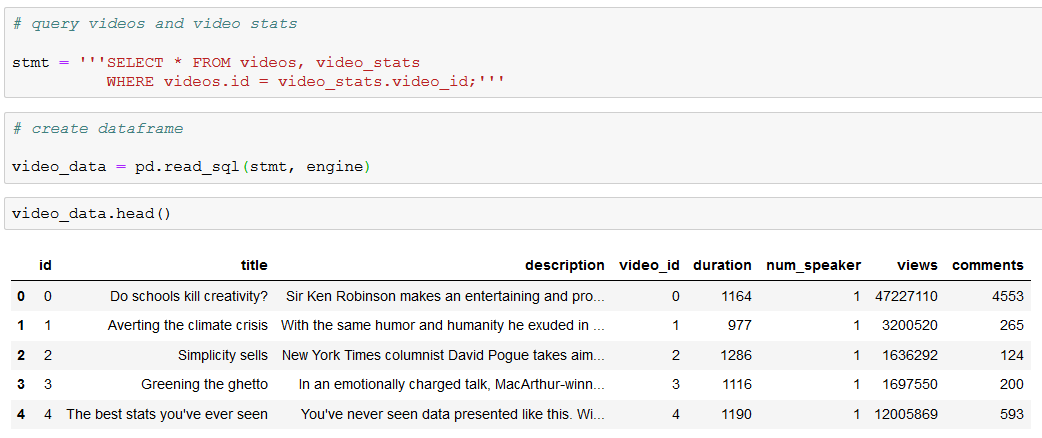
Since our client is interested in understanding the aspects of our project that relate to video views, we have taken preliminary steps to conduct an exploratory data analysis for features that may be related to the outcome variable. To simulate how a real database user would conduct an analysis, we are using a combination of Postgres and Python, including the SQL Alchemy, Pandas, Matplotlib, and Seaborn packages. Data is queried from the database using SQL Alchemy, stored in a Pandas dataframe, and visualized with Matplotlib and/or Seaborn. Specific parts of the EDA are detailed below:

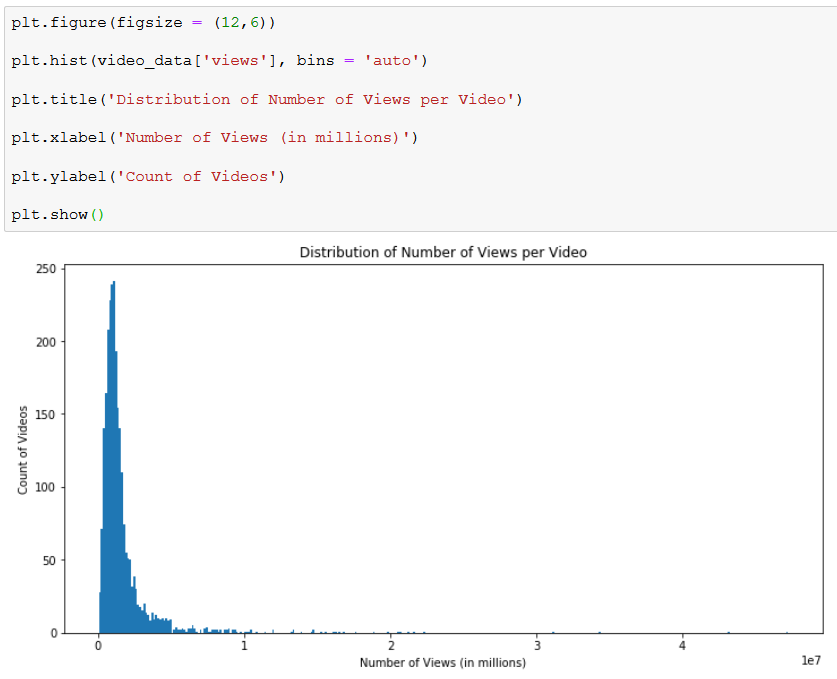
***Procedure 1: What is the distribution of the outcome variable – views?***

To help the client with their goal of producing popular online learning videos, we must first understand how many views the TED Talks in our database generated. Let’s examine the views variable from the video\_stats table.

