# Wrangling and Analyze Data Project

## Table of Contents



## Data Gathering

### 3 different data gathering methods are executed.

In [493]:

```
# Import Libraries for functionality and features.
import pandas as pd
import numpy as np
import tweepy
import seaborn as sns
from tweepy import OAuthHandler
import json
import re
from timeit import default_timer as timer
import requests
import os
import warnings
import matplotlib.pyplot as plt
%matplotlib inline

# Prevents future warning messages.
warnings.simplefilter(action = 'ignore', category = FutureWarning)
```

In [494]:
```
# Creates a dataframe, reads data from the
# file in, and displays the first 2 lines.
df_twit  = pd.read_csv('twitter_archive_enhanced.csv')

df_twit.head(2)
```

Out[494]:

| | tweet_id | in_reply_to_status_id | in_reply_to_user_id | timestamp | |
|---|---|---|---|---|---|
| **0** | 892420643555336193 | NaN | NaN | 2017-08-01 16:23:56 +0000 | href="http://twit |
| **1** | 892177421306343426 | NaN | NaN | 2017-08-01 00:17:27 +0000 | href="http://twit |

In [495]:
```
# Verify the number of rows and columns.
df_twit.shape
```

Out[495]: (2356, 17)

In [496]:
```
# Displays the datatypes and null values within the dataframe.
df_twit.info()
```

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

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

In [497]:
```
# Creates a folder and then Downloads the image
# predictions file using the requests.get() method.
img_fldr = 'img_prdctns'

if not os.path.exists(img_fldr):
    os.makedirs(img_fldr)
    url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2

# Stores the file in the response variable.
response = requests.get(url)
```

In [498]:
```
# Displays the request has succeeded code (200)
# for storing file in response variable.
response
```

Out[498]:
```
<Response [200]>
```

In [499]:
```
# Displays the raw data content.
response.content
```

Out[499]:
```
b"tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_dog
\tp3\tp3_conf\tp3_dog\n666020888022790149\thttps://pbs.twimg.com/medi
a/CT4udn0WwAA0aMy.jpg\t1\tWelsh_springer_spaniel\t0.465074\tTrue\tcoll
ie\t0.156665\tTrue\tShetland_sheepdog\t0.0614285\tTrue\n66602928500262
0928\thttps://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg\t1\tredbone\t0.5
06826\tTrue\tminiature_pinscher\t0.07419169999999999\tTrue\tRhodesian_
ridgeback\t0.07201\tTrue\n666033412701032449\thttps://pbs.twimg.com/me
dia/CT4521TWwAEvMyu.jpg\t1\tGerman_shepherd\t0.596461\tTrue\tmalinois
\t0.13858399999999998\tTrue\tbloodhound\t0.11619700000000001\tTrue\n66
6044226329800704\thttps://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg\t1\t
Rhodesian_ridgeback\t0.408143\tTrue\tredbone\t0.360687\tTrue\tminiatur
e_pinscher\t0.222752\tTrue\n666049248165822465\thttps://pbs.twimg.com/
media/CT5IQmsXIAAKY4A.jpg\t1\tminiature_pinscher\t0.560311\tTrue\tRott
weiler\t0.243682\tTrue\tDoberman\t0.154629\tTrue\n666050758794694657\t
https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg\t1\tBernese_mountain_d
og\t0.651137\tTrue\tEnglish_springer\t0.263788\tTrue\tGreater_Swiss_Mo
untain_dog\t0.0161992\tTrue\n666051853826850816\thttps://pbs.twimg.co
m/media/CT5KoJ1WoAAJash.jpg\t1\tbox_turtle\t0.9330120000000001\tFalse
\tmud_turtle\t0.04588540000000001\tFalse\tterrapin\t0.0178852999999999
```

In [500]:
```
# Writes the raw data content to the file.
with open(os.path.join(img_fldr, url.split('/')[-1]), mode = 'wb') as file
    file.write(response.content)
```

In [501]:
```
# Creates a dataframe and reads the data in from the file and displays the
img_predictns = pd.read_csv('img_prdctns\image-predictions.tsv', sep = '\t

img_predictns.head(3)
```

Out[501]:

| | tweet_id | jpg_url | img_num | |
|---|---|---|---|---|
| 0 | 666020888022790149 | https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg | 1 | Welsh_s |
| 1 | 666029285002620928 | https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg | 1 | |
| 2 | 666033412701032449 | https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg | 1 | Ger |

In [502]:
```
# Verify the number of rows and columns.
img_predictns.shape
```

Out[502]:
```
(2075, 12)
```

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

In [503]:

```
# Query Twitter API for each tweet in the Twitter archive and save JSON in
# 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 STUDENT WITH MOBILE VERIFICATION ISSUES:
# df_1 is a DataFrame with the twitter_archive_enhanced.csv file. You may
# change line 17 to match the name of your DataFrame with twitter_archive_
# NOTE TO REVIEWER: this student had mobile verification issues so the fol
# 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_twit.tweet_id.values
# len(tweet_ids)

# Query Twitter's API for JSON data for each tweet ID in the Twitter archi
# 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
    # 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)
```

In [504]:
```
# Creates a list and reads text in from the file.
in_list = []

with open('tweet_json.txt') as file:
    for line in file:
        data = json.loads(line)

        # Creates elements in the list for columns data.
        twt_id = data.get('id_str')
        retwt_count = data.get('retweet_count')
        fav_count = data.get('favorite_count')

        # Appends the columns data elements to the list.
        in_list.append({'tweet_id': twt_id,
        'retwt_count': retwt_count,
        'fav_count': fav_count})
```

In [505]:
```
# Creates dataframe from the lists and displays.
twts_cnts = pd.DataFrame(in_list, columns = ['tweet_id', 'retwt_count', 'f
twts_cnts.head(3)
```

Out[505]:

| | tweet_id | retwt_count | fav_count |
|---|---|---|---|
| **0** | 892420643555336193 | 8853 | 39467 |
| **1** | 892177421306343426 | 6514 | 33819 |
| **2** | 891815181378084864 | 4328 | 25461 |

In [506]:
```
# Displays the length of the dataframe.
len(twts_cnts)
```

Out[506]: 2354

# Assessing Data

### 9 quality issues are defined.

# Quality Issues

### 1. Several incorrect datatypes for columns in df_twit2, img_predictns2, and twts_cnts2.

### - Change datatypes of columns that would be more appropriate for

**cleaning and analysis.**

**2. Many incorrect or mislabeled dog names in df_twit2.**

**- Remove incorrect names.**

**3. The 'floofer' stage name is incorrect in df_twit2.**

**- Change the column name.**

**4. Only original ratings should be in df_twit2.**

**- Remove non-original ratings.**

**5. Inconsistent Numerator and Denominators in the rating system in df_twit2.**

**- Remove incorrect/outlier values.**

**6. There are non dog entries in img_predictns2.**

**- remove non-dog entries.**

**7. HTML tags are in source text in df_twit2.**

**- Remove the HTML tags.**

**8. Dog breeds in 'p1', 'p2', & 'p3' columns in img_predictns2 dataframe have inconsistent capitalization.**

In [507]:

```
# Make copies of original dataframes.
df_twit2 = df_twit.copy()
img_predictns2 = img_predictns.copy()
twts_cnts2 = twts_cnts.copy()
```

## Cleaning Data

**9 quality issues are defined, resolved, and tested for verification.**

**----------**

### Issue #1:

### Define: Several incorrect datatypes for columns in df_twit2, img_predictns2, & twts_cnts2.

### Solution: Change datatypes of columns that would be more appropriate for cleaning and analysis.

### Code

In [508]:
```
# Changes the datatype of the timestamp columns.
df_twit2[['timestamp',
      'retweeted_status_timestamp']] = df_twit2[['timestamp',
      'retweeted_status_timestamp']].apply(pd.to_datetime)
```

