Wrangling report

Data wrangling process.

Stage (A) : Gather the information from 3 variant sources:

- First source: It is a csv file on hand file called 'twitter-archive-enhanced.csv' , that was the most trivial part of the gathering process because all it needs is reading g via pandas data frame in a file called archive_df.

- Second source: It is a tsv file was already saved in Udacity database and all we have to do is uploading this file programmatically and extracting Image-predictions.tsv from the URL provided. Then reading it via pandas Data frame in a file called image_predictions_df.

```
# reading Image Predictions File and extracting Image-predictions.tsv

folder_name = 'image_predictions_file'

if not os.path.exists(folder_name):

   os.makedirs(folder_name)




url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-
predictions/image-predictions.tsv'

response = requests.get(url)




with open(os.path.join(folder_name,

                  url.split('/')[-1]), mode = 'wb') as file:

   file.write(response.content)
```

- Third source: I had to do the second choice which is using the file that already attached in the classroom and I studied the code related to this well. Anyway, I have done this and then read this file line by line to get the data , finally importing them into pandas data frame called api_df. That was the most challenging part in the gathering process.

```
# reading (tweet-json.txt) file line by line


df_list = []


with open('tweet-json.txt' ,'r') as file:
    for line in file:
        tweet = json.loads(line)
        tweet_id = tweet['id']
        retweet_count = tweet['retweet_count']
        fav_count = tweet['favorite_count']
        user_count = tweet['user']['followers_count']


        # Append to list of dictionaries.


        df_list.append({'tweet_id': tweet_id,
                        'retweet_count': retweet_count,
                        'favorite_count': fav_count,
                        'user_count': user_count})
```

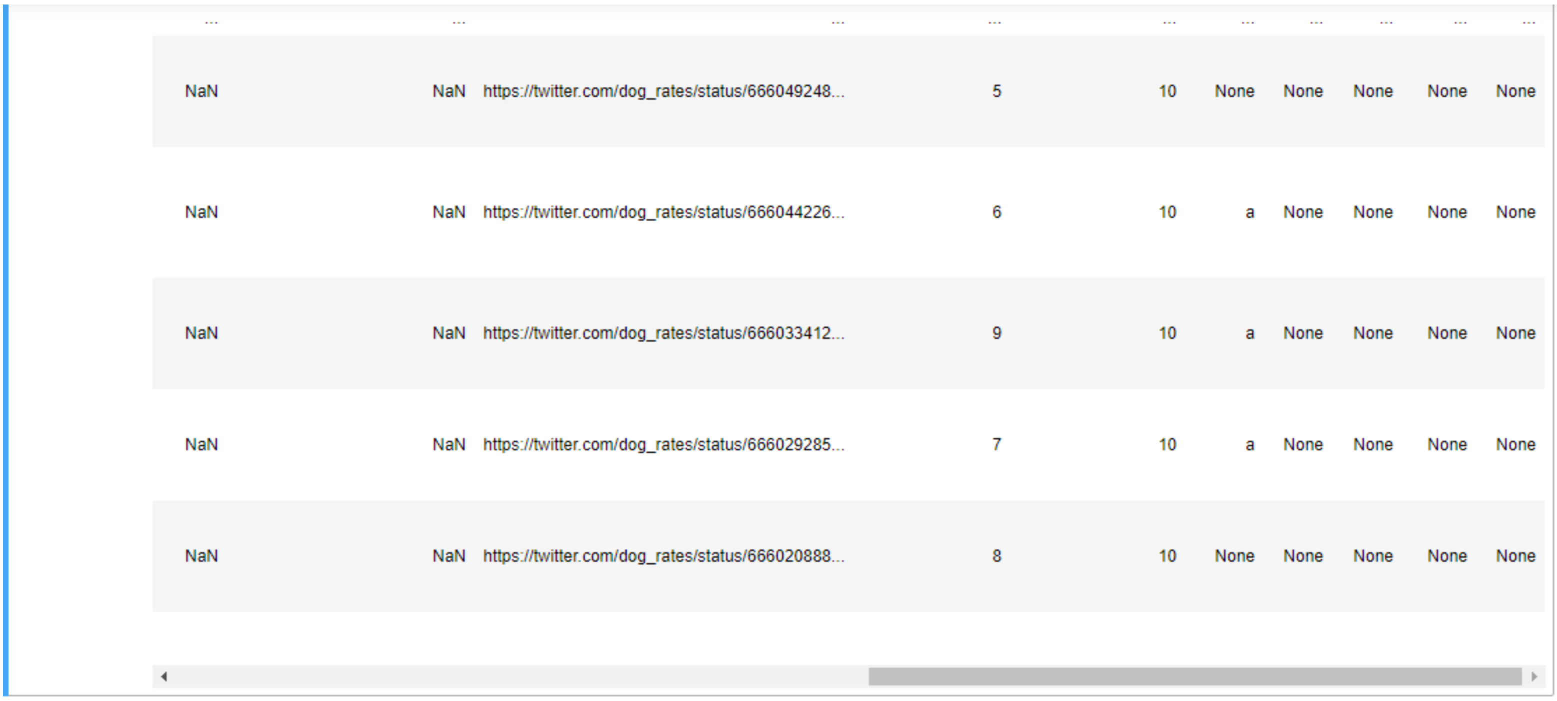
Reading files.

```
# reading Enhanced Twitter Archive (.csv) and creating pandas dataframe.
archive_df = pd.read_csv('twitter-archive-enhanced.csv')


# reading Image-predictions (.tsv) and creating pandas dataframe.
image_predictions_df = pd.read_csv('image_predictions_file/image-predictions.tsv', sep='\t')


# # JSON objects to DataFrame:
api_df = pd.DataFrame(df_list , columns = ['tweet_id', 'retweet_count',
'favorite_count','user_count'])
```

Stage (B) : Assessing .

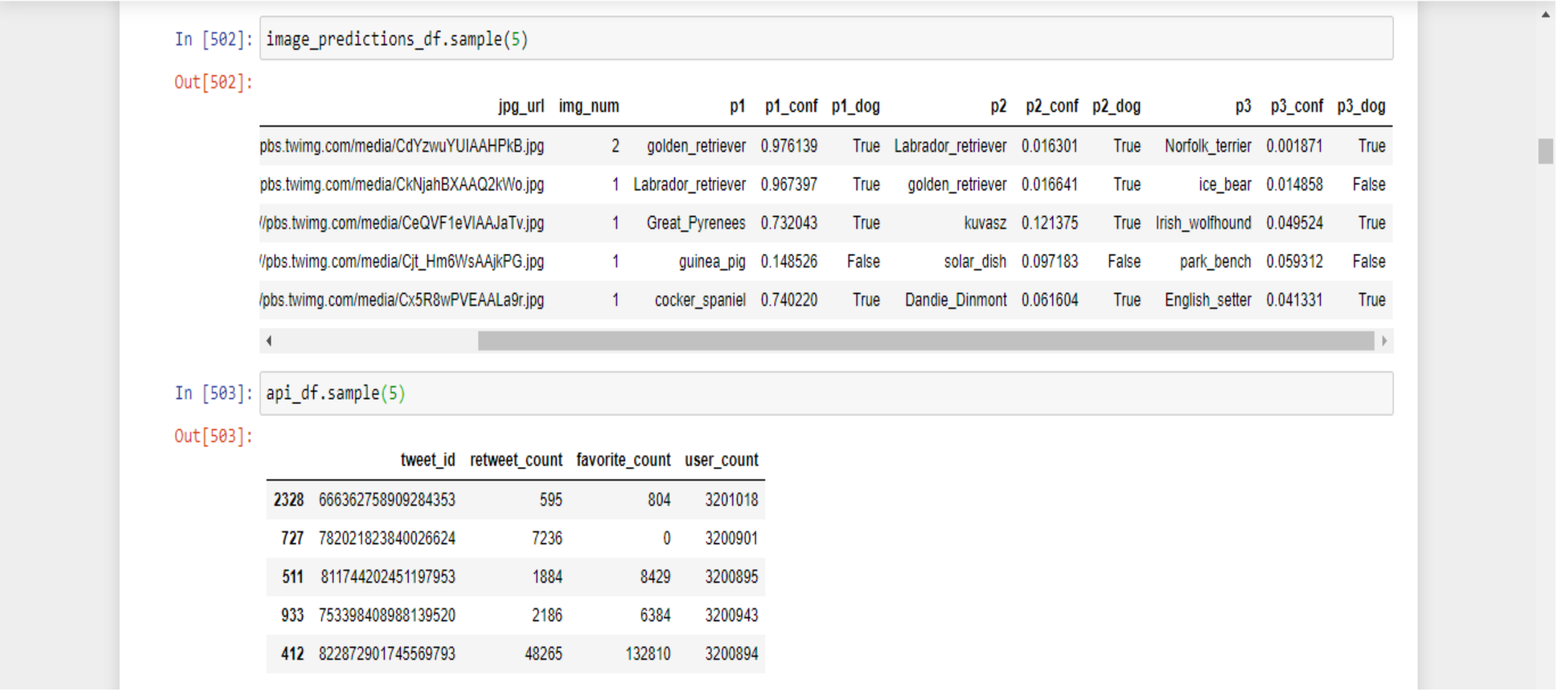
- **First, visually**

archive_df



...	...	...	...	...	...	...	...	...	...
NaN	NaN	https://twitter.com/dog_rates/status/666049248...	5	10	None	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/666044226...	6	10	a	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/666033412...	9	10	a	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/666029285...	7	10	a	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/666020888...	8	10	None	None	None	None	None

- there is an invalid name like (a)
- 'None' instead of NaN in missing values.



