
Millie	1
Kial	1
Berb	1
Bradlay	1
Colin	1
Brian	1
Yoda	1
Coopson	1
Carter	1
Dook	1
Grady	1
Jessiga	1
Eazy	1
Murphy	1
Kaia	1
Fillup	1
Miley	1
Daniel	1
Philippe	1
Charl	1
Reagan	1
Yukon	1
Crouton	1
Hall	1
Fwed	1
Cuddles	1

```
Claude              1
Filup               1
Kobe                1
Durg                1
Vinscent            1
Hazel               1
Lolo                1
Eriq                1
Erik                1
Rudy                1
Rambo               1
Cleopatricia        1
Oddie               1
Socks               1
Maxwell             1
Ralphson            1
Keet                1
Geoff               1
Covach              1
Lucia               1
Cedrick             1
Stu                 1
Bilbo               1
Fiji                1
Rilo                1
Freddery            1
Bodie               1
Dunkin              1
Tedrick             1
Tupawc              1
Clybe               1
Amber               1
Edgar               1
Teddy               1
Kingsley            1
Brockly             1
Richie              1
Brandy              1
Molly               1
Bobb                1
Patch               1
Lulu                1
Finnegus            1
Henry               1
Bobbay              1
Mitch               1
Creg                1
Antony              1
Trigger             1
Kaiya               1
Acro                1
Mason               1
Arnold              1
```

```
Andy          1
Aiden         1
Sage          1
Obie          1
Amy           1
Jett          1
Dot           1
Shnuggles     1
life          1
Chesney       1
Traviss       1
Jonah         1
Damon         1
Herb          1
Willy         1
Mollie        1
Edd           1
Laela         1
Bloo          1
Pluto         1
Ester         1
Danny         1
Tedders       1
Jeffrie       1
Superpup      1
Rufio         1
Gin           1
Jeb           1
Rodman        1
Kendall       1
Holly         1
Jeffri        1
Norman        1
Scott         1
Dylan         1
Taz           1
Darby         1
Jackie        1
light         1
Jazz          1
Franq         1
Pippin        1
Rolf          1
Batdog        1
Snickers      1
Ridley        1
Cal           1
Bradley       1
Bubba         1
Mojo          1
Caryl         1
Leonidas      1
Steve         1
```

```
space          1
Mac            1
Fletcher       1
Anthony        1
Kenzie         1
Sparky         1
Tanner         1
Pumpkin        1
Julius         1
Schnozz        1
Gustaf         1
Pip            1
Scruffers      1
Cheryl         1
Alejandro      1
JD             1
Ed             1
Mark           1
Zuzu           1
Buddah         1
Darrel         1
Kirk           1
Cheesy         1
Moofasa        1
Jaspers        1
Hector         1
Goliath        1
Clarq          1
Kawhi          1
Ralf           1
by             1
Peanut         1
Sully          1
Emmie          1
Tessa          1
Willie         1
Dug            1
Rinna          1
Mike           1
William        1
Samsom         1
Harrison       1
Griswold       1
Ozzie          1
Amélie         1
Taco           1
Joey           1
Ruffles        1
Lambeau        1
Todo           1
Tess           1
Ulysses        1
Toffee         1
```

```
Apollo          1
Tuco            1
Jaycob          1
Glacier         1
Chuck           1
Jeremy          1
Champ           1
Chaz            1
Saydee          1
Dwight          1
Billl           1
Humphrey        1
Tommy           1
Juckson         1
Liam            1
Randall         1
Chuq            1
Tyrus           1
Karl            1
Godzilla        1
Fabio           1
Vinnie          1
Sandra          1
Bert            1
Striker         1
Donny           1
Butters         1
Pepper          1
Nigel           1
Tassy           1
Ben             1
Evy             1
Ralph           1
officially      1
Rascal          1
Ronduh          1
Kollin          1
Skye            1
Shawwn          1
Kloey           1
Andru           1
Tug             1
Tango           1
Julio           1
Grizz           1
Jerome          1
Crumpet         1
Jessifer        1
Raphael         1
Izzy            1
Asher           1
Huxley          1
Clarkus         1
```

```
Autumn          1
Sundance        1
Flash           1
Howie           1
Jazzy           1
Anna            1
Wafer           1
Tom             1
Florence        1
Dido            1
Duchess         1
Eugene          1
Strudel         1
Tebow           1
Chloe           1
Timber          1
Binky           1
Dudley          1
Comet           1
Harlso          1
Crawford        1
Smiley          1
Sailer          1
Emanuel         1
Kuyu            1
Dutch           1
Pete            1
Mutt            1
Doobert         1
Beebop          1
Alexander       1
Brutus          1
Sweet           1
Kona            1
Boots           1
Ralphie         1
Cupid           1
Pawnd           1
Pilot           1
Ike             1
Mo              1
Akumi           1
Alf             1
Chubbs          1
Dobby           1
Sobe            1
Longfellow      1
Iroh            1
Pancake         1
Snicku          1
Mack            1
Nimbus          1
Laika           1
```

Maude	1
Sky	1
Newt	1
Nida	1
Robin	1
Monster	1
BeBe	1
Remus	1
Mabel	1
Mosby	1
Chef	1
Mauve	1
Bones	1
Ronnie	1
Eleanor	1
Baron	1
Bauer	1
Swagger	1
Brandi	1
Mary	1
Halo	1
Augie	1
Craig	1
Pavlov	1
Ito	1
incredibly	1
Major	1
Shooter	1
Diogi	1
Sonny	1
Severus	1
Miguel	1
Eevee	1
Brownie	1
Napolean	1
Goose	1
Nugget	1
Jed	1
Monkey	1
Harry	1
Kody	1
Lassie	1
Rover	1
Rumble	1
Ginger	1
Clifford	1
Dewey	1
Shikha	1
Lili	1
Meatball	1
Zooey	1
Jersey	1
Burt	1

Venti	1
Dante	1
Orion	1
Roscoe	1
Darla	1
Bruno	1
Stuart	1
Jim	1
Ralphus	1
such	1
Maya	1
Mingus	1
Jimbo	1
Aja	1
Maisey	1
Noah	1
Grizzwald	1
Alfy	1
Koko	1
Rey	1
Duddles	1
Snoopy	1
Jordy	1
Milky	1
Thor	1
Mookie	1
Jarvis	1
Mimosa	1
Brady	1
Margo	1
Tycho	1
Monty	1
Sojourner	1
Arlo	1
Meera	1
Callie	1
Tobi	1
Sunshine	1
Lipton	1
Gabby	1
Bronte	1
Poppy	1
Rhino	1
Willow	1
Vixen	1
Barney	1
Tuck	1
Furzey	1
Cermet	1
Marlee	1
Arya	1
Einstein	1
Rumpole	1

Benny	1
Jarod	1
Wiggles	1
General	1
Sailor	1
Iggy	1
Snoop	1
Noosh	1
Odin	1
Georgie	1
Rontu	1
Cannon	1
Ralphy	1
Stella	1
Jiminus	1
Lillie	1
Coleman	1
Enchilada	1
all	1
Rueben	1
Cilantro	1
Karll	1
Sprout	1
Bloop	1
Ashleigh	1
Puff	1
Luther	1
Ivar	1
Jangle	1
Schnitzel	1
Berkeley	1
Ralphé	1
Charleson	1
Clyde	1
Flurpson	1
Pupcasso	1
Kayla	1
Kellogg	1
Aqua	1
Chase	1
Rorie	1
Simba	1
Bayley	1
Storkson	1
Remy	1
Chadrick	1
Buckley	1
Stefan	1
Livvie	1
Hermione	1
Ralpher	1
Aldrick	1
this	1

```
unacceptable        1
Rooney              1
Ziva                1
Harnold             1
Sid                 1
Carper              1
Vince               1
Sora                1
Beemo               1
Gunner              1
infuriating         1
Lacy                1
Tater               1
Olaf                1
Cecil               1
Karma               1
Bowie               1
Billy               1
Walker              1
Rodney              1
Malikai             1
Bobble              1
River               1
Jebberson           1
Farfle              1
Staniel             1
Kane                1
Opie                1
Lance               1
Alexanderson        1
Suki                1
Barclay             1
Skittle             1
Ebby                1
Link                1
Jennifur            1
Bluebert            1
Stephanus           1
old                 1
Zeus                1
Bertson             1
Nico                1
Michelangelope      1
Siba                1
Travis              1
Kanu                1
Edmund              1
Beya                1
Tonks               1
Rupert              1
Grey                1
Willem              1
Dixie               1
```

Al	1
Carbon	1
DonDon	1
Chevy	1
Tito	1
Brudge	1
Heinrich	1
Shadoe	1
Angel	1
Brat	1
Tove	1
my	1
Aubie	1
Kota	1
Glenn	1
Shakespeare	1
Sprinkles	1
Stark	1
Godi	1
Dale	1
Rizzo	1
Pinot	1
Dallas	1
Hero	1
Stormy	1
Mairi	1
Loomis	1
Deacon	1
Blue	1
Timmy	1
Combo	1
Jay	1
Mya	1
Strider	1
Wesley	1
Huck	1
O	1
Shelby	1
Sephie	1
Bonaparte	1
Tilly	1
Spanky	1
Jameson	1
Blu	1
Dietrich	1
Divine	1
Tripp	1
his	1
Cora	1
Bookstore	1
Wishes	1
Linus	1
Shiloh	1

```
Gustav              1
Arlen               1
Lenox               1
Harvey              1
Blanket             1
Geno                1
Lilah               1
Stewie              1
Zoe                 1
Gilbert             1
Rose                1
Theo                1
Fido                1
Emma                1
Spencer             1
Lilli               1
Boston              1
Brandonald          1
Corey               1
Leonard             1
Beckham             1
Devón               1
Gert                1
Dex                 1
Ace                 1
Tayzie              1
Grizzie             1
Christoper          1
Name: count, dtype: int64
```

In [227…
```
# How many times 'doggo' appears
df_archive.doggo.value_counts()
```

Out[227]:
```
doggo
doggo    97
Name: count, dtype: int64
```

In [228…
```
# How many times 'floofer' appears
df_archive.floofer.value_counts()
```

Out[228]:
```
floofer
floofer    10
Name: count, dtype: int64
```

In [229…
```
# How many times 'pupper' appears
df_archive.pupper.value_counts()
```

Out[229]:
```
pupper
pupper    257
Name: count, dtype: int64
```

In [230…
```
# How many times 'puppo' appears
df_archive.puppo.value_counts()
```

Out[230]:
```
puppo
puppo    30
Name: count, dtype: int64
```

In [231…
```
# See how many different denominators exist and their value counts
df_archive.rating_denominator.value_counts()
```

Out[231]:
```
rating_denominator
10     2333
11        3
50        3
20        2
80        2
70        1
7         1
15        1
150       1
170       1
0         1
90        1
40        1
130       1
110       1
16        1
120       1
2         1
Name: count, dtype: int64
```

In [232…
```
# See how the different numerators that exist and their value counts
df_archive.rating_numerator.value_counts()
```

Out[232]:
```
rating_numerator
12      558
11      464
10      461
13      351
9       158
8       102
7        55
14       54
5        37
6        32
3        19
4        17
2         9
1         9
75        2
15        2
420       2
0         2
80        1
144       1
17        1
26        1
20        1
121       1
143       1
44        1
60        1
45        1
50        1
99        1
204       1
1776      1
165       1
666       1
27        1
182       1
24        1
960       1
84        1
88        1
Name: count, dtype: int64
```

In [233…
```
# Create and view a df that holds rows that have a numerator other than '10'
other_than_10 = df_archive[df_archive['rating_denominator'] != 10]
other_than_10
```

Out[233]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
313	835246439529840640	8.352460e+17	2.625958e+07	2017-02-24 21:54:03 +0000	href="http

342	832088576586297345	8.320875e+17	3.058208e+07	2017-02-16 04:45:50 +0000	href="http
433	820690176645140481	NaN	NaN	2017-01-15 17:52:40 +0000	href="http
516	810984652412424192	NaN	NaN	2016-12-19 23:06:23 +0000	href="http
784	775096608509886464	NaN	NaN	2016-09-11 22:20:06 +0000	href="http
902	758467244762497024	NaN	NaN	2016-07-28 01:00:57 +0000	href="http
1068	740373189193256964	NaN	NaN	2016-06-08 02:41:38 +0000	href="http
1120	731156023742988288	NaN	NaN	2016-05-13 16:15:54 +0000	href="http
1165	722974582966214656	NaN	NaN	2016-04-21 02:25:47 +0000	href="http
1202	716439118184652801	NaN	NaN	2016-04-03 01:36:11 +0000	href="http
1228	713900603437621249	NaN	NaN	2016-03-27 01:29:02 +0000	href="http
1254	710658690886586372	NaN	NaN	2016-03-18 02:46:49 +0000	href="http
1274	709198395643068416	NaN	NaN	2016-03-14 02:04:08 +0000	href="http

1351	704054845121142784	NaN	NaN	2016-02-28 21:25:30 +0000	href="http
1433	697463031882764288	NaN	NaN	2016-02-10 16:51:59 +0000	href="http
1598	686035780142297088	6.860340e+17	4.196984e+09	2016-01-10 04:04:10 +0000	href="http
1634	684225744407494656	6.842229e+17	4.196984e+09	2016-01-05 04:11:44 +0000	href="http
1635	684222868335505415	NaN	NaN	2016-01-05 04:00:18 +0000	href="http
1662	682962037429899265	NaN	NaN	2016-01-01 16:30:13 +0000	href="http
1663	682808988178739200	6.827884e+17	4.196984e+09	2016-01-01 06:22:03 +0000	href="http
1779	677716515794329600	NaN	NaN	2015-12-18 05:06:23 +0000	href="http
1843	675853064436391936	NaN	NaN	2015-12-13 01:41:41 +0000	href="http
2335	666287406224695296	NaN	NaN	2015-11-16 16:11:11 +0000	href="http

Visusal assessment of df_archive

>

1. Retweets need to be removed
2. Several columns will not be used, therefore need to be removed ('in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp', 'expanded_urls')
3. Some dog names are 'non-names', such as 'a', 'an', 'the', etc. Non-names are not capitalized and will be removed.

