Wrangle Report

**A. Gathering**

> We utilise 3 datasets, but here, the first task comes in.

- dataset 01: 'twitter\_archive\_enhanced.csv' (I named it ‘arch’)

- dataset 02: 'image\_predictions.tsv' (I named it ‘image\_p’

- dataset 03: **We need to obtain some additional information: 1)‘favotite count’ and 2)‘retweet\_count’ by querying the Twitter API.**(I named it ‘tw\_json’)

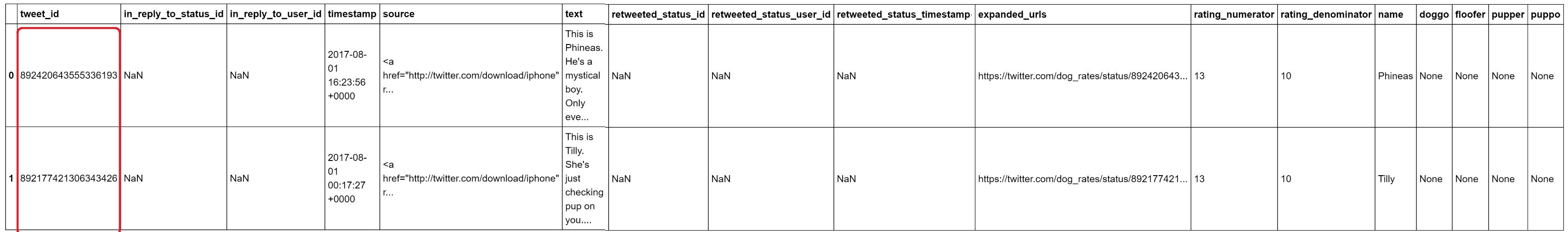
>Task\_01: Collect ‘favorite\_count’ and ‘retweet\_count’

- First, after signing up twitter account, we filled out the application form, which is followed by authentication of use for ‘the Consumer Key’, ‘Consumer Secret’, ‘Access Token’ and ‘Access Token Secret’. Then we were able to create ‘authentication object’ as follows.

 - Next, using this ‘authentication object’, we could access specific tweets by ‘their id’ when querying the Tweeter API.

- We already had ‘their id’ to use, and took a look at them from one of our given dataset-'twitter\_archive\_enhanced.csv'.

- Here, via ‘tweet\_id’ using Tweepy library, we could get tweet JSON data.

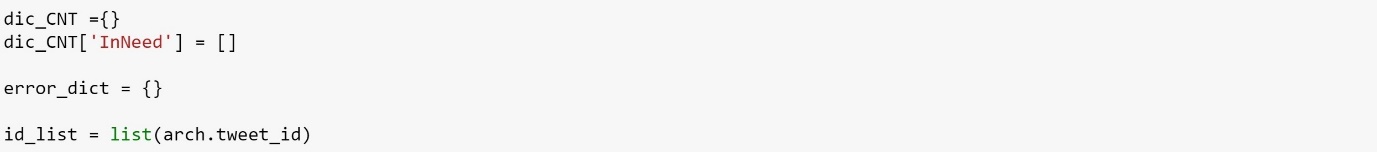


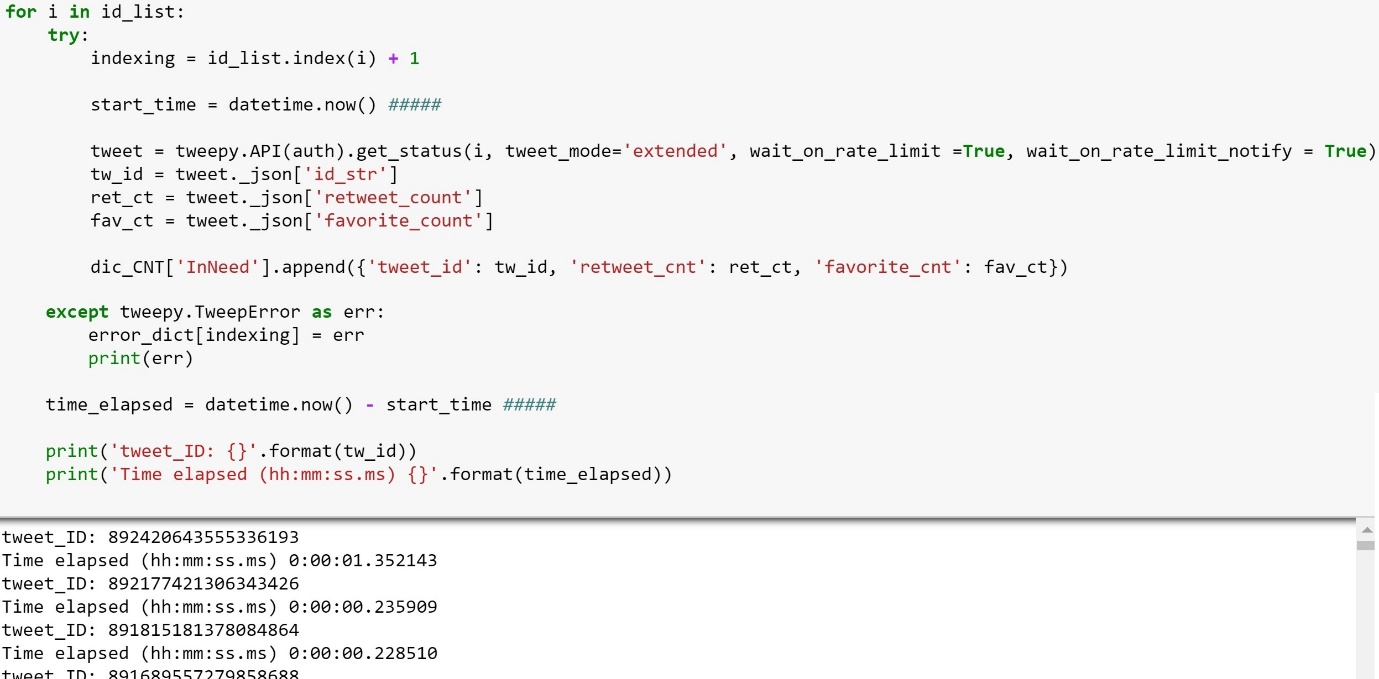
- From the instruction, there are some IDs in which tweet does not exist and it would result in some errors. We need to capture and handle that. This gives us a hint that we need to use **‘try ~ except’** statement.

- We begin our codes with setting up two empty dictionaries, the one for storing our extracted data (tweet\_id, favorite\_count, retweet\_count), and the other for the error.

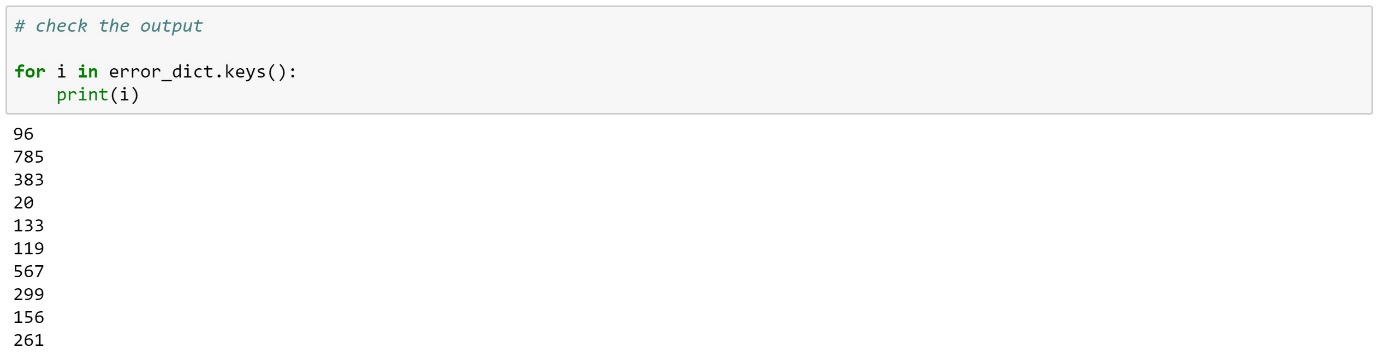
- ‘dic\_CNT’

- ‘error\_dict’

I did care the indexing of the erroneous records to be stored in ‘error\_dict’. So I converted a series of ‘tweet\_id’ into a list.



And we can check those erroneous records stored in ‘error\_dict’.

Now we write and read our json data.