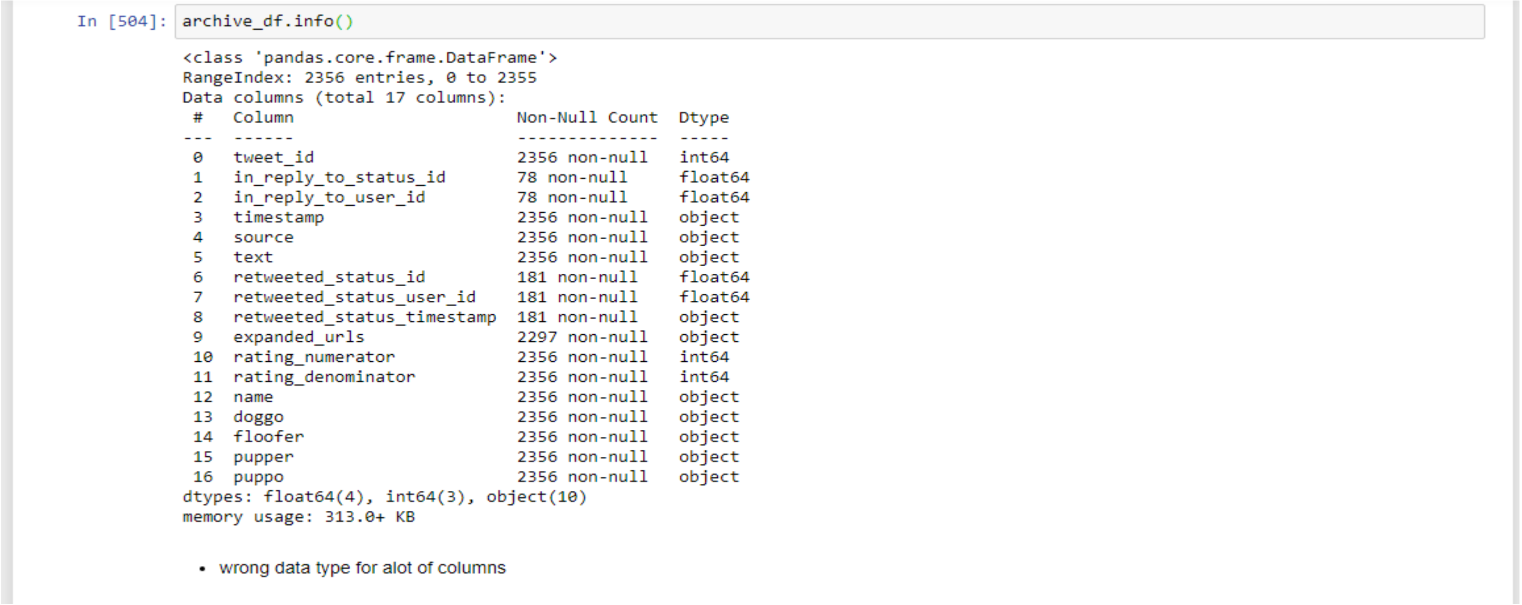
```
In [502]: image_predictions_df.sample(5)
Out[502]:
```

jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog
pbs.twimg.com/media/CdYzwuYUIAAHPkB.jpg	2	golden_retriever	0.976139	True	Labrador_retriever	0.016301	True	Norfolk_terrier	0.001871	True
pbs.twimg.com/media/CkNjahBXAAQ2kWo.jpg	1	Labrador_retriever	0.967397	True	golden_retriever	0.016641	True	ice_bear	0.014858	False
//pbs.twimg.com/media/CeQVF1eVIAAJaTv.jpg	1	Great_Pyrenees	0.732043	True	kuvasz	0.121375	True	Irish_wolfhound	0.049524	True
//pbs.twimg.com/media/Cjt_Hm6WsAAjkPG.jpg	1	guinea_pig	0.148526	False	solar_dish	0.097183	False	park_bench	0.059312	False
/pbs.twimg.com/media/Cx5R8wPVEAALa9r.jpg	1	cocker_spaniel	0.740220	True	Dandie_Dinmont	0.061604	True	English_setter	0.041331	True

```
In [503]: api_df.sample(5)
Out[503]:
```

	tweet_id	retweet_count	favorite_count	user_count
2328	666362758909284353	595	804	3201018
727	782021823840026624	7236	0	3200901
511	811744202451197953	1884	8429	3200895
933	753398408988139520	2186	6384	3200943
412	822872901745569793	48265	132810	3200894

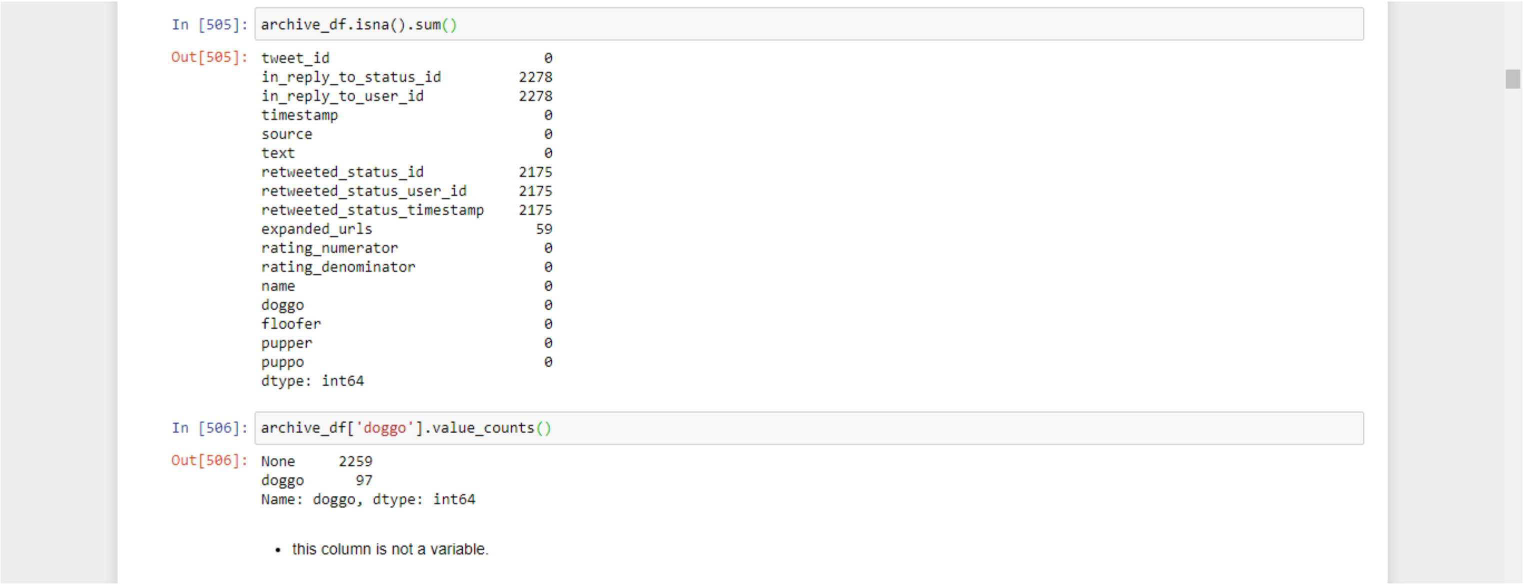
Second: Programmatically



```
In [504]: archive_df.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
```