Programmatic assessment of df_archive

>

1. 'tweet_id' column needs to be converted to 'string' format.
2. 'timestamp' column needs to be converted to 'datetime' format., and '+0000' needs to be removed from the end of each value.

Assessing df_image

In [234…
```
df_image.shape
```

Out[234]:
```
(2075, 12)
```

In [235…
```
df_image.sample(10)
```

Out[235]:

	tweet_id	jpg_url	img_num
1211	742528092657332225	https://pbs.twimg.com/media/Ck39W0JWUAApgnH.jpg	2
440	674447403907457024	https://pbs.twimg.com/media/CVweVUfW4AACPwI.jpg	1
1820	834458053273591808	https://pbs.twimg.com/media/C5SXK89XUAQg7GX.jpg	1
454	674764817387900928	https://pbs.twimg.com/media/CV0_BSuWIAIvE9k.jpg	2
1594	798697898615730177	https://pbs.twimg.com/media/CeRoBaxWEAABi0X.jpg	1
361	672828477930868736	https://pbs.twimg.com/media/CVZd7ttWcAEs2wP.jpg	1
2033	883482846933004288	https://pbs.twimg.com/media/DELC9dZXUAADqUk.jpg	1
437	674416750885273600	https://pbs.twimg.com/media/CVwCdCFW4AUHY4D.jpg	1
708	685198997565345792	https://pbs.twimg.com/media/CYJQxvJW8AAkkws.jpg	1
638	681281657291280384	https://pbs.twimg.com/media/CXRmDfWWMAADCdc.jpg	1

In [236…
```
df_image.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 [237…
```
df_image.describe()
```

Out[237]:

	tweet_id	img_num	p1_conf	p2_conf	p3_conf
count	2.075000e+03	2075.000000	2075.000000	2.075000e+03	2.075000e+03
mean	7.384514e+17	1.203855	0.594548	1.345886e-01	6.032417e-02
std	6.785203e+16	0.561875	0.271174	1.006657e-01	5.090593e-02
min	6.660209e+17	1.000000	0.044333	1.011300e-08	1.740170e-10
25%	6.764835e+17	1.000000	0.364412	5.388625e-02	1.622240e-02
50%	7.119988e+17	1.000000	0.588230	1.181810e-01	4.944380e-02
75%	7.932034e+17	1.000000	0.843855	1.955655e-01	9.180755e-02
max	8.924206e+17	4.000000	1.000000	4.880140e-01	2.734190e-01

In [238…
```
df_image.duplicated().sum()
```

Out[238]:
```
0
```

In [239…
```
df_image.p1.value_counts()
```

Out[239]:
```
p1
golden_retriever             150
Labrador_retriever           100
Pembroke                      89
Chihuahua                     83
pug                           57
chow                          44
Samoyed                       43
toy_poodle                    39
Pomeranian                    38
cocker_spaniel                30
malamute                      30
French_bulldog                26
miniature_pinscher            23
Chesapeake_Bay_retriever      23
seat_belt                     22
Siberian_husky                20
German_shepherd               20
Staffordshire_bullterrier     20
Cardigan                      19
web_site                      19
Eskimo_dog                    18
Maltese_dog                   18
Shetland_sheepdog             18
teddy                         18
beagle                        18
Lakeland_terrier              17
Rottweiler                    17
Shih-Tzu                      17
Italian_greyhound             16
```

```
kuvasz                            16
West_Highland_white_terrier       14
Great_Pyrenees                    14
American_Staffordshire_terrier    13
basset                            13
Pekinese                          13
dalmatian                         13
vizsla                            13
Airedale                          12
Border_collie                     12
Old_English_sheepdog              12
kelpie                            11
Blenheim_spaniel                  11
soft-coated_wheaten_terrier       11
schipperke                        10
Bernese_mountain_dog              10
English_springer                  10
collie                            10
boxer                             10
whippet                            9
dingo                              9
malinois                           9
Great_Dane                         9
borzoi                             9
tennis_ball                        9
Boston_bull                        9
flat-coated_retriever              8
standard_poodle                    8
Yorkshire_terrier                  8
doormat                            8
Norwegian_elkhound                 8
Doberman                           8
papillon                           8
miniature_poodle                   8
English_setter                     8
Siamese_cat                        7
Border_terrier                     7
bath_towel                         7
basenji                            7
German_short-haired_pointer        7
tub                                7
hamster                            7
swing                              7
bloodhound                         7
Norfolk_terrier                    7
Brittany_spaniel                   7
Saint_Bernard                      7
home_theater                       6
llama                              6
ice_bear                           6
Irish_terrier                      6
Irish_setter                       6
car_mirror                         6
```

Dandie_Dinmont	6
redbone	6
porcupine	5
shopping_cart	5
Tibetan_mastiff	5
Walker_hound	5
bull_mastiff	5
Lhasa	5
ox	5
minivan	5
Bedlington_terrier	5
hippopotamus	5
Newfoundland	5
brown_bear	4
hog	4
bow_tie	4
keeshond	4
wombat	4
Norwich_terrier	4
jigsaw_puzzle	4
barrow	4
miniature_schnauzer	4
Rhodesian_ridgeback	4
Arctic_fox	4
patio	4
Mexican_hairless	4
bluetick	4
Gordon_setter	4
Afghan_hound	4
Saluki	4
bathtub	4
Weimaraner	4
guinea_pig	4
Tibetan_terrier	4
goose	4
balloon	3
muzzle	3
briard	3
prison	3
wood_rabbit	3
mousetrap	3
Greater_Swiss_Mountain_dog	3
refrigerator	3
sea_lion	3
toilet_tissue	3
stone_wall	3
white_wolf	3
Leonberg	3
motor_scooter	3
space_heater	3
ram	3
cowboy_hat	3
Irish_water_spaniel	3

Scottish_deerhound	3
dishwasher	3
Welsh_springer_spaniel	3
vacuum	3
triceratops	3
Christmas_stocking	3
komondor	3
ski_mask	3
common_iguana	3
washbasin	3
curly-coated_retriever	3
Arabian_camel	3
cairn	3
window_shade	3
Ibizan_hound	3
jack-o'-lantern	3
comic_book	3
seashore	3
Brabancon_griffon	3
giant_schnauzer	3
gas_pump	2
jellyfish	2
shower_curtain	2
lakeside	2
black-and-tan_coonhound	2
tusker	2
bustard	2
acorn_squash	2
hen	2
paper_towel	2
koala	2
bubble	2
leatherback_turtle	2
Angora	2
wire-haired_fox_terrier	2
cash_machine	2
wallaby	2
paddle	2
frilled_lizard	2
street_sign	2
toyshop	2
dough	2
upright	2
birdhouse	2
tabby	2
axolotl	2
snorkel	2
dogsled	2
hermit_crab	2
snail	2
Australian_terrier	2
hyena	2
meerkat	2

```
toy_terrier                    2
geyser                         2
feather_boa                    2
Loafer                         2
Appenzeller                    2
weasel                         2
chimpanzee                     2
laptop                         2
badger                         2
sorrel                         2
Sussex_spaniel                 2
wool                           2
ostrich                        2
box_turtle                     2
gondola                        2
tiger_shark                    1
bib                            1
binoculars                     1
sulphur-crested_cockatoo       1
coil                           1
agama                          1
wild_boar                      1
traffic_light                  1
hotdog                         1
handkerchief                   1
espresso                       1
cup                            1
bonnet                         1
polecat                        1
alp                            1
coho                           1
hammer                         1
studio_couch                   1
cliff                          1
timber_wolf                    1
nail                           1
lawn_mower                     1
three-toed_sloth               1
sunglasses                     1
desktop_computer               1
rapeseed                       1
hand_blower                    1
peacock                        1
American_black_bear            1
loupe                          1
school_bus                     1
cowboy_boot                    1
jersey                         1
wooden_spoon                   1
leopard                        1
mortarboard                    1
teapot                         1
military_uniform               1
```

washer	1
coffee_mug	1
fountain	1
pencil_box	1
barbell	1
grille	1
revolver	1
envelope	1
syringe	1
marmot	1
pole	1
basketball	1
tricycle	1
convertible	1
limousine	1
restaurant	1
shield	1
rotisserie	1
bookcase	1
conch	1
skunk	1
bookshop	1
radio_telescope	1
cougar	1
African_grey	1
coral_reef	1
lion	1
maillot	1
Madagascar_cat	1
Egyptian_cat	1
silky_terrier	1
giant_panda	1
long-horned_beetle	1
clumber	1
sundial	1
padlock	1
pool_table	1
quilt	1
beach_wagon	1
remote_control	1
bakery	1
pedestal	1
four-poster	1
cheeseburger	1
otter	1
suit	1
killer_whale	1
terrapin	1
cuirass	1
microwave	1
starfish	1
sandbar	1
leaf_beetle	1

lynx	1
water_bottle	1
toilet_seat	1
shopping_basket	1
robin	1
crash_helmet	1
slug	1
soccer_ball	1
African_crocodile	1
tick	1
ocarina	1
bearskin	1
bow	1
carton	1
candle	1
bee_eater	1
china_cabinet	1
banana	1
dhole	1
sea_urchin	1
lacewing	1
ping-pong_ball	1
platypus	1
scorpion	1
flamingo	1
microphone	1
mud_turtle	1
pitcher	1
African_hunting_dog	1
boathouse	1
picket_fence	1
pot	1
zebra	1
piggy_bank	1
park_bench	1
prayer_rug	1
stove	1
king_penguin	1
tailed_frog	1
snowmobile	1
ibex	1
electric_fan	1
sliding_door	1
damselfly	1
hare	1
fiddler_crab	1
bannister	1
crane	1
Scotch_terrier	1
bighorn	1
standard_schnauzer	1
bison	1
ice_lolly	1