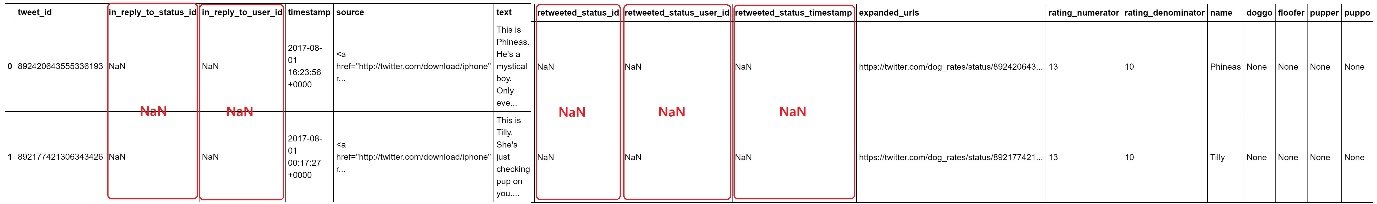
**B. Assessing**

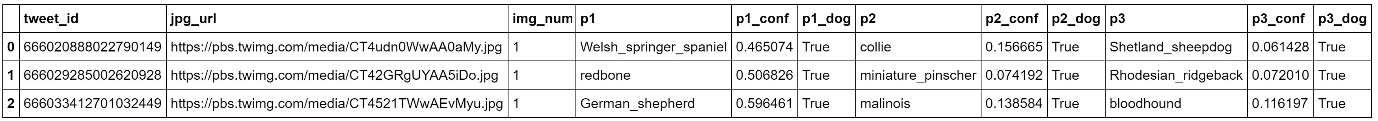
> We have 3 datasets.

- dataset 01: 'twitter\_archive\_enhanced.csv' (I named it ‘arch’)

- dataset 02: 'image\_predictions.tsv' (I named it ‘image\_p’

- dataset 03: **‘**tweet\_json.txt’(I named it ‘tw\_json’)







> We found so many problems in each.

**In ‘arch’ dataset:**

--------------------------------------------------------------------------------------------------------------------------------------

- As Quality Issues:

(a)\_‘tweet\_id’ value was an integer,

(b)\_Each ‘timestamp’ value was just a string,

(c)\_There were lots of inappropriate name, missing names in ’name’ column.

- 'None'?, 'a'?, 'an'?, 'very'?, 'the'?, 'not'? 'quite'? 'just'? 'mad'? 'actually'?

(d)\_’rating\_numerator' and 'rating\_denominator', values were confusing as well. From the instruction, each value in ‘rating\_denominator’ can be ‘10’ altogether.

(e)\_ when taking a look at the numerator - in the text field, there seems to be fractional numerators, and it does not match up to the value in the 'rating\_numerator' field.

- As Tidiness Issues:

(a)\_there are columns with too many NaN.

- in\_reply\_to\_status\_id 2278 / 2356

- in\_reply\_to\_user\_id 2278 / 2356

- retweeted\_status\_id 2175 / 2356

- retweeted\_status\_user\_id 2175 / 2356

- retweeted\_status\_timestamp 2175 / 2356

(b)\_**Prior to this, we should discard all rows which represent retweets.**

- retweeted\_status\_id 181 non-null float64

- retweeted\_status\_user\_id 181 non-null float64

- retweeted\_status\_timestamp 181 non-null object

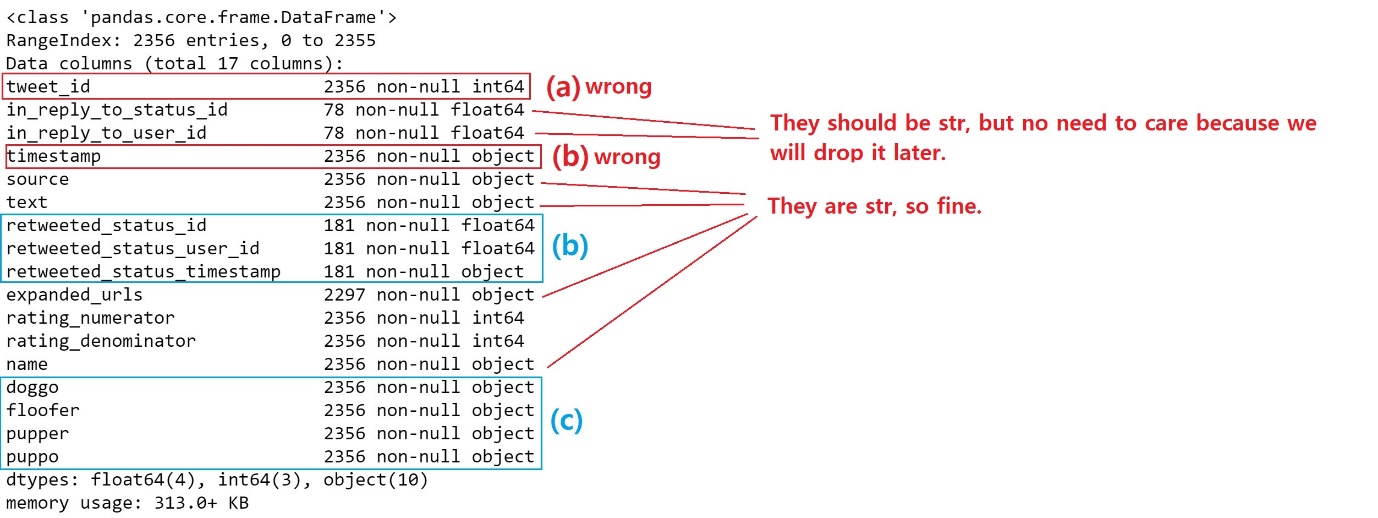
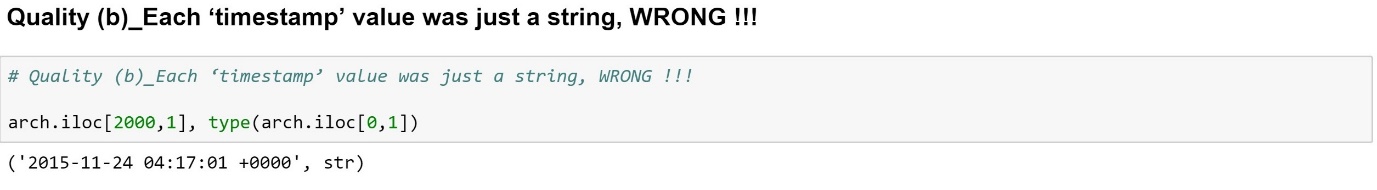
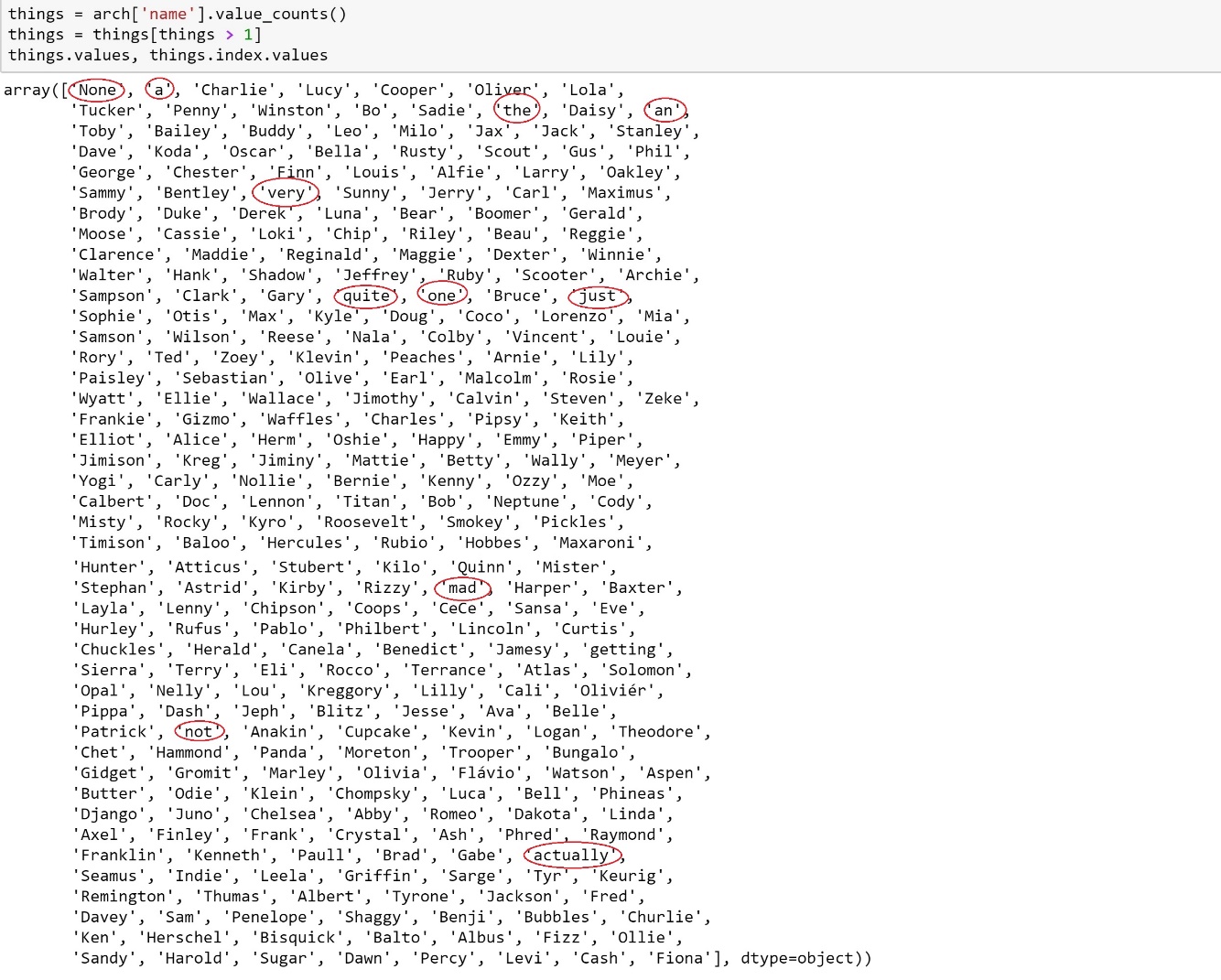
(c)\_Its wide format seems problematic.