- wrong data type for alot of columns



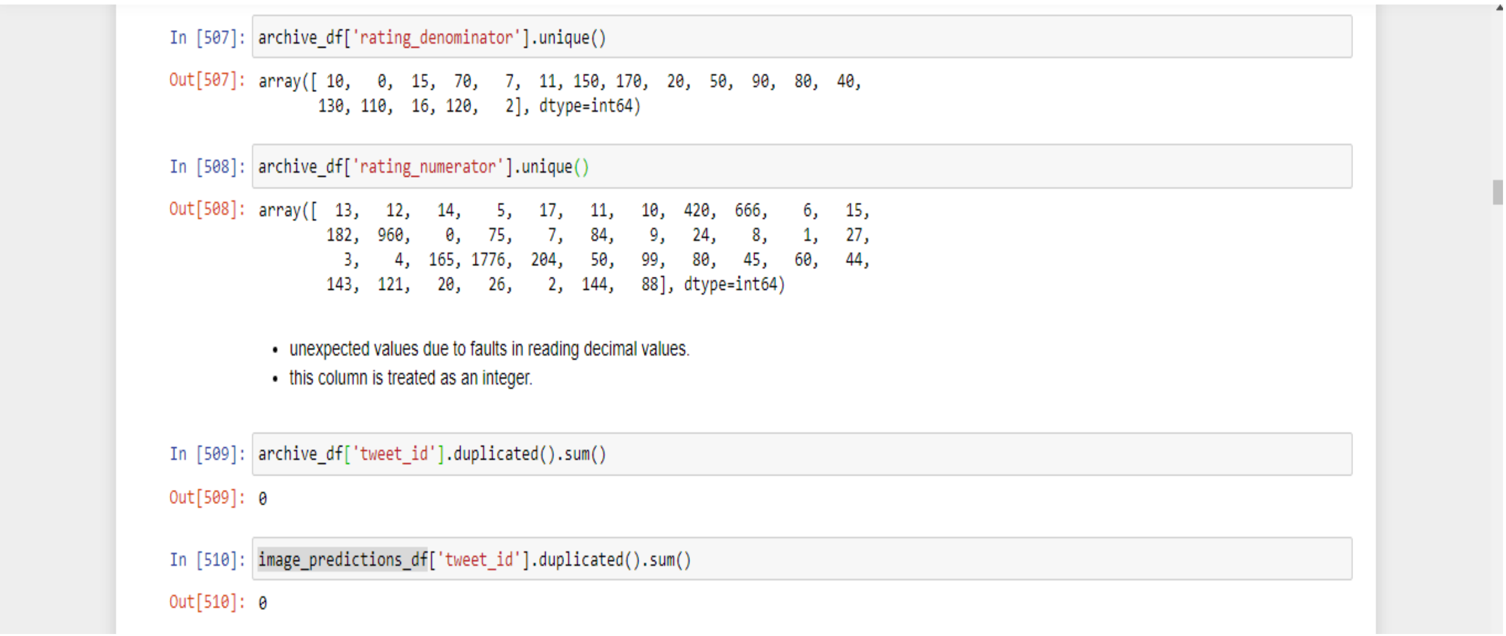
```
In [505]: archive_df.isna().sum()
```

```
Out[505]: 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                             0
          doggo                            0
          floofer                          0
          pupper                           0
          puppo                            0
          dtype: int64
```

```
In [506]: archive_df['doggo'].value_counts()
```

```
Out[506]: None     2259
          doggo      97
          Name: doggo, dtype: int64
```

- this column is not a variable.



```
In [507]: archive_df['rating_denominator'].unique()
```

```
Out[507]: array([ 10,   0,  15,  70,   7,  11, 150, 170,  20,  50,  90,  80,  40,
                 130, 110,  16, 120,   2], dtype=int64)
```

```
In [508]: archive_df['rating_numerator'].unique()
```

```
Out[508]: array([  13,   12,   14,    5,   17,   11,   10,  420,  666,    6,   15,
                  182,  960,    0,   75,    7,   84,    9,   24,    8,    1,   27,
                    3,    4,  165, 1776,  204,   50,   99,   80,   45,   60,   44,
                  143,  121,   20,   26,    2,  144,   88], dtype=int64)
```

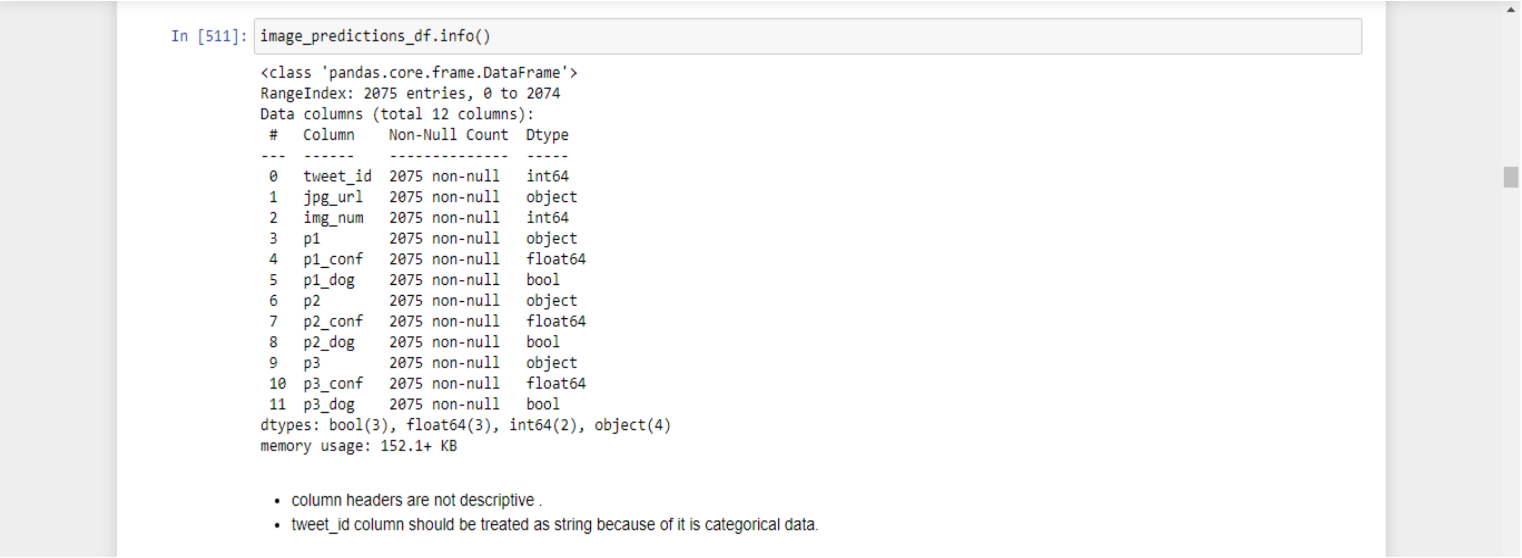
- unexpected values due to faults in reading decimal values.
- this column is treated as an integer.

```
In [509]: archive_df['tweet_id'].duplicated().sum()
```

```
Out[509]: 0
```

```
In [510]: image_predictions_df['tweet_id'].duplicated().sum()
```

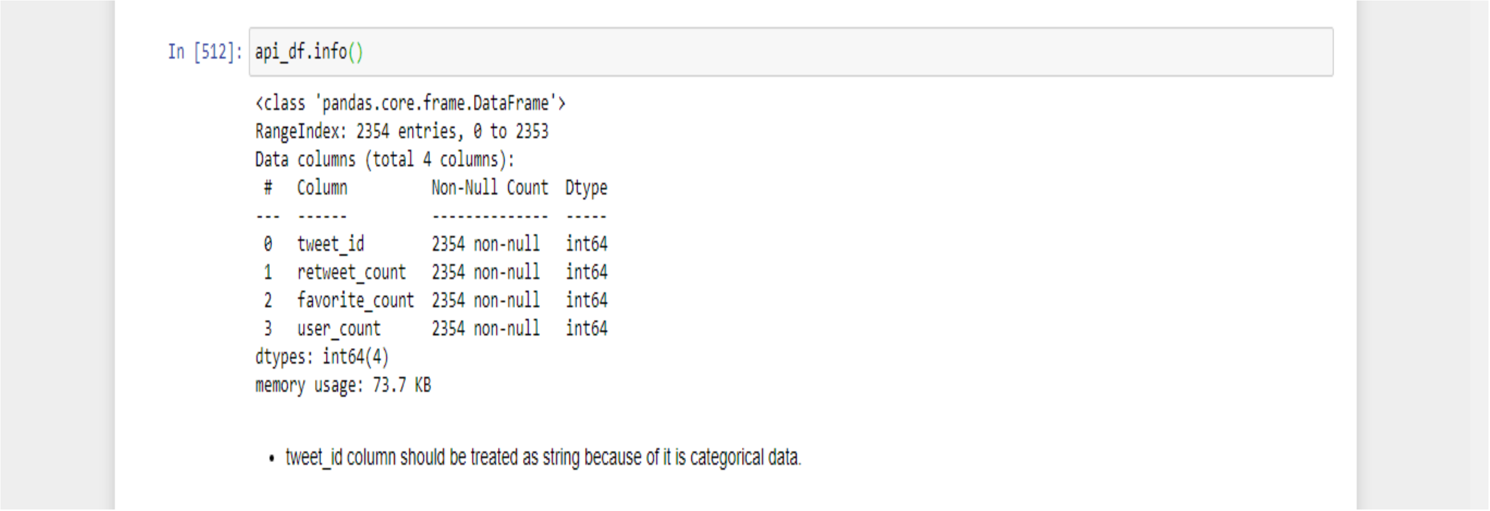
```
Out[510]: 0
```



In [511]:
```
image_predictions_df.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
```

- column headers are not descriptive .
- tweet_id column should be treated as string because of it is categorical data.



In [512]:
```
api_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2354 non-null   int64
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
 3   user_count      2354 non-null   int64
dtypes: int64(4)
memory usage: 73.7 KB
```

- tweet_id column should be treated as string because of it is categorical data.

After giving the data more investigation through two ways manually (display it on excel) and programmatically [using info( ), describe( ), etc.] and what have come in our summary was as following:

Assessment summary.

