Report: wrangle_report

1. Data gathering

Gathering the three pieces of data for this project and load to the notebook.

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

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

```
#import Tweet image predictions:
twt_image_url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.t
response = requests.get(twt_image_url)

if response.status_code == 200:
    with open("image_predictions.tsv", "wb") as file:
        file.write(response.content)
        print("Datasheet downloaded successfully!")
else:
    print("Failed to download datasheet. Status code:", response.status_code)
```

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

Request Twitter APIs:

```
consumer_key =
consumer_secret
access_token =
access_secret =

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

api = tweepy.API(auth,parser = tweepy.parsers.JSONParser())
✓ 0.0s
```

Try querying the first tweet by its ID:

```
#try querying the first tweet:
tweet1 = api.get_status(892420643555336193)
print(tweet1.text)
```

Result:

```
---------------------------------------------------------------------------
Forbidden                                 Traceback (most recent call last)
Cell In[1946], line 2
      1 #try querying the first tweet:
----> 2 tweet1 = api.get_status(892420643555336193)
      3 print(tweet1.text)

File c:\Users\latong\Anaconda3\envs\myenv\lib\site-packages\tweepy\api.py:46, in payload..decorator..wrapper(*args, **kwargs
     44 kwargs['payload_list'] = payload_list
     45 kwargs['payload_type'] = payload_type
---> 46 return method(*args, **kwargs)

File c:\Users\latong\Anaconda3\envs\myenv\lib\site-packages\tweepy\api.py:739, in API.get_status(self, id, **kwargs)
    706 @payload('status')
    707 def get_status(self, id, **kwargs):
...
    272 if resp.status_code == 404:
    273     raise NotFound(resp)

Forbidden: 403 Forbidden
453 - You currently have access to a subset of Twitter API v2 endpoints and limited v1.1 endpoints (e.g. media post, oauth)
```

It looks like free level access to Twitter APIs no longer provide the access to *get_status* API. Hence, I used the provided '*tweet-json.txt*' file instead:

```
tweet_list = []
with open('tweet-json.txt', 'r') as json_text:
    data = json_text.read()
    json_objects = data.split('\n')
    parsed_objects = []
    for json_object in json_objects:
        if json_object.strip():
            parsed_objects.append(json.loads(json_object))
for obj in parsed_objects:
        tweet_id = obj['id']
        favorite_count = obj['favorite_count']
        retweet_count = obj['retweet_count']

        tweet_list.append({'tweet_id': str(tweet_id),
                           'favorite_count': int(favorite_count),
                           'retweet_count': int(retweet_count),
                          })
        tweet_json = pd.DataFrame(tweet_list, columns = ['tweet_id', 'favorite_count','retweet_count'])
```

2. Accessing Data:

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

a. Inspecting *'Twitter_archive_enhance'* dataframe:

Visual accessment:

```
twitter_archive_enchanced
✓ 0.0s
```

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	Twitter for iPhone	This is F mystical appears i https://t.co/M
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	Twitter for iPhone	This is checkin Hopes you not, she pats, snu wi https://t.c
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	Twitter for iPhone	This is Arch Norwe Corgo. L grass. Yo when c https://t.co Th

Scanning the 'name' column to detect quality issues:

```
[1825]  twitter_archive_enchanced['name'].unique()
        ✓ 0.0s
...
array(['Phineas', 'Tilly', 'Archie', 'Darla', 'Franklin', 'None', 'Jax',
       'Zoey', 'Cassie', 'Koda', 'Bruno', 'Ted', 'Stuart', 'Oliver',
       'Jim', 'Zeke', 'Ralphus', 'Canela', 'Gerald', 'Jeffrey', 'such',
       'Maya', 'Mingus', 'Derek', 'Roscoe', 'Waffles', 'Jimbo', 'Maisey',
       'Lilly', 'Earl', 'Lola', 'Kevin', 'Yogi', 'Noah', 'Bella',
       'Grizzwald', 'Rusty', 'Gus', 'Stanley', 'Alfy', 'Koko', 'Rey',
       'Gary', 'a', 'Elliot', 'Louis', 'Jesse', 'Romeo', 'Bailey',
       'Duddles', 'Jack', 'Emmy', 'Steven', 'Beau', 'Snoopy', 'Shadow',
       'Terrance', 'Aja', 'Penny', 'Dante', 'Nelly', 'Ginger', 'Benedict',
       'Venti', 'Goose', 'Nugget', 'Cash', 'Coco', 'Jed', 'Sebastian',
       'Walter', 'Sierra', 'Monkey', 'Harry', 'Kody', 'Lassie', 'Rover',
       'Napolean', 'Dawn', 'Boomer', 'Cody', 'Rumble', 'Clifford',
       'quite', 'Dewey', 'Scout', 'Gizmo', 'Cooper', 'Harold', 'Shikha',
       'Jamesy', 'Lili', 'Sammy', 'Meatball', 'Paisley', 'Albus',
       'Neptune', 'Quinn', 'Belle', 'Zooey', 'Dave', 'Jersey', 'Hobbes',
       'Burt', 'Lorenzo', 'Carl', 'Jordy', 'Milky', 'Trooper', 'Winston',
       'Sophie', 'Wyatt', 'Rosie', 'Thor', 'Oscar', 'Luna', 'Callie',
       'Cermet', 'George', 'Marlee', 'Arya', 'Einstein', 'Alice',
       'Rumpole', 'Benny', 'Aspen', 'Jarod', 'Wiggles', 'General',
       'Sailor', 'Astrid', 'Iggy', 'Snoop', 'Kyle', 'Leo', 'Riley',
```