```
hay                         1
dining_table                1
groenendael                 1
beaver                      1
swab                        1
grey_fox                    1
hummingbird                 1
clog                        1
fire_engine                 1
minibus                     1
cheetah                     1
walking_stick               1
canoe                       1
trombone                    1
book_jacket                 1
rain_barrel                 1
black-footed_ferret         1
guenon                      1
Japanese_spaniel            1
water_buffalo               1
maze                        1
harp                        1
panpipe                     1
mailbox                     1
EntleBucher                 1
earthstar                   1
pillow                      1
carousel                    1
bald_eagle                  1
lorikeet                    1
orange                      1
Name: count, dtype: int64
```

In [240…
```
df_image.tweet_id.duplicated().sum()
```

Out[240]:
```
0
```

In [241…
```
# if 'jpg_url' is duplicated, perhaps it is a retweet. lets check
df_image.jpg_url.duplicated().sum()
```

Out[241]:
```
66
```

In [242…
```
# See the rows where the 'jpg_url' is duplicated
df_image[df_image['jpg_url'].duplicated()]
```

Out[242]:

	tweet_id	jpg_url	img_num
1297	752309394570878976	https://pbs.twimg.com/ext_tw_video_thumb/67535...	1
1315	754874841593970688	https://pbs.twimg.com/media/CWza7kpWcAAdYLc.jpg	1
1333	757729163776290825	https://pbs.twimg.com/media/CWyD2HGUYAQ1Xa7.jpg	2

1345	759159934323924993	https://pbs.twimg.com/media/CU1zsMSUAAAS0qW.jpg	1
1349	759566828574212096	https://pbs.twimg.com/media/CkNjahBXAAQ2kWo.jpg	1
1364	761371037149827077	https://pbs.twimg.com/tweet_video_thumb/CeBym7...	1
1368	761750502866649088	https://pbs.twimg.com/media/CYLDikFWEAAIy1y.jpg	1
1387	766078092750233600	https://pbs.twimg.com/media/ChK1tdBWwAQ1flD.jpg	1
1407	770093767776997377	https://pbs.twimg.com/media/CkjMx99UoAM2B1a.jpg	1
1417	771171053431250945	https://pbs.twimg.com/media/CVgdFjNWEAAxmbq.jpg	3
1427	772615324260794368	https://pbs.twimg.com/media/Cp6db4-XYAAMmqL.jpg	1
1446	775898661951791106	https://pbs.twimg.com/media/CiyHLocU4AI2pJu.jpg	1
1453	776819012571455488	https://pbs.twimg.com/media/CW88XN4WsAAIo8r.jpg	3
1456	777641927919427584	https://pbs.twimg.com/media/CmoPdmHW8AAi8BI.jpg	1
1463	778396591732486144	https://pbs.twimg.com/media/CcG07BYW0AErrC9.jpg	1
1476	780496263422808064	https://pbs.twimg.com/media/Ck2d7tJWUAEPTL3.jpg	1
1487	782021823840026624	https://pbs.twimg.com/media/CdHwZd0VIAA4792.jpg	1
1495	783347506784731136	https://pbs.twimg.com/media/CVuQ2LeUsAAIe3s.jpg	1
1510	786036967502913536	https://pbs.twimg.com/media/CtKHLuCWYAA2TTs.jpg	1
1522	788070120937619456	https://pbs.twimg.com/media/Co-hmcYXYAASkiG.jpg	1
1538	790723298204217344	https://pbs.twimg.com/media/CvaYgDOWgAEfjIs.jpg	1
1541	791026214425268224	https://pbs.twimg.com/media/CpmyNumW8AAAJGj.jpg	1
1564	793614319594401792	https://pbs.twimg.com/media/CvyVxQRWEAAdSZS.jpg	1
1569	794355576146903043	https://pbs.twimg.com/media/CvJCabcWgAIoUxW.jpg	1
1571	794983741416415232	https://pbs.twimg.com/media/CvT6IV6WEAQhhV5.jpg	3
1579	796177847564038144	https://pbs.twimg.com/media/Cwx99rpW8AMk_Ie.jpg	1
1588	798340744599797760	https://pbs.twimg.com/media/CrXhIqBW8AA6Bse.jpg	1
1589	798628517273620480	https://pbs.twimg.com/media/CUN4Or5UAAAa5K4.jpg	1
1590	798644042770751489	https://pbs.twimg.com/media/CU3mITUWIAAfyQS.jpg	1
1591	798665375516884993	https://pbs.twimg.com/media/CVMOlMiWwAA4Yxl.jpg	1
1592	798673117451325440	https://pbs.twimg.com/media/CV_cnjHWUAADc-c.jpg	1
1593	798694562394996736	https://pbs.twimg.com/media/Cbs3DOAXIAAp3Bd.jpg	1
1594	798697898615730177	https://pbs.twimg.com/media/CeRoBaxWEAABi0X.jpg	1
1601	799774291445383169	https://pbs.twimg.com/media/CsGnz64WYAEIDHJ.jpg	1
1605	800443802682937345	https://pbs.twimg.com/media/CsVO7ljW8AAckRD.jpg	1

1615	802247111496568832	https://pbs.twimg.com/media/Cs_DYr1XEAA54Pu.jpg	1	
1619	802624713319034886	https://pbs.twimg.com/media/CsrjryzWgAAZY00.jpg	1	
1624	803692223237865472	https://pbs.twimg.com/media/CZhn-QAWwAASQan.jpg	1	
1627	804413760345620481	https://pbs.twimg.com/media/CuRDF-XWcAIZSer.jpg	1	
1634	805958939288408065	https://pbs.twimg.com/media/CtzKC7zXEAALfSo.jpg	1	
1636	806242860592926720	https://pbs.twimg.com/media/Ct72q9jWcAAhlnw.jpg	2	
1640	807059379405148160	https://pbs.twimg.com/media/Ct2qO5PXEAE6eB0.jpg	1	
1645	808134635716833280	https://pbs.twimg.com/media/Cx5R8wPVEAALa9r.jpg	1	
1652	809808892968534016	https://pbs.twimg.com/media/CwS4aqZXUAAe3IO.jpg	1	
1683	813944609378369540	https://pbs.twimg.com/media/Cveg1-NXgAASaaT.jpg	1	
1693	816014286006976512	https://pbs.twimg.com/media/CiibOMzUYAA9Mxz.jpg	1	
1699	816829038950027264	https://pbs.twimg.com/media/CvoBPWRWgAA4het.jpg	1	
1703	817181837579653120	https://pbs.twimg.com/ext_tw_video_thumb/81596...	1	
1712	818588835076603904	https://pbs.twimg.com/media/Crwxb5yWgAAX5P_.jpg	1	
1717	819015331746349057	https://pbs.twimg.com/media/C12x-JTVIAAzdfl.jpg	4	
1718	819015337530290176	https://pbs.twimg.com/media/C12whDoVEAALRxa.jpg	1	
1727	820446719150292993	https://pbs.twimg.com/media/CxqsX-8XUAAEvjD.jpg	3	
1736	821813639212650496	https://pbs.twimg.com/media/CtVAvX-WIAAcGTf.jpg	1	
1742	822647212903690241	https://pbs.twimg.com/media/C2oRbOuWEAAbVSl.jpg	1	
1746	823269594223824897	https://pbs.twimg.com/media/C2kzTGxWEAEOpPL.jpg	1	
1755	824796380199809024	https://pbs.twimg.com/media/CwiuEJmW8AAZnit.jpg	2	
1789	829878982036299777	https://pbs.twimg.com/media/C3nygbBWQAAjwcW.jpg	1	
1803	832040443403784192	https://pbs.twimg.com/media/Cq9guJ5WgAADfpF.jpg	1	
1804	832215726631055365	https://pbs.twimg.com/media/CwJR1okWIAA6XMp.jpg	1	
1858	841833993020538882	https://pbs.twimg.com/ext_tw_video_thumb/81742...	1	
1864	842892208864923648	https://pbs.twimg.com/ext_tw_video_thumb/80710...	1	
1903	851953902622658560	https://pbs.twimg.com/media/C4KHj-nWQAA3poV.jpg	1	S
1944	861769973181624320	https://pbs.twimg.com/media/CzG425nWgAAnP7P.jpg	2	
1992	873697596434513921	https://pbs.twimg.com/media/DA7iHL5U0AA1OQo.jpg	1	
2041	885311592912609280	https://pbs.twimg.com/media/C4bTH6nWMAAX_bJ.jpg	1	
2055	888202515573088257	https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg	2	

In [243…
```
# View the duplicate URLs, they can be explored with the clickable links
duplicate_jpg = df_image.groupby('jpg_url').size()
duplicates = duplicate_jpg[duplicate_jpg >1]
duplicates
```

Out[243]:
```
jpg_url
https://pbs.twimg.com/ext_tw_video_thumb/675354114423808004/pu/img/qL1R_nGL
qa6lmkOx.jpg    2
https://pbs.twimg.com/ext_tw_video_thumb/807106774843039744/pu/img/8XZg1xW3
5Xp2J6JW.jpg    2
https://pbs.twimg.com/ext_tw_video_thumb/815965888126062592/pu/img/JleSw4wR
hgKDWQj5.jpg    2
https://pbs.twimg.com/ext_tw_video_thumb/817423809049493505/pu/img/5OFW0yue
Fu9oTUiQ.jpg    2
https://pbs.twimg.com/media/C12whDoVEAALRxa.jpg
2
https://pbs.twimg.com/media/C12x-JTVIAAzdfl.jpg
2
https://pbs.twimg.com/media/C2kzTGxWEAEOpPL.jpg
2
https://pbs.twimg.com/media/C2oRbOuWEAAbVSl.jpg
2
https://pbs.twimg.com/media/C3nygbBWQAAjwcW.jpg
2
https://pbs.twimg.com/media/C4KHj-nWQAA3poV.jpg
2
https://pbs.twimg.com/media/C4bTH6nWMAAX_bJ.jpg
2
https://pbs.twimg.com/media/CU1zsMSUAAAS0qW.jpg
2
https://pbs.twimg.com/media/CU3mITUWIAAfyQS.jpg
2
https://pbs.twimg.com/media/CUN4Or5UAAAa5K4.jpg
2
https://pbs.twimg.com/media/CVMOlMiWwAA4Yxl.jpg
2
https://pbs.twimg.com/media/CV_cnjHWUAADc-c.jpg
2
https://pbs.twimg.com/media/CVgdFjNWEAAxmbq.jpg
2
https://pbs.twimg.com/media/CVuQ2LeUsAAIe3s.jpg
2
https://pbs.twimg.com/media/CW88XN4WsAAlo8r.jpg
2
https://pbs.twimg.com/media/CWyD2HGUYAQ1Xa7.jpg
2
https://pbs.twimg.com/media/CWza7kpWcAAdYLc.jpg
2
https://pbs.twimg.com/media/CYLDikFWEAAIy1y.jpg
2
https://pbs.twimg.com/media/CZhn-QAWwAASQan.jpg
2
https://pbs.twimg.com/media/Cbs3DOAXIAAp3Bd.jpg
```

```
2
https://pbs.twimg.com/media/CcG07BYW0AErrC9.jpg
2
https://pbs.twimg.com/media/CdHwZd0VIAA4792.jpg
2
https://pbs.twimg.com/media/CeRoBaxWEAABi0X.jpg
2
https://pbs.twimg.com/media/ChK1tdBWwAQ1flD.jpg
2
https://pbs.twimg.com/media/CiibOMzUYAA9Mxz.jpg
2
https://pbs.twimg.com/media/CiyHLocU4AI2pJu.jpg
2
https://pbs.twimg.com/media/Ck2d7tJWUAEPTL3.jpg
2
https://pbs.twimg.com/media/CkNjahBXAAQ2kWo.jpg
2
https://pbs.twimg.com/media/CkjMx99UoAM2B1a.jpg
2
https://pbs.twimg.com/media/CmoPdmHW8AAi8BI.jpg
2
https://pbs.twimg.com/media/Co-hmcYXYAASkiG.jpg
2
https://pbs.twimg.com/media/Cp6db4-XYAAMmqL.jpg
2
https://pbs.twimg.com/media/CpmyNumW8AAAJGj.jpg
2
https://pbs.twimg.com/media/Cq9guJ5WgAADfpF.jpg
2
https://pbs.twimg.com/media/CrXhIqBW8AA6Bse.jpg
2
https://pbs.twimg.com/media/Crwxb5yWgAAX5P_.jpg
2
https://pbs.twimg.com/media/CsGnz64WYAEIDHJ.jpg
2
https://pbs.twimg.com/media/CsVO7ljW8AAckRD.jpg
2
https://pbs.twimg.com/media/Cs_DYr1XEAA54Pu.jpg
2
https://pbs.twimg.com/media/CsrjryzWgAAZY00.jpg
2
https://pbs.twimg.com/media/Ct2qO5PXEAE6eB0.jpg
2
https://pbs.twimg.com/media/Ct72q9jWcAAhlnw.jpg
2
https://pbs.twimg.com/media/CtKHLuCWYAA2TTs.jpg
2
https://pbs.twimg.com/media/CtVAvX-WIAAcGTf.jpg
2
https://pbs.twimg.com/media/CtzKC7zXEAALfSo.jpg
2
https://pbs.twimg.com/media/CuRDF-XWcAIZSer.jpg
2
```