Quality issues:

In `archive_df`:

- `NaN` is mistakenly written as **None** in: `doggo`,`floofer`,`pupper`,`puppo`, and `name`.
- `tweet_id` type is **integer** instead of **str**.
- `in_reply_to_status_id` type is **float** instead of **str**.
- `in_reply_to_user_id` type is **float** instead of **str**.
- `timestamp` type is **str** instead of **DateTime**.
- `source` type is **str** instead of **category**.
- `retweeted_status_id` type is **float** instead of **str**.
- `retweeted_status_user_id` type is **float** instead of **str**.
- `retweeted_status_timestamp` type is **str** instead of **DateTime**.
- `name` invalid names.
- `rating_numerator` Ratings with decimal values incorrectly extracted
- `rating_numerator` dtype is int instead of float.

In `image_predictions_df`:

- `tweet_id` type is **integer** instead of **str**.
- columns headers are values not variable and not descriptive.

In`api_df`:

- `tweet_id` type is **integer** instead of **str**

Tidiness Issues:

In `archive_df`:

- `doggo`,`floofer`,`pupper`, and `puppo` should all be one column called e.g.`dog_stage`.
- Some records are irrelevant (i.e. retweets or have replies).
- some recordes in `archive_df` don't have images in `image_predictions_df` so I should filter the 3 datasets according to the records in `image_predictions_df` to get the original tweets with images.
- Data from the 3 datasets (`archive_df`, `image_predictions_df`, and `api_df`) can be combined in one DataFrame for simplicity.

Stage (C) : Cleaning .

First, we need to take a copies for our datasets to avoid any bad thing for our original data frames that we get from Gathering step:

```
tweet_clean = archive_df.copy()

image_clean = image_predictions_df.copy()

api_clean = api_df.copy()
```

Second, we are going to have every single issue in Assessment Summary and solve it through applying [Define , Code , Test] strategy

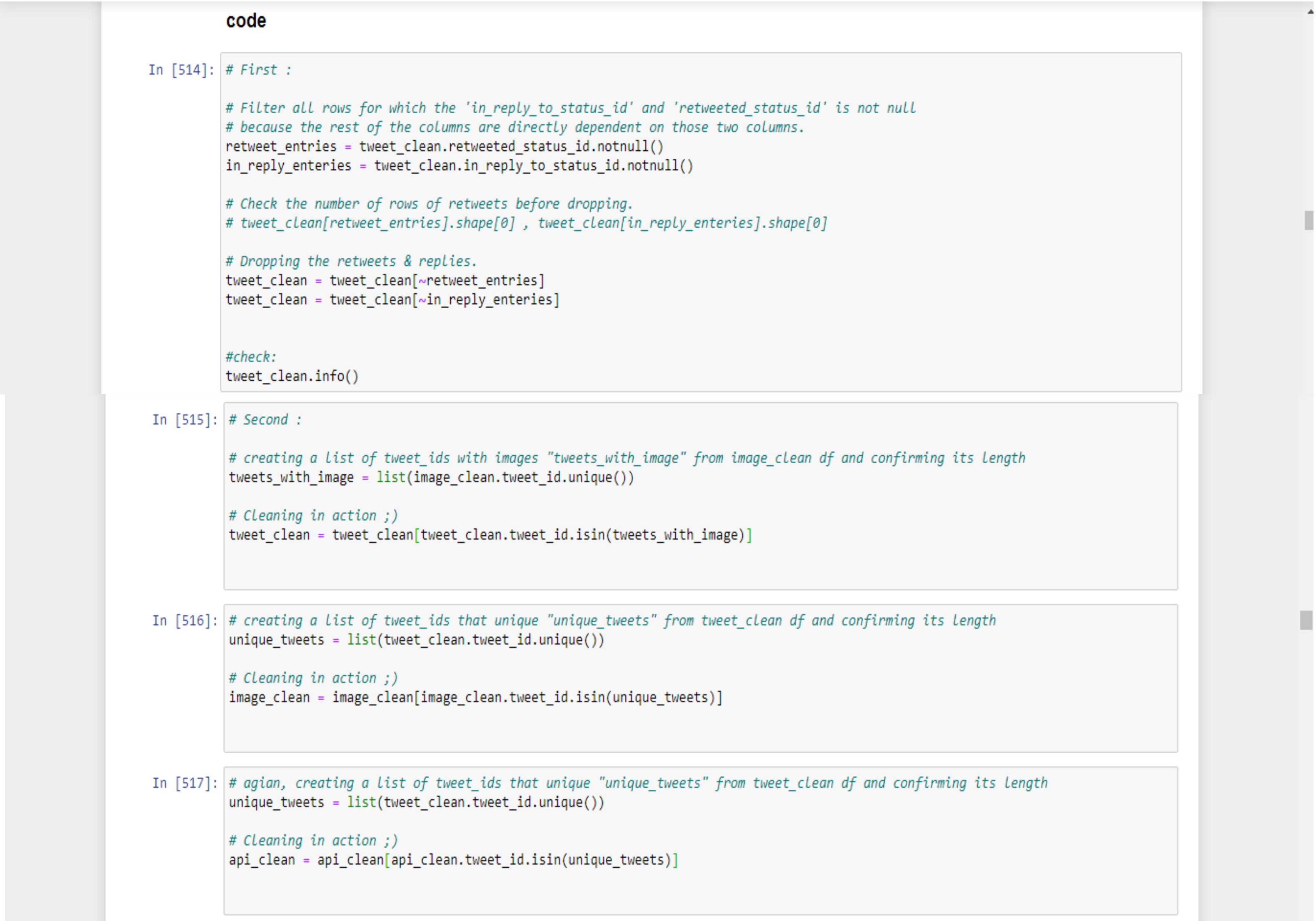
Define

In `archive_df:

- Some records are irrelevant (i.e. retweets or have replies).
- some recordes in `archive_df` don't have images in `image_predictions_df` so I should filter the 3 datasets according to the records in `image_predictions_df` to get the original tweets with images.

Solution

- Drop every row that is `retweeted_status_id.notnull()` & `in_reply_to_status_id.notnull()`
- use the `image_predictions_df` to drop all irrelevant records from the other 2 datasets.
- through creating a list of tweet_ids with images "tweets_with_image" and confirming itslength &use to get rid of tweets without images.



code

```
In [514]: # First :

          # Filter all rows for which the 'in_reply_to_status_id' and 'retweeted_status_id' is not null
          # because the rest of the columns are directly dependent on those two columns.
          retweet_entries = tweet_clean.retweeted_status_id.notnull()
          in_reply_enteries = tweet_clean.in_reply_to_status_id.notnull()

          # Check the number of rows of retweets before dropping.
          # tweet_clean[retweet_entries].shape[0] , tweet_clean[in_reply_enteries].shape[0]

          # Dropping the retweets & replies.
          tweet_clean = tweet_clean[~retweet_entries]
          tweet_clean = tweet_clean[~in_reply_enteries]


          #check:
          tweet_clean.info()
```

```
In [515]: # Second :

          # creating a list of tweet_ids with images "tweets_with_image" from image_clean df and confirming its length
          tweets_with_image = list(image_clean.tweet_id.unique())

          # Cleaning in action ;)
          tweet_clean = tweet_clean[tweet_clean.tweet_id.isin(tweets_with_image)]
```

```
In [516]: # creating a list of tweet_ids that unique "unique_tweets" from tweet_clean df and confirming its length
          unique_tweets = list(tweet_clean.tweet_id.unique())

          # Cleaning in action ;)
          image_clean = image_clean[image_clean.tweet_id.isin(unique_tweets)]
```

```
In [517]: # agian, creating a list of tweet_ids that unique "unique_tweets" from tweet_clean df and confirming its length
          unique_tweets = list(tweet_clean.tweet_id.unique())

          # Cleaning in action ;)
          api_clean = api_clean[api_clean.tweet_id.isin(unique_tweets)]
```

Test

tweet_clean.shape , image_clean.shape , api_clean.shape

```
((1971, 17), (1971, 12), (1971, 4))
```