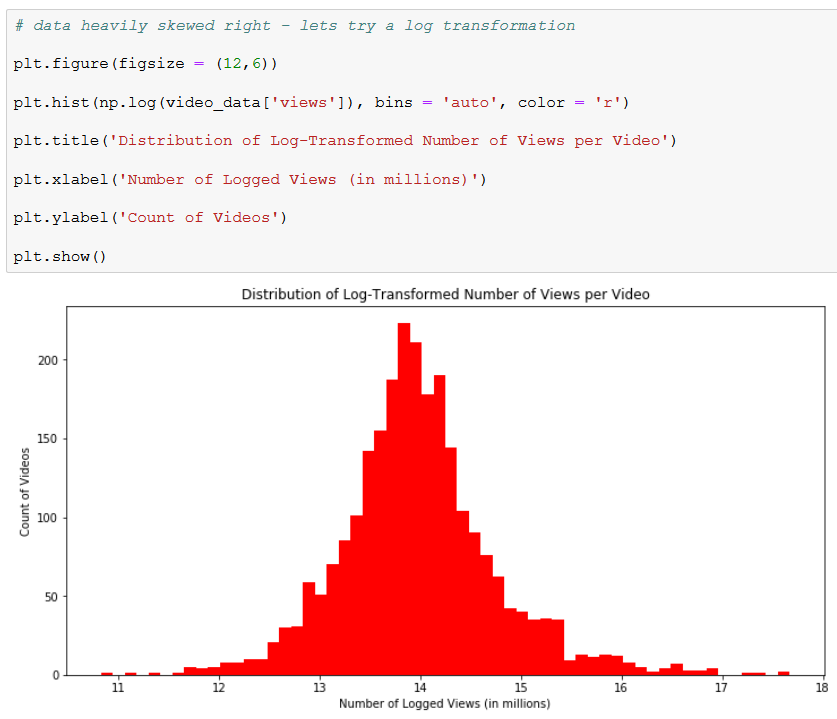
Data retrieval:



Data visualization:



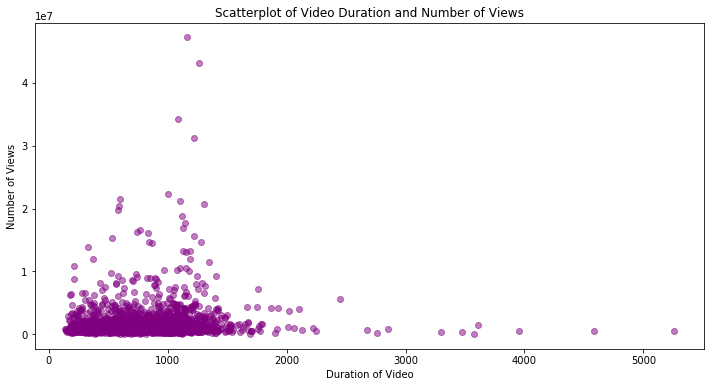
***Procedure 2: The outcome variable looks highly skewed to the right – would it look better under a log transformation?***



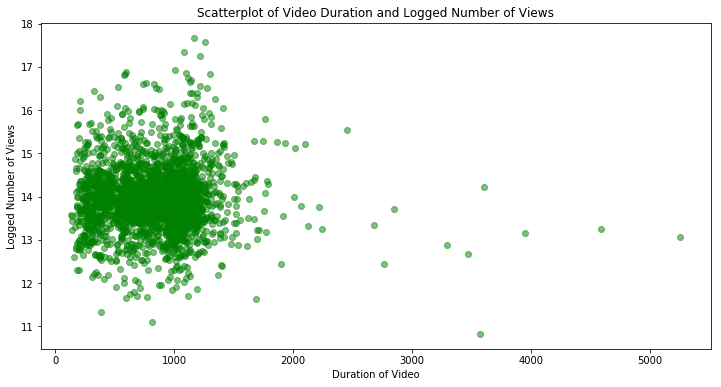
The distribution of the outcome variable takes on a much more Gaussian form when we apply a log transformation. If we decide to use a parametric method to build a machine learning model, we should probably apply this transformation before we do so. This will help to eliminate outliers from the training examples and ultimately generate predictions about video view statistics that are more accurate. Let’s now check out some of the predictor variables at our disposal.