In [509]:
```
# Changes and displays the datatypes of several columns.
df_twit2 = df_twit2.astype({'tweet_id': 'object',
                            'rating_numerator': 'object',
                            'rating_denominator': 'object',
                            'in_reply_to_status_id': 'object',
                            'in_reply_to_user_id': 'object',
                            'retweeted_status_id': 'object',
                            'retweeted_status_user_id': 'object'}).copy()

df_twit2.dtypes
```

Out[509]:
```
tweet_id                                     object
in_reply_to_status_id                        object
in_reply_to_user_id                          object
timestamp                      datetime64[ns, UTC]
source                                       object
text                                         object
retweeted_status_id                          object
retweeted_status_user_id                     object
retweeted_status_timestamp     datetime64[ns, UTC]
expanded_urls                                object
rating_numerator                             object
rating_denominator                           object
name                                         object
doggo                                        object
floofer                                      object
pupper                                       object
puppo                                        object
dtype: object
```

In [510]:
```
# Changes and displays the datatypes of 2 columns.
img_predictns2 = img_predictns2.astype({'tweet_id': 'object',
                                        'img_num': 'object'}).copy()

img_predictns2.dtypes
```

Out[510]:
```
tweet_id     object
jpg_url      object
img_num      object
p1           object
p1_conf     float64
p1_dog         bool
p2           object
p2_conf     float64
p2_dog         bool
p3           object
p3_conf     float64
p3_dog         bool
dtype: object
```

In [511]:
```
# Changes and displays the datatypes of several columns.
twts_cnts2 = twts_cnts2.astype({'retwt_count': 'object',
                                'fav_count': 'object'}).copy()

twts_cnts2.dtypes
```

Out[511]:
```
tweet_id       object
retwt_count    object
fav_count      object
dtype: object
```

### Issue #2:

### Define: Many incorrect or mislabeled dog names in df_twit2.

### Solution: Remove incorrect names.

### Code

In [512]:
```
# Creates a series of unique names from the name column,
# sorts values and displays the first 100 names.
df_name = df_twit2.name.sort_values().unique().copy()

df_name[:100]
```

Out[512]:
```
array(['Abby', 'Ace', 'Acro', 'Adele', 'Aiden', 'Aja', 'Akumi', 'Al',
       'Albert', 'Albus', 'Aldrick', 'Alejandro', 'Alexander',
       'Alexanderson', 'Alf', 'Alfie', 'Alfy', 'Alice', 'Amber',
       'Ambrose', 'Amy', 'Amélie', 'Anakin', 'Andru', 'Andy', 'Angel',
       'Anna', 'Anthony', 'Antony', 'Apollo', 'Aqua', 'Archie', 'Arlen',
       'Arlo', 'Arnie', 'Arnold', 'Arya', 'Ash', 'Asher', 'Ashleigh',
       'Aspen', 'Astrid', 'Atlas', 'Atticus', 'Aubie', 'Augie', 'Autumn',
       'Ava', 'Axel', 'Bailey', 'Baloo', 'Balto', 'Banditt', 'Banjo',
       'Barclay', 'Barney', 'Baron', 'Barry', 'Batdog', 'Bauer', 'Baxte
r',
       'Bayley', 'BeBe', 'Bear', 'Beau', 'Beckham', 'Beebop', 'Beemo',
       'Bell', 'Bella', 'Belle', 'Ben', 'Benedict', 'Benji', 'Benny',
       'Bentley', 'Berb', 'Berkeley', 'Bernie', 'Bert', 'Bertson',
       'Betty', 'Beya', 'Biden', 'Bilbo', 'Billl', 'Billy', 'Binky',
       'Birf', 'Bisquick', 'Blakely', 'Blanket', 'Blipson', 'Blitz',
       'Bloo', 'Bloop', 'Blu', 'Blue', 'Bluebert', 'Bo'], dtype=object)
```

In [513]:
```
# Creates a series from a slice of the last 30 names and displays.
df_name2 = df_name[-30:].copy()

df_name2
```

Out[513]:
```
array(['Ziva', 'Zoe', 'Zoey', 'Zooey', 'Zuzu', 'a', 'actually', 'all',
       'an', 'by', 'getting', 'his', 'incredibly', 'infuriating', 'just',
       'life', 'light', 'mad', 'my', 'not', 'officially', 'old', 'one',
       'quite', 'space', 'such', 'the', 'this', 'unacceptable', 'very'],
      dtype=object)
```

### At the end of the alphabetical list, there are 25 names that seem to be incorrect.

In [514]:

```
# Creates a dataframe with records of all the incorrect names
# and displays the number of entries for compoarison.
test_loc = df_twit2.loc[((df_twit2['name'] == 'a') |
                         (df_twit2['name'] == 'actually') |
                         (df_twit2['name'] == 'all') |
                         (df_twit2['name'] == 'an') |
                         (df_twit2['name'] == 'by') |
                         (df_twit2['name'] == 'getting') |
                         (df_twit2['name'] == 'his') |
                         (df_twit2['name'] == 'incredibly') |
                         (df_twit2['name'] == 'infuriating') |
                         (df_twit2['name'] == 'just') |
                         (df_twit2['name'] == 'life') |
                         (df_twit2['name'] == 'light') |
                         (df_twit2['name'] == 'mad') |
                         (df_twit2['name'] == 'my') |
                         (df_twit2['name'] == 'None') |
                         (df_twit2['name'] == 'not') |
                         (df_twit2['name'] == 'officially') |
                         (df_twit2['name'] == 'old') |
                         (df_twit2['name'] == 'one') |
                         (df_twit2['name'] == 'quite') |
                         (df_twit2['name'] == 'space') |
                         (df_twit2['name'] == 'such') |
                         (df_twit2['name'] == 'the') |
                         (df_twit2['name'] == 'this') |
                         (df_twit2['name'] == 'unacceptable') |
                         (df_twit2['name'] == 'very'))]

test_loc.name.count()
```

Out[514]: 854

In [515]:
```
# Removes all of the records with incorrect names.
df_twit2 = df_twit2.loc[~((df_twit2['name'] == 'a') |
                          (df_twit2['name'] == 'actually') |
                          (df_twit2['name'] == 'all') |
                          (df_twit2['name'] == 'an') |
                          (df_twit2['name'] == 'by') |
                          (df_twit2['name'] == 'getting') |
                          (df_twit2['name'] == 'his') |
                          (df_twit2['name'] == 'incredibly') |
                          (df_twit2['name'] == 'infuriating') |
                          (df_twit2['name'] == 'just') |
                          (df_twit2['name'] == 'life') |
                          (df_twit2['name'] == 'light') |
                          (df_twit2['name'] == 'mad') |
                          (df_twit2['name'] == 'my') |
                          (df_twit2['name'] == 'None') |
                          (df_twit2['name'] == 'not') |
                          (df_twit2['name'] == 'officially') |
                          (df_twit2['name'] == 'old') |
                          (df_twit2['name'] == 'one') |
                          (df_twit2['name'] == 'quite') |
                          (df_twit2['name'] == 'space') |
                          (df_twit2['name'] == 'such') |
                          (df_twit2['name'] == 'the') |
                          (df_twit2['name'] == 'this') |
                          (df_twit2['name'] == 'unacceptable') |
                          (df_twit2['name'] == 'very'))].copy()
```

## Test

In [516]:
```
# Boolean expression to compare the length of
# the dataframe after the removal of records.
test_loc.name.count() == len(df_twit) - len(df_twit2)
```

Out[516]: True

### Issue #3:

### Define: The 'floofer' stage name is incorrect (as stated in the dogtionary) in df_twit2.

### Solution: Change the column name to 'floof'.

### Code

In [517]:
```
# Rename the 'floofer' stage column 'floof'S.
df_twit2.rename(columns = {'floofer':'floof'}, inplace = True)
```

