CPSC 573 Data Cleaning Document

Bonnie Wu
UCID: 30392461

Introduction

My dataset is on Anime that is accessed from myAnimeList (also known as MAL), a volunteer run social networking and social cataloging application, through the Jikan API. As someone who grew up watching and interacting with Anime, I thought it would be interesting to see how Anime has changed over the years, especially within the past couple of years. As I have interacted with Anime when I was younger, I have seen or experienced negative stigma associated with it. Recently however, it seems that that stigma has been diminishing as more people are enjoying this type of media and as Anime seems to become more mainstream. One of my goals is to identify if there is any data from a general anime database that might indicate that there has been an increase in popularity of viewership. In addition, I plan to see if there is any data that could indicate if an Anime is popular. Here are my following initial analysis questions that I want to explore with this dataset.

Initial Analysis Questions

These are the initial questions that I had thought of before I started the data cleaning process. Over the course of cleaning and profiling my data, my questions have changed slightly.

1. **What is the distribution of reviews over time?**
2. **What genres tend to have more reviews? What themes tend to have more reviews?**
3. **What animes tend to seem the most popular?**
4. **Can we use reviews as a way to see if animes are popular?**
5. **Do anime themes tend to be part of a certain demographic or genre? E.g. Is Action more associated with the Shounen demographic?**
6. **Does the source material for the anime affect it's viewership/reviews?**

Data Cleaning

I have three main files, animeData.json, topAnime.json, and animeReviews.json. I will go over the process of cleaning all of these files which were done with a mixture of python scripts and Tableau Prep.

Cleaning data from animeData.json

The animeData JSON file had a lot of measures that I did not need, such as image urls and trailer urls. Some of the measures, such as 'relations' or 'streaming', were empty so I excluded them. The following are the measures that I decided to keep. While not all directly pertain to the analysis questions that I have chosen, I decided to keep them in in case.

- anime_id: myAnimeList ID for the Anime title (named mal_id in the json data)
- url: the URL where the Anime title can be found
- title: the default title of the Anime on myAnimeList
- title_english: the title of the Anime in English
- source: the source material for the anime
- episodes: how many episodes the anime has
- status: current airing status of the anime
- aired start: the date when the anime started airing
- aired finish: the date when the anime finished airing if applicable
- duration: the duration of each anime episode
- rating: the suitability rating for audiences
- score: the overall score of the anime rated by people
- scored by: the amount of people who have scored the anime
- rank: the ranking of the anime if applicable
- popularity: the popularity rank of the anime
- season: the season when the anime aired
- year: the year when the anime started airing
- broadcast day: the day of the week when the anime airs/aired
- producers: a list of the producers for the anime
- studios: a list of the studios for the anime
- genres: a list of the genres for the anime
- themes: a list of the themes for the anime
- demographics: a list of the demographics for the anime

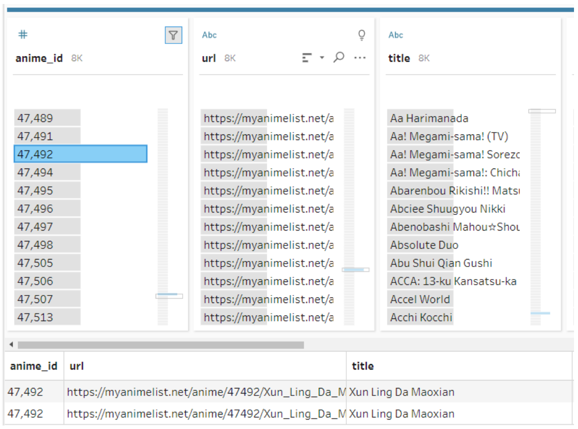
Some measures I encountered, such as 'type' contained values that I did not expect. While I was cleaning, I realized that the API request that I used (getAnimeByID) returned not only broadcasted Animes, but also the following types 'OVA', 'ONA', 'Manga', and 'Music'. I excluded 'Manga' and 'Music' as those are different than Anime. OVAs and ONAs are considered anime, however the difference is that they were distributed as home and network videos respectively. Additionally, OVAs and ONAs are usually, but not always, related to a broadcasted anime series and may not follow the main plot of the anime. Therefore, they might not be canonically related and due to the distribution method, not everyone has access to them like they do for broadcasted anime. For viewers in North America, streaming services such as Crunchyroll(https://www.crunchyroll.com/) may not stream the OVAs/ONAs as well. For those reasons, I have decided to exclude them and only focus on broadcasted anime that is generally more accessible to the general public. To ensure this, I only included values that had 'TV' as their 'type' value.

There were multiple measures for the title of the anime. I decided to choose 'title' and 'title_english' as the measures to keep as I thought these two measures were enough information for my purposes. I excluded ‘titles’, ‘title_japanese’, and ‘title_synonyms’. An anime can have slightly different titles on how it’s translated, however I decided to trust the judgement of myAnimeList to choose a default title for animes so I decided to choose ‘title’. I also kept the ‘title_english’ measure as some default anime titles are not actually the title in English, but the Romanized version of the title so there may be some discrepancies. While I was doing some light analysis, I did find myself using only ‘title’, but I still decided to keep ‘title_english’ in case.

I also realized that episodes will return null for titles that are currently airing. This makes sense as there is no way to know how many episodes will air for those titles. I decided to keep them as null values.

I did most of my measure selection in python. Some measures that I got rid of later, I did in Tableau. For now, here is my python code which pulled the data that I wanted from the animeData.json file into a csv file.

The following cleaning of the animeData is mostly done in Tableau.



anime_id 8K	url 8K	title 8K
47,489	https://myanimelist.net/a	Aa Harimanada
47,491	https://myanimelist.net/a	Aa! Megami-sama! (TV)
47,492	https://myanimelist.net/a	Aa! Megami-sama! Sorezc
47,494	https://myanimelist.net/a	Aa! Megami-sama!: Chichi
47,495	https://myanimelist.net/a	Abarenbou Rikishi!! Matsı
47,496	https://myanimelist.net/a	Abciee Shuugyou Nikki
47,497	https://myanimelist.net/a	Abenobashi Mahou☆Shou
47,498	https://myanimelist.net/a	Absolute Duo
47,505	https://myanimelist.net/a	Abu Shui Qian Gushi
47,506	https://myanimelist.net/a	ACCA: 13-ku Kansatsu-ka
47,507	https://myanimelist.net/a	Accel World
47,513	https://myanimelist.net/a	Acchi Kocchi

anime_id	url	title
47,492	https://myanimelist.net/anime/47492/Xun_Ling_Da_M	Xun Ling Da Maoxian
47,492	https://myanimelist.net/anime/47492/Xun_Ling_Da_M	Xun Ling Da Maoxian

Figure 1: Deleting duplicate in Tableau Prep

For measures such as ‘anime_id’ and ‘url’, I used Tableau Prep to make sure that there were no duplicates. There was only one duplicate and I cleaned it up by deleting the duplicate. You can see this in Figure 1.

For time measures, such as ‘aired start’ and ‘aired finish’ I just changed the data type from string to date. Something I did notice was there were more null values in ‘year’ than in ‘aired start’ which I thought was odd. I compared the values and realized that ‘aired start’ usually had the values for those that were null. I decided to calculate the year based on the ‘aired start’ measure and was able to get rid of some of the null. In Figure 2. you can see how some of the null values in ‘year’ were now classified in ‘calculated year’.

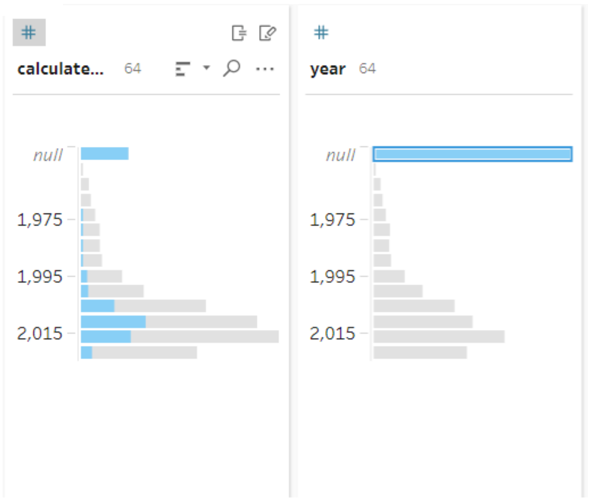


Figure 2: Null values in year compared to calculated year

To make sure that my calculations were correct, I decided to make a simple check, named 'SameYear?' to see if 'calculated year' had the same values as 'year'. The check returns a boolean if the year is the same. I expected to get only True and null values in return. To my surprise, there were 82 False returns. You can see this in Figure 3.

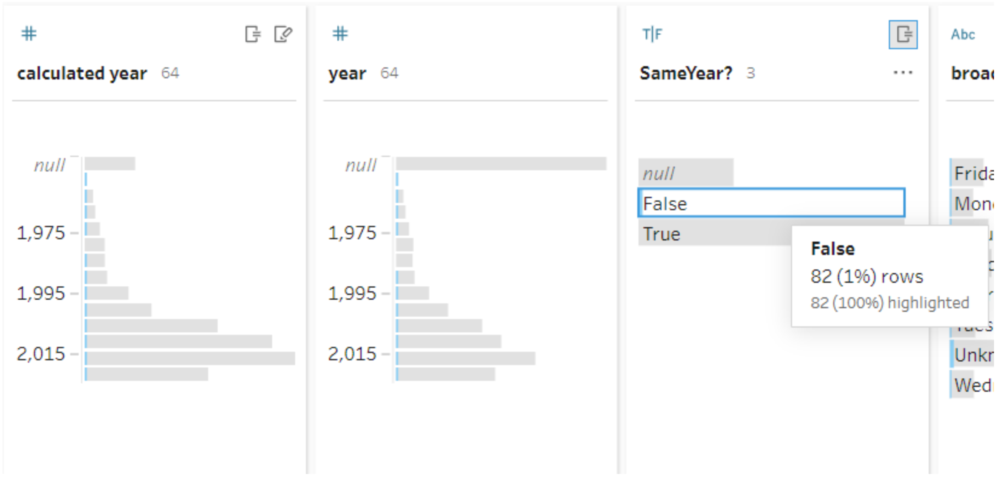
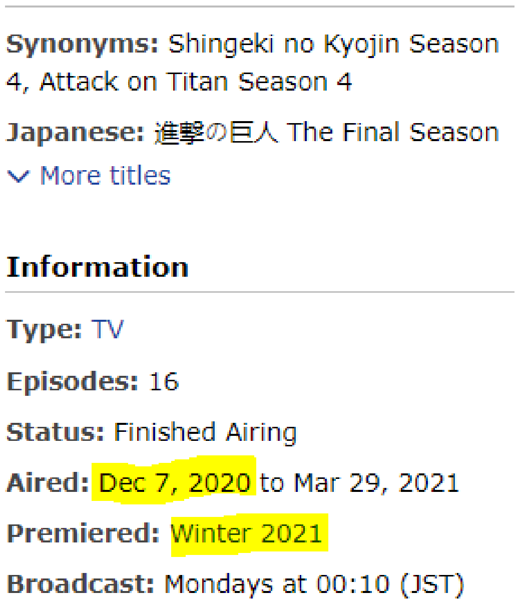


Figure 3: Checking False results after comparing the values for year and calculated year

I clicked on some of the titles that returned 'False' in my check and realized that they all aired in the month of December. I looked at the URL and realized that 'aired start' is slightly different than 'year'. It seems that 'year' actually refers to the season and year that it released and not the actual air date. See Figure 4. For an example of this.



Synonyms: Shingeki no Kyojin Season 4, Attack on Titan Season 4

Japanese: 進撃の巨人 The Final Season

More titles

Information

Type: TV

Episodes: 16

Status: Finished Airing

Aired: Dec 7, 2020 to Mar 29, 2021

Premiered: Winter 2021

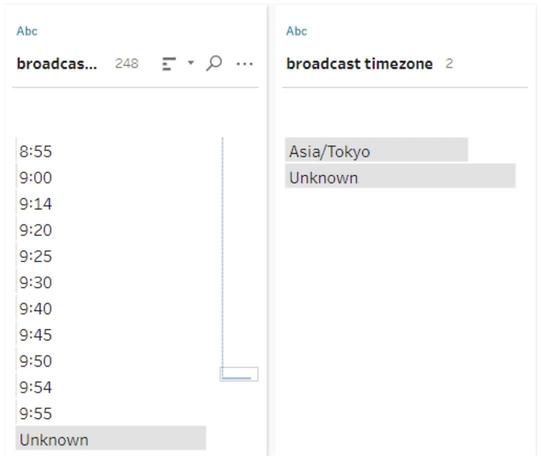
Broadcast: Mondays at 00:10 (JST)

Figure 4: Anime example showing the discrepancy between Aired and Premiered

We can see that it did air in 2020 but the premiere date is in 2021. I decided to keep both but for my analysis I will use calculated year. I think this is more consistent with the airing start date and will also help make my analysis more consistent.

While looking at 'aired start', I also noticed that there were many null values in 'aired end'. Some of the values are because the anime is still currently airing, but there were also null values for animes whose status was 'Finished Airing'. There are around 2000 anime titles that have finished airing but also have a null value for 'aired finish'. As MAL is volunteer based, I feel that this information was just forgotten unfortunately. I will not get rid of it but when I do analysis with the 'aired finish' measure, I will need to keep this in mind.

I decided to only keep the ‘broadcast day’ measure as opposed to ‘broadcast day’, ‘broadcast time’, and ‘broadcast timezone’. Before I loaded it into Tableau Prep, I changed all the null values to ‘Unknown’. While looking at the data in Tableau Prep, I decided there wasn’t enough meaningful data for me to use. Below you can see how many ‘Unknown’ values there are for ‘broadcast time’ and ‘broadcast timezone’. Figure 5. Shows the amount of ‘Unknown’s compared to other values.