```
https://pbs.twimg.com/media/CvJCabcWgAIoUxW.jpg
2
https://pbs.twimg.com/media/CvT6IV6WEAQhhV5.jpg
2
https://pbs.twimg.com/media/CvaYgDOWgAEfjls.jpg
2
https://pbs.twimg.com/media/Cveg1-NXgAASaaT.jpg
2
https://pbs.twimg.com/media/CvoBPWRWgAA4het.jpg
2
https://pbs.twimg.com/media/CvyVxQRWEAAdSZS.jpg
2
https://pbs.twimg.com/media/CwJR1okWIAA6XMp.jpg
2
https://pbs.twimg.com/media/CwS4aqZXUAAe3IO.jpg
2
https://pbs.twimg.com/media/CwiuEJmW8AAZnit.jpg
2
https://pbs.twimg.com/media/Cwx99rpW8AMk_Ie.jpg
2
https://pbs.twimg.com/media/Cx5R8wPVEAALa9r.jpg
2
https://pbs.twimg.com/media/CxqsX-8XUAAEvjD.jpg
2
https://pbs.twimg.com/media/CzG425nWgAAnP7P.jpg
2
https://pbs.twimg.com/media/DA7iHL5U0AA1OQo.jpg
2
https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg
2
https://pbs.twimg.com/tweet_video_thumb/CeBym7oXEAEWbEg.jpg
2
dtype: int64
```

In [244…

```
df_image[df_image['jpg_url'] == 'https://pbs.twimg.com/media/C12whDoVEAALRxa
```

Out[244]:

	tweet_id	jpg_url	img_num	
1715	819004803107983360	https://pbs.twimg.com/media/C12whDoVEAALRxa.jpg	1	star
1718	819015337530290176	https://pbs.twimg.com/media/C12whDoVEAALRxa.jpg	1	star

Visual assessment of df_image

>

1. We will only use rows where columns 'p1_dog', 'p2_dog', and 'p3_dog' are not 'False', as the image prediction algorithm suggests these are NOT dogs. We will then filter the dataframe and keep only the values that correspond to the highest confidence level
2. There are duplicates in the 'jpg_url' column, indicating they are possibly retweets.
3. We will get rid of unnecessary columns for our analysis
4. The dog breeds need to be displayed in a standard, consistent format.

Programmatic assessment of df_image

1. The dataframe contains duplicates in the 'jpg_url' column, and are likely retweets and should be removed.

Assessing df_json

In [245…
```
df_json.shape
```

Out[245]:
```
(2354, 3)
```

In [246…
```
df_json.sample(10)
```

Out[246]:

	id	retweet_count	favorite_count
1350	703774238772166656	526	2020
921	755955933503782912	3285	8092
1331	705442520700944385	1859	4877
1707	680609293079592961	816	2906
406	823699002998870016	2772	13826
1959	673359818736984064	728	1558
268	841320156043304961	6080	21402
2185	668979806671884288	382	842
1090	737322739594330112	907	3953
492	813202720496779264	2090	10192

In [247…
```
df_json.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              2354 non-null   int64
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
dtypes: int64(3)
memory usage: 55.3 KB
```

In [248…
```
df_json.describe()
```

Out[248]:

	id	retweet_count	favorite_count
count	2.354000e+03	2354.000000	2354.000000
mean	7.426978e+17	3164.797366	8080.968564
std	6.852812e+16	5284.770364	11814.771334
min	6.660209e+17	0.000000	0.000000
25%	6.783975e+17	624.500000	1415.000000
50%	7.194596e+17	1473.500000	3603.500000
75%	7.993058e+17	3652.000000	10122.250000
max	8.924206e+17	79515.000000	132810.000000

In [249…
```
df_json.duplicated().sum()
```

```
Out[249]: 0
```

Visual assessment of df_json

>

1. rename 'id' column 'tweet_id'

Quality issues

df_archive

>

1. Retweets need to be removed
2. Several columns will not be used, therefore need to be removed ('in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp', 'expanded_urls')
3. Some dog names are 'non-names', such as 'a', 'an', 'the', etc. Non-names are not capitalized and will be removed.
4. 'tweet_id' column needs to be converted to 'string' format.
5. 'timestamp' column needs to be converted to 'datetime' format., and '+0000' needs to be removed from the end of each value.

df_image

1. We will only use rows where columns 'p1_dog', 'p2_dog', and 'p3_dog' are not 'False', as the image prediction algorithm suggests these are NOT dogs.
2. We will keep only the dog breed that was guessed with the highest confidence level. We will place these values in a column called 'dog_breed'.
3. Next we will drop unnecessary columns from df_image_clean.
4. The dog breeds need to be displayed in a standard, consistent format (we will use snake_case).
5. The dataframe contains duplicates in the 'jpg_url' column, and are likely retweets and should be removed.

df_json

1. The 'id' column needs to be renamed 'tweet_id'.

> 2. The new 'tweet_id' data type needs to be converted to string from int.

Tidiness issues

>

> 1. Four columns, doggo, floofer, puppo and pupper need to be one column, making the values ‘doggo’, ‘floofer’, ‘puppo’, and ‘pupper’ categorical values.
> 2. All three tables need to be combined into one master table.

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 [250…
```
# Make copies of original pieces of data
df_archive_clean = df_archive.copy()
df_image_clean = df_image.copy()
df_json_clean = df_json.copy()
```

Issue #1:

> Remove all retweets and replies.

Define:

> We will keep only original tweets. If the columns 'retweeted_status_id' and 'in_reply_to_status_id' are NOT null, they will be filtered out.

Code

In [251…
```
# Keep only rows with null values in the 'retweeted_status_id' column.
df_archive_clean = df_archive_clean[df_archive_clean.retweeted_status_id.isn

# Keep only rows with null values in the 'in_reply_to_status_id' column
df_archive_clean = df_archive_clean[df_archive_clean.in_reply_to_status_id.i
```

Test

In [252…
```
df_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2097 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2097 non-null   int64  
 1   in_reply_to_status_id       0 non-null      float64
 2   in_reply_to_user_id         0 non-null      float64
 3   timestamp                   2097 non-null   object 
 4   source                      2097 non-null   object 
 5   text                        2097 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               2094 non-null   object 
 10  rating_numerator            2097 non-null   int64  
 11  rating_denominator          2097 non-null   int64  
 12  name                        1494 non-null   object 
 13  doggo                       83 non-null     object 
 14  floofer                     10 non-null     object 
 15  pupper                      230 non-null    object 
 16  puppo                       24 non-null     object 
dtypes: float64(4), int64(3), object(10)
memory usage: 294.9+ KB
```

Issue #2:

> Get rid of unnecesary columns for our analysis

Define

> Drop columns related to retweets, source, and expanded_urls

Code

In [253…
```
# Create a list of columns to drop
columns_to_drop = ['in_reply_to_status_id', 'in_reply_to_user_id', 'retweete
                   'source', 'retweeted_status_user_id', 'retweeted_status_t
                   'expanded_urls']

# Drop the columns contained in the list
df_archive_clean.drop(columns=columns_to_drop, axis=1, inplace=True)
```

Test

In [254…
```
df_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2097 entries, 0 to 2355
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   tweet_id            2097 non-null   int64 
 1   timestamp           2097 non-null   object
 2   text                2097 non-null   object
 3   rating_numerator    2097 non-null   int64 
 4   rating_denominator  2097 non-null   int64 
 5   name                1494 non-null   object
 6   doggo               83 non-null     object
 7   floofer             10 non-null     object
 8   pupper              230 non-null    object
 9   puppo               24 non-null     object
dtypes: int64(3), object(7)
memory usage: 180.2+ KB
```

Issue #3:

> Some values in the 'name' column are not names, such as 'a', 'an' 'the', 'get', ect. All values that are not actual names are not capitalized. All real names are capitalized.

Define

> We will replace all non-capitalized words in the 'name' column with 'None'.

Code

In [255…
```
# Replace non-capitalized values (these are not actual names) with 'None'
df_archive_clean['name'] = df_archive_clean['name'].apply(lambda x: x if (st
```

Test

In [256…
```
df_archive_clean.name.value_counts()
```

Out[256]:

name	
None	104
Lucy	11
Charlie	11
Cooper	10
Oliver	10
Penny	9
Tucker	9
Winston	8
Lola	8
Sadie	8
Toby	7
Daisy	7
Jax	6
Koda	6
Bella	6
Oscar	6
Bailey	6
Stanley	6
Bo	6
Scout	5
Rusty	5
Leo	5
Louis	5
Dave	5
Chester	5
Buddy	5
Milo	5
Bentley	5
Cassie	4
Scooter	4
Jack	4
Reggie	4
Larry	4
Clarence	4
Bear	4
Boomer	4
Clark	4
Dexter	4
Duke	4
Alfie	4
Archie	4
Sophie	4
Finn	4
Brody	4
Oakley	4
Gus	4
Derek	4
Sammy	4
Phil	4

Maggie	4
Jeffrey	4
Jerry	4
Chip	4
Winnie	4
George	4
Gary	4
Loki	3
Otis	3
Wyatt	3
Nala	3
Carl	3
Walter	3
Rosie	3
Malcolm	3
Mia	3
Samson	3
Doug	3
Ruby	3
Bruce	3
Louie	3
Riley	3
Maximus	3
Wilson	3
Luna	3
Hank	3
Lily	3
Max	3
Kyle	3
Vincent	3
Gizmo	3
Zeke	3
Beau	3
Steven	3
Shadow	3
Moose	3
Jimothy	3
Calvin	3
Gerald	3
Olive	3
Wallace	3
Waffles	3
Sunny	3
Ted	3
Reese	3
Klevin	3
Earl	3
Zoey	3
Reginald	3
Ellie	3
Sebastian	3
Axel	2
Jeph	2

Oliviér	2
Charles	2
Atlas	2
Watson	2
Oshie	2
Rufus	2
Ash	2
Bob	2
Juno	2
Layla	2
Olivia	2
Rocky	2
Eli	2
Rocco	2
Tyr	2
Ava	2
Opal	2
Baxter	2
Terry	2
Hunter	2
Moe	2
Keith	2
Theodore	2
Titan	2
Mister	2
Keurig	2
Jimison	2
Gabe	2
Luca	2
Marley	2
Abby	2
Lenny	2
Percy	2
Doc	2
Hammond	2
Django	2
Sugar	2
Peaches	2
Bernie	2
Kilo	2
Kenneth	2
Atticus	2
Maxaroni	2
Piper	2
Sansa	2
Fiona	2
Herald	2
Albert	2
Lennon	2
Fred	2
Wally	2
Phred	2
Brad	2

Herm	2
Ollie	2
Phineas	2
Misty	2
Odie	2
Harold	2
Frankie	2
Ozzy	2
Kreg	2
Patrick	2
Smokey	2
Flávio	2
Cody	2
Bisquick	2
Arnie	2
Bubbles	2
Churlie	2
Coco	2
Roosevelt	2
Benji	2
Albus	2
Neptune	2
Belle	2
Pippa	2
Cash	2
Benedict	2
Rory	2
Yogi	2
Franklin	2
Dash	2
Solomon	2
Chipson	2
Remington	2
Sampson	2
Chuckles	2
Kevin	2
Cupcake	2
Penelope	2
Thumas	2
Elliot	2
Jesse	2
Romeo	2
Finley	2
Curtis	2
Calbert	2
Nelly	2
Hobbes	2
Paisley	2
Sam	2
Sandy	2
Alice	2
Sarge	2
Aspen	2

Kreggory	2
Griffin	2
Jackson	2
Lorenzo	2
Chet	2
Colby	2
Blitz	2
Coops	2
CeCe	2
Jiminy	2
Raymond	2
Lou	2
Kirby	2
Indie	2
Linda	2
Pickles	2
Lincoln	2
Panda	2
Dakota	2
Frank	2
Maddie	2
Trooper	2
Crystal	2
Obi	1
Asher	1
Horace	1
Tino	1
Ulysses	1
Kulet	1
Carly	1
Sweets	1
Toffee	1
Apollo	1
Lupe	1
Mona	1
Gerbald	1
Gordon	1
Perry	1
Darrel	1
Oreo	1
Bruiser	1
Lulu	1
Hubertson	1
Petrick	1
Brooks	1
Taco	1
Marty	1
Crimson	1
Joey	1
Brandy	1
Lorelei	1
Banjo	1
Birf	1

Bobby	1
Tess	1
Tiger	1
Todo	1
Freddery	1
Tyrone	1
Murphy	1
Fiji	1
Rilo	1
Bilbo	1
Coopson	1
Yoda	1
Millie	1
Crouton	1
Daniel	1
Kaia	1
Dotsy	1
Rambo	1
Eazy	1
Fillup	1
Miley	1
Charl	1
Reagan	1
Yukon	1
Cuddles	1
Claude	1
Jessiga	1
Rudy	1
Socks	1
Barry	1
Finnegus	1
Rodney	1
Malikai	1
Bobble	1
River	1
Jebberson	1
Farfle	1
Jiminus	1
Harper	1
Clarkus	1
Kathmandu	1
Ralphson	1
Katie	1
Kara	1
Adele	1
Zara	1
Ambrose	1
Bode	1
Terrenth	1
Chesterson	1
Lucia	1
Carter	1
Ole	1

Pherb	1
Ferg	1
Eriq	1
Oddie	1
Maxwell	1
Geoff	1
Covach	1
Durg	1
Fynn	1
Ricky	1
Lucky	1
Trip	1
Blipson	1
Hamrick	1
Pubert	1
Frönq	1
Derby	1
Lizzie	1
Ember	1
Blakely	1
Marq	1
Kramer	1
Lolo	1
Hazel	1
Cedrick	1
Vinscent	1
Reptar	1
Trevith	1
Berb	1
Colin	1
Brian	1
Grady	1
Kobe	1
Chuck	1
Bodie	1
Dunkin	1
Tupawc	1
Amber	1
Herschel	1
Edgar	1
Teddy	1
Kingsley	1
Brockly	1
Richie	1
Molly	1
Glacier	1
Rufio	1
Champ	1
Skye	1
Fabio	1
Randall	1
Liam	1
Tommy	1