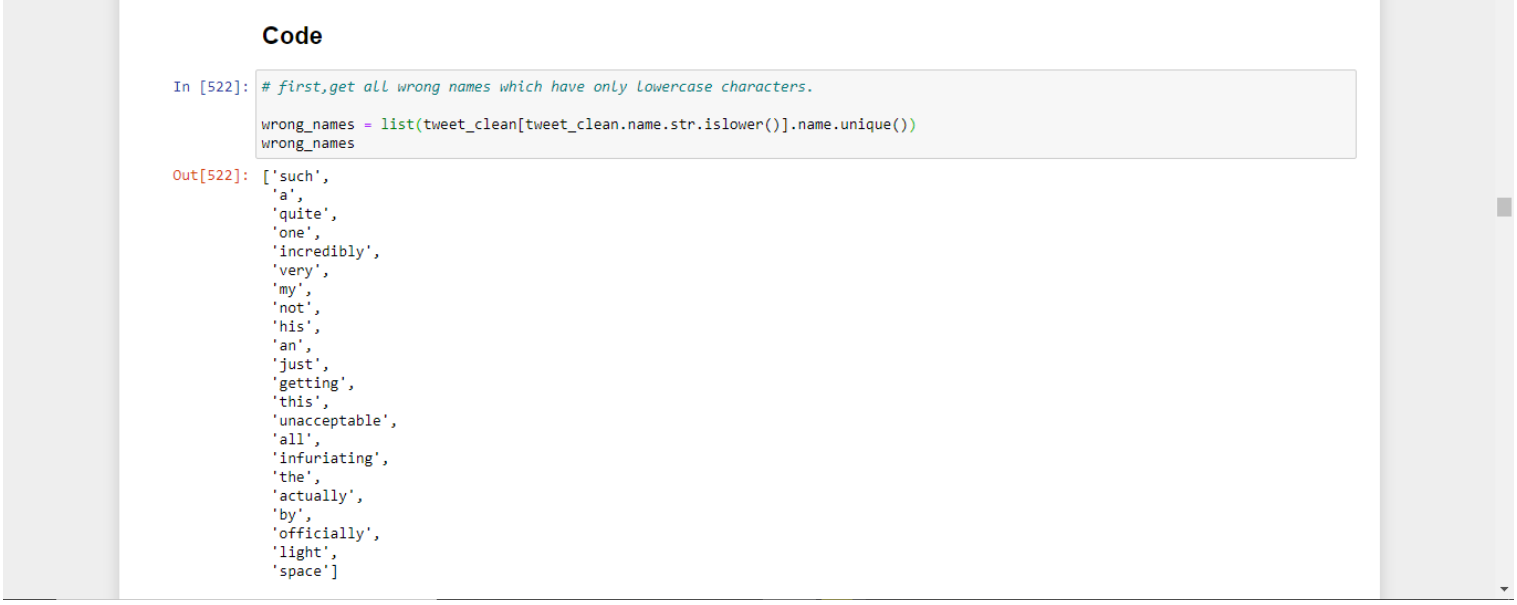
Define

In `archive_df:

- invalid names in name column.
- wrong representation for missing values as 'None'.

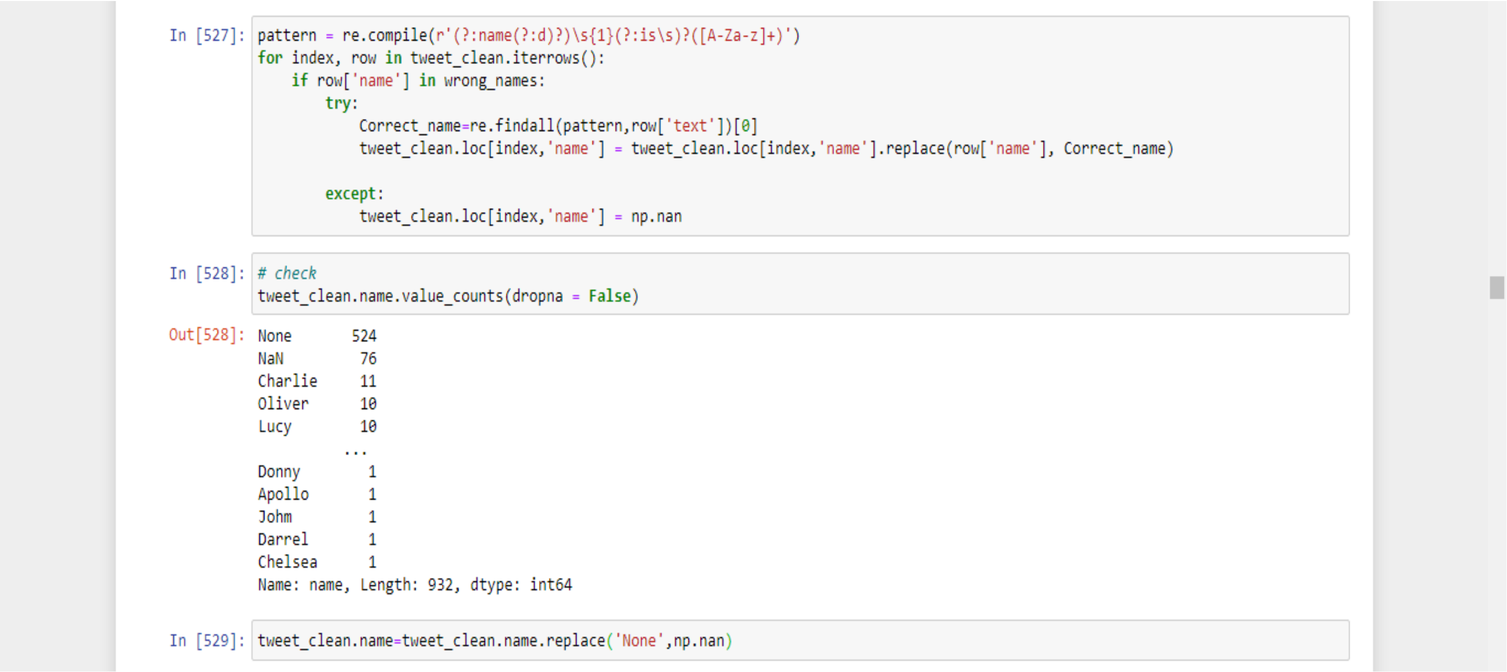
solution:

- try extracting the right name from the text otherwise assign NaN to this value.
- use . replace() to convret them to NaNs.



Code

```
In [522]: # first,get all wrong names which have only lowercase characters.

          wrong_names = list(tweet_clean[tweet_clean.name.str.islower()].name.unique())
          wrong_names

Out[522]: ['such',
           'a',
           'quite',
           'one',
           'incredibly',
           'very',
           'my',
           'not',
           'his',
           'an',
           'just',
           'getting',
           'this',
           'unacceptable',
           'all',
           'infuriating',
           'the',
           'actually',
           'by',
           'officially',
           'light',
           'space']
```



```
In [527]: pattern = re.compile(r'(?:name(?:d)?)\s{1}(?:is\s)?([A-Za-z]+)')
          for index, row in tweet_clean.iterrows():
              if row['name'] in wrong_names:
                  try:
                      Correct_name=re.findall(pattern,row['text'])[0]
                      tweet_clean.loc[index,'name'] = tweet_clean.loc[index,'name'].replace(row['name'], Correct_name)

                  except:
                      tweet_clean.loc[index,'name'] = np.nan

In [528]: # check
          tweet_clean.name.value_counts(dropna = False)

Out[528]: None       524
          NaN         76
          Charlie     11
          Oliver      10
          Lucy        10
                     ...
          Donny        1
          Apollo       1
          Johm         1
          Darrel       1
          Chelsea      1
          Name: name, Length: 932, dtype: int64

In [529]: tweet_clean.name=tweet_clean.name.replace('None',np.nan)
```

Test

tweet_clean.name.value_counts(dropna = False)

```
NaN          600
Charlie       11
Oliver        10
Cooper        10
Lucy          10
            ...
Ralphie        1
Pawnd          1
Harry          1
Goliath        1
Shelby         1
Name: name, Length: 931, dtype: int64
```

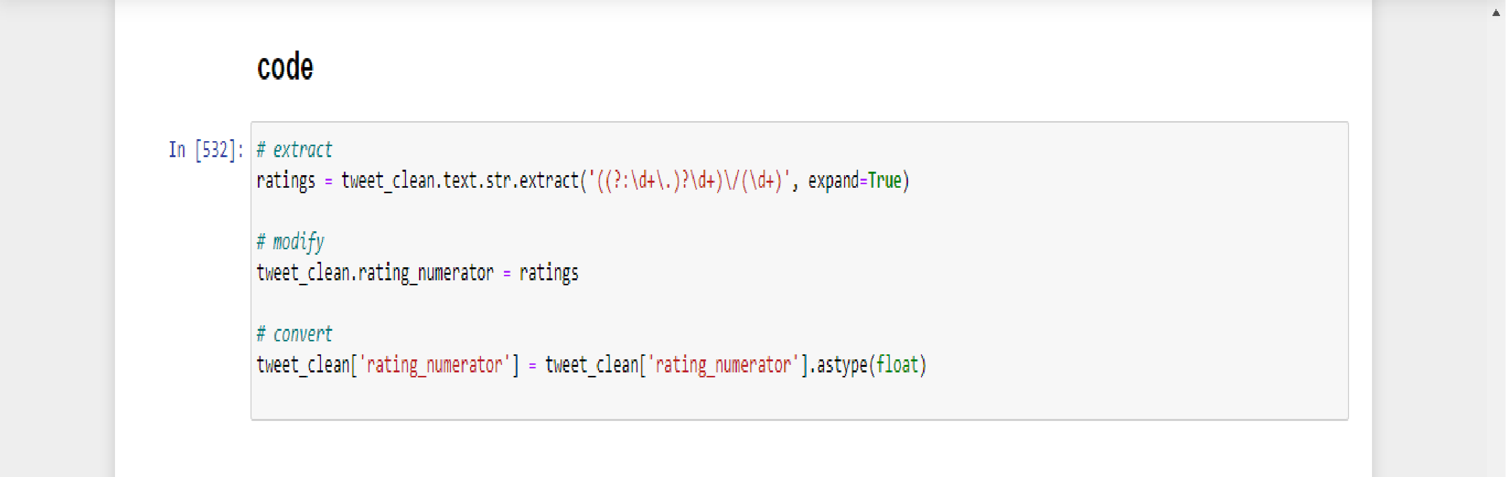
Define

In `archive_df`:

- Ratings with decimal values incorrectly extracted in `rating_numerator` column
- `rating_numerator` dtype is int instead of float.

solution:

- extract the right decimal value using `str.extract( )`.
- convert `rating_numerator` to float using `astype()`.



code

In [532]:

```
# extract
ratings = tweet_clean.text.str.extract('((?:\d+\.)?\d+)\/(\d+)', expand=True)

# modify
tweet_clean.rating_numerator = ratings

# convert
tweet_clean['rating_numerator'] = tweet_clean['rating_numerator'].astype(float)
```

Test

tweet_clean.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
```

```
---  ------                      --------------  -----
 0   tweet_id                    1971 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                   1971 non-null   object
 4   source                      1971 non-null   object
 5   text                        1971 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               1971 non-null   object
 10  rating_numerator            1971 non-null   float64
 11  rating_denominator          1971 non-null   int64
 12  name                        1371 non-null   object
 13  doggo                       1971 non-null   object
 14  floofer                     1971 non-null   object
 15  pupper                      1971 non-null   object
 16  puppo                       1971 non-null   object
dtypes: float64(5), int64(2), object(10)
memory usage: 357.2+ KB
```

```
tweet_clean['rating_numerator'].unique()
```

```
array([1.300e+01, 1.200e+01, 1.400e+01, 1.350e+01, 1.100e+01, 6.000e+00
,
       1.000e+01, 0.000e+00, 8.400e+01, 2.400e+01, 9.750e+00, 5.000e+00
,
       1.127e+01, 3.000e+00, 7.000e+00, 8.000e+00, 9.000e+00, 4.000e+00
,
       1.650e+02, 1.776e+03, 2.040e+02, 5.000e+01, 9.900e+01, 8.000e+01
,
       4.500e+01, 6.000e+01, 4.400e+01, 1.210e+02, 1.126e+01, 2.000e+00
,
       1.440e+02, 8.800e+01, 1.000e+00, 4.200e+02])
```

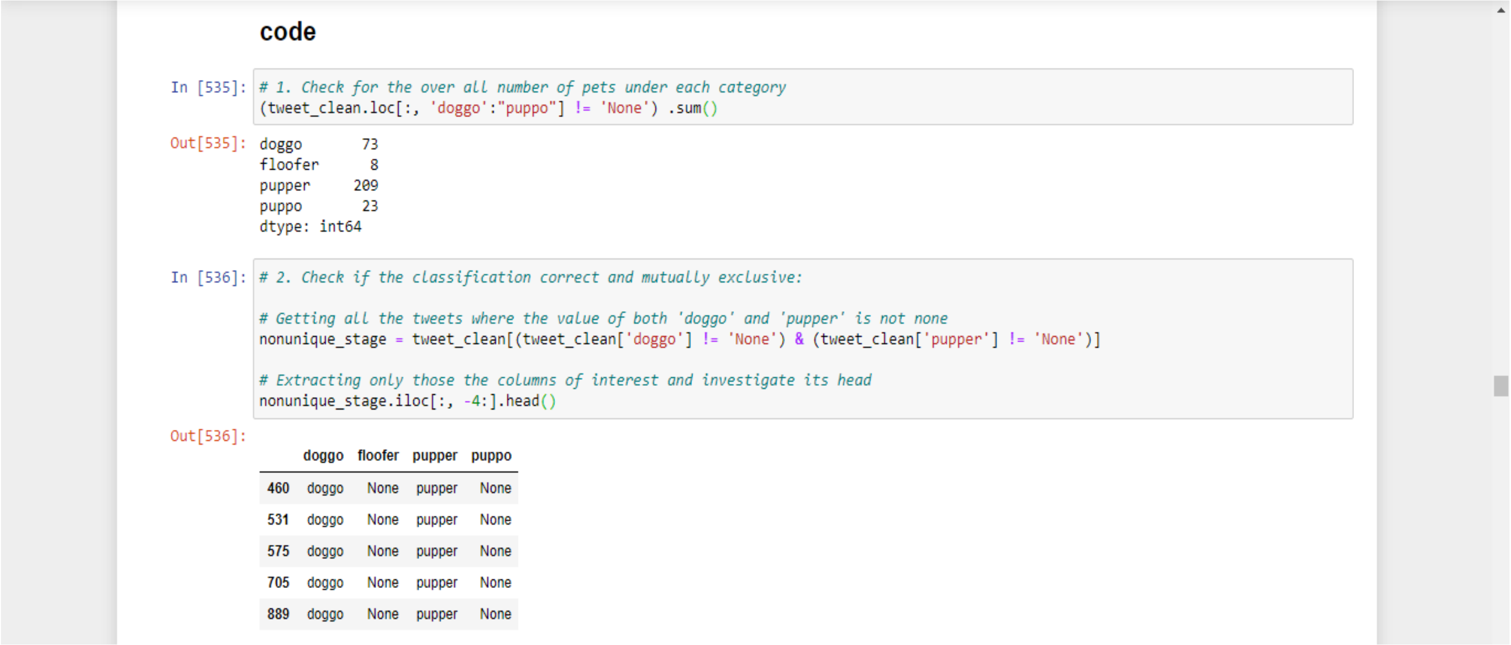
Define

In `archive_df`:

- `NaN` is mistakenly written as **None** in: `doggo`,`floofer`,`pupper`, and `puppo`.
- doggo,floofer,pupper, and puppo should all be one column called e.g.dog_stage.

Solution

- merge the last 4 columns to create new column called `dog_stage` (further investegation needed)

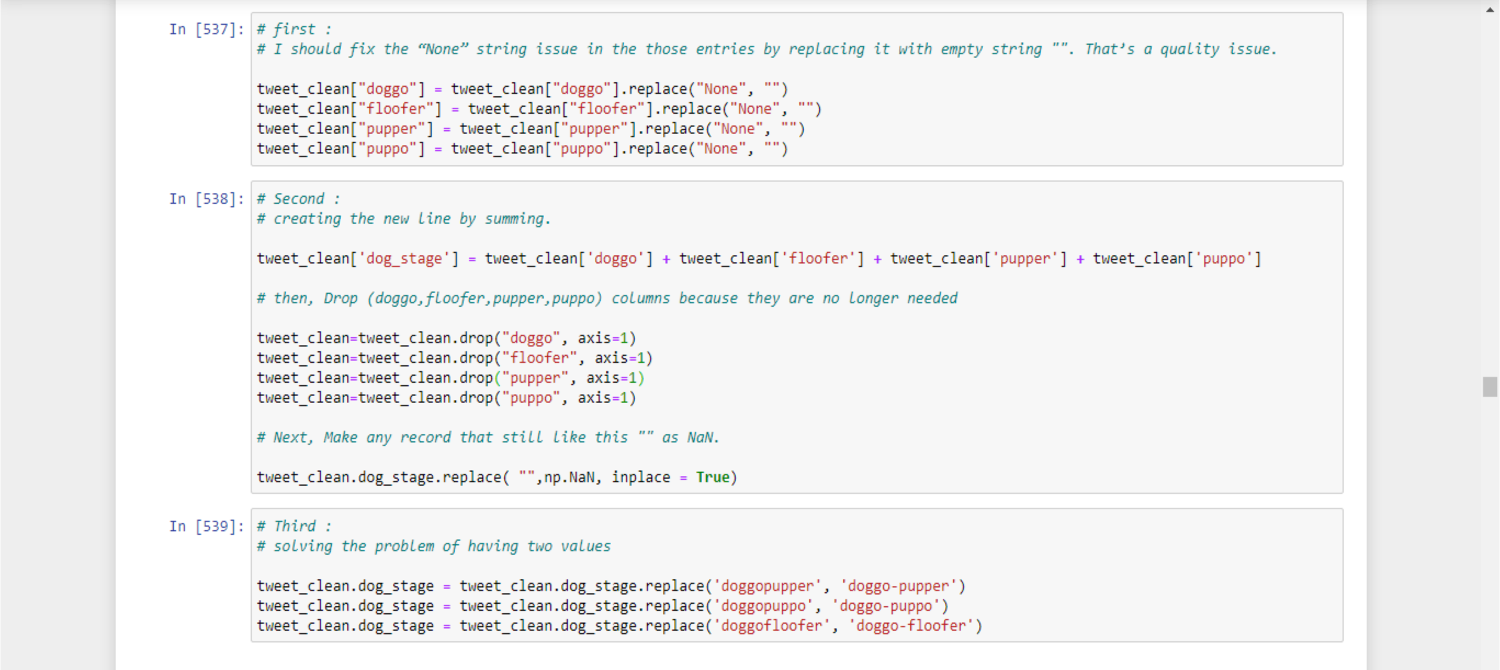


code

```
In [535]: # 1. Check for the over all number of pets under each category
          (tweet_clean.loc[:, 'doggo':"puppo"] != 'None') .sum()

Out[535]: doggo       73
          floofer      8
          pupper     209
          puppo       23
          dtype: int64
```

```
In [536]: # 2. Check if the classification correct and mutually exclusive:

          # Getting all the tweets where the value of both 'doggo' and 'pupper' is not none
          nonunique_stage = tweet_clean[(tweet_clean['doggo'] != 'None') & (tweet_clean['pupper'] != 'None')]

          # Extracting only those the columns of interest and investigate its head
          nonunique_stage.iloc[:, -4:].head()

Out[536]:
```

	doggo	floofer	pupper	puppo
460	doggo	None	pupper	None
531	doggo	None	pupper	None
575	doggo	None	pupper	None
705	doggo	None	pupper	None
889	doggo	None	pupper	None



```
In [537]: # first :
          # I should fix the “None” string issue in the those entries by replacing it with empty string "". That’s a quality issue.

          tweet_clean["doggo"] = tweet_clean["doggo"].replace("None", "")
          tweet_clean["floofer"] = tweet_clean["floofer"].replace("None", "")
          tweet_clean["pupper"] = tweet_clean["pupper"].replace("None", "")
          tweet_clean["puppo"] = tweet_clean["puppo"].replace("None", "")
```

```
In [538]: # Second :
          # creating the new line by summing.

          tweet_clean['dog_stage'] = tweet_clean['doggo'] + tweet_clean['floofer'] + tweet_clean['pupper'] + tweet_clean['puppo']

          # then, Drop (doggo,floofer,pupper,puppo) columns because they are no longer needed

          tweet_clean=tweet_clean.drop("doggo", axis=1)
          tweet_clean=tweet_clean.drop("floofer", axis=1)
          tweet_clean=tweet_clean.drop("pupper", axis=1)
          tweet_clean=tweet_clean.drop("puppo", axis=1)

          # Next, Make any record that still like this "" as NaN.

          tweet_clean.dog_stage.replace( "",np.NaN, inplace = True)
```

```
In [539]: # Third :
          # solving the problem of having two values

          tweet_clean.dog_stage = tweet_clean.dog_stage.replace('doggopupper', 'doggo-pupper')
          tweet_clean.dog_stage = tweet_clean.dog_stage.replace('doggopuppo', 'doggo-puppo')
          tweet_clean.dog_stage = tweet_clean.dog_stage.replace('doggofloofer', 'doggo-floofer')
```

Test

tweet_clean.dog_stage.value_counts(dropna=False)

```
NaN                1668
pupper              201
doggo                63
puppo                22
doggo-pupper          8
floofer               7
doggo-floofer         1
doggo-puppo           1
Name: dog_stage, dtype: int64
```

Define

In `archive_df`:

- `tweet_id` type is **integer** instead of **str**.
- `in_reply_to_status_id` type is **float** instead of **str**.
- `in_reply_to_user_id` type is **float** instead of **str**.
- `timestamp` type is **str** instead of **DateTime**.
- `source` type is **str** instead of **category**.
- `retweeted_status_id` type is **float** instead of **str**.
- `retweeted_status_user_id` type is **float** instead of **str**.
- `retweeted_status_timestamp` type is **str** instead of **DateTime**.

In `image_predictions_df`:

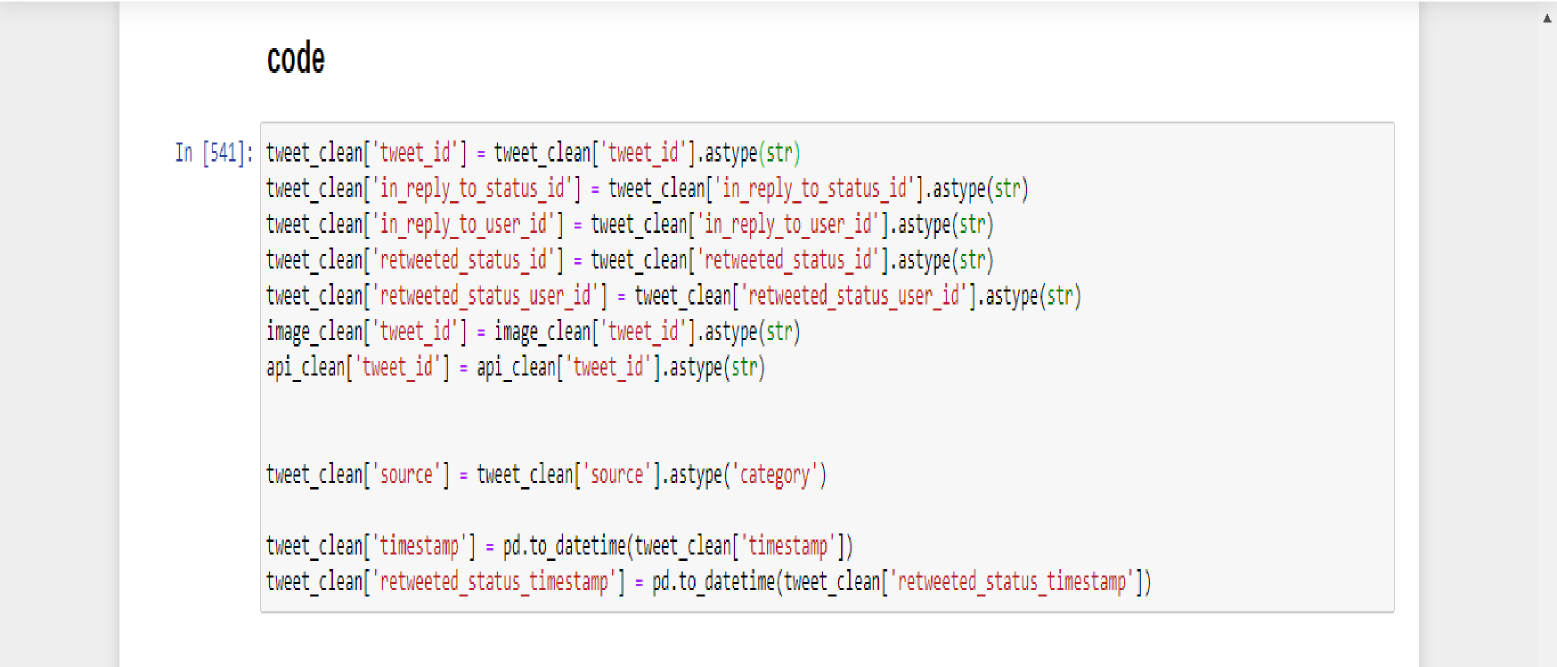
- `tweet_id` type is **integer** instead of **str**.

In `api_df`:

- `tweet_id` type is **integer** instead of **str**

Solution:

- Change values to `str` using `.astype()` method.
- Change values to `category` using `.astype()` method.
- Change values to `datetime` using `pd.to_datetime()` method.



code

In [541]:

```
tweet_clean['tweet_id'] = tweet_clean['tweet_id'].astype(str)
tweet_clean['in_reply_to_status_id'] = tweet_clean['in_reply_to_status_id'].astype(str)
tweet_clean['in_reply_to_user_id'] = tweet_clean['in_reply_to_user_id'].astype(str)
tweet_clean['retweeted_status_id'] = tweet_clean['retweeted_status_id'].astype(str)
tweet_clean['retweeted_status_user_id'] = tweet_clean['retweeted_status_user_id'].astype(str)
image_clean['tweet_id'] = image_clean['tweet_id'].astype(str)
api_clean['tweet_id'] = api_clean['tweet_id'].astype(str)


tweet_clean['source'] = tweet_clean['source'].astype('category')

tweet_clean['timestamp'] = pd.to_datetime(tweet_clean['timestamp'])
tweet_clean['retweeted_status_timestamp'] = pd.to_datetime(tweet_clean['retweeted_status_timestamp'])
```

Test

```
# test 1

assert tweet_clean['tweet_id'].dtype == 'O'

assert tweet_clean['in_reply_to_status_id'].dtype  == 'O'

assert tweet_clean['in_reply_to_user_id'].dtype  == 'O'

assert tweet_clean['retweeted_status_id'].dtype  == 'O'

assert tweet_clean['retweeted_status_user_id'].dtype  == 'O'

assert image_clean['tweet_id'].dtype  == 'O'

assert api_clean['tweet_id'].dtype  == 'O'



assert tweet_clean['source'].dtype == 'category'

# test 2

tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 2355
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    1971 non-null   object
 1   in_reply_to_status_id       1971 non-null   object
 2   in_reply_to_user_id         1971 non-null   object
 3   timestamp                   1971 non-null   datetime64[ns, UTC]
 4   source                      1971 non-null   category
 5   text                        1971 non-null   object
 6   retweeted_status_id         1971 non-null   object
 7   retweeted_status_user_id    1971 non-null   object
 8   retweeted_status_timestamp  0 non-null      datetime64[ns]
 9   expanded_urls               1971 non-null   object
 10  rating_numerator            1971 non-null   float64
 11  rating_denominator          1971 non-null   int64
 12  name                        1371 non-null   object
 13  dog_stage                   303 non-null    object
dtypes: category(1), datetime64[ns, UTC](1), datetime64[ns](1), float64
(1), int64(1), object(9)
memory usage: 297.6+ KB
```