- 'doggo, floofer, pupper, puppo' these 4 variables should be combined into one categorical variable which would make it easy to use groupby() method.

(d)\_In 'source' column with wrong format, some unnecessary characters should be stripped off.

--------------------------------------------------------------------------------------------------------------------------------------

**#Quality (a),(b)**

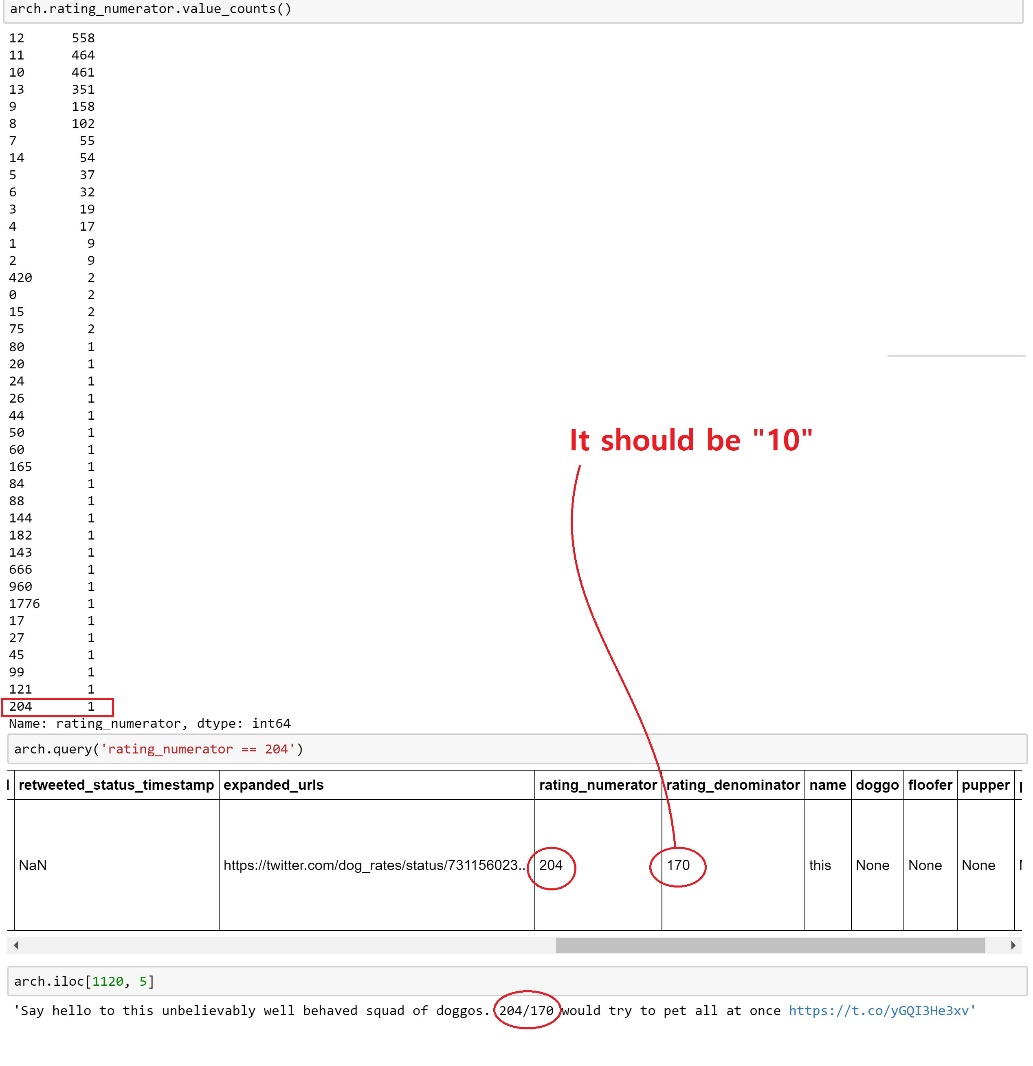
#**Quality (c)**In ‘name’ column, 'None'?, 'a'?, 'an'?, 'very'?, 'the'?, 'not'? 'quite'? 'just'? 'mad'? 'actually'? do not seem right. We want to investigate this, but rows are collapsed so cannot check all values via ‘value\_counts()’. So we do this. We check each row of these suspected names.

And as we suspected, most of them are not about dogs.



**#Quality (d)**

From the instruction, each value in ‘rating\_denominator’ can be ‘10’ altogether, BUT 'rating numerator' has no limit.

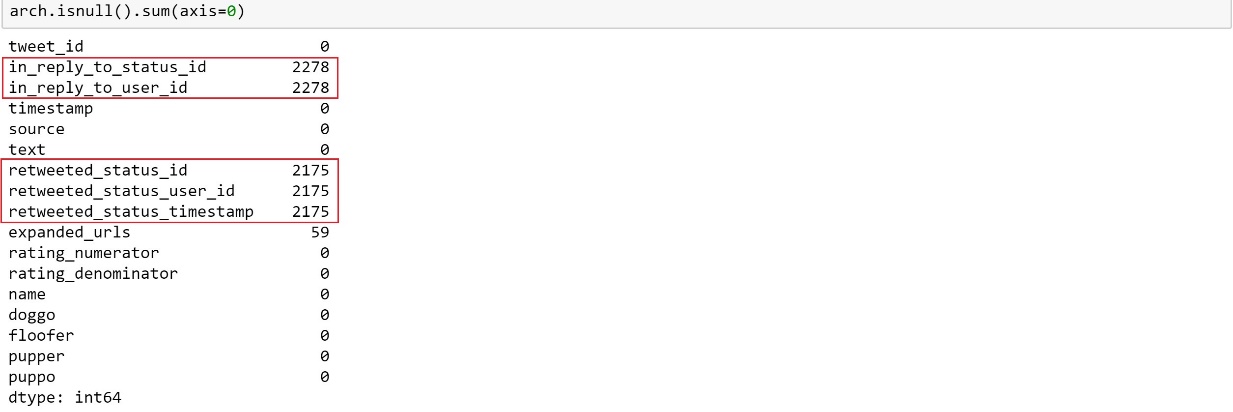


**#Quality (e)**

when taking a look at the numerator - in the text field, there seems to be fractional numerators, and it does not match up to the value in the 'rating\_numerator' field.