```
Ben                1
Raphael            1
Julio              1
Andru              1
Kloey              1
Shawwn             1
Kollin             1
Harrison           1
Ronduh             1
Billl              1
Saydee             1
Dug                1
Sully              1
Kirk               1
Ralf               1
Clarq              1
Jaspers            1
Samsom             1
Sandra             1
Butters            1
Nigel              1
Peanut             1
Holly              1
Jett               1
Amy                1
Sage               1
Andy               1
Mason              1
Trigger            1
Antony             1
Creg               1
Traviss            1
Gin                1
Jeffrie            1
Danny              1
Ester              1
Pluto              1
Bloo               1
Edd                1
Paull              1
Willy              1
Herb               1
Damon              1
Terrance           1
Chaz               1
Ozzie              1
Fwed               1
Stu                1
Tedrick            1
Shaggy             1
Filup              1
Kial               1
Naphaniel          1
```

Dook	1
Hall	1
Philippe	1
Biden	1
Genevieve	1
Jeremy	1
Joshwa	1
Timison	1
Bradlay	1
Pipsy	1
Clybe	1
Keet	1
Carll	1
Jockson	1
Josep	1
Lugan	1
Erik	1
Cleopatricia	1
Skittles	1
Ron	1
Jaycob	1
Lambeau	1
Ruffles	1
Amélie	1
Bobb	1
Banditt	1
Kevon	1
Winifred	1
Hanz	1
Zeek	1
Timofy	1
Maks	1
Jomathan	1
Kallie	1
Marvin	1
Spark	1
Gòrdón	1
Jo	1
DayZ	1
Jareld	1
Torque	1
Anthony	1
Sparky	1
Tanner	1
Donny	1
Humphrey	1
Tassy	1
Juckson	1
Chuq	1
Tyrus	1
Karl	1
Godzilla	1
Vinnie	1

Bert	1
Striker	1
Pepper	1
Julius	1
Buddah	1
Arnold	1
Zuzu	1
Mollie	1
Laela	1
Tedders	1
Superpup	1
Billy	1
Jeb	1
Rodman	1
Ralph	1
Izzy	1
Jessifer	1
Crumpet	1
Griswold	1
Cheesy	1
Moofasa	1
Hector	1
Goliath	1
Kawhi	1
Emmie	1
Willie	1
Rinna	1
Mike	1
William	1
Dwight	1
Evy	1
Hurley	1
Rubio	1
Chompsky	1
Rascal	1
Tug	1
Tango	1
Grizz	1
Jerome	1
Jonah	1
Chesney	1
Kenny	1
Scott	1
Darby	1
Jackie	1
Jazz	1
Franq	1
Pippin	1
Rolf	1
Snickers	1
Ridley	1
Cal	1
Bradley	1

Bubba	1
Tuco	1
Patch	1
Mojo	1
Batdog	1
Dylan	1
Mark	1
JD	1
Alejandro	1
Scruffers	1
Pip	1
Taz	1
Caryl	1
Henry	1
Norman	1
Bobbay	1
Mitch	1
Kaiya	1
Acro	1
Aiden	1
Obie	1
Dot	1
Shnuggles	1
Kendall	1
Jeffri	1
Steve	1
Eve	1
Mac	1
Fletcher	1
Kenzie	1
Pumpkin	1
Schnozz	1
Gustaf	1
Cheryl	1
Ed	1
Leonidas	1
Walker	1
Bell	1
Karma	1
Pablo	1
Duchess	1
Harlso	1
Sundance	1
Flash	1
Howie	1
Jazzy	1
Anna	1
Wafer	1
Tom	1
Florence	1
Autumn	1
Dido	1
Eugene	1

Ken	1
Strudel	1
Tebow	1
Chloe	1
Timber	1
Binky	1
Crawford	1
Sweet	1
Comet	1
Mo	1
Emanuel	1
Kuyu	1
Dutch	1
Pete	1
Lilly	1
Astrid	1
Mutt	1
Doobert	1
Beebop	1
Alexander	1
Sailer	1
Brutus	1
Kona	1
Boots	1
Ralphie	1
Cupid	1
Pawnd	1
Pilot	1
Ike	1
Dudley	1
Akumi	1
Smiley	1
Ronnie	1
Mauve	1
Chef	1
Sobe	1
Longfellow	1
Iroh	1
Pancake	1
Snicku	1
Mack	1
Nimbus	1
Laika	1
Dobby	1
Moreton	1
Maude	1
Newt	1
Nida	1
Robin	1
Monster	1
BeBe	1
Remus	1
Bones	1

Severus	1
Alf	1
Sonny	1
Chubbs	1
Sky	1
Eleanor	1
Baron	1
Bauer	1
Swagger	1
Brandi	1
Mary	1
Halo	1
Augie	1
Craig	1
Pavlov	1
Kyro	1
Ito	1
Seamus	1
Stephan	1
Major	1
Shooter	1
Diogi	1
Miguel	1
Eevee	1
Vince	1
Ginger	1
Goose	1
Nugget	1
Jed	1
Sierra	1
Monkey	1
Harry	1
Kody	1
Lassie	1
Rover	1
Napolean	1
Rumble	1
Clifford	1
Dewey	1
Shikha	1
Lili	1
Jamesy	1
Meatball	1
Quinn	1
Zooey	1
Venti	1
Dante	1
Burt	1
Aja	1
Darla	1
Bruno	1
Stuart	1
Jim	1

Ralphus	1
Canela	1
Maya	1
Mingus	1
Roscoe	1
Jimbo	1
Maisey	1
Noah	1
Grizzwald	1
Alfy	1
Koko	1
Rey	1
Duddles	1
Snoopy	1
Emmy	1
Jersey	1
Jordy	1
Orion	1
Barney	1
Jarvis	1
Mimosa	1
Brady	1
Margo	1
Tycho	1
Dawn	1
Monty	1
Sojourner	1
Arlo	1
Mookie	1
Meera	1
Tobi	1
Sunshine	1
Lipton	1
Bronte	1
Poppy	1
Gidget	1
Rhino	1
Willow	1
Vixen	1
Tuck	1
Milky	1
Furzey	1
Thor	1
Callie	1
Cermet	1
Marlee	1
Arya	1
Einstein	1
Rumpole	1
Benny	1
Jarod	1
Wiggles	1
General	1

Sailor	1
Iggy	1
Snoop	1
Noosh	1
Odin	1
Georgie	1
Rontu	1
Cannon	1
Levi	1
Mabel	1
Betty	1
Rorie	1
Bayley	1
Storkson	1
Remy	1
Chadrick	1
Kellogg	1
Buckley	1
Livvie	1
Hermione	1
Ralpher	1
Aldrick	1
Rooney	1
Ziva	1
Stefan	1
Pupcasso	1
Puff	1
Flurpson	1
Coleman	1
Enchilada	1
Rueben	1
Simba	1
Nollie	1
Karll	1
Chase	1
Tripp	1
Cora	1
Huxley	1
Bookstore	1
Linus	1
Shiloh	1
Gustav	1
Arlen	1
Lenox	1
Harvey	1
Blanket	1
Geno	1
Stark	1
Beya	1
Kayla	1
Tilly	1
Edmund	1
Aqua	1

```
Baloo              1
Cilantro           1
Sprout             1
Mosby              1
Bluebert           1
Zeus               1
Bertson            1
Nico               1
Michelangelope     1
Siba               1
Travis             1
Kanu               1
Lance              1
Opie               1
Stubert            1
Kane               1
Staniel            1
Sora               1
Beemo              1
Gunner             1
Lacy               1
Tater              1
Olaf               1
Cecil              1
Stephanus          1
Jennifur           1
Bloop              1
Link               1
Lillie             1
Ashleigh           1
Luther             1
Ivar               1
Jangle             1
Schnitzel          1
Berkeley           1
Ralphé             1
Charleson          1
Clyde              1
Harnold            1
Sid                1
Carper             1
Bowie              1
Alexanderson       1
Suki               1
Barclay            1
Skittle            1
Ebby               1
Divine             1
Dietrich           1
Blu                1
Timmy              1
Hercules           1
Jay                1
```

Mya	1
Strider	1
Wesley	1
Huck	1
O	1
Blue	1
Anakin	1
Sprinkles	1
Heinrich	1
Shakespeare	1
Chelsea	1
Bungalo	1
Grey	1
Willem	1
Davey	1
Fizz	1
Dixie	1
Combo	1
Deacon	1
Jameson	1
Cali	1
Happy	1
Ralphy	1
Brownie	1
Rizzy	1
Stella	1
Butter	1
Tonks	1
Logan	1
Dale	1
Rizzo	1
Mattie	1
Pinot	1
Dallas	1
Hero	1
Stormy	1
Balto	1
Mairi	1
Loomis	1
Godi	1
Al	1
Carbon	1
Klein	1
DonDon	1
Spencer	1
Lilli	1
Boston	1
Brandonald	1
Corey	1
Leonard	1
Beckham	1
Devón	1
Gert	1

```
Dex                 1
Ace                 1
Tayzie              1
Grizzie             1
Gilbert             1
Meyer               1
Zoe                 1
Stewie              1
Lilah               1
Spanky              1
Emma                1
Fido                1
Theo                1
Tove                1
Chevy               1
Tito                1
Philbert            1
Rupert              1
Brudge              1
Shadoe              1
Angel               1
Brat                1
Gromit              1
Rose                1
Aubie               1
Kota                1
Leela               1
Glenn               1
Shelby              1
Sephie              1
Bonaparte           1
Wishes              1
Christoper          1
Name: count, dtype: int64
```

Issue #4:

> In df_archive_clean and df_image_clean, the 'tweet_id' column needs to be converted to 'string' format.

Define

> Convert column 'tweet_id' in dataframes df_archive_clean and df_image_clean to 'string' data type.

Code

In [257…
```
# Convert 'tweet_id' data type to 'object'.
df_archive_clean['tweet_id'] = df_archive_clean['tweet_id'].astype(object)

df_image_clean['tweet_id'] = df_image_clean['tweet_id'].astype(object)
```

Test

In [258…
```
df_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2097 entries, 0 to 2355
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   tweet_id            2097 non-null   object
 1   timestamp           2097 non-null   object
 2   text                2097 non-null   object
 3   rating_numerator    2097 non-null   int64 
 4   rating_denominator  2097 non-null   int64 
 5   name                1494 non-null   object
 6   doggo               83 non-null     object
 7   floofer             10 non-null     object
 8   pupper              230 non-null    object
 9   puppo               24 non-null     object
dtypes: int64(2), object(8)
memory usage: 180.2+ KB
```

In [259…
```
df_image_clean.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   object
 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(1), object(5)
memory usage: 152.1+ KB
```

Issue #5:

> The 'timestamp' column has five unnecessary characters ('+0000') after each value. The column is not of the 'datetime' data type.

Define

> We need to remove the final five characters ('+0000') from each value in the 'timestamp column. Then we will convert the column to 'datetime' data type.

Code

In [260…
```
# Strip the final five characters (+0000) from the timestamp values
df_archive_clean['timestamp'] = df_archive_clean['timestamp'].str[:-5]
```

In [261…
```
# Convert 'timestamp' to datetime data type
df_archive_clean['timestamp'] = pd.to_datetime(df_archive_clean['timestamp']
```

Test

In [262…
```
df_archive_clean.head()
```

Out[262]:

	tweet_id	timestamp	text	rating_numerator	rating_denominator	r
0	892420643555336193	2017-08-01 16:23:56	This is Phineas. He's a mystical boy. Only eve...	13	10	Ph
1	892177421306343426	2017-08-01 00:17:27	This is Tilly. She's just checking pup on you....	13	10	
2	891815181378084864	2017-07-31 00:18:03	This is Archie. He is a rare Norwegian Pouncin...	12	10	A
3	891689557279858688	2017-07-30 15:58:51	This is Darla. She commenced a snooze mid meal...	13	10	
4	891327558926688256	2017-07-29 16:00:24	This is Franklin. He would like you to stop ca...	12	10	Fra

In [263…

