This report will note the data cleaning and transformation efforts effected on the WeRateDogs Twitter archive data, the tweet image prediction data, and the archive supplemental data downloaded via Twitter's API. Upon downloading the three files, I discovered a number of quality and tidiness errors. They are noted below, sorted by source and error type:

From the WeRateDogs Twitter archive data:

Quality

- timestamp data is an object, should be a timestamp
- all status_ids, and user_ids are floats, should be objects
- name column contains values that are not names
- value in floofer field should be a yes/no, and category type

Tidiness

- doggo, pupper, puppo - should be one column and category type

From tweet image prediction data:

Quality

- tweet_id is an int, should be an object
- rename columns p1, p1_conf, and p1_dog to more clear column names
- for p1 column values, replace underscore with space and make title case

From the Twitter API:

Quality

- id is an integer, should be an object
- Retweet and favorite counts are changed to decimal, revert to integer

Tidiness

- all data is in a tweepy.model.resultset and must be converted to a useful json before it can be loaded into pandas

Correcting the WeRateDogs Twitter archive data:

Quality:

Changing the timestamp datatype was a simple use of the to_timestamp function from pandas.

Changing the data type for the ids, floofer, and type was done via the astype function.

Since the main dataset should only contain tweets from the account and not retweets or replies, the columns about retweets and replies can be dropped. In addition, the columns doggo, pupper, and puppo can be dropped. It would be useful to check that those columns do not contain any data before dropping them.

Reviewing the results of the value_counts() method on the name column, it becomes clear that the script that assigned the name inadvertently caught articles and other words that are not the dog's actual name. To locate those, I identified the words in the name column that had four or fewer letters. I separated out those rows and applied a function to the tweet text column that used regular expressions to extract the word that comes after 'named', which I treated as the probable name. The original archive was then updated using that corrected dataframe.

Tidiness:

Entries identifying the dog as a doggo, pupper, or puppo were entered into a single column called type using a function that returned the first non-blank field from the three columns. Otherwise it returned the value 'none' as some of the values were blank.

Correcting the WeRateDogs tweet image prediction data:

Quality:

Renaming the columns was done using the rename function.

Removing the underscore and changing the case of the values was done by creating a function that took in the column value and replacing the underscore with a space and changing the case to title case. The function was applied across the entire column.

Correcting the WeRateDogs tweet data from the Twitter API:

Tidiness:

To tidy the twitter data, I had to extract the relevant JSON data from the API return before I could turn it into a pandas dataframe. I extracted the JSON and stored it in variable temp, but when attempting to load it using the read_json function, it returned an error:

Protocol not known: [{"created_at": "Sun Jun 18 16:57:37 +0000 2017", "id": 876484053909872640, "id_str": "876484053909872640", "text": "This is Benedict. He wants to thank you for this delightful urban walk. Hopes you know he loves you. 13/10 super du\u2026 https

According to this Stack Overflow question, Pandas read json ValueError: Protocol not known, current pandas read_json looks for file-like objects and by using StringIO, it allows read_json to read the string object.

Quality:

Attempting to change **retweet_count** and **favorite_count** to integers results in the error ValueError: Cannot convert non-finite values (NA or inf) to integer

According to this Stack Overflow question, Data type conversion error: ValueError: Cannot convert non-finite values (NA or inf) to integer, this may be because there were missing values. After confirming the existence of null values using isnull().sum(), I used the Int64 type, which allows NaN support for integer types, as indicated in these Stack Overflow questions,

According to NumPy or Pandas: Keeping array type as integer while having a NaN value and in the Pandas documentation here.