**#Tidiness (a)**

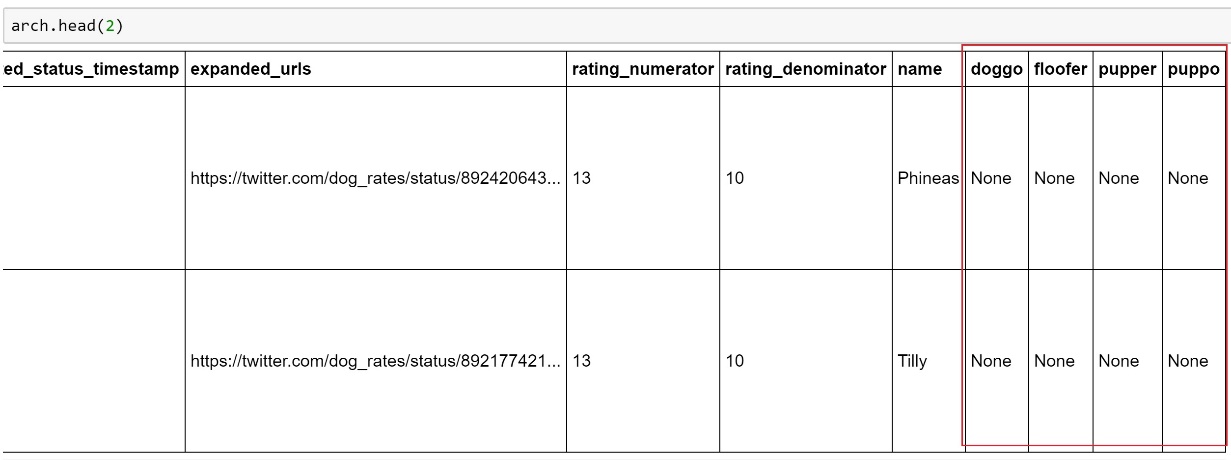
Columns with too many NaN



**#Tidiness (b)**

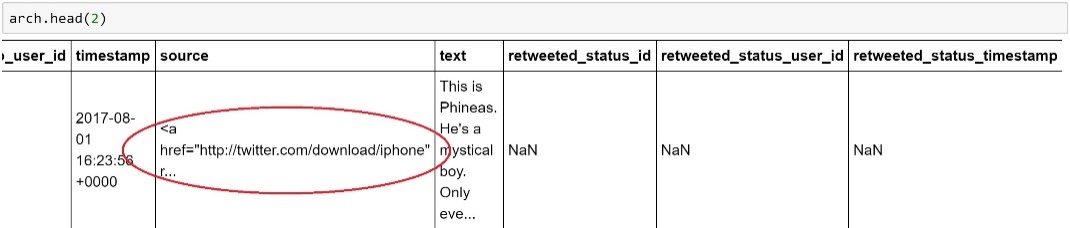
Unnecessary ‘retweet’

**#Tidiness (c)**

'doggo, floofer, pupper, puppo' these 4 variables make ‘groupby()’ hard.

**#Tidiness (d)**

'source' column with messy format.



**In ‘image\_p’ dataset:**

--------------------------------------------------------------------------------------------------------------------------------------

- As Quality Issues:

there are just two simple datatype mis-matchings:

- (a)\_'tweet\_id' should be string

- (b)\_'img\_num' should be string

- As Tidiness Issues:

its wide format seems problematic as well.

- (a)\_'p1, p2, p3' necessary be combined into one categorical variable.

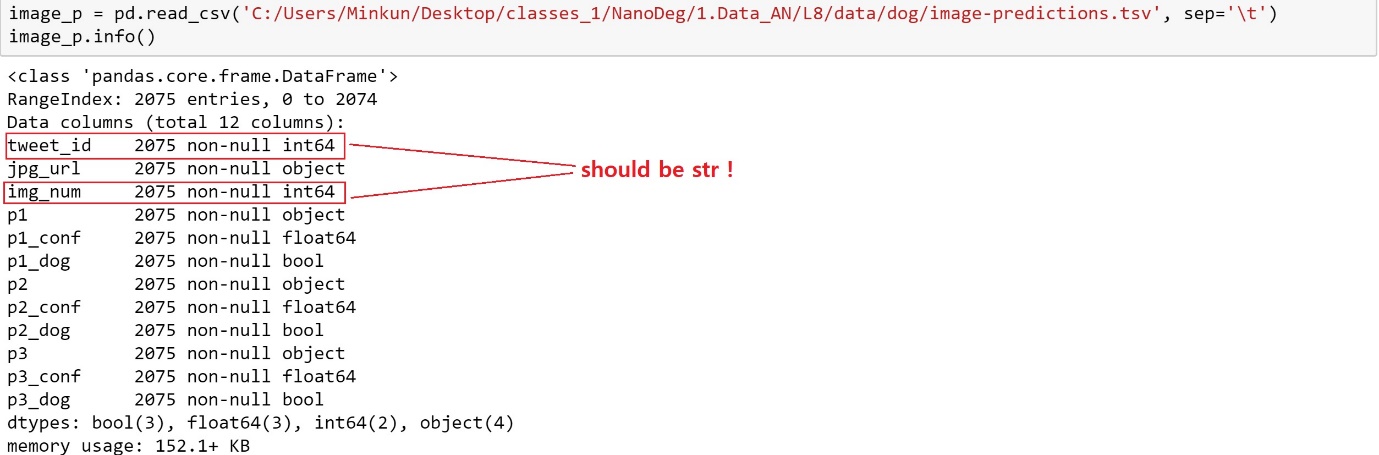
- (b)\_'p1\_conf, p2\_conf, p3\_conf' necessary be combined into one categorical variable.

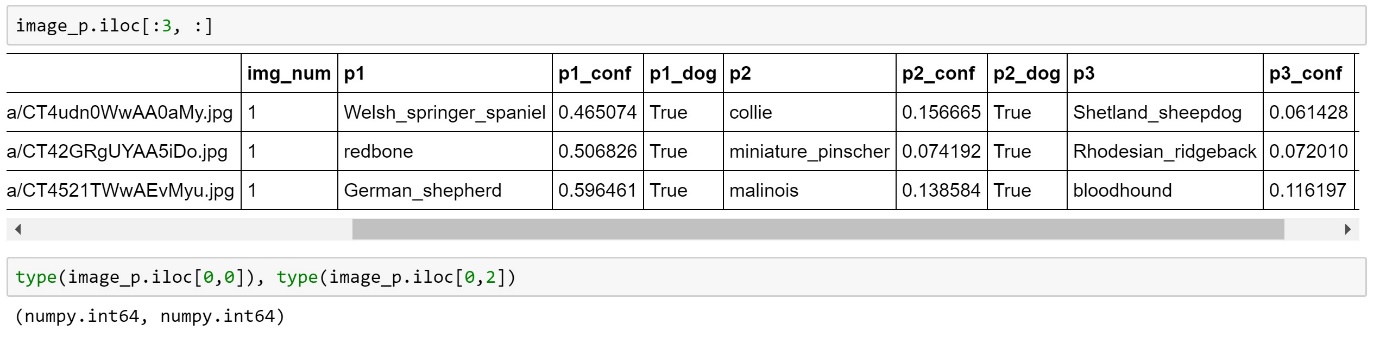
- (c)\_'p1\_dog, p2\_dog, p3\_dog' necessary be combined into one categorical variable.

And we should screen out the rows do not speak about dogs. We decide ‘dog-True' & ' highest probability > 0.8 will survive.

--------------------------------------------------------------------------------------------------------------------------------------

**#Quality (a),(b)**



#**Tidiness (a),(b),(c)**

**In ‘tw\_json’ dataset:**

--------------------------------------------------------------------------------------------------------------------------------------

- As Tidiness Issues:

(a) merging

- archive dataset,

- image dataset,

- tweet\_json dataset

(b) we should deal with duplicates in our master dataset later on.