### Test

In [518]:
```
# Displays the first row of the dataframe
# to verify the column name has been changed.
df_twit2.head(1)
```

Out[518]:

| | tweet_id | in_reply_to_status_id | in_reply_to_user_id | timestamp | |
|---|---|---|---|---|---|
| **0** | 892420643555336193 | NaN | NaN | 2017-08-01 16:23:56+00:00 | href="http:/ |

### Issue #4:

### Define: Only original ratings should be in df_twit2 dataset.

### Solution: Remove all retweet ratings.

### Code

In [519]:
```
# Stores and displays length of dataframe for comparison.
df_t2_len = len(df_twit2)

df_t2_len
```

Out[519]: 1502

In [520]:
```
# Displays the number of retweeted entries for comparison.
rt_count = df_twit2.retweeted_status_id.count()

rt_count
```

Out[520]: 111

In [521]:
```
# Removes all records with a retweeted status.
df_twit2 = df_twit2.loc[~((df_twit2['retweeted_status_id'] > 0))].copy()
```

### Test

In [522]:
```
# Displays length of dataframe for comparison.
len(df_twit2)
```

Out[522]: 1391

In [523]:
```
# Boolean expression to compare the length of
# the dataframe after the removal of records.
df_t2_len == len(df_twit2) + rt_count
```

Out[523]: True

### Issue #5:

### Define: Inconsistent rating numerators and denominators in the rating system of df_twit2.

### Solution: Remove incorrect/outlier values.

### Code

In [524]:
```
# Displays the different values for rating_denominator.
df_twit2['rating_denominator'].value_counts()
```

Out[524]:
```
10    1388
7        1
50       1
11       1
Name: rating_denominator, dtype: int64
```

### There are 3 rating denominator outlier entries.

In [525]:
```
# Displays the different values for rating_denominator.
df_twit2['rating_numerator'].value_counts()
```

Out[525]:
```
12      340
11      314
10      278
13      183
9       101
8        71
7        33
14       17
6        17
5        14
3        10
4         5
2         3
75        1
24        1
1776      1
50        1
27        1
Name: rating_numerator, dtype: int64
```

### There are consistent numbers from 2 - 14 for the rating numerator, with many outliers.

In [526]:
```
# Removes incorrect/outlier values from the dataframe.
df_twit2 = df_twit2.loc[~((df_twit2['rating_numerator'] > 15) |
                (df_twit2['rating_denominator'] != 10))].copy()
```

### Test

In [527]:
```
# Displays the values for rating_numerator.
df_twit2['rating_numerator'].value_counts()
```

Out[527]:
```
12    340
11    314
10    278
13    183
9     101
8      71
7      32
14     17
6      17
5      14
3      10
4       5
2       3
Name: rating_numerator, dtype: int64
```

### Only rating numerator numbers 2 - 14 (range 1 - 15) are in the dataframe.

In [528]:
```
# Displays the value(s) for rating_numerator.
df_twit2['rating_denominator'].value_counts()
```

Out[528]:
```
10    1385
Name: rating_denominator, dtype: int64
```

### Only the rating denominator number 10 rating is in the datframe.

### Issue #6:

### Define: There are non dog entries in img_predictns2.

### Solution: Remove all non dog entries.

### Code

In [529]:
```
# Stores and displays the length
# of the datframe for comparison.
img_len = len(img_predictns2)

img_len
```

Out[529]:
```
2075
```

In [530]:
```
# Displays the last 5 records of the different values
# for the p1 column (Which are not dog entries).
p1 = img_predictns2['p1'].value_counts()

p1.tail(5)
```

Out[530]:
```
pillow        1
carousel      1
bald_eagle    1
lorikeet      1
orange        1
Name: p1, dtype: int64
```

In [531]:
```
# Count, store, and display the number of non-dog entries.
non_dog = img_predictns2.loc[((img_predictns2['p1_dog'] == False) |
                   (img_predictns2['p2_dog'] == False) |
                   (img_predictns2['p3_dog'] == False))].count().mean().as

non_dog
```

Out[531]:
```
832
```

In [532]:
```
#Remove all non-dog entries from the dataframe.
img_predictns2 = img_predictns2.loc[~((img_predictns2['p1_dog'] == False)
                   (img_predictns2['p2_dog'] == False) |
                   (img_predictns2['p3_dog'] == False))].copy()
```

### Test

In [533]:
```
# Stores and displays the length
# of the datframe for comparison.
img_pred2_len = len(img_predictns2)

img_pred2_len
```

Out[533]:
```
1243
```

In [534]:
```
# Boolean expression to compare the length of
# the dataframe after the removal of records.
img_len = img_pred2_len == img_pred2_len + non_dog

img_len

print('\nIt is', img_len, 'that all non-dog entries have been removed.\n')
```

```
It is False that all non-dog entries have been removed.
```

### Issue #7:

### Define: HTML tags are in source text in df_twit2.

### Solution: Remove all HTML tags.

### Code

In [535]:
```
# Displays the first entry for the source column.
df_twit2.source[1]
```

Out[535]:
```
'<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for
iPhone</a>'
```

In [536]:
```
# Displays the number of unique
# values for the source column.
df_twit2.source.nunique()
```

Out[536]:
```
4
```

In [537]:
```
# Removing the HTML tags from the
# source column using regular expressions.
df_twit2.source = df_twit2.source.apply(lambda x: re.sub('<[^<]+?>', '', x
```

### Test

In [538]:
```
# Displays the first record in the source
# column for comparison to verify changes.
df_twit2.source[1]
```

Out[538]:
```
'Twitter for iPhone'
```

### The HTML tags have been removed.

### Issue #8:

### Define: Dog breeds in 'p1', 'p2', & 'p3' columns in img_predictns2 dataframe have inconsistent capitalization.

### Solution: Change all uppercase letters to lowercase.

In [539]:
```
# Store and display the first entry
# of the p1 column for comparison.
cap_check = img_predictns2.p1.head(1).copy()

cap_check
```

Out[539]:
```
0    Welsh_springer_spaniel
Name: p1, dtype: object
```

### Code

In [540]:
```
# Changes all uppercase letters to lowercase in the p1, p2, & p3 columns.
img_predictns2['p1'] = img_predictns2['p1'].apply(str.lower).copy()
img_predictns2['p2'] = img_predictns2['p2'].apply(str.lower).copy()
img_predictns2['p3'] = img_predictns2['p3'].apply(str.lower).copy()
```

### Test

In [541]:
```
# Display the first entry of
# the p1 column for comparison.
img_predictns2.p1.head(1)
```

Out[541]:
```
0    welsh_springer_spaniel
Name: p1, dtype: object
```

## Tidiness issues

### 1. Merge df_twit2, img_predictns2, and twts_cnts2 dataframes.

### 2. Condense the 4 specific dog stage columns into 1.

### 3. Remove duplicate columns after joining/merging dataframes.

### 4. Remove unnecessary columns.

### Issue #1: Merge df_twit2, img_predictns2, and twts_cnts2 dataframes.

### Code

In [542]:
```
# Displays the rows and columns for comparison.
img_predictns2.shape
```

Out[542]:
```
(1243, 12)
```

In [543]:
```
# Displays the rows and columns for comparison.
df_twit2.shape
```

Out[543]:
```
(1385, 17)
```

In [544]:
```
twts_cnts2.shape
```

Out[544]:
```
(2354, 3)
```

In [545]:
```
twts_cnts2.tweet_id = twts_cnts2.tweet_id.astype('int64').copy()
```

In [546]:
```
# Merge the df_twit2 and img_predictns2 dataframes.
df_fin1 = pd.merge(df_twit2, img_predictns2, on = 'tweet_id', how = 'inner
```