It looks like beside from 'None' the incorrect names like 'old', 'a', 'my' ,etc. are lowercase.

Inspecting the ‘rating_numerator’ and ‘rating_denominator’ columns for outliners:

```
twitter_archive_enchanced['rating_numerator'].value_counts()
✓ 0.0s

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
...
24       1
960      1
84       1
88       1
Name: rating_numerator, dtype: int64
```

```
twitter_archive_enchanced['rating_denominator'].value_counts()
✓ 0.0s

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: rating_denominator, dtype: int64
```

Programmatic accessment:

```
twitter_archive_enchanced.info()
✓ 0.0s

<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
...
 15  pupper                      2356 non-null   object
 16  puppo                       2356 non-null   object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Accessing categorized dog stages:

```
"""
:return -  total of dog in the stage
:param  - (stage) = 'pupper','puppo','doggo','floofer'
"""
def catgorized_stage(stage):
    number = (twitter_archive_enchanced[stage] == stage).sum()
    return number

stage = ('pupper','puppo','doggo','floofer')
categorized = 0
for dog in stage:
    print(f'Number of {dog}: {catgorized_stage(dog)}')
    categorized += catgorized_stage(dog)
print(f'Number of dog arent sorted to any stages: {twitter_archive_enchanced.shape[0] - categorized}')
```

✓ 0.0s

```
Number of pupper: 257
Number of puppo: 30
Number of doggo: 97
Number of floofer: 10
Number of dog arent sorted to any stages: 1962
```

Accessing the lowercase names:

```
lowercase_names = twitter_archive_enchanced[twitter_archive_enchanced['name'].str.islower()]
suspect_name = lowercase_names['name'].unique()
suspect_name
```

✓ 0.0s

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

Those lowercase names are incorrect names for dogs.

Checking the text discriptions of tweets with abnomal rating numerator:

```
#with pd.set_option('display.max_colwidth', None):
rating = (204,143,666,1776,144)
for i in rating:
    print_tweet(col_name='rating_numerator',value=i)
```

✓ 0.0s Python

```
1120    Say hello to this unbelievably well behaved squad of doggos. 204/170 would try to pet all at once https://t.co/yGQI
Name: text, dtype: object
1634    Two sneaky puppers were not initially seen, moving the rating to 143/130. Please forgive us. Thank you https://t.co
Name: text, dtype: object
189    @s8n You tried very hard to portray this good boy as not so good, but you have ultimately failed. His goodness shine
Name: text, dtype: object
979    This is Atticus. He's quite simply America af. 1776/10 https://t.co/GRXwMxLBkh
Name: text, dtype: object
1779    IT'S PUPPERGEDDON. Total of 144/120 ...I think https://t.co/ZanVtAtvIq
Name: text, dtype: object
```

b. Inspecting *df_image_predictions* dataframe:

Visual accessment:

```
df_image_predictions
✓ 0.0s                                                                    Python
```

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074	True	
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826	True	mi
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.596461	True	
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridgeback	0.408143	True	
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pinscher	0.560311	True	
...	...	...	...	...	...	...	
2070	891327558926688256	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	2	basset	0.555712	True	
2071	891689557279858688	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	1	paper_towel	0.170278	False	La
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	Chihuahua	0.716012	True	
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	Chihuahua	0.323581	True	
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	orange	0.097049	False	

2075 rows × 12 columns

Programmatic accessment:

```
[1834]  df_image_predictions.info()
        ✓ 0.0s
```

```
<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
```

Checking for duplicated:

```
[1835]  df_image_predictions.duplicated().sum()
        ✓ 0.0s
```

```
0
```

Checking the images of the tweets failed to be recognized dog by all 3 algorithms:

```
#check the images that was failed to recognized dogs with all 3 algorithm:
condition = (df_image_predictions['p1_dog'] == False) & (df_image_predictions['p2_dog'] == False) & (df
filtered_df = df_image_predictions[condition]

filtered_df['jpg_url']
```

[1836] ✓ 0.0s

```
6                                                 https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg
17                                                https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg
18                                                https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg
21                                                https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg
25                                                https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg
                                                                          ...
2021                                              https://pbs.twimg.com/media/DDm2Z5aXUAEDS2u.jpg
2022                                              https://pbs.twimg.com/media/DDrk-f9WAAI-WQv.jpg
2046                                              https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg
2052    https://pbs.twimg.com/ext_tw_video_thumb/887517108413886465/pu/img/WanJKwssZj4VJvL9.jpg
2074                                              https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg
Name: jpg_url, Length: 324, dtype: object
```

Cross-checking with the tweets' text and URLs:

```
merge_tw = pd.merge(filtered_df,twitter_archive_enchanced, on='tweet_id')
```

[1837] ✓ 0.0s Python

```
#check the contents and URL of those tweets with no dogs in the picture:
merge_tw['text']
```

[1838] ✓ 0.0s Python

```
0                          This is an odd dog. Hard on the outside but loving on the inside. Petting still fun. Doesn't
1                              Not familiar with this breed. No tail (weird). Only 2 legs. Doesn't bark. Surprisingly qu
2                                                                      Very concerned about fellow dog trapp
3                          This is a funny dog. Weird toes. Won't come down. Loves branch. Refuses to eat his food. Hard
4                        Unique dog here. Very small. Lives in container of Frosted Flakes (?). Short legs. Must be rare
                                                                                    ...
319                                                                This is Louis. He's crossing. It's a big deal. 13/1
320    Meet Elliot. He's a Canadian Forrest Pup. Unusual number of antlers for a dog. Sneaky tongue slip to celebrate #Canad
321                                                                 This is Derek. He's late for a dog meeting. 13/10
322                                  I've yet to rate a Venezuelan Hover Wiener. This is such an honor. 14/10 paw-inspiring
323                                                       This is Phineas. He's a mystical boy. Only ever appears in the h
Name: text, Length: 324, dtype: object
```

It looks like some of the images in this list still have dogs, whereas large amounts of them (that all 3 algorithms returned: False) don't have any dogs. But they are still tweets from 'WeRateDogs' account, so I won't clean them. But If the purpose of the analysis is focused on dogs, we should clean this issue.

c. Inspecting tweet_json dataframe:

Visual accessment:

```
tweet_json
```

[1949] ✓ 0.0s

	tweet_id	favorite_count	retweet_count
0	892420643555336193	39467	8853
1	892177421306343426	33819	6514
2	891815181378084864	25461	4328
3	891689557279858688	42908	8964
4	891327558926688256	41048	9774
...	...	...	...
2349	666049248165822465	111	41
2350	666044226329800704	311	147
2351	666033412701032449	128	47
2352	666029285002620928	132	48
2353	666020888022790149	2535	532

2354 rows × 3 columns

Programmatic accessment:

```
tweet_json.info()
```

[1840] ✓ 0.0s

```
<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   favorite_count  2354 non-null   int64 
 2   retweet_count   2354 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 55.3+ KB
```

d. Issues Found:

Quality issues:

1. twitter_archive_enhance: 'name' some names are incorrect.
2. twitter_archive_enhance: dataframe: some of 'rating_numerator' and 'rating_denominator' are not correct.
3. twitter_archive_enhance: 'timestamp' are string objects
4. twitter_archive_enhance: 'tweet_id' are int type
5. df_image_predictions: 'tweet_id' are int type

6. twitter_archive_enhance: 1962 dogs are not shorted into any of the 4 stages, 14 dogs are sorted into 2 types
7. twitter_archive_enhance: 'retweeted_status_timestamp' are string objects
8. twitter_archive_enhance: 'reply_to_status_id', 'in_reply_to_user_id' ,'retweeted_status_id' and 'retweeted_status_user_id' are float.

Tidiness Issues:

1. twitter_archive_enhance: 'Dog stages' (doggo,puppo,etc) are stored in multiple columns
2. twitter_archive_enhance: 'text' contains both tweet's content and its URL
3. twitter_archive_enhance: 2 columns for the 'rating' values: 'numerator' and 'denominator'

3. Cleaning Data:

Make copies of original pieces of data.

a. Cast *tweet_id* of *twitter_archive_clean and df_image_predictions_clean* to string type

b. Merge 3 dataframes by *tweet_id* into one dataframe *df*

c. Change *timestamp* and *retweeted_status_timestamp* to correct *datetime* format

d. Correct incorrect names: Delete names in those don't provide names in the tweets, and replace the right name for some.