***Procedure 3: How does the duration of a video affect its number of views?***

One of the other features in the queried table is video duration. Let’s check it out in a scatterplot to see if we can observe any relationships in the data.



Unfortunately, most observations are bunched in the bottom right and we cannot easily see a relationship in the data. Does it look any different under a log transformation as performed in Procedure 2?

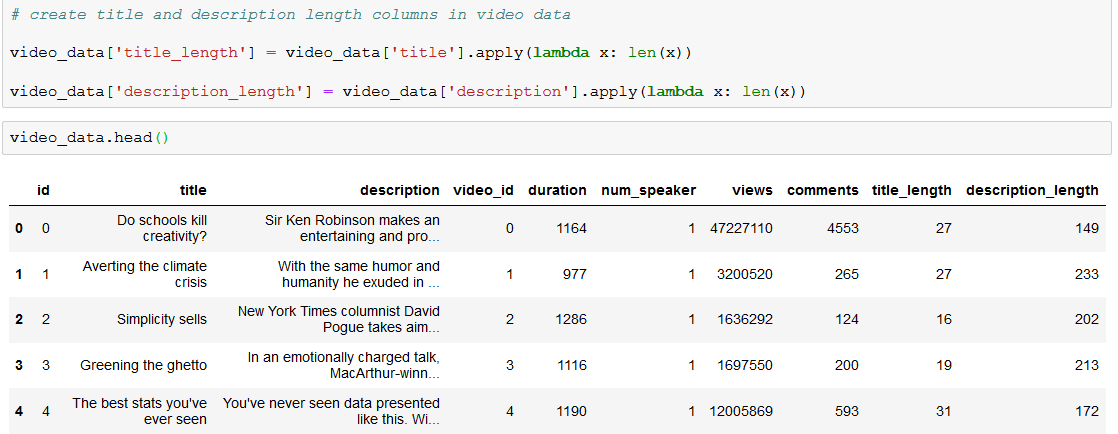


Not really – the observations are certainly more spread out than they were previously, but there are no easily identifiable trends that could benefit us for predictive modeling.

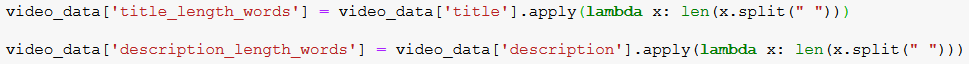
Let’s do some feature engineering to create new variables.

***Procedure 4: Feature Engineering on Video Data***

The flexibility of Python and Pandas allows us to easily create new columns using a few short lines of code. In this case, we are interested in seeing whether the length of a video title or description have an effect on its view count. In this part, I go back to the video\_data table from previous procedures to make new columns for the length of their titles and descriptions (measured in number of characters).



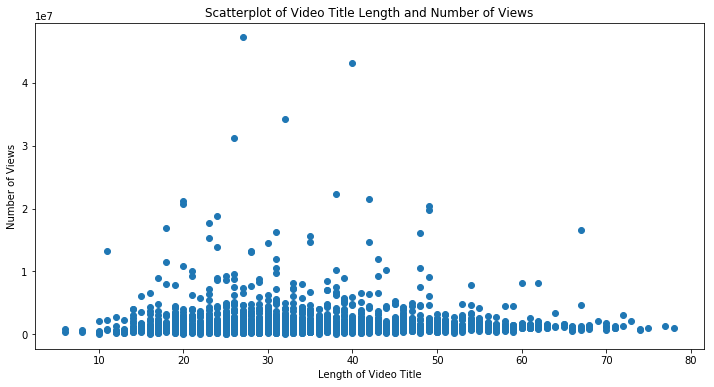
Looking at the current data, it may be helpful to also include new features that count the number of words, not characters, that appear in each field. We update the table below:



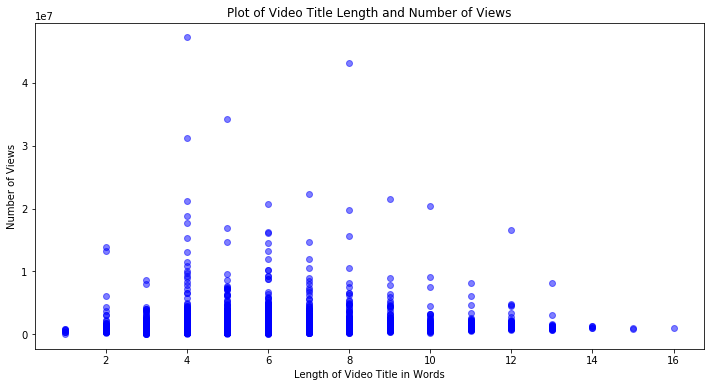
Now that the new features are ready, we can continue to explore them through plots and graphs.