```
df_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2097 entries, 0 to 2355
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2097 non-null   object
 1   timestamp           2097 non-null   datetime64[ns]
 2   text                2097 non-null   object
 3   rating_numerator    2097 non-null   int64
 4   rating_denominator  2097 non-null   int64
 5   name                1494 non-null   object
 6   doggo               83 non-null     object
 7   floofer             10 non-null     object
 8   pupper              230 non-null    object
 9   puppo               24 non-null     object
dtypes: datetime64[ns](1), int64(2), object(7)
memory usage: 180.2+ KB
```

Issue #6:

> Some rows in df_image_clean have data where the prediction algorithm determined that the photo in the row does not contain a dog.

Define

> We need to remove all rows where 'p1', 'p2', and 'p3' are all 'False', which means the photo associated with the row is not an image of a dog.

Code

In [264…
```
# Create a condition to filter df_image_clean.
not_a_dog = (df_image_clean['p1_dog'] == False) & \
            (df_image_clean['p2_dog'] == False) & \
            (df_image_clean['p3_dog'] == False)

# Drop rows where the condition is True
df_image_clean = df_image_clean[~not_a_dog]
```

Test

In [265…
```
# Test programmatically if this worked.
len(df_image_clean[not_a_dog])
```

```
/var/folders/bf/x15pjvv529b_8tj_93vxbw780000gn/T/ipykernel_1562/2902436842.p
y:2: UserWarning: Boolean Series key will be reindexed to match DataFrame in
dex.
  len(df_image_clean[not_a_dog])
```

Out[265]:
```
0
```

In [266…
```
# Test visually to see if this worked
df_image_clean.tail(10)
```

Out[266]:

	tweet_id	jpg_url	img_num	
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	C
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	

Issue #7:

> We need to only use the dog breed that the prediction algorithm chose with the highest confidence level.

Define

> We will create a column called 'dog_breed' and populate it with values from the highest confidence level.

Code

In [267…

```
# create a function to select the dog breed with the highest confidence leve
# and place that value in a new column called dog breed.
def best_guess_breed(row):
    if row['p1_dog']:
        return row['p1']
    elif row['p2_dog']:
        return row['p2']
    elif row['p3_dog']:
        return row['p3']
    else:
        return None

# Adding the selected dog breed to the 'dog_breed' column.
df_image_clean['dog_breed'] = df_image_clean.apply(best_guess_breed, axis=1)
```

Test

In [268…
```
# Visually compare this to the 'test' code used in issue #6.
df_image_clean.tail(10)
```

Out[268]:

	tweet_id	jpg_url	img_num	
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	1	
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	1	
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	1	
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	2	
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	1	
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	1	C
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	

Issue #8:

There are now unnecessary columns in the df_image_clean dataframe.

Define

> We will drop the columns we no longer need for our analysis from df_image_clean.

Code

In [269…
```
# Create a list of columns to drop
columns_to_drop = ['img_num', 'p1', 'p1_conf',
                   'p1_dog', 'p2', 'p2_conf', 'p2_dog',
                   'p3', 'p3_conf', 'p3_dog']

# Drop the columns contained in the list
df_image_clean.drop(columns=columns_to_drop, axis=1, inplace=True)
```

Test

In [270…
```
# Visually test
df_image_clean.tail(10)
```

Out[270]:

	tweet_id	jpg_url	
2064	890006608113172480	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	
2065	890240255349198849	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	
2066	890609185150312448	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	
2067	890729181411237888	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	
2068	890971913173991426	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	
2069	891087950875897856	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	Chesapeake_
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	Labra
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	

Issue #9.

> There are duplicates in the 'jpg_url' column, indicating that they are possibly retweets.

Define

> We will drop these dupliceate values from the dataframe.

Code

In [271…
```
# Remove the duplicates in 'jpg_url'
df_image_clean.drop_duplicates(subset='jpg_url', inplace=True)
```

Test

In [272…
```
# Check to see if there are any duplicates
df_image_clean.jpg_url.duplicated().sum()
```

Out[272]: 0

Issue #10:

> The different dog breeds are not displayed in a consistent format (some are capitalized, some are not).

Define

> We want to display the dog breeds in a standard format, replaceing spaces or hyphens ('-') with underscores ('_'), and have all letters be lowercase.

Code

In [273…
```
# Replace all uppercase letters with lowercase letter
df_image_clean['dog_breed'] = df_image_clean['dog_breed'].str.lower()
```

Test

In [274…
```
df_image_clean.dog_breed.value_counts()
```

Out[274]:
```
dog_breed
golden_retriever                158
labrador_retriever              108
pembroke                         95
chihuahua                        91
pug                              63
toy_poodle                       51
chow                             48
samoyed                          42
pomeranian                       42
malamute                         33
french_bulldog                   31
chesapeake_bay_retriever         31
cocker_spaniel                   30
miniature_pinscher               25
eskimo_dog                       22
cardigan                         21
german_shepherd                  21
staffordshire_bullterrier        21
beagle                           20
shih-tzu                         20
siberian_husky                   20
maltese_dog                      19
kuvasz                           19
shetland_sheepdog                19
rottweiler                       19
lakeland_terrier                 18
italian_greyhound                17
```

```
basset                            17
west_highland_white_terrier       16
american_staffordshire_terrier    16
old_english_sheepdog              15
great_pyrenees                    15
soft-coated_wheaten_terrier       15
vizsla                            14
pekinese                          14
kelpie                            13
schipperke                        13
border_collie                     12
dalmatian                         12
airedale                          12
boston_bull                       12
blenheim_spaniel                  11
standard_poodle                   11
boxer                             11
norwegian_elkhound                11
great_dane                        11
borzoi                            11
collie                            11
whippet                           11
malinois                          11
bernese_mountain_dog              11
yorkshire_terrier                 10
english_springer                  10
doberman                           9
basenji                            9
english_setter                     8
german_short-haired_pointer        8
miniature_poodle                   8
brittany_spaniel                   8
flat-coated_retriever              8
border_terrier                     7
dandie_dinmont                     7
bloodhound                         7
norfolk_terrier                    7
saint_bernard                      7
mexican_hairless                   7
newfoundland                       7
papillon                           7
bedlington_terrier                 6
redbone                            6
irish_terrier                      6
bull_mastiff                       5
ibizan_hound                       5
norwich_terrier                    5
irish_setter                       5
walker_hound                       5
lhasa                              5
miniature_schnauzer                5
tibetan_mastiff                    4
weimaraner                         4
```

```
bluetick                           4
saluki                             4
welsh_springer_spaniel             4
giant_schnauzer                    4
tibetan_terrier                    4
rhodesian_ridgeback                4
scottish_deerhound                 4
gordon_setter                      4
keeshond                           4
greater_swiss_mountain_dog         3
cairn                              3
briard                             3
toy_terrier                        3
curly-coated_retriever             3
leonberg                           3
komondor                           3
afghan_hound                       3
brabancon_griffon                  3
irish_water_spaniel                3
wire-haired_fox_terrier            2
sussex_spaniel                     2
black-and-tan_coonhound            2
australian_terrier                 2
appenzeller                        2
groenendael                        2
silky_terrier                      1
clumber                            1
irish_wolfhound                    1
scotch_terrier                     1
entlebucher                        1
japanese_spaniel                   1
standard_schnauzer                 1
bouvier_des_flandres               1
Name: count, dtype: int64
```

Issue #11:

> The 'id' column of 'df_json_clean' needs to be renamed 'tweet_id'.

Define

> We will rename the column 'tweet_id' so the dataframes can be joined using this column.

Code

In [275…

```
# Rename the 'id' column 'tweet_id'
df_json_clean.rename(columns={'id': 'tweet_id'}, inplace=True)
```

Test

In [276…
```
df_json_clean.head()
```

Out[276]:

	tweet_id	retweet_count	favorite_count
0	892420643555336193	8853	39467
1	892177421306343426	6514	33819
2	891815181378084864	4328	25461
3	891689557279858688	8964	42908
4	891327558926688256	9774	41048

Issue #12:

> The 'tweet_id' column of df_json_clean needs to be converted to 'object' data type.

Define

> Convert 'tweet_id' from 'int64' to 'object' data type.

Code

In [277…
```
# Convert 'tweet_id' data type to 'object' data type.
df_json_clean['tweet_id'] = df_json_clean['tweet_id'].astype(object)
```

Test

In [278…
```
df_json_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   tweet_id        2354 non-null   object
 1   retweet_count   2354 non-null   int64 
 2   favorite_count  2354 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 55.3+ KB
```

Tidiness Issues

Issue #1:

>

> The four columns 'doggo', 'floofer', 'puppo' and 'pupper' need to be combined into one column called 'dog_stage', which will hold categorical data about what stage of life the dog is in. This is not scientific whatsoever:) A 'floofer' and 'doggo' would be older than a 'pupper' or 'puppo'.

Define

> We will use 'melt()' to combine the columns 'doggo', 'floofer', 'puppo', and 'pupper' into one column. Then we will drop the old columns.

Code

In [279…
```
# Combine these columns :
columns_to_merge = ['doggo', 'floofer', 'pupper', 'puppo']

# Create a df called 'df_melted' that holds the data.
df_melted = pd.melt(df_archive_clean, value_vars=columns_to_merge, value_nam

df_melted.drop('variable', axis=1, inplace=True)

# View info summary
df_melted.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 347 entries, 9 to 7158
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   dog_stage  347 non-null    object
dtypes: object(1)
memory usage: 5.4+ KB
```

In [280…
```
df_melted.value_counts()
```

Out[280]:
```
dog_stage
pupper       230
doggo         83
puppo         24
floofer       10
Name: count, dtype: int64
```

In [281…
```
# use 'concat' to merge the two dfs.
df_archive_clean = pd.concat([df_archive_clean, df_melted['dog_stage']], axi
```

In [282…
```
df_archive_clean.head()
```

Out[282]:

	tweet_id	timestamp	text	rating_numerator	rating_denominator	r
0	892420643555336193	2017-08-01 16:23:56	This is Phineas. He's a mystical boy. Only eve...	13.0	10.0	Ph
1	892177421306343426	2017-08-01 00:17:27	This is Tilly. She's just checking pup on you....	13.0	10.0	
2	891815181378084864	2017-07-31 00:18:03	This is Archie. He is a rare Norwegian Pouncin...	12.0	10.0	A
3	891689557279858688	2017-07-30 15:58:51	This is Darla. She commenced a snooze mid meal...	13.0	10.0	
4	891327558926688256	2017-07-29 16:00:24	This is Franklin. He would like you to stop ca...	12.0	10.0	Fra

In [283…
```
df_archive_clean['dog_stage'].value_counts()
```

Out[283]:
```
dog_stage
pupper      230
doggo        83
puppo        24
floofer      10
Name: count, dtype: int64
```

In [284…
```
# Drop the merged columns
columns_to_drop = ['doggo', 'floofer', 'pupper', 'puppo']