--------------------------------------------------------------------------------------------------------------------------------------

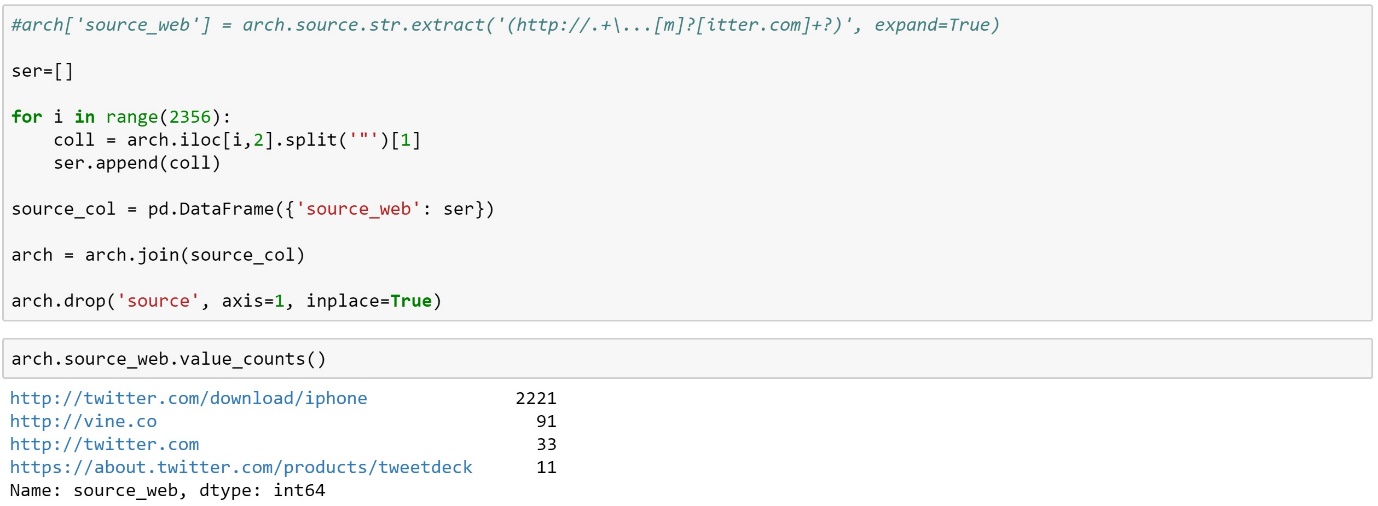
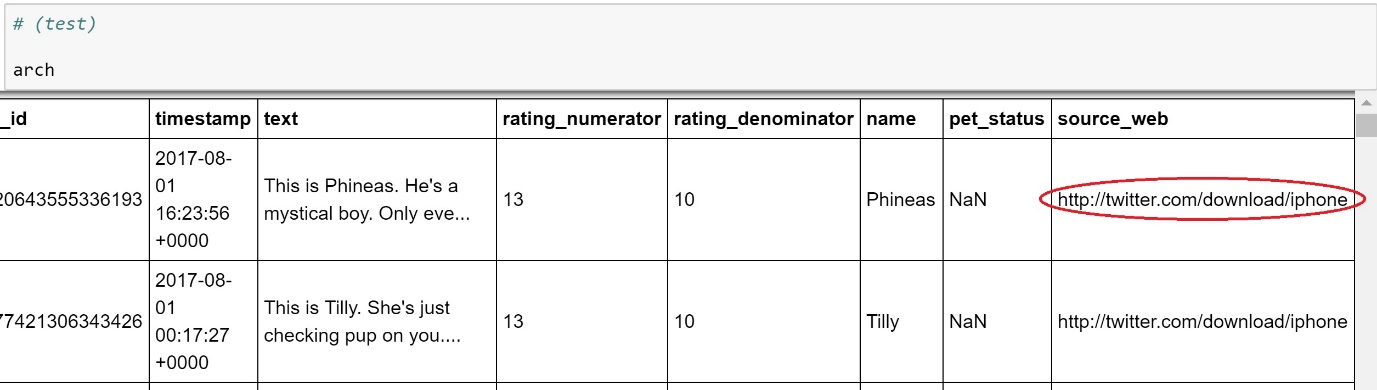
**C. Cleaning**

**[Data 01] - twitter-archive-enhanced.csv**

**# Tidiness(d)**

Define) Clean up the messy website address column.

Code)

test)

**# Tidiness(a)**

Define) Drop the rows with retweets.

Code)

**# Tidiness(b)**

Define) Drop the columns with too many NaN.

Code)

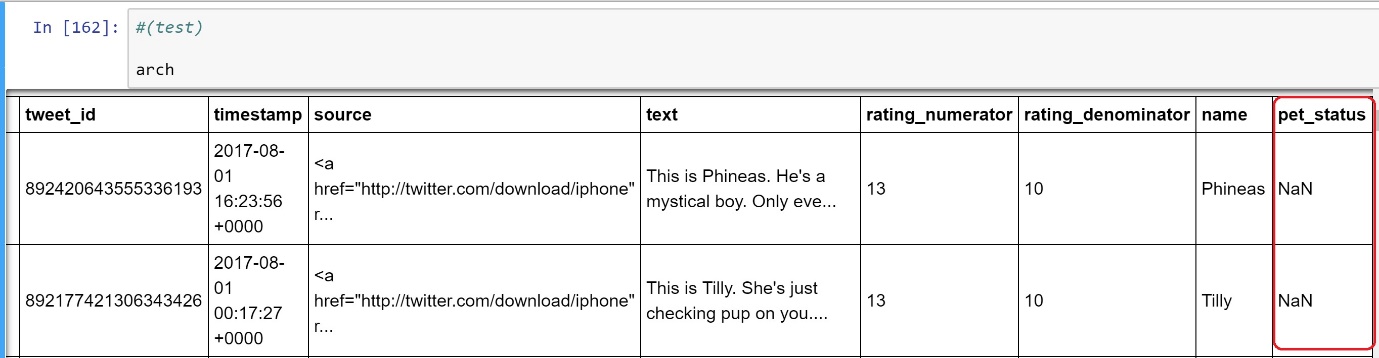


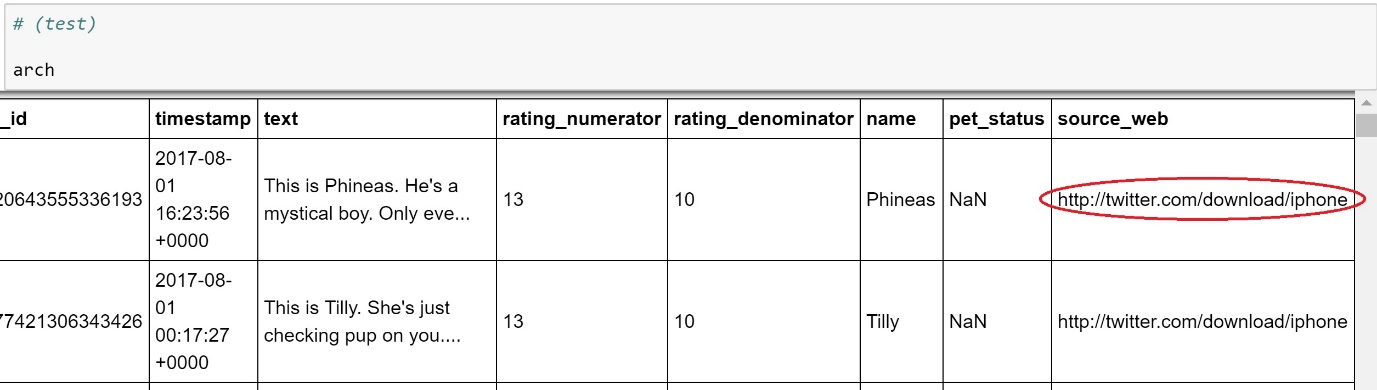
**# Tidiness(c)**

Define) Convert to the long format:

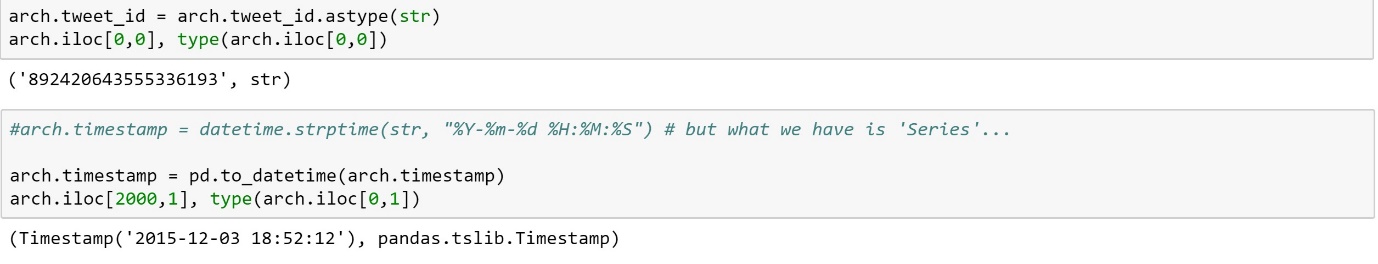
\_In order to combine 'doggo, floofer, pupper, puppo' these 4 variables, what I tried is using ‘lamba function’ and combining all records horizontally within a comma separated format. Then using one of ‘RegExp functions’ - re.sub( ), I removed or replaced all unnecessary patterns, in this case ‘None’, and all commas. What I yielded is a single column called ‘pet\_status’ containing some ‘NaN’ values and 'doggo, floofer, pupper, puppo'.