```
# test 3

image_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   tweet_id  1971 non-null   object
 1   jpg_url   1971 non-null   object
 2   img_num   1971 non-null   int64
 3   p1        1971 non-null   object
 4   p1_conf   1971 non-null   float64
 5   p1_dog    1971 non-null   bool
 6   p2        1971 non-null   object
 7   p2_conf   1971 non-null   float64
 8   p2_dog    1971 non-null   bool
 9   p3        1971 non-null   object
 10  p3_conf   1971 non-null   float64
 11  p3_dog    1971 non-null   bool
dtypes: bool(3), float64(3), int64(1), object(5)
memory usage: 159.8+ KB
```

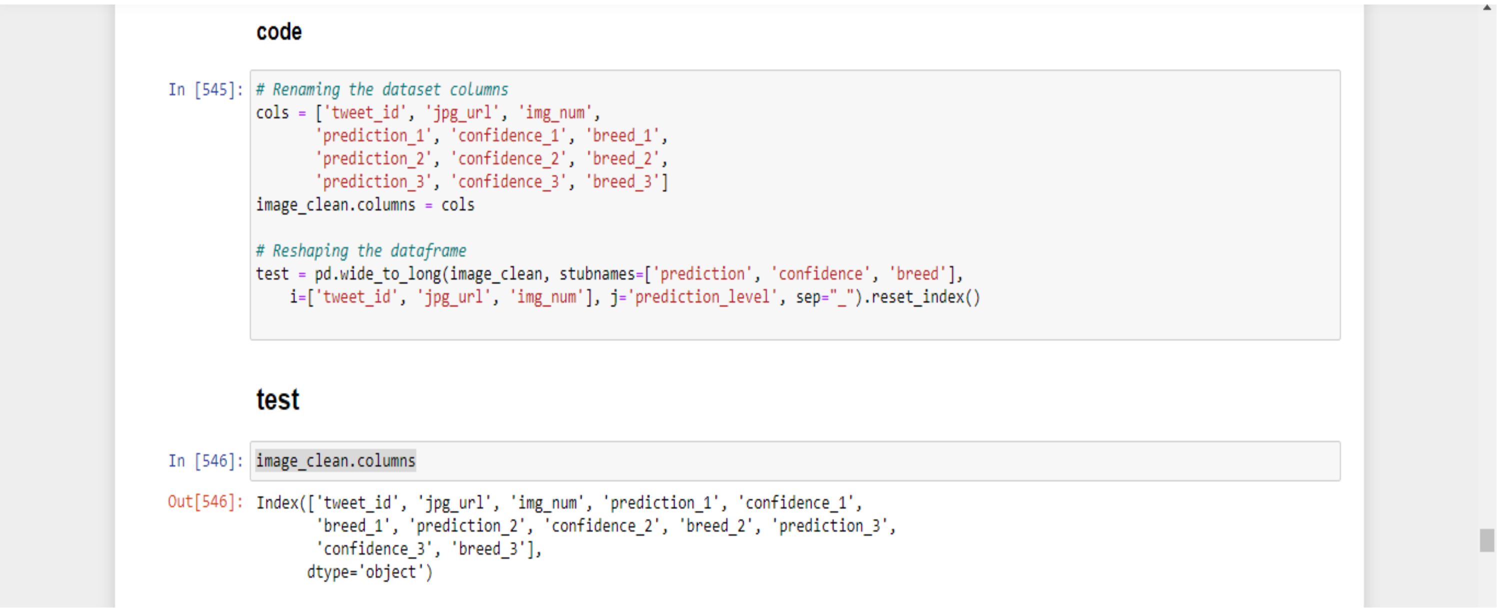
Define

In `image_predictions_df`:

- columns headers are values not variable

solution:

- change columns headers using `pd.wide_to_long()`



code

In [545]:
```python
# Renaming the dataset columns
cols = ['tweet_id', 'jpg_url', 'img_num',
       'prediction_1', 'confidence_1', 'breed_1',
       'prediction_2', 'confidence_2', 'breed_2',
       'prediction_3', 'confidence_3', 'breed_3']
image_clean.columns = cols

# Reshaping the dataframe
test = pd.wide_to_long(image_clean, stubnames=['prediction', 'confidence', 'breed'],
    i=['tweet_id', 'jpg_url', 'img_num'], j='prediction_level', sep="_").reset_index()
```

test

In [546]:
```python
image_clean.columns
```

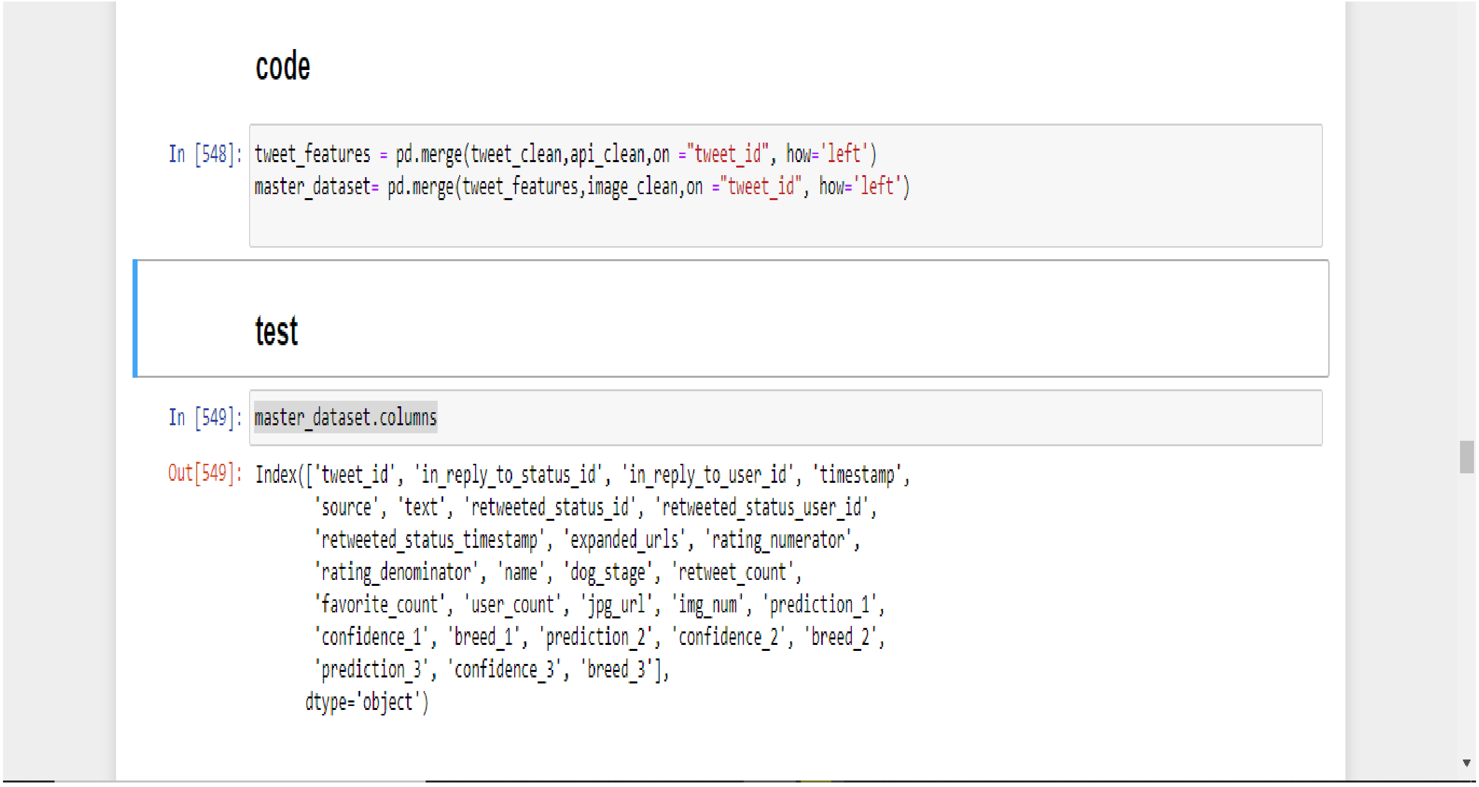
Out[546]:
```
Index(['tweet_id', 'jpg_url', 'img_num', 'prediction_1', 'confidence_1',
       'breed_1', 'prediction_2', 'confidence_2', 'breed_2', 'prediction_3',
       'confidence_3', 'breed_3'],
      dtype='object')
```

Define

- Data from the 3 datasets (`archive_df`, `image_predictions_df`, and `api_df`) can be combined in one DataFrame for simplicity.

Solution

- while our 3 data sets have the same number of record we can use `pd.merge()`



code

```
In [548]: tweet_features = pd.merge(tweet_clean,api_clean,on ="tweet_id", how='left')
          master_dataset= pd.merge(tweet_features,image_clean,on ="tweet_id", how='left')
```

test

```
In [549]: master_dataset.columns

Out[549]: Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
                 'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
                 'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
                 'rating_denominator', 'name', 'dog_stage', 'retweet_count',
                 'favorite_count', 'user_count', 'jpg_url', 'img_num', 'prediction_1',
                 'confidence_1', 'breed_1', 'prediction_2', 'confidence_2', 'breed_2',
                 'prediction_3', 'confidence_3', 'breed_3'],
                dtype='object')
```