***Procedure 5: Visualizing relationship between title length and view count***

We seek to know if the length of a video’s title has any significant effect on its view count. To better understand the relationship, we can once again use a scatterplot in Matplotlib to visualize the data.

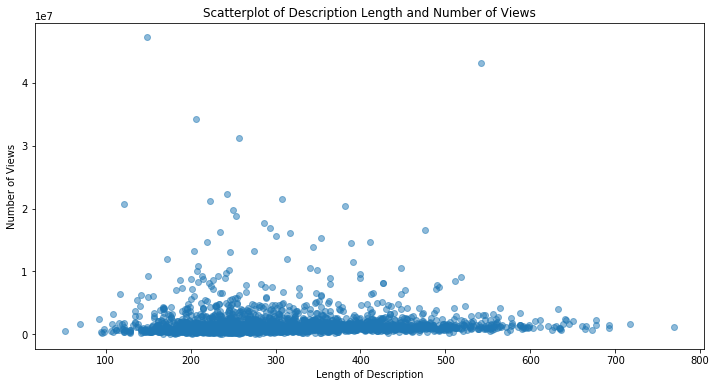


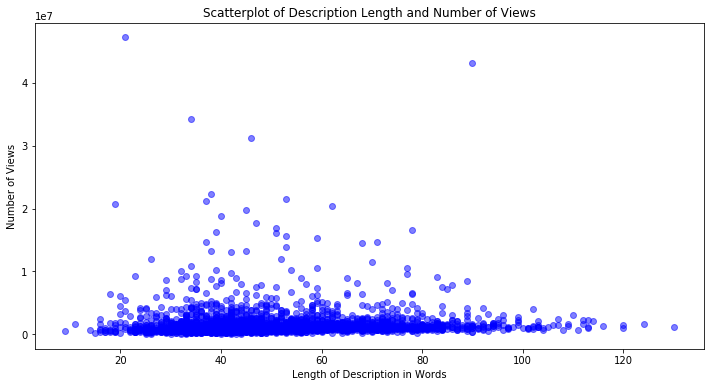
We cannot see any significant increase or decrease in the outcome variable at title length increases. Let’s try again with new feature that measures the number of words used instead of characters:



Despite the new feature engineering efforts, the plot still shows no strong relationship between title length and number of views. Let’s check out the descriptions to see if we get better results.

***Procedure 6: Visualizing relationship between description length and view count***



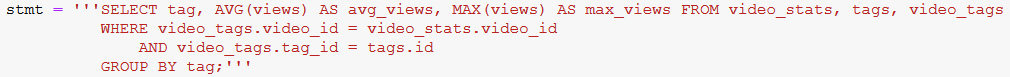


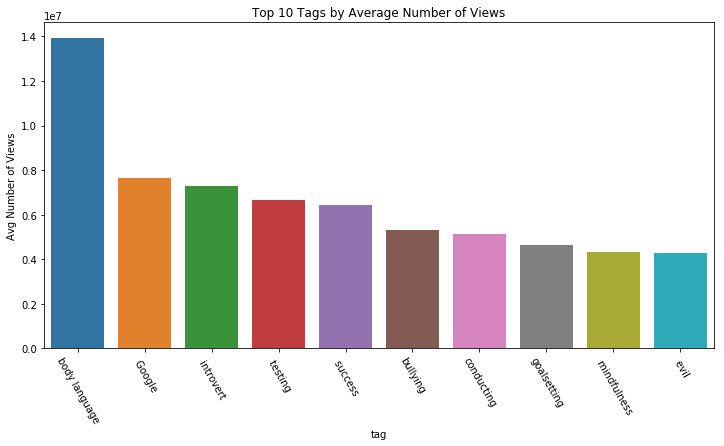
The plots above show a slight relationship between description lengths and view counts. Through the similar distributions in both plots, we can see that videos with the highest views tended to have description lengths between 25 and 60 words, or alternatively, between 150 and 400 characters. These insights could be actionable for our client and help generate some initial best practices for the videos they post.

Let’s continue to explore the data and check out the tags on each video.

***Procedure 7: Which tags were included on the videos with the highest average number of views?***

One of the features captured in our database is tags, which has 594 distinct values. We can combine the video stats and tags data to generate summary statistics about each tag and see which has the highest average view counts.

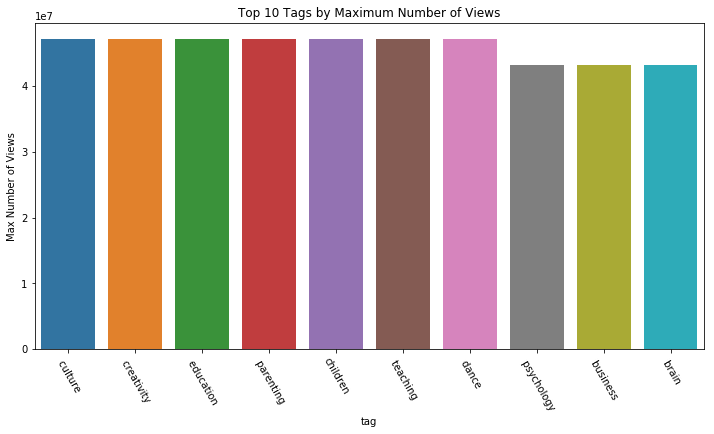




Interestingly, body language had an exceptionally high average view count at ~1.4 million. The next highest topic, Google, was nearly half that amount at ~800k. It is worthwhile to note that body language is a rather unique tag, so a few outlier videos may be pulling this number up significantly compared to other tags.

***Procedure 8: Which tags had the highest maximum number of views?***

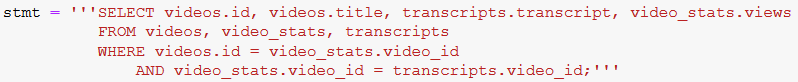
We can take a similar approach as we did on the above visual to examine which tags were included on the videos with the most views in our database.

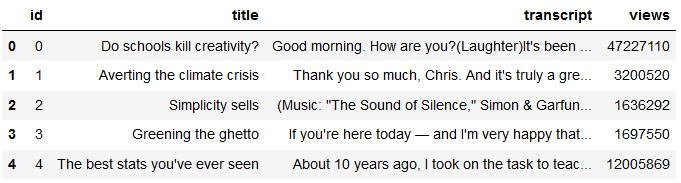


This visual is less informative than the summary by average, since all it tells us is that the most viewed video had the seven tags on the left included on it. Nonetheless, it tells us that certain unique topics, like children and dance, may have potential to generate lots of interest in the online learning community.

***Procedure 9: Clean transcripts text data***

One of the most important features in our database is the transcripts table, which tells us exactly what was said during each TED Talk. We can pull this info in with videos and video stats to make a new data frame in Python.

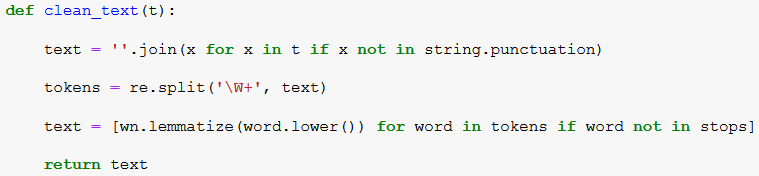




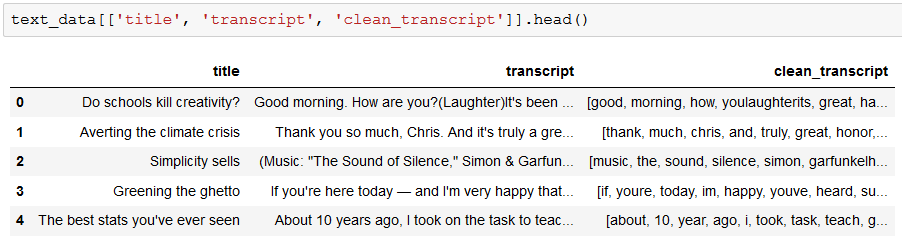
Now that the important information is all in one place, we need to clean the transcript data. Every NLP project is at least slightly different, so our team decided to take a few major steps to clean the data in a way that is usually successful in other similar projects. The steps to clean the text data included:

* Remove stop words using NLTK
* Lemmatizing words using NLTK’s Wordnet Lemmatizer
* Making all letters lower case
* Splitting words into a list

This was all accomplished in one function, as shown below:

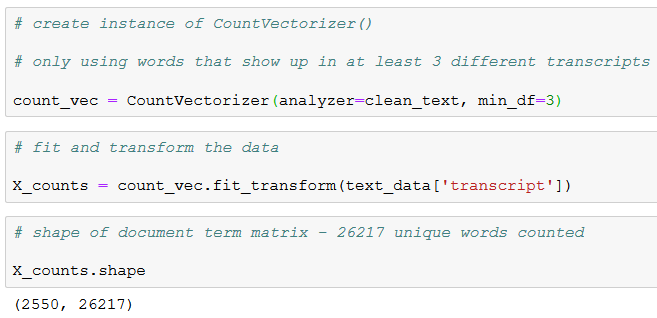


Once ready, this function was applied to the transcripts column to make a new column – clean transcripts. A sample of this column is shown below next to the original title and transcript columns:

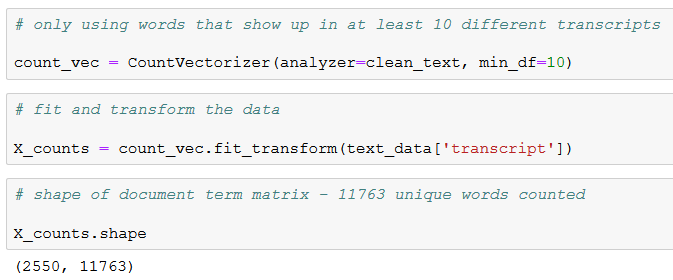


***Procedure 10: Vectorizing transcripts using Scikit Learn***

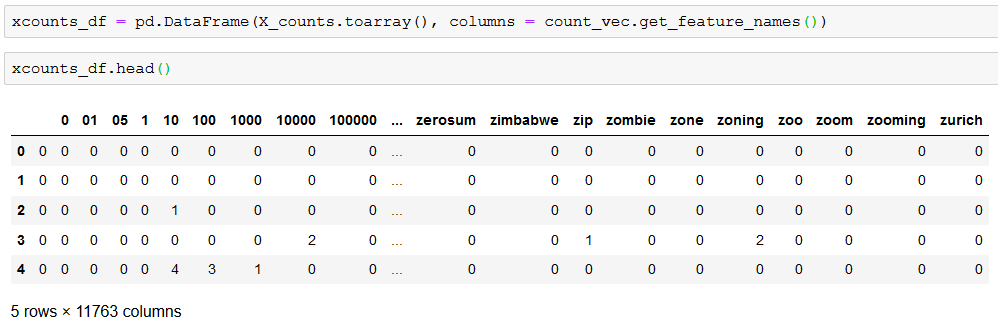
Another task that would be highly relevant to our project goal would be vectorizing the words in the transcript. Vectorization is a process that converts text data into sparse matrices by recording where certain values exist. In our case, we used a Count Vectorizer from the Scikit Learn package to vectorize the transcript data. The parameters specified that transcripts would be cleaned with our clean\_text function and that words would only be included if they showed up in at least 3 videos; 26217 words were included in the output matrix:



Since we probably don’t want to include all 26000+ columns in a predictive model, we increased the threshold to 10 and examined how much that reduced the size of the matrix.



Our threshold increase was successful, as the number of columns was reduced to 11763. Now that we have a more management count vectorized matrix, we can convert it to a Pandas dataframe for further analysis:



With our data vectorized, predictive models may now be run on this data. Other types of vectorization, such as TF-IDF, should also be explored in the future to see how the results compare against each other.