Code)

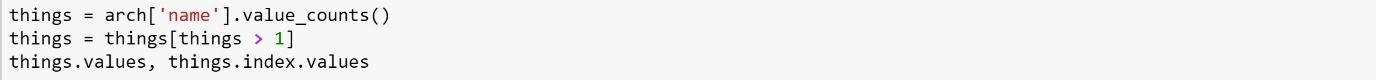
Test)

**# Quality(a),(b)**

Define) Fix the datatype: ‘tweeter\_id’, ‘timestamp’

Code) & test)

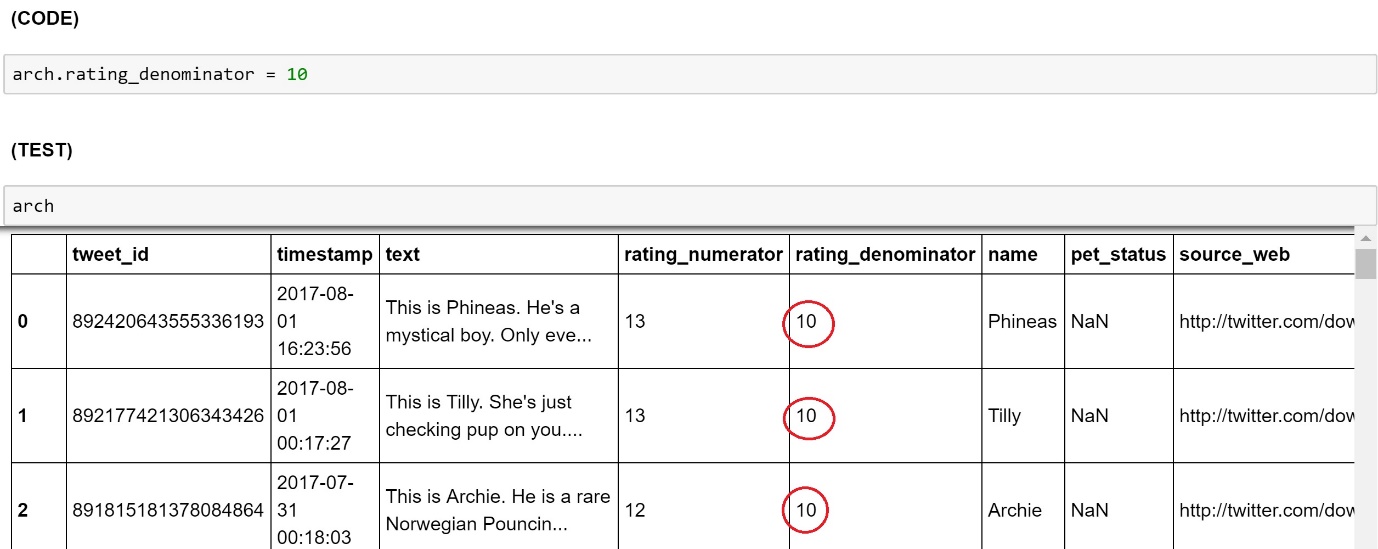
**# Quality(c)**

Define) Fix wrong 'name', inappropriate, duplicated, and missing name values: What was tricky is that when trying to take a look at the names column, lots of rows that show name are collapsed thus we could not check each individual name strings. So except some names present as unique values, 

And 'None', 'a', 'an', 'very', 'the', 'not', 'quite', 'actually' are suspected. We decided to put their name as 'NaN'. We decided this because when bringing up the ‘text’ in accordance with such suspected names, it is clear that those records have nothing to do with ‘dog’. For example,

So we replaced all suspected names with NaNs.

**# Quality(d)**

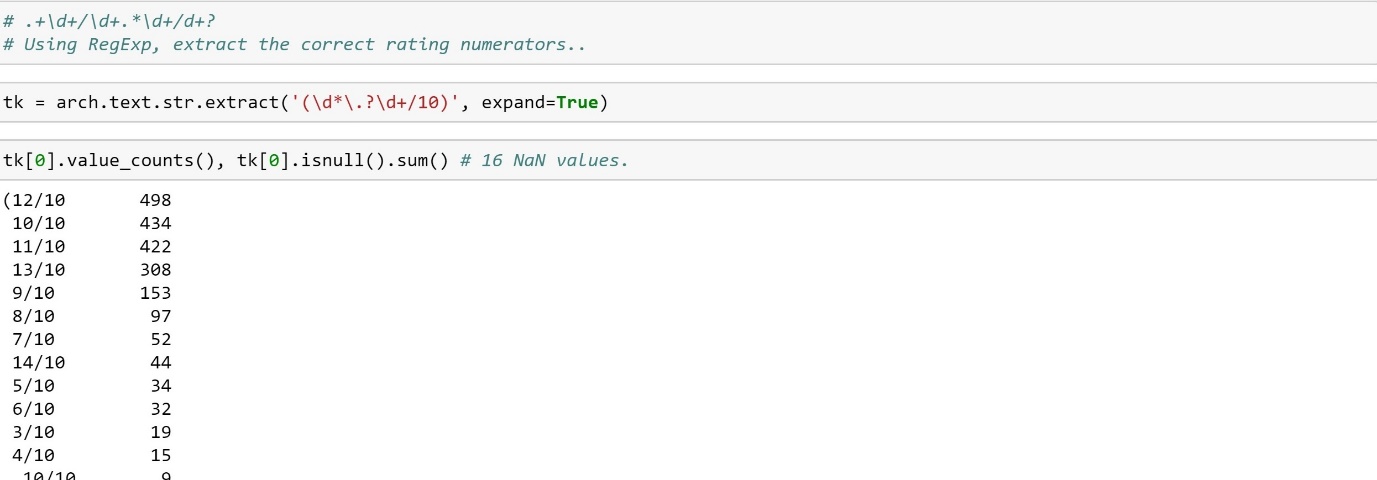
Define) Fix ‘rating\_denominator’ value. It should be 10 consistently. But we decided not to fix ‘rating\_numerator’ values although it shows lots of suspected outliers. This is because we believe there is no limit in rating. 'rating\_denominator = 10'

**# Quality (e)**

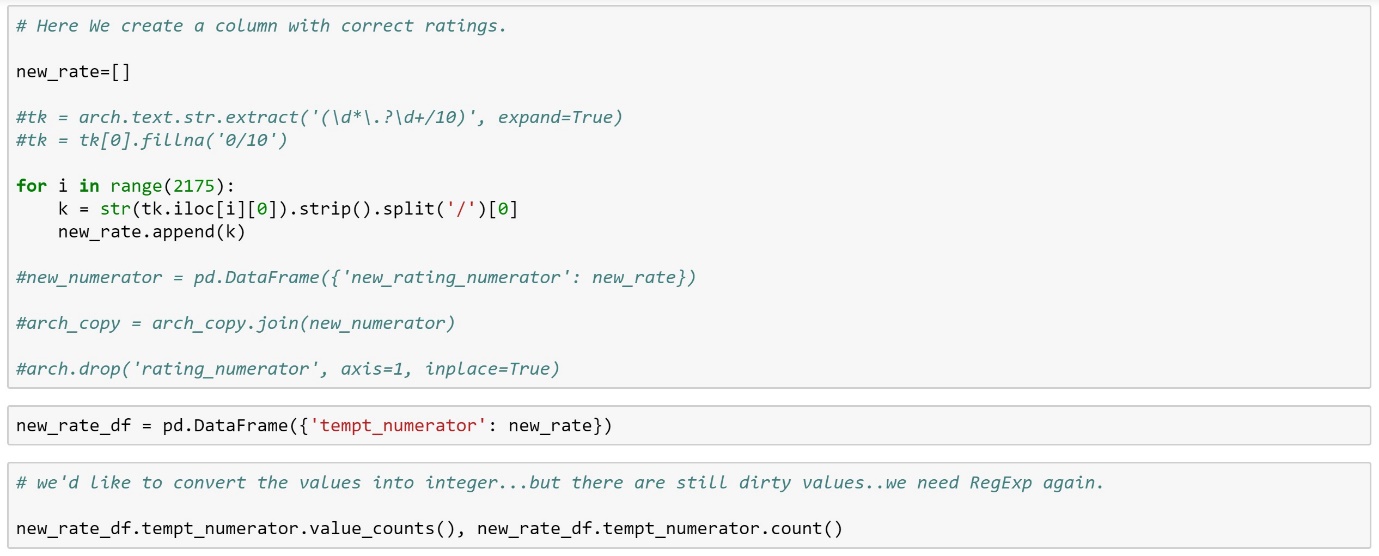
Define)