In [547]:
```
# Merge the df_fin1 and twts_cnts2 dataframes.
df_final = pd.merge(df_fin1, twts_cnts2, on = 'tweet_id', how = 'outer').c
```

### Test

In [548]:
```
# Displays the rows and columns
# after merge for comparison.
df_final.shape
```

Out[548]:
```
(2354, 30)
```

In [549]:
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2354 entries, 0 to 2353
Data columns (total 30 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2354 non-null   object
 1   in_reply_to_status_id       0 non-null      object
 2   in_reply_to_user_id         0 non-null      object
 3   timestamp                   849 non-null    datetime64[ns, UTC]
 4   source                      849 non-null    object
 5   text                        849 non-null    object
 6   retweeted_status_id         0 non-null      object
 7   retweeted_status_user_id    0 non-null      object
 8   retweeted_status_timestamp  0 non-null      datetime64[ns, UTC]
 9   expanded_urls               849 non-null    object
 10  rating_numerator            849 non-null    object
 11  rating_denominator          849 non-null    object
 12  name                        849 non-null    object
 13  doggo                       849 non-null    object
 14  floof                       849 non-null    object
 15  pupper                      849 non-null    object
 16  puppo                       849 non-null    object
 17  jpg_url                     849 non-null    object
 18  img_num                     849 non-null    object
 19  p1                          849 non-null    object
 20  p1_conf                     849 non-null    float64
 21  p1_dog                      849 non-null    object
 22  p2                          849 non-null    object
 23  p2_conf                     849 non-null    float64
 24  p2_dog                      849 non-null    object
 25  p3                          849 non-null    object
 26  p3_conf                     849 non-null    float64
 27  p3_dog                      849 non-null    object
 28  retwt_count                 2354 non-null   object
 29  fav_count                   2354 non-null   object
dtypes: datetime64[ns, UTC](2), float64(3), object(25)
memory usage: 570.1+ KB
```

## Issue #2: Condense the 4 specific dog stage columns into 1.

### Code

In [550]:
```
# First replace None in stage columns with empty string as follows.
df_final.doggo.replace('None', '', inplace = True)
df_final.puppo.replace('None', '', inplace = True)
df_final.floof.replace('None', '', inplace = True)
df_final.pupper.replace('None', '', inplace = True)

# Concatenates the 4 columns into 1.
df_final['stage'] = df_final.doggo + df_final.floof + df_final.pupper + df

# Locates and changes double-entry grouped sytrings into 2 entries.
df_final.loc[df_final.stage == 'doggopupper', 'stage'] = 'doggo, pupper'
df_final.loc[df_final.stage == 'doggopuppo', 'stage'] = 'doggo, puppo'
df_final.loc[df_final.stage == 'doggofloof', 'stage'] = 'doggo, floof'
```

### Test

In [551]:
```
df_final.stage.value_counts()
```

Out[551]:
```
                  739
pupper             68
doggo              22
puppo              13
floofer             4
doggo, pupper       3
Name: stage, dtype: int64
```

In [552]:
```
df_final.stage[:10]
```

Out[552]:
```
0
1
2
3
4    doggo
5
6
7    puppo
8
9
Name: stage, dtype: object
```

### Issue #3: Remove duplicate columns after joining/merging dataframes.

### Code

In [553]:
```
# Stores and displays the number
# of columns for comparison.
df_shp_cols = df_final.shape[1]

df_shp_cols
```

Out[553]: 31

In [554]:
```
# Removes duplicate columns while using the transpose() method.
df_final = df_final.drop_duplicates().copy()
```

### Test

In [555]:
```
# Stores and displays the number of columns of
# the dataframe after dropping duplicate columns.
df_shp_cols2 = df_final.shape[1]

df_shp_cols2
```

Out[555]: 31

In [556]:
```
# Calculates and displays the number
# of duplicate columns removed.
df_shp_cols3 = df_shp_cols - df_shp_cols2

print('\n', df_shp_cols3, 'columns were removed from the dataframe.\n')
```

```
 0 columns were removed from the dataframe.
```

### 6 duplicate columns were dropped from the dataframe.

### Issue #4: Remove unnecessary columns.

In [557]:

```
# Displays the columns, null values,
# and datatypes of the dataframe.
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2354 entries, 0 to 2353
Data columns (total 31 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2354 non-null   object
 1   in_reply_to_status_id       0 non-null      object
 2   in_reply_to_user_id         0 non-null      object
 3   timestamp                   849 non-null    datetime64[ns, UTC]
 4   source                      849 non-null    object
 5   text                        849 non-null    object
 6   retweeted_status_id         0 non-null      object
 7   retweeted_status_user_id    0 non-null      object
 8   retweeted_status_timestamp  0 non-null      datetime64[ns, UTC]
 9   expanded_urls               849 non-null    object
 10  rating_numerator            849 non-null    object
 11  rating_denominator          849 non-null    object
 12  name                        849 non-null    object
 13  doggo                       849 non-null    object
 14  floof                       849 non-null    object
 15  pupper                      849 non-null    object
 16  puppo                       849 non-null    object
 17  jpg_url                     849 non-null    object
 18  img_num                     849 non-null    object
 19  p1                          849 non-null    object
 20  p1_conf                     849 non-null    float64
 21  p1_dog                      849 non-null    object
 22  p2                          849 non-null    object
 23  p2_conf                     849 non-null    float64
 24  p2_dog                      849 non-null    object
 25  p3                          849 non-null    object
 26  p3_conf                     849 non-null    float64
 27  p3_dog                      849 non-null    object
 28  retwt_count                 2354 non-null   object
 29  fav_count                   2354 non-null   object
 30  stage                       849 non-null    object
dtypes: datetime64[ns, UTC](2), float64(3), object(26)
memory usage: 588.5+ KB
```

### Code

In [558]:
```
# Stores and displays the number
# of columns for comparison.
df_f_cols = df_final.shape[1]

df_f_cols
```

Out[558]: 31

In [559]:
```
# Removes unnecessary columns.
df_final.drop(['in_reply_to_status_id', 'p1_dog'], axis = 1, inplace = Tru
```

In [560]:
```
# Stores and displays the number
# of columns for comparison.
df_f_cols2 = df_final.shape[1]

df_f_cols2
```

Out[560]: 29

In [561]:
```
# Displays the number of columns removed.
df_f_cols3 = df_f_cols - df_f_cols2

print('\n', df_f_cols3, 'unnecessary columns were removed from the datafra
```

```
 2 unnecessary columns were removed from the dataframe.
```

### 2 columns were removed from the dataframe.

## Storing Data

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

In [562]:
```
# Save the df_final dataframe to a csv file (twitter_archive_master.csv).
df_final.to_csv('twitter_archive_master.csv', index = False)
```

# Analyze Data

In [563]:

```
# Read the twitter_archive_master.csv file into the
# df_twit_mstr dataframe and displays the first row.
df_twit_mstr = pd.read_csv("twitter_archive_master.csv")

df_twit_mstr.head(1)
```

Out[563]:

| | tweet_id | in_reply_to_user_id | timestamp | source | text | retweeted_status_id |
|---|---|---|---|---|---|---|
| **0** | 8.921774e+17 | NaN | 2017-08-01 00:17:27+00:00 | Twitter for iPhone | This is Tilly. She's just checking pup on you.... | NaN |

1 rows × 29 columns

In [564]:

```
# Creates copies of the dataframe to analyze/visualize.
df_twit_mstr1 = df_twit_mstr
df_twit_mstr2 = df_twit_mstr
df_twit_mstr3 = df_twit_mstr
df_twit_mstr4 = df_twit_mstr
df_twit_mstr5 = df_twit_mstr
df_twit_mstr6 = df_twit_mstr
df_twit_mstr7 = df_twit_mstr
```

## General Analysis:

In [565]:

```
# Displays the columns, null values,
# and dtypes of the dataframe.
df_twit_mstr1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 29 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2354 non-null   float64
 1   in_reply_to_user_id         0 non-null      float64
 2   timestamp                   849 non-null    object 
 3   source                      849 non-null    object 
 4   text                        849 non-null    object 
 5   retweeted_status_id         0 non-null      float64
 6   retweeted_status_user_id    0 non-null      float64
 7   retweeted_status_timestamp  0 non-null      float64
 8   expanded_urls               849 non-null    object 
 9   rating_numerator            849 non-null    float64
 10  rating_denominator          849 non-null    float64
 11  name                        849 non-null    object 
 12  doggo                       25 non-null     object 
 13  floof                       4 non-null      object 
 14  pupper                      71 non-null     object 
 15  puppo                       13 non-null     object 
 16  jpg_url                     849 non-null    object 
 17  img_num                     849 non-null    float64
 18  p1                          849 non-null    object 
 19  p1_conf                     849 non-null    float64
 20  p2                          849 non-null    object 
 21  p2_conf                     849 non-null    float64
 22  p2_dog                      849 non-null    object 
 23  p3                          849 non-null    object 
 24  p3_conf                     849 non-null    float64
 25  p3_dog                      849 non-null    object 
 26  retwt_count                 2354 non-null   int64  
 27  fav_count                   2354 non-null   int64  
 28  stage                       110 non-null    object 
dtypes: float64(11), int64(2), object(16)
memory usage: 533.5+ KB
```

In [566]:
```
# Displays calculations of the values within the datframe.
df_twit_mstr.describe()
```

Out[566]:

| | tweet_id | in_reply_to_user_id | retweeted_status_id | retweeted_status_user_id | retv |
|---|---|---|---|---|---|
| **count** | 2.354000e+03 | 0.0 | 0.0 | 0.0 | |
| **mean** | 7.426978e+17 | NaN | NaN | NaN | |
| **std** | 6.852812e+16 | NaN | NaN | NaN | |
| **min** | 6.660209e+17 | NaN | NaN | NaN | |
| **25%** | 6.783975e+17 | NaN | NaN | NaN | |
| **50%** | 7.194596e+17 | NaN | NaN | NaN | |
| **75%** | 7.993058e+17 | NaN | NaN | NaN | |
| **max** | 8.924206e+17 | NaN | NaN | NaN | |

# Insights:

### 1. The Top 5 Most Common of the Highest Rated Dog Breeds.

### 2. The Top 10 Most Popular Dog Names.

### 3. The Top 10 Most Favorited Dog Breeds.

### 4. The Top 10 Most Retweeted Dog Breeds.

# Visualizations:

### Insight #1: The Top 5 Most Common of the Highest Rated Dog Breeds.

In [567]:
```
# Creates and displays the first row of a dataframe containing the necessa
df_final_merge1 = df_twit_mstr[['tweet_id', 'rating_numerator', 'p1', 'p2'

df_final_merge1.head(1)
```

Out[567]:

| | tweet_id | rating_numerator | p1 | p2 | p3 |
|---|---|---|---|---|---|
| **0** | 8.921774e+17 | 13.0 | chihuahua | pekinese | papillon |

In [568]:
```
# Displays the length of the dataframe.
len(df_final_merge1)
```

Out[568]:
```
2354
```

In [569]:
```
# A stacked list is created from the combined values of
# p1, p2, & p3 columns and displays the first 5 elements.
df_groups = df_final_merge1[['p1','p2', 'p3']].stack().value_counts().copy

df_groups.index[:5]
```

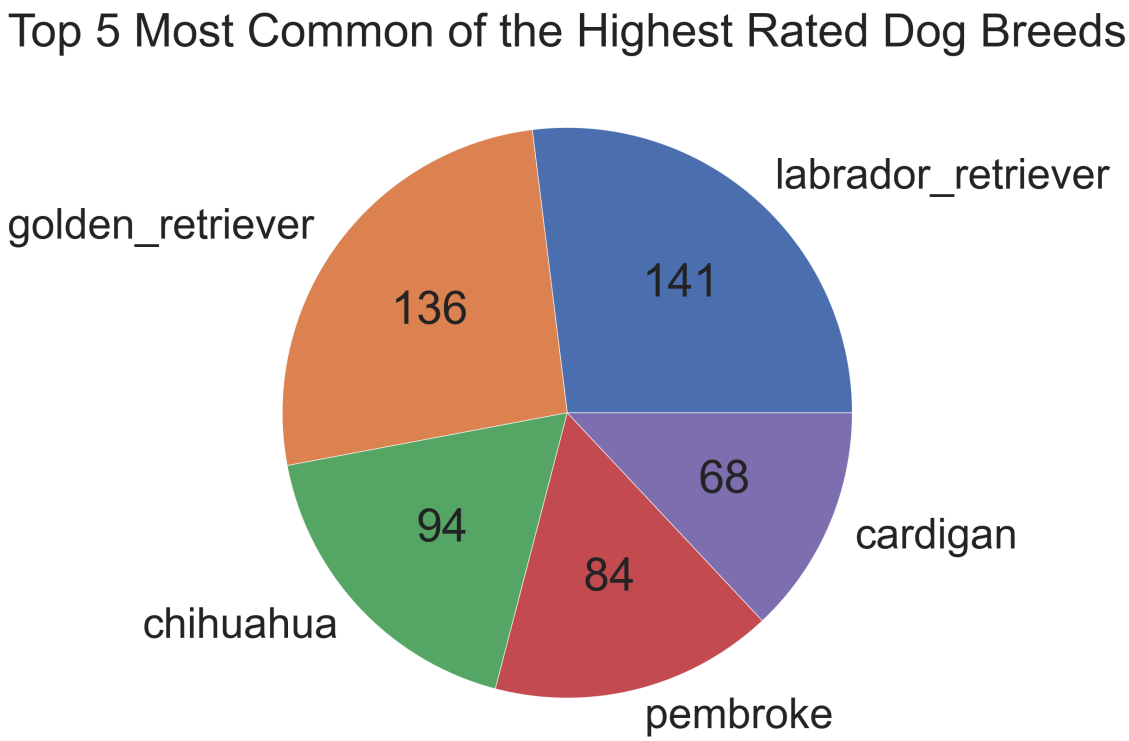
Out[569]:
```
Index(['labrador_retriever', 'golden_retriever', 'chihuahua', 'pembroke',
       'cardigan'],
      dtype='object')
```

In [570]:
```
# Creates and displays a list from
# a slice of the first 5 elements.
df_grp_cnt = df_groups[:5].copy()

df_grp_cnt
```

Out[570]:
```
labrador_retriever    141
golden_retriever      136
chihuahua              94
pembroke               84
cardigan               68
dtype: int64
```

In [571]:

```
# Function to calculate the values for display within the pie chart.
def abs_val(val):
    a  = np.round(val/100.*df_grp_cnt.sum(), 0).astype('int')
    return a