df_archive_clean.drop(columns=columns_to_drop, axis=1, inplace=True)
```

In [287…
```
df_archive_clean.head(2)
```

Out[287]:

	tweet_id	timestamp	text	rating_numerator	rating_denominator	nam
0	892420643555336193	2017-08-01 16:23:56	This is Phineas. He's a mystical boy. Only eve...	13.0	10.0	Phine
1	892177421306343426	2017-08-01 00:17:27	This is Tilly. She's just checking pup on you....	13.0	10.0	Til

Issue #2:

> We now have three cleaned dataframes that need to be joined together to create a master dataframe from which we will perform our analysis.

Define

> Join the three data sets on the 'tweet_id' column to make the master dataframe.

Code

In [288…
```
# Combine df_archive_clean and df_image_clean, call new df 'df_merged'.
df_merged = pd.merge(df_archive_clean, df_image_clean, on='tweet_id', how='i
```

In [291…
```
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1666 entries, 0 to 1665
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1666 non-null   object
 1   timestamp           1666 non-null   datetime64[ns]
 2   text                1666 non-null   object
 3   rating_numerator    1666 non-null   float64
 4   rating_denominator  1666 non-null   float64
 5   name                1266 non-null   object
 6   dog_stage           50 non-null     object
 7   jpg_url             1666 non-null   object
 8   dog_breed           1666 non-null   object
dtypes: datetime64[ns](1), float64(2), object(6)
memory usage: 117.3+ KB
```

In [292…
```
df_merged.head(2)
```

Out[292]:

	tweet_id	timestamp	text	rating_numerator	rating_denominator	nam
0	892177421306343426	2017-08-01 00:17:27	This is Tilly. She's just checking pup on you....	13.0	10.0	Til
1	891815181378084864	2017-07-31 00:18:03	This is Archie. He is a rare Norwegian Pouncin...	12.0	10.0	Archi

In [293…
```
df_merged.shape
```

Out[293]:
```
(1666, 9)
```

In [294…
```
df_merged.isnull().sum()
```

Out[294]:
```
tweet_id                0
timestamp               0
text                    0
rating_numerator        0
rating_denominator      0
name                  400
dog_stage            1616
jpg_url                 0
dog_breed               0
dtype: int64
```

In [295…
```
df_json_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2354 non-null   object
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
dtypes: int64(2), object(1)
memory usage: 55.3+ KB
```

In [296…
```
#Combine df_merged with df_json_clean.
df_merged = pd.merge(df_merged, df_json_clean, on='tweet_id', how='inner')
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1666 entries, 0 to 1665
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1666 non-null   object
 1   timestamp           1666 non-null   datetime64[ns]
 2   text                1666 non-null   object
 3   rating_numerator    1666 non-null   float64
 4   rating_denominator  1666 non-null   float64
 5   name                1266 non-null   object
 6   dog_stage           50 non-null     object
 7   jpg_url             1666 non-null   object
 8   dog_breed           1666 non-null   object
 9   retweet_count       1666 non-null   int64
 10  favorite_count      1666 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(2), object(6)
memory usage: 143.3+ KB
```

In [297…
```
df_merged.shape
```

Out[297]:
```
(1666, 11)
```

In [300…
```
df_merged.head(2)
```

Out[300]:

	tweet_id	timestamp	text	rating_numerator	rating_denominator	nam
0	892177421306343426	2017-08-01 00:17:27	This is Tilly. She's just checking pup on you....	13.0	10.0	Til
1	891815181378084864	2017-07-31 00:18:03	This is Archie. He is a rare Norwegian Pouncin...	12.0	10.0	Archi

Storing Data

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

In [298…
```
# Save df_merged as 'twitter_archive_master.csv'
df_merged.to_csv('twitter_archive_master.csv', index = False)
```

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.** >

> All analysis and visualization will be performed using df_merged, our cleaned, master dataframe.

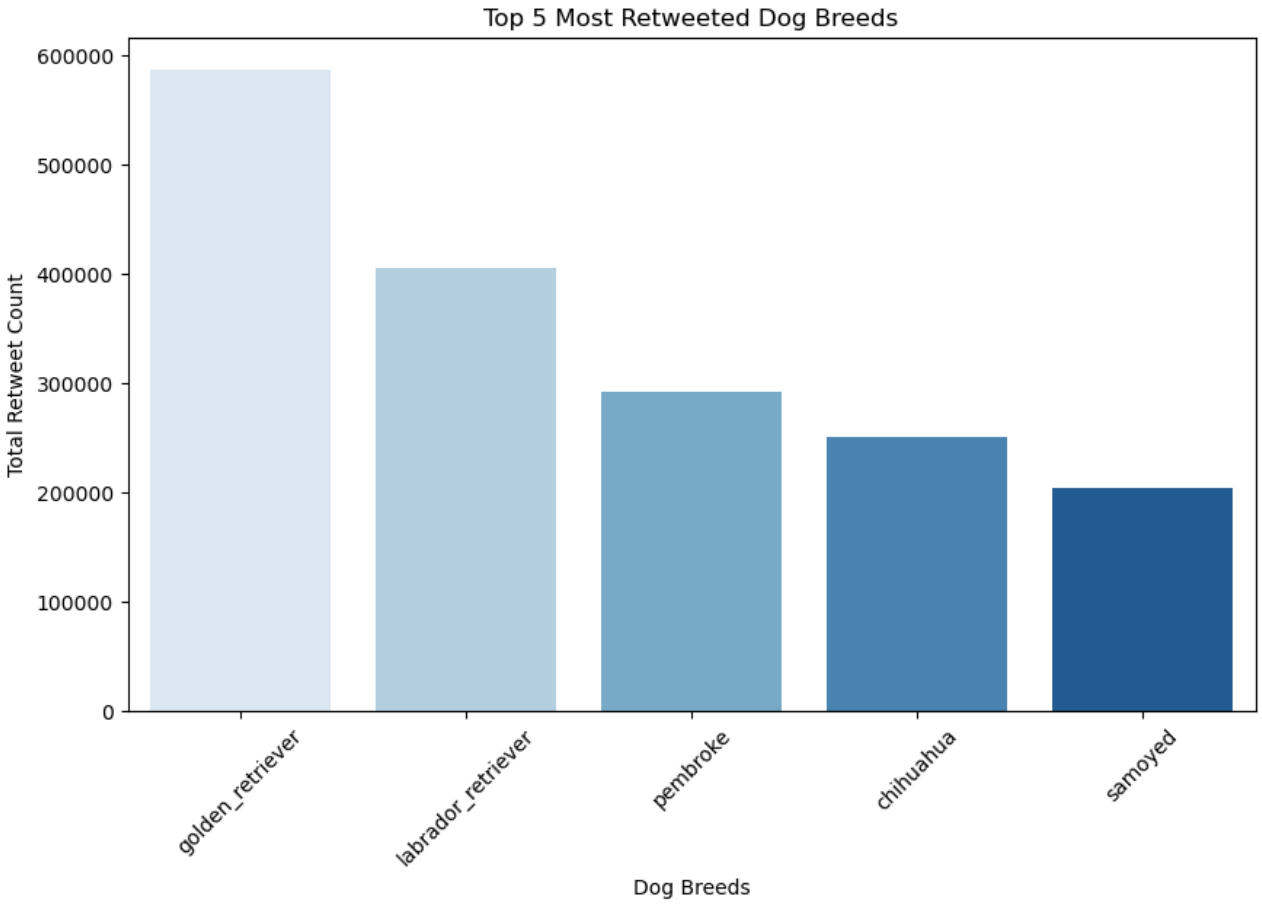
> What are the top 10 dog breeds to appear in our dataframe?

In [343…
```
# view 10 dog breeds with the most value counts
df_merged['dog_breed'].value_counts().sort_values(ascending=False).head(10)
```

Out[343]:
```
dog_breed
golden_retriever      156
labrador_retriever    106
pembroke               94
chihuahua              90
pug                    62
toy_poodle             50
chow                   48
samoyed                42
pomeranian             41
malamute               33
Name: count, dtype: int64
```

> Let's now make a barplot to view the top 5 most retweeted dog breeds.

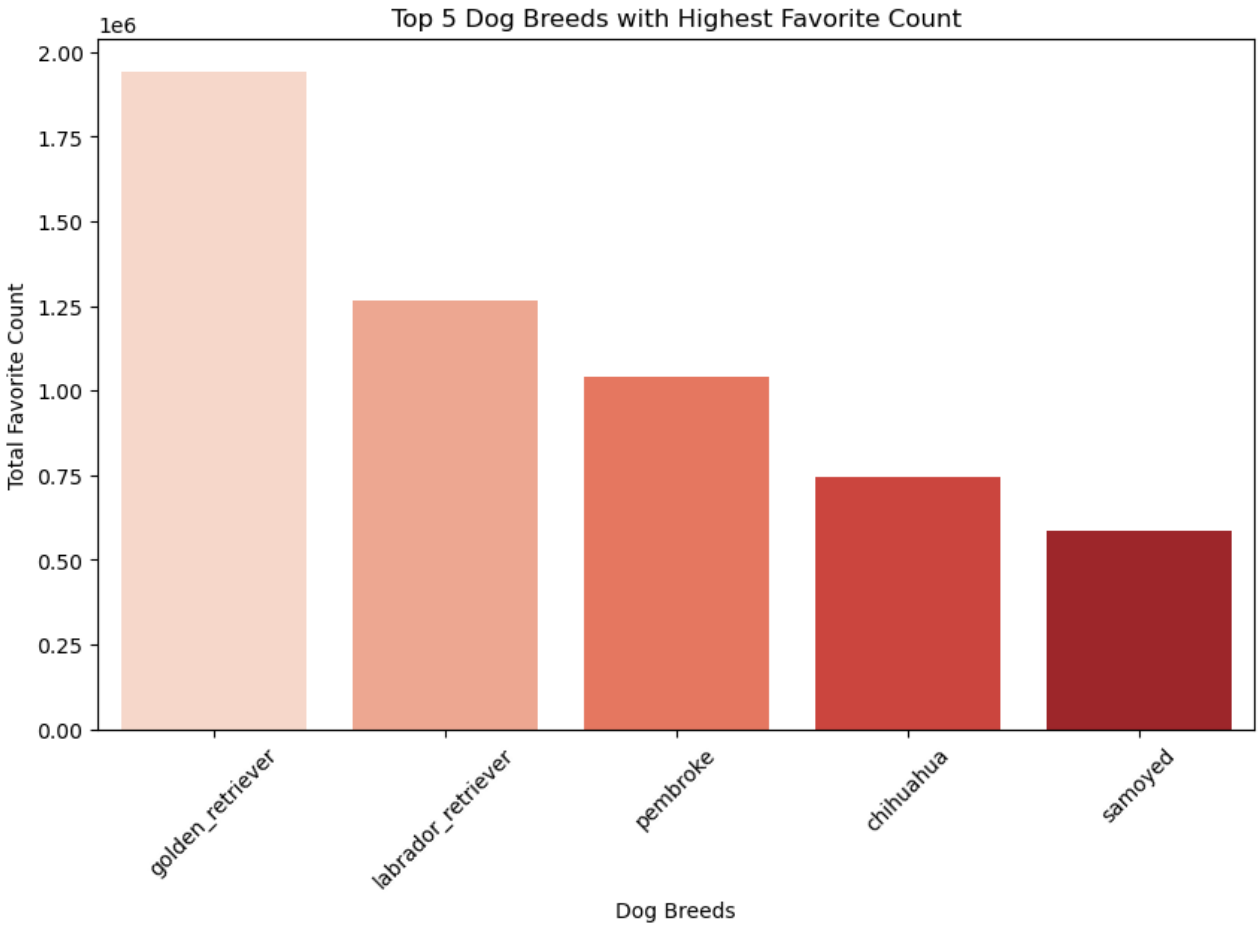
In [344…
```
# Group by dog breeds and sum the retweet counts
retweet_counts = df_merged.groupby('dog_breed')['retweet_count'].sum()

# Sort the retweet counts in descending order and select the top 5
top5_retweeted = retweet_counts.sort_values(ascending=False).head(5)

# Plotting the top 5 most retweeted dog breeds using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=top5_retweeted.index, y=top5_retweeted.values, palette='Blues'
plt.xlabel('Dog Breeds')
plt.ylabel('Total Retweet Count')
plt.title('Top 5 Most Retweeted Dog Breeds')
plt.xticks(rotation=45)
# Save the plot as an image file
plt.savefig('5_most_retweeted.png')
plt.show()
```



> And now, how does it compare to the top five dog breeds with the most favorite counts?

In [346…

```
# Group by dog breeds and sum the favorite counts
favorite_counts = df_merged.groupby('dog_breed')['favorite_count'].sum()

# Sort the favorite counts in descending order and select the top 5
top5_favorite = favorite_counts.sort_values(ascending=False).head(5)