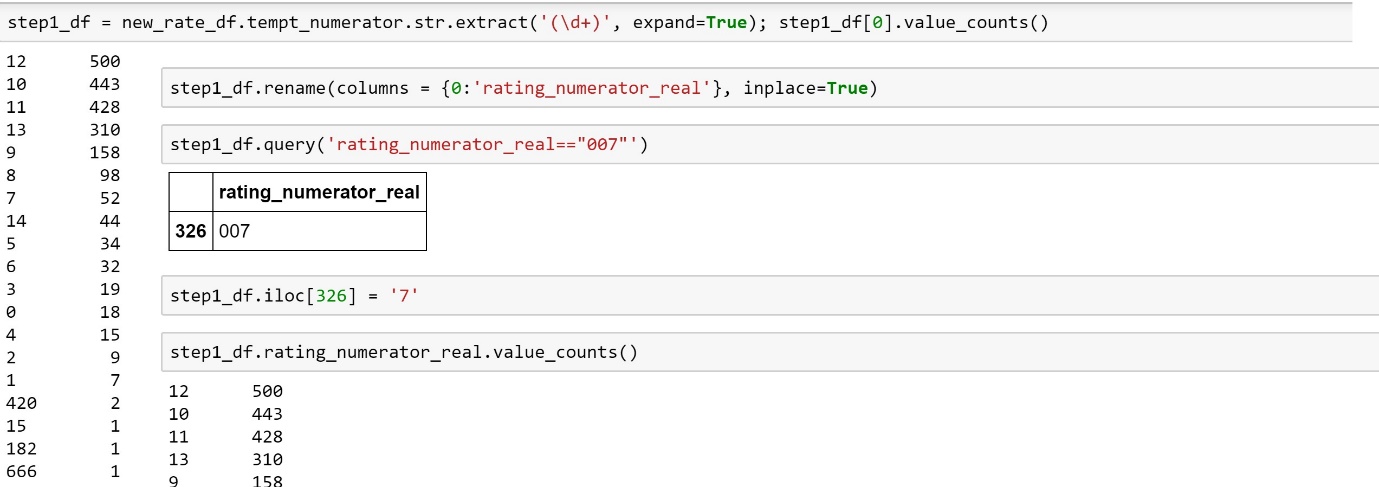
Using RegExp, parse and extract the correct rating numerators in the 'text' field. It seems extremely baffling at the first glance, but we can eventually see the pattern of the real rating numbers…which comes right before ‘/10’.

Code)

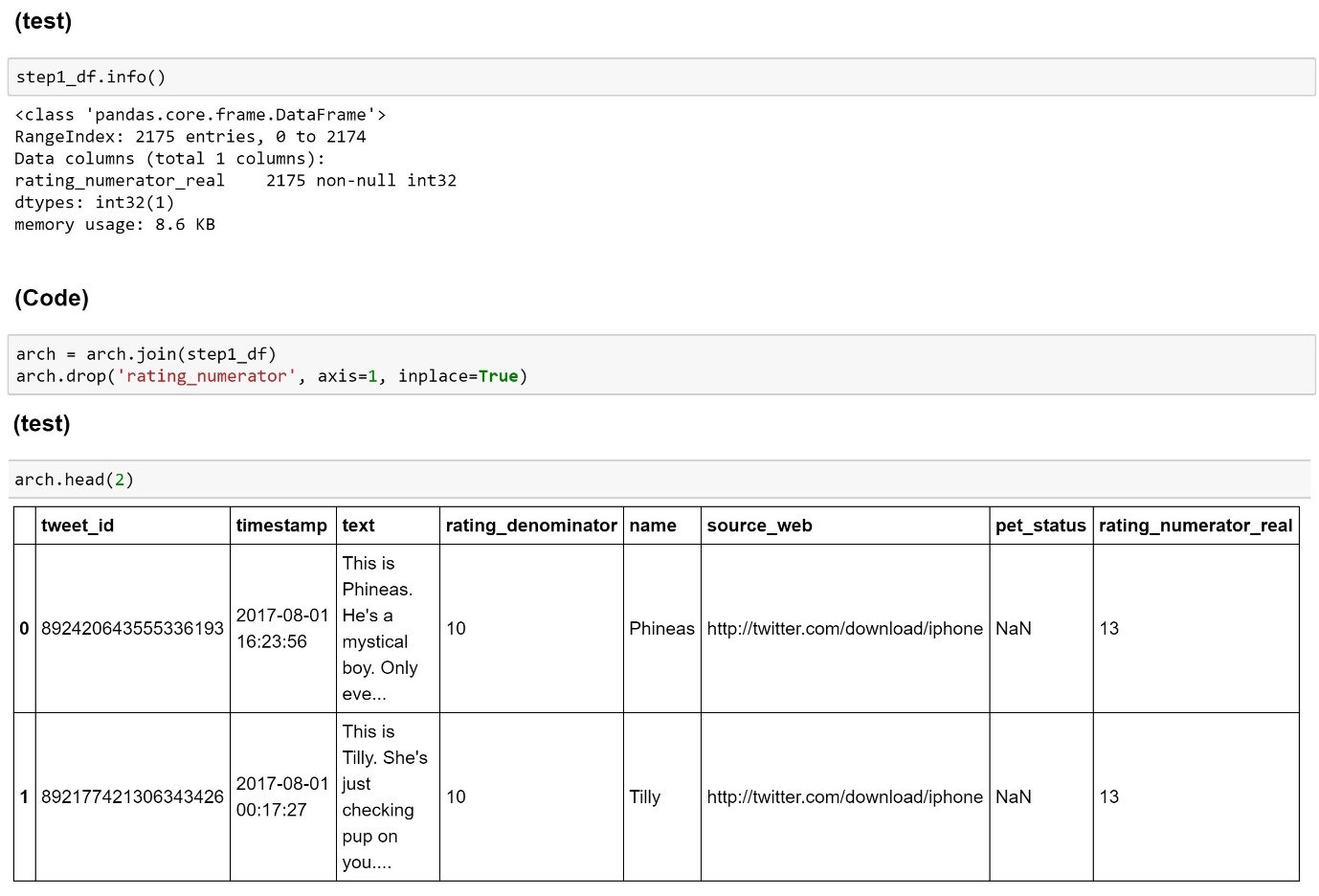


Imputing the NaN value is very important here because otherwise, we should consider atteching this new column of real numerators to the original dataset. Here, NaN values will be replaced with ‘0/10’.





Test)



**[Data 02] - image-predictions.tsv**

**# Tidiness(a)**

Define) Convert to the long format.

'p1, p2, p3' These 3 variables shoule be combined into one categorical variable.

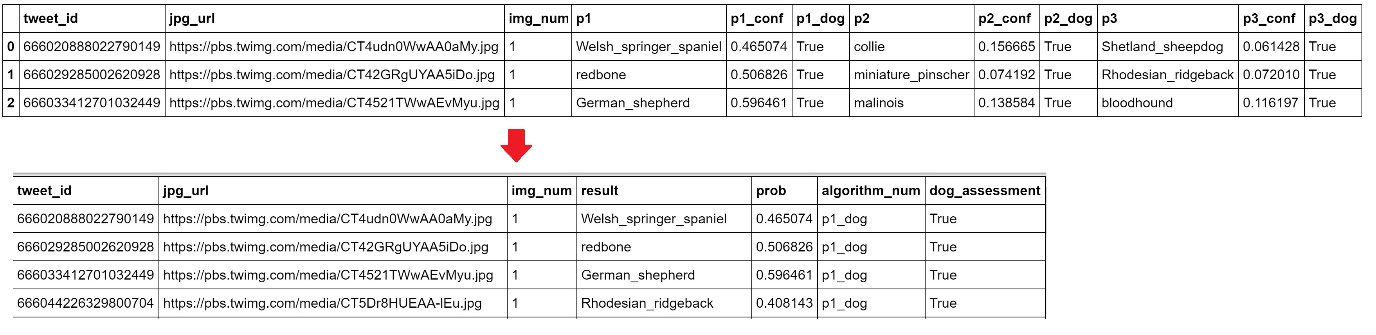
'p1\_conf, p2\_conf, p3\_conf' These 3 variables shoule be combined into one categorical variable.