# Creates amd displays a pie chart.

az = sns.set(font_scale = 6)
az = plt.figure(figsize = (40, 20))
az = plt.pie(df_grp_cnt, labels = df_grp_cnt.index, autopct = abs_val)
az = plt.title('Top 5 Most Common of the Highest Rated Dog Breeds')

az = plt.savefig("Top 5 Most Common of the Highest Rated Dog Breeds.jpg",
```



## Insight #2: The Top 10 Most Popular Dog Names.

In [572]:
```
# Counts and sorts the values of the name column as well as stores and dis
df_twit_mstr2 = df_twit_mstr.name.value_counts().sort_values(ascending = F

df_twit_mstr2
```

Out[572]:

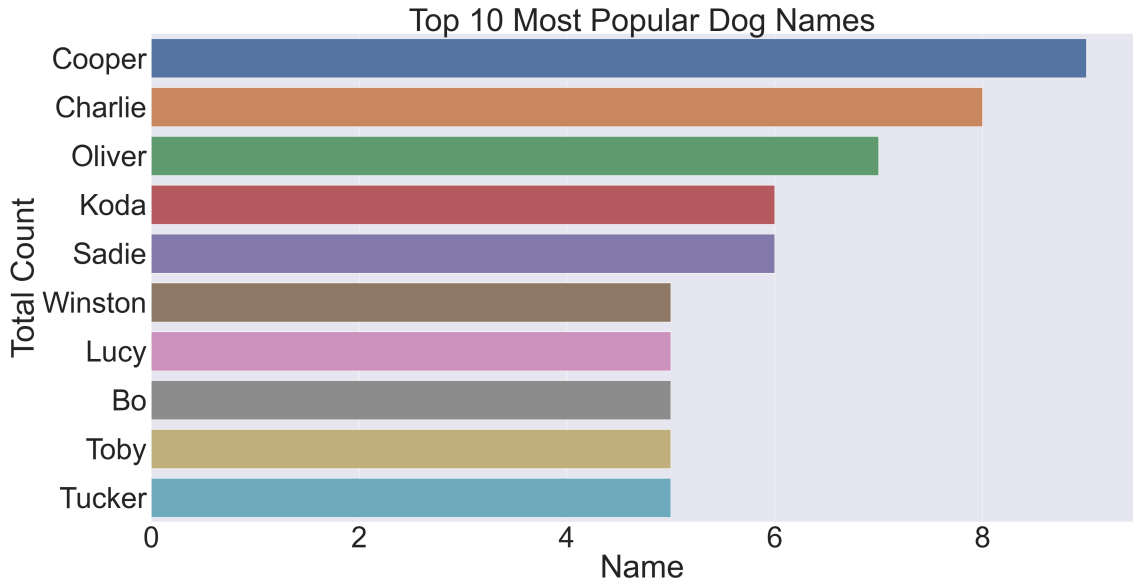
| | index | name |
|---|---|---|
| **0** | Cooper | 9 |
| **1** | Charlie | 8 |
| **2** | Oliver | 7 |
| **3** | Koda | 6 |
| **4** | Sadie | 6 |
| **5** | Winston | 5 |
| **6** | Lucy | 5 |
| **7** | Bo | 5 |
| **8** | Toby | 5 |
| **9** | Tucker | 5 |

In [573]:
```
# Renames the columns and displays.
df_twit_mstr2.rename(columns = {'index': 'Name','name':'Count'}, inplace =

df_twit_mstr2
```

Out[573]:

| | Name | Count |
|---|---|---|
| **0** | Cooper | 9 |
| **1** | Charlie | 8 |
| **2** | Oliver | 7 |
| **3** | Koda | 6 |
| **4** | Sadie | 6 |
| **5** | Winston | 5 |
| **6** | Lucy | 5 |
| **7** | Bo | 5 |
| **8** | Toby | 5 |
| **9** | Tucker | 5 |

In [574]:
```
# Creates and displays a bar graph.

ax = sns.set(font_scale = 6)

ax = sns.barplot(x = 'Count',
            y = 'Name',
            data = df_twit_mstr2,
            orient = 'h')

ax.set(xlabel = 'Name',
       ylabel = 'Total Count',
       title = 'Top 10 Most Popular Dog Names')

ax.figure.set_size_inches(40, 20)

ax = plt.savefig("Top 10 Most Popular Dog Names.jpg", bbox_inches = 'tight
```




## Insight #3: The Top 10 Most Favorited Dog Breeds.

In [575]:
```
# Changes the tweet_id datatype to integer (for merge), and displays.
twts_cnts2.tweet_id = twts_cnts2.tweet_id.astype('int64').copy()

twts_cnts2.dtypes
```

Out[575]:
```
tweet_id        int64
retwt_count    object
fav_count      object
dtype: object
```

In [576]:
```
# Creates dataframe from specific columns.
df_twit_mstr3 = img_predictns2[['tweet_id', 'p1', 'p2', 'p3']].copy()

#  Changes the tweet_id datatype to integer (for merge), and displays.
df_twit_mstr3.tweet_id = df_twit_mstr3.tweet_id.astype('int64').copy()

df_twit_mstr3.dtypes
```

Out[576]:
```
tweet_id     int64
p1          object
p2          object
p3          object
dtype: object
```

In [577]:
```
# Merge 2 dataframes on tweet_id, with only common tweet_ids.
df_twit_mstr4 = pd.merge(df_twit_mstr3, twts_cnts2, on = 'tweet_id', how =

# Removes the retwt_count column.
df_twit_mstr4 = df_twit_mstr4.drop('retwt_count', axis = 1).copy()

# Changes the dataype of the fav_count column to integer.
df_twit_mstr4.fav_count = df_twit_mstr4.fav_count.astype('int64').copy()

# Displays the first row to verify changes.
df_twit_mstr4.head(1)
```

Out[577]:

| | tweet_id | p1 | p2 | p3 | fav_count |
|---|---|---|---|---|---|
| **0** | 666020888022790149 | welsh_springer_spaniel | collie | shetland_sheepdog | 2535 |

## The retwt_count column has been dropped.

In [578]:

```
# Creates a copy of the dataframe with the index reset.
df_groups2 = df_groups.reset_index().copy()

# Rename the columns.
df_groups2.rename(columns = {'index': 'breed'}, inplace = True)

# Creates a series from the breed column.
df_groups3 = df_groups2.breed.copy()

df_groups3
```

Out[578]:

```
0      labrador_retriever
1        golden_retriever
2               chihuahua
3                pembroke
4                cardigan
              ...
111      sealyham_terrier
112    kerry_blue_terrier
113         affenpinscher
114        scotch_terrier
115       irish_wolfhound
Name: breed, Length: 116, dtype: object
```

In [579]:

```
# Creates an empty dataframe.
df_breeds = pd.DataFrame()

# Loop to locate and append records that
# match the breed in series comparison.
for favs in range(len(df_groups3)):
        df_breeds = df_breeds.append(df_twit_mstr4.loc[df_twit_mstr4['p1']
        df_breeds = df_breeds.append(df_twit_mstr4.loc[df_twit_mstr4['p2']
        df_breeds = df_breeds.append(df_twit_mstr4.loc[df_twit_mstr4['p3']

df_breeds.head(3)
```

Out[579]:

| | tweet_id | p1 | p2 | p3 | fav_count |
|---|---|---|---|---|---|
| **30** | 666701168228331520 | labrador_retriever | chihuahua | french_bulldog | 449 |
| **54** | 667453023279554560 | labrador_retriever | french_bulldog | staffordshire_bullterrier | 327 |
| **57** | 667502640335572993 | labrador_retriever | golden_retriever | beagle | 563 |

In [580]:

```
# Creates a copy of the series p1, is grouped by
# breeds, and calculates the sum of the fav_count.
df_breeds3 = df_breeds.groupby('p1').agg({'fav_count': ['sum']}).copy()

df_breeds3.head(5)
```

Out[580]:

| | fav_count |
|---|---|
| | sum |
| p1 | |
| afghan_hound | 73149 |
| airedale | 184914 |
| american_staffordshire_terrier | 294822 |
| appenzeller | 7782 |
| australian_terrier | 66765 |

In [581]:

```
# Creates a copy of the series p2, is grouped by
# breeds, and calculates the sum of the fav_count.
df_breeds4 = df_breeds.groupby('p2').agg({'fav_count': ['sum']}).copy()

df_breeds4.head(5)
```

Out[581]:

| | fav_count |
|---|---|
| | sum |
| p2 | |
| affenpinscher | 10872 |
| afghan_hound | 213888 |
| airedale | 86787 |
| american_staffordshire_terrier | 703599 |
| appenzeller | 102123 |

In [582]:

```
# Creates a copy of the series p3, is grouped by
# breeds, and calculates the sum of the fav_count.
df_breeds5 = df_breeds.groupby('p3').agg({'fav_count': ['sum']}).copy()

df_breeds5.head(5)
```

Out[582]:

| | fav_count |
|---|---|
| | sum |
| p3 | |
| affenpinscher | 15282 |
| afghan_hound | 11772 |
| airedale | 193794 |
| american_staffordshire_terrier | 371343 |
| appenzeller | 165186 |

In [583]:

```
# Concatenates 3 dataframes.
df_breeds6 = df_breeds3 + df_breeds4 + df_breeds5

# Sorts the values, sums the total of fav_count for each breed, unstacked,
df_breeds6 = df_breeds6.sort_values([('fav_count', 'sum')], ascending = Fa

df_breeds6.head(10)
```

Out[583]:

| | level_0 | level_1 | level_2 | 0 |
|---|---|---|---|---|
| 0 | fav_count | sum | labrador_retriever | 7436421.0 |
| 1 | fav_count | sum | golden_retriever | 6789165.0 |
| 2 | fav_count | sum | pembroke | 4074438.0 |
| 3 | fav_count | sum | cardigan | 3725439.0 |
| 4 | fav_count | sum | chihuahua | 3454161.0 |
| 5 | fav_count | sum | pomeranian | 2698848.0 |
| 6 | fav_count | sum | french_bulldog | 2085513.0 |
| 7 | fav_count | sum | chesapeake_bay_retriever | 2064642.0 |
| 8 | fav_count | sum | cocker_spaniel | 2001651.0 |
| 9 | fav_count | sum | eskimo_dog | 1946985.0 |

In [584]:

```
# Renames columns.
df_breeds6.rename(columns = {df_breeds6.columns[2]: 'breed', df_breeds6.co

# Removes level_0 and level_1 columns.
df_breeds6.drop(['level_0', 'level_1'], axis = 'columns', inplace = True)

df_breeds6.head(10)
```

Out[584]:

| | breed | fav_count |
|---|---|---|
| **0** | labrador_retriever | 7436421.0 |
| **1** | golden_retriever | 6789165.0 |
| **2** | pembroke | 4074438.0 |
| **3** | cardigan | 3725439.0 |
| **4** | chihuahua | 3454161.0 |
| **5** | pomeranian | 2698848.0 |
| **6** | french_bulldog | 2085513.0 |
| **7** | chesapeake_bay_retriever | 2064642.0 |
| **8** | cocker_spaniel | 2001651.0 |
| **9** | eskimo_dog | 1946985.0 |

In [585]:

```
# Creates a dataframe of the top 10 highest
# fav_count entries of dog breeds.
df_breeds_final = df_breeds6.head(10).copy()

df_breeds_final
```

Out[585]:

| | breed | fav_count |
|---|---|---|
| **0** | labrador_retriever | 7436421.0 |
| **1** | golden_retriever | 6789165.0 |
| **2** | pembroke | 4074438.0 |
| **3** | cardigan | 3725439.0 |
| **4** | chihuahua | 3454161.0 |
| **5** | pomeranian | 2698848.0 |
| **6** | french_bulldog | 2085513.0 |
| **7** | chesapeake_bay_retriever | 2064642.0 |
| **8** | cocker_spaniel | 2001651.0 |
| **9** | eskimo_dog | 1946985.0 |

In [586]:

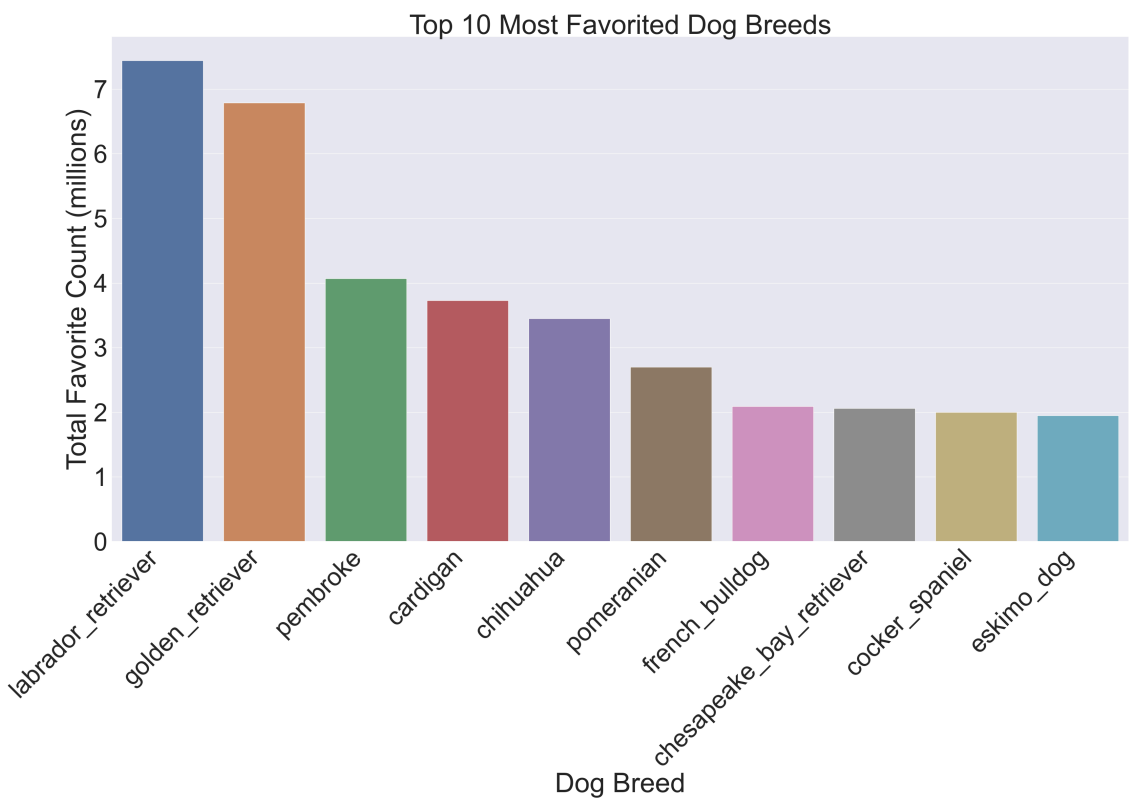
```
# Divide the fav_count column by millions for visualization.
df_breeds_final.fav_count = df_breeds_final.fav_count.div(10**6).round(2).

df_breeds_final
```

Out[586]:

| | breed | fav_count |
|---|---|---|
| **0** | labrador_retriever | 7.44 |
| **1** | golden_retriever | 6.79 |
| **2** | pembroke | 4.07 |
| **3** | cardigan | 3.73 |
| **4** | chihuahua | 3.45 |
| **5** | pomeranian | 2.70 |
| **6** | french_bulldog | 2.09 |
| **7** | chesapeake_bay_retriever | 2.06 |
| **8** | cocker_spaniel | 2.00 |
| **9** | eskimo_dog | 1.95 |

In [587]:

```
# Creates and displays a bar graph.
av = sns.set(font_scale = 5)

av = sns.barplot(x = 'breed',
            y = 'fav_count',
            data = df_breeds_final)

av.set(xlabel = 'Dog Breed',
       ylabel = 'Total Favorite Count (millions)',
       title = 'Top 10 Most Favorited Dog Breeds')

av.figure.set_size_inches(40, 20)
av = plt.xticks(rotation = 45, ha = 'right')

av = plt.savefig("Top 10 Most Favorited Dog Breeds.jpg", bbox_inches = 'ti
```




## Insight #4: The Top 10 Most Retweeted Dog Breeds.

In [588]:

```
# Merge 2 dataframes.
df_twit_mstr5 = pd.merge(df_twit_mstr3, twts_cnts2, on = 'tweet_id', how =

# Removes the fav_count column.
df_twit_mstr5 = df_twit_mstr5.drop('fav_count', axis = 1).copy()

df_twit_mstr5.head(1)
```

Out[588]:

| | tweet_id | p1 | p2 | p3 | retwt_count |
|---|---|---|---|---|---|
| **0** | 666020888022790149 | welsh_springer_spaniel | collie | shetland_sheepdog | 532 |

In [589]:

```
# Creates a copy with a reset index.
df_groups2 = df_groups.reset_index().copy()

# Renames the index column to breed.
df_groups2.rename(columns = {'index': 'breed'}, inplace = True)

# Creates a copy of the breed series.
df_groups3 = df_groups2.breed.copy()

df_groups3
```

Out[589]:

```
0      labrador_retriever
1        golden_retriever
2               chihuahua
3                pembroke
4                cardigan
              ...
111      sealyham_terrier
112    kerry_blue_terrier
113         affenpinscher
114        scotch_terrier
115       irish_wolfhound
Name: breed, Length: 116, dtype: object
```

In [590]:

```
# Creates dataframe and appends the grouped breeds.
df_breeds7 = pd.DataFrame()

for favs in range(len(df_groups3)):
        df_breeds7 = df_breeds7.append(df_twit_mstr5.loc[df_twit_mstr5['p1
        df_breeds7 = df_breeds7.append(df_twit_mstr5.loc[df_twit_mstr5['p2
        df_breeds7 = df_breeds7.append(df_twit_mstr5.loc[df_twit_mstr5['p3

df_breeds7.head(5)
```

Out[590]:

| | tweet_id | p1 | p2 | p3 | retwt_co |
|---|---|---|---|---|---|
| **30** | 666701168228331520 | labrador_retriever | chihuahua | french_bulldog | |
| **54** | 667453023279554560 | labrador_retriever | french_bulldog | staffordshire_bullterrier | |
| **57** | 667502640335572993 | labrador_retriever | golden_retriever | beagle | |
| **72** | 668204964695683073 | labrador_retriever | golden_retriever | chesapeake_bay_retriever | |
| **81** | 668528771708952576 | labrador_retriever | kuvasz | english_setter | |

In [591]:
```
# Groups the p1 column into breeds, sums the values, and displays.
df_breeds8 = df_breeds7.groupby('p1').agg({'retwt_count': ['sum']}).copy()

df_breeds8.head(5)
```

Out[591]:

| | retwt_count |
|---|---|
| | sum |
| p1 | |
| afghan_hound | 21615 |
| airedale | 47532 |
| american_staffordshire_terrier | 85992 |
| appenzeller | 2274 |
| australian_terrier | 18471 |

In [592]:
```
# Groups the p2 column into breeds, sums the values, and displays.
df_breeds9 = df_breeds7.groupby('p2').agg({'retwt_count': ['sum']}).copy()

df_breeds9.head(5)
```

Out[592]:

| | retwt_count |
|---|---|
| | sum |
| p2 | |
| affenpinscher | 3750 |
| afghan_hound | 49686 |
| airedale | 20574 |
| american_staffordshire_terrier | 220320 |
| appenzeller | 24099 |

In [593]:

```
# Groups the p3 column into breeds, sums the values, and displays.
df_breeds10 = df_breeds7.groupby('p3').agg({'retwt_count': ['sum']}).copy(

df_breeds10.head(5)
```

Out[593]:

| | retwt_count |
|---|---|
| | sum |
| p3 | |
| affenpinscher | 3678 |
| afghan_hound | 5304 |
| airedale | 88806 |
| american_staffordshire_terrier | 100812 |
| appenzeller | 40137 |

In [594]:

```
# Concatenates 3 dataframes.
df_breeds11 = df_breeds8 + df_breeds9 + df_breeds10

# Sorts the values, sums the total of retwt_count for each breed, unstacke
df_breeds11 = df_breeds11.sort_values([('retwt_count', 'sum')], ascending

df_breeds11.head(10)
```

Out[594]:

| | level_0 | level_1 | level_2 | 0 |
|---|---|---|---|---|
| 0 | retwt_count | sum | labrador_retriever | 2168580 |
| 1 | retwt_count | sum | golden_retriever | 2111784 |
| 2 | retwt_count | sum | chihuahua | 1350330 |
| 3 | retwt_count | sum | pembroke | 1256196 |
| 4 | retwt_count | sum | pomeranian | 1177071 |
| 5 | retwt_count | sum | cardigan | 1172985 |
| 6 | retwt_count | sum | toy_poodle | 738726 |
| 7 | retwt_count | sum | cocker_spaniel | 697551 |
| 8 | retwt_count | sum | eskimo_dog | 662823 |
| 9 | retwt_count | sum | chesapeake_bay_retriever | 656448 |

In [595]:

```
# Renames columns.
df_breeds11.rename(columns = { df_breeds11.columns[2]: 'breed', df_breeds1

# Removes level_0 and level_1 columns.
df_breeds11.drop(['level_0', 'level_1'], axis = 'columns', inplace = True)

df_breeds11.head(10)
```

Out[595]:

| | breed | retwt_count |
|---|---|---|
| **0** | labrador_retriever | 2168580 |
| **1** | golden_retriever | 2111784 |
| **2** | chihuahua | 1350330 |
| **3** | pembroke | 1256196 |
| **4** | pomeranian | 1177071 |
| **5** | cardigan | 1172985 |
| **6** | toy_poodle | 738726 |
| **7** | cocker_spaniel | 697551 |
| **8** | eskimo_dog | 662823 |
| **9** | chesapeake_bay_retriever | 656448 |

In [596]:

```
# Creates a dataframe copy of the first 5 rows.
df_breeds_final2 = df_breeds11.head(10).copy()

df_breeds_final2
```

Out[596]:

| | breed | retwt_count |
|---|---|---|
| **0** | labrador_retriever | 2168580 |
| **1** | golden_retriever | 2111784 |
| **2** | chihuahua | 1350330 |
| **3** | pembroke | 1256196 |
| **4** | pomeranian | 1177071 |
| **5** | cardigan | 1172985 |
| **6** | toy_poodle | 738726 |
| **7** | cocker_spaniel | 697551 |
| **8** | eskimo_dog | 662823 |
| **9** | chesapeake_bay_retriever | 656448 |

In [597]:

```
# Divide the retwt_count column by hundred thousands for visualization.
df_breeds_final2.retwt_count = df_breeds_final2.retwt_count.div(10**5).sor

df_breeds_final2
```

Out[597]:

| | breed | retwt_count |
|---|---|---|
| **0** | labrador_retriever | 21.69 |
| **1** | golden_retriever | 21.12 |
| **2** | chihuahua | 13.50 |
| **3** | pembroke | 12.56 |
| **4** | pomeranian | 11.77 |
| **5** | cardigan | 11.73 |
| **6** | toy_poodle | 7.39 |
| **7** | cocker_spaniel | 6.98 |
| **8** | eskimo_dog | 6.63 |
| **9** | chesapeake_bay_retriever | 6.56 |

In [598]:

```
# Creates and displays a bar graph.
ar = sns.set(font_scale = 5)

ar = sns.barplot(x = 'breed',
            y = 'retwt_count',
            data = df_breeds_final2)

ar.set(xlabel = 'Dog Breed',
       ylabel = 'Total Retweet Count (hundred thousands)',
       title = 'Top 10 Most Retweeted Dog Breeds.')

ar.figure.set_size_inches(40, 20)
ar = plt.xticks(rotation = 45, ha = 'right')

ar = plt.savefig("Top 10 Most Retweeted Dog Breeds.jpg", bbox_inches = 'ti
```