# Plotting the top 5 dog breeds with highest favorite count using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=top5_favorite.index, y=top5_favorite.values, palette='Reds')
plt.xlabel('Dog Breeds')
plt.ylabel('Total Favorite Count')
plt.title('Top 5 Dog Breeds with Highest Favorite Count')
plt.xticks(rotation=45)
# Save the plot as an image file
plt.savefig('5_highest_favorite_count.png')
plt.show()
```



Insight #1:

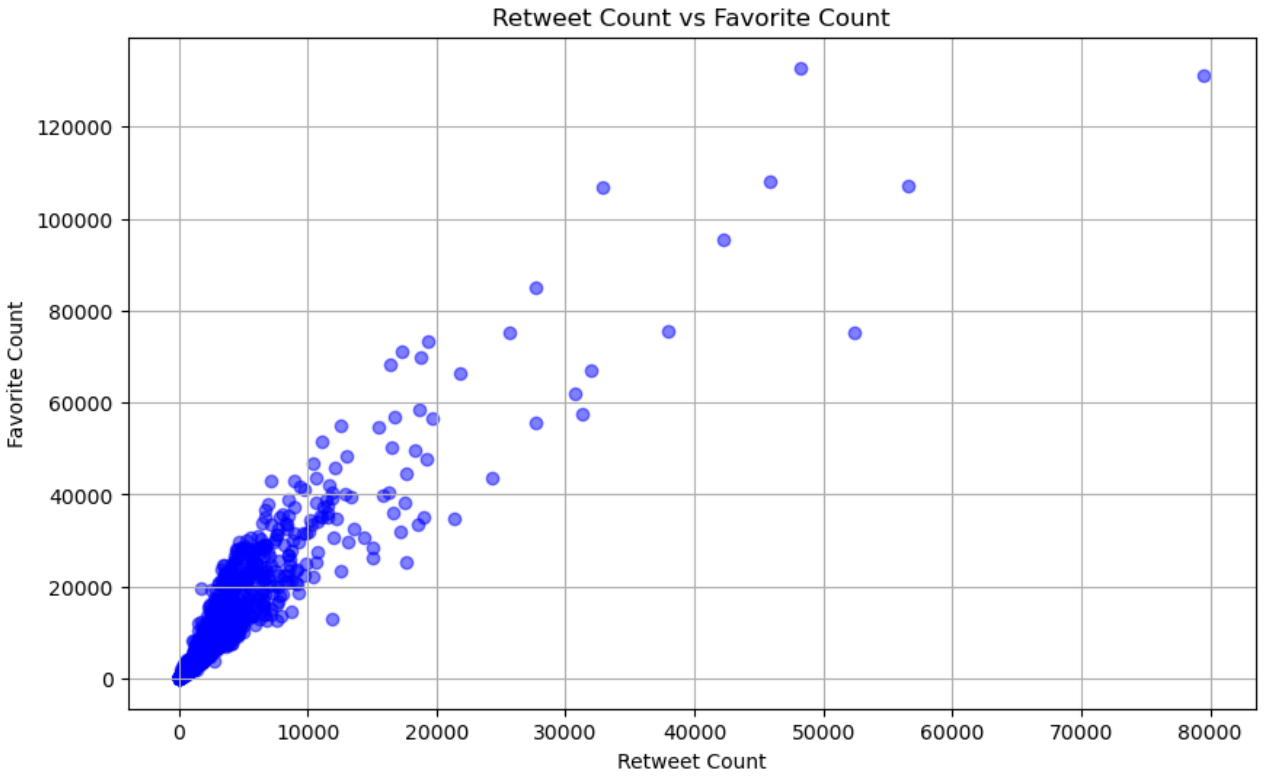
>

> The top five breeds with the most retweets and favorite counts are the same.
>
> The top four breeds are also the top four most common breeds to appear in our dataframe.
>
> However, the breed with the fifth most retweets and favorite counts, the samoyed, is actually the eighth most common breed to appear in out dataframe.

> The barplots for retweet counts and favorite counts look very similar.
>
> Below we will make a scatterplot to see if there is a correlation between retweet counts and favorite counts.

In [345…

```
# Scatter plot for 'retweet_count' vs 'favorite_count'
plt.figure(figsize=(10, 6))
plt.scatter(df_merged['retweet_count'], df_merged['favorite_count'], alpha=0
plt.xlabel('Retweet Count')
plt.ylabel('Favorite Count')
plt.title('Retweet Count vs Favorite Count')
plt.grid(True)
# Save the plot as an image file
plt.savefig('retweet_vs_favorite_scatterplot.png')
plt.show()
```



Insight #2:

>

> It appears that there is indeed a correlation between retweet counts and favorite counts.

> Continuing with what we discovered with 'Insight #1', lets explore which dog breeds have the highest ratio for total engagement:
>
> (retweets + favorites) / appearances

In [342…

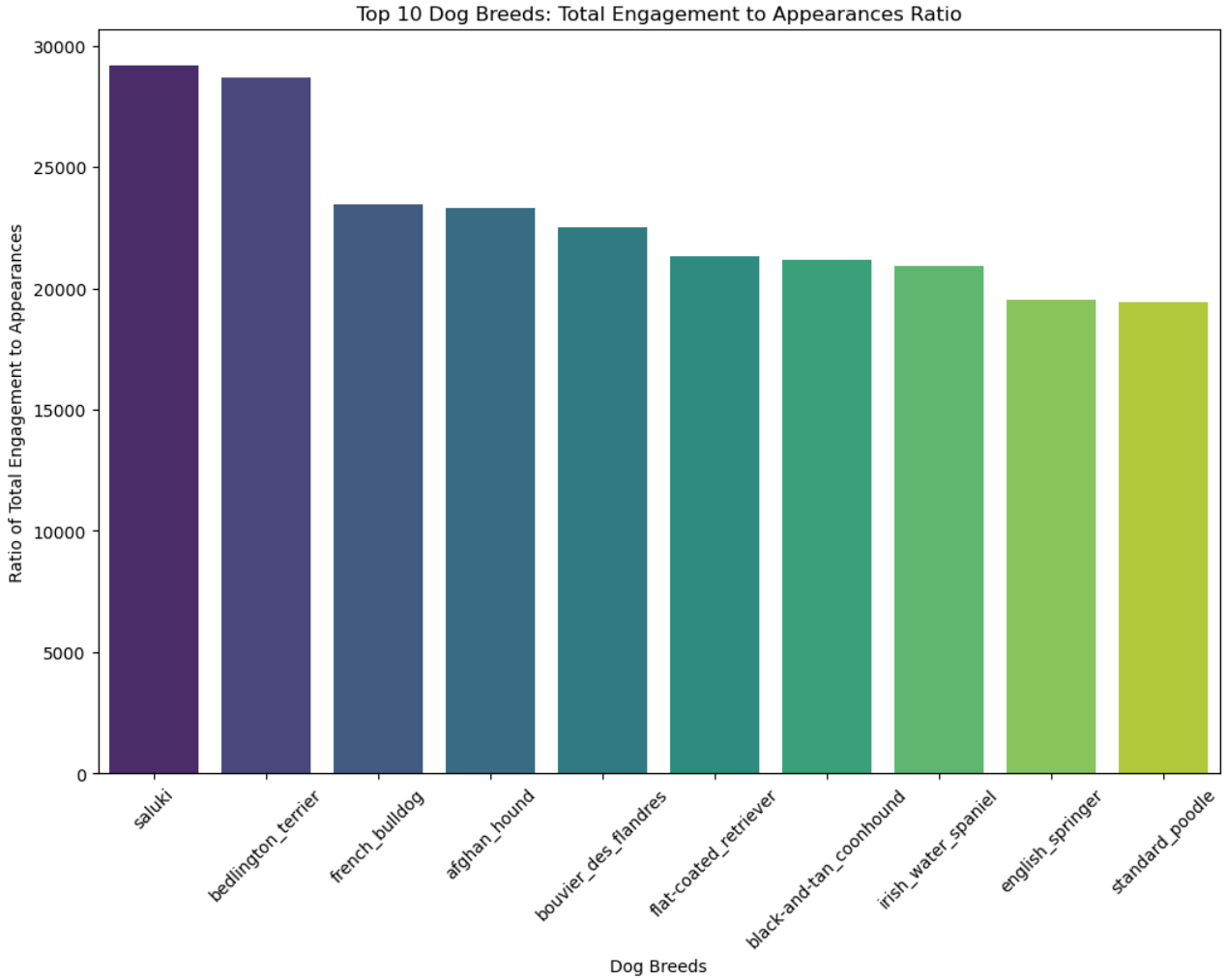
```
# Group by dog breeds and calculate the sum of favorite counts, sum of retwe
breed_counts = df_merged.groupby('dog_breed').agg({
    'favorite_count': 'sum',
    'retweet_count': 'sum',
    'dog_breed': 'count'
})

# Rename the count column to 'appearance_count'
breed_counts = breed_counts.rename(columns={'dog_breed': 'appearance_count'}

# Calculate the total engagement (favorite_count + retweet_count) for each b
breed_counts['total_engagement'] = breed_counts['favorite_count'] + breed_co

# Calculate the ratio (total_engagement / appearance_count) for each breed
breed_counts['engagement_ratio'] = breed_counts['total_engagement'] / breed_

# Sort the breeds by the ratio in descending order
breed_counts_sorted = breed_counts.sort_values(by='engagement_ratio', ascend

# Display the top 20 breeds based on the ratio of total engagement to appear
breed_counts_sorted.head(20)
```

Out[342]:

dog_breed	favorite_count	retweet_count	appearance_count	total_engagement	e
saluki	96240	20535	4	116775	
bedlington_terrier	126919	45061	6	171980	
french_bulldog	570921	156230	31	727151	
afghan_hound	51980	17928	3	69908	
bouvier_des_flandres	18032	4479	1	22511	
flat-coated_retriever	134333	36162	8	170495	
black-and-tan_coonhound	34024	8329	2	42353	
irish_water_spaniel	49200	13502	3	62702	
english_springer	141383	54016	10	195399	
standard_poodle	153041	60598	11	213639	
leonberg	44803	11591	3	56394	
samoyed	583906	203446	42	787352	
cocker_spaniel	407412	145743	30	553155	
mexican_hairless	95134	29784	7	124918	
whippet	142026	53243	11	195269	
great_pyrenees	186847	68008	15	254855	
border_terrier	95046	23640	7	118686	
cardigan	269644	79765	21	349409	
tibetan_mastiff	48694	16772	4	65466	
golden_retriever	1943782	586657	156	2530439	

In [347…

```
# Plotting the top 10 breeds with the highest ratio of total engagement to a
plt.figure(figsize=(12, 8))
sns.barplot(x=breed_counts_sorted.head(10).index, y=breed_counts_sorted.head
plt.xlabel('Dog Breeds')
plt.ylabel('Ratio of Total Engagement to Appearances')
plt.title('Top 10 Dog Breeds: Total Engagement to Appearances Ratio')
plt.xticks(rotation=45)
# Save the plot as an image file
plt.savefig('top_10_engagement_ratio.png')
plt.show()
```


Top 10 Dog Breeds: Total Engagement to Appearances Ratio
Ratio of Total Engagement to Appearances
30000
25000
20000
15000
10000
5000
0
saluki
bedlington_terrier
french_bulldog
afghan_hound
bouvier_des_flandres
flat-coated_retriever
black-and-tan_coonhound
irish_water_spaniel
english_springer
standard_poodle
Dog Breeds

Insight #3:

>

> We see a striking difference between this visualization and the barplots of breeds with the most retweets and favorite counts.
>
> We can see that golden retrievers are the most common dog breed that appears in the dataframe, and also holds the most retweet counts and favorite counts. We can contrast this with the saluki, which only appears four times but has almost double the engagement ratio. In fact, none of the dogs with the top 10 highest engagement ratios appear in our charts of breeds with the most favorites or retweets.
>
> The data shows that there are far less common dogs that have a much higher engagement ratio than the top five dog breeds that have the highest retweet and favorite counts.

In [349…
```
df_merged.describe()
```

Out[349]:

	timestamp	rating_numerator	rating_denominator	retweet_count	favorite_c
count	1666	1666.000000	1666.000000	1666.000000	1666.00(
mean	2016-06-05 17:09:11.599639808	11.388355	10.468788	2834.429772	9288.18
min	2015-11-15 22:32:08	0.000000	2.000000	16.000000	81.00(
25%	2015-12-17 18:06:03.249999872	10.000000	10.000000	650.500000	2176.00(
50%	2016-03-27 19:50:21.500000	11.000000	10.000000	1440.500000	4464.00(
75%	2016-10-31 20:45:18.500000	12.000000	10.000000	3261.000000	11734.75(
max	2017-08-01 00:17:27	165.000000	150.000000	79515.000000	132810.00(
std	NaN	7.489081	6.343940	4831.960790	12641.538

In [ ]:
```
```