```
#Delete all dogs named in suspect_name
df['name'] = df['name'].replace(suspect_name,np.nan)
```
✓ 0.0s

```
#Delete all dogs named 'None'
df['name'] = df['name'].replace('None', np.nan)
```
✓ 0.0s

```
#Correct some names:
df.iloc[22, df.columns.get_loc('name')] = 'Roxy'
df.iloc[649, df.columns.get_loc('name')] = 'Forrest'
df.iloc[801, df.columns.get_loc('name')] = 'Galapagos Speed Panda'
df.iloc[1853, df.columns.get_loc('name')] = 'Wylie'
```
✓ 0.0s

e. Correct ratings *in rating_numerator* and *rating_denominator*, by the index of those rows (Highlighted rows need to correct manually while the rest is score of multiple dogs

Correct *rating_denominator:*

```
rates = df['rating_numerator']/df['rating_denominator']
rates
```

✓ 0.0s

```
0       1.3
1       1.3
2       1.2
3       1.3
4       1.2
       ...
2351    0.5
2352    0.6
2353    0.9
2354    0.7
2355    0.8
Length: 2356, dtype: float64
```

```
loc = [784, 1068, 1662, 1202, 1274, 1351, 1165, 1254, 1843, 433, 516, 342, 1120, 1228, 1433, 1634, 16
for loc_index in loc:
    df.iloc[loc, df.columns.get_loc('rating_denominator')] = 10
```

✓ 0.0s

```
df['rating_denominator'].value_counts()
```

✓ 0.0s

```
10    2356
Name: rating_denominator, dtype: int64
```

Correcting *rating_numerator* manually*:*

```
num_loc =  [784, 1068, 1662, 1202, 1165, 516, 342, 313, 1663, 2335, 1598]
rate_num = [14, 14, 10, 11, 13, 10, 0, 13, 13, 9, 2]

for index in range(len(num_loc)):
    df.at[num_loc[index], 'rating_numerator'] = rate_num[index]
```

✓ 0.0s

```
df[df['rating_numerator'] == 26]['text']
```

[1867] ✓ 0.0s

```
1712    Here we have uncovered an entire battalion of holiday puppers. Average of 11.26/10 https://t.co/e
Name: text, dtype: object
```

```
df.at[1712, 'rating_numerator'] = 11
```

[1868] ✓ 0.0s

Correcting *rating_numerator* by standardizing the rating for multiple dogs:

```
rates = df['rating_numerator']/df['rating_denominator']
rates
```
[1857] ✓ 0.0s

```
rates_array = np.array(rates*10)
```
[1860] ✓ 0.0s

```
rates_array = rates_array.astype(int)
rates_array
```
[1861] ✓ 0.0s

f. Change *reply_to_status_id*, *in_reply_to_user_id* , *retweeted_status_id* and *retweeted_status_user_id* to string.

g. Separate the *text* column into 'text' and 'tweet_URL' columns

```
url_pattern = r'(https?://\S+)'
df['urls'] = df['text'].str.extract(url_pattern)
```
[1887] ✓ 0.0s Python

```
text_patern = r'(.+)(?:https?://\S+)'
df['Text'] = df['text'].str.extract(text_patern)
```
[1888] ✓ 0.0s Python

h. Standardize the Ratings by dropping the *rating_denominator* column and rename *rating_numerator* to *rating_score_(out_of_10)*

```
#Drop the rating_denominator column:
df = df.drop(columns='rating_denominator')
```
[1893] ✓ 0.0s

```
#rename rating_numerator
df = df.rename(columns={'rating_numerator': 'rating_score_(out_of_10)'})
```
[1894] ✓ 0.0s

i. Create one column *stage* for the dog stages: *doggo, floofer, pupper, puppo*

```
stages = ['puppo', 'doggo', 'floofer', 'pupper']
def drop_none(col):
    df[col] = df[col].replace('None',np.nan)
for stage in stages:
    drop_none(stage)
```
[1895] ✓ 0.0s Python

```
df['stage'] = df['puppo'].fillna('') + df['doggo'].fillna('') + df['floofer'].fillna('')+ df['pupp
```

4. **Test:**

   Using programmtic method to recheck the cleaned dataframe. Result: The cleaned dataframe tackles all the issues in Define step.