'p1\_dog, p2\_dog, p3\_dog' These 3 variables shoule be combined into one categorical variable.

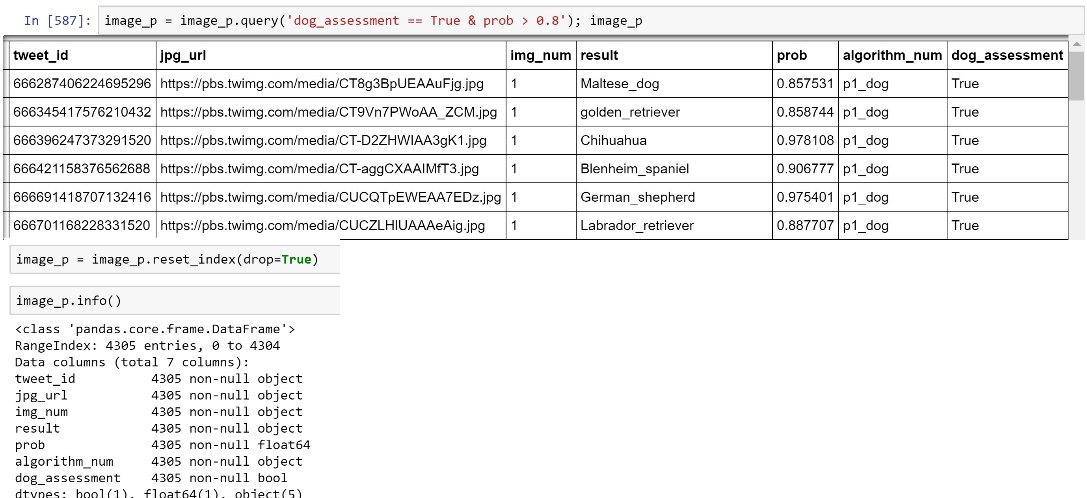
Code)



Test)

****

Code), test)

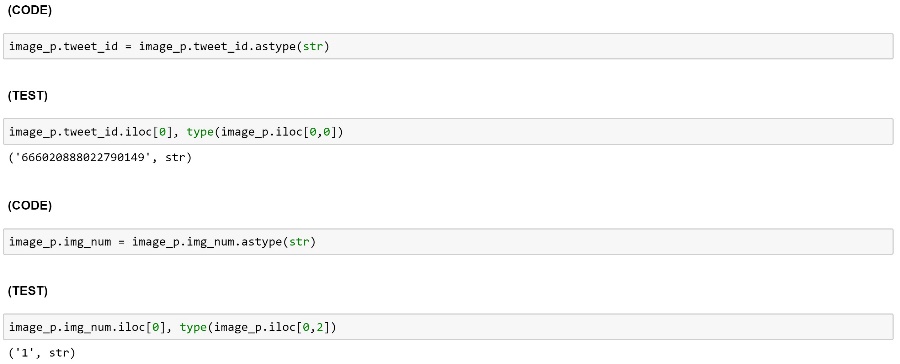
****

**#Quality(a)(b)**

Define)

- 'tweet\_id' should be str

- 'img\_num' should be str

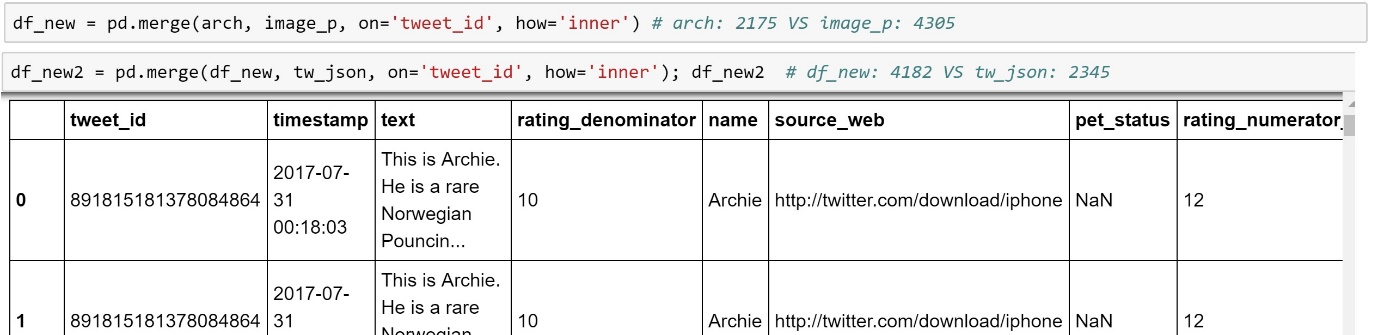


**[Data 03] - tweet\_json.txt**

**# Tidiness**

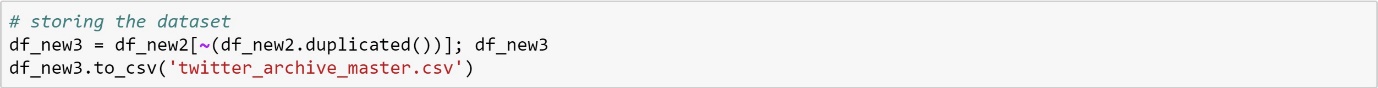
Define) ‘arch’ dataset, ‘image\_p dataset, ‘tw\_json’ dataset should be all combined into one. And then we need to get rid of some duplicated rows which is our last step of our data wrangling.

Code) & test)

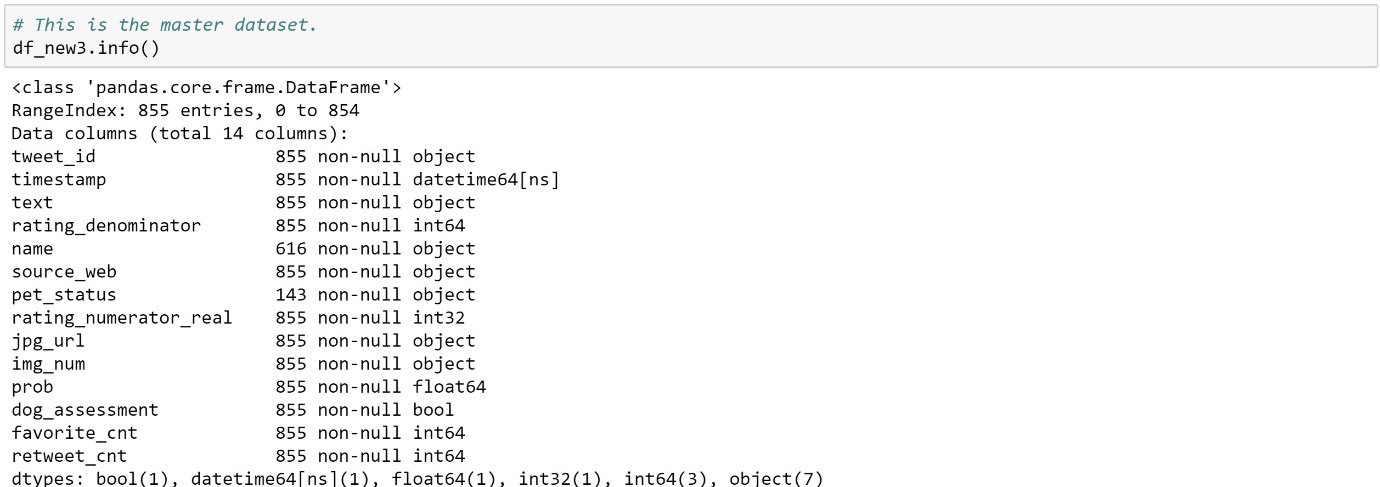


Here, for the first merge, we went for ‘inner join’ because we don’t want to create NaN values in the ratings column. For the second merge, we went for ‘inner join’ again because otherwise, we might get some missing values in ‘retweet\_count’ and ‘favorite\_count’ which is very problematic giving us the datatype ‘float.’ We‘d like to assume that ‘retweet\_count’ and ‘favorite\_count’ are our response variables in this analysis.

Then, Remove duplicated rows and save the master dataset in our local machine.

(Code)

(test)



Now we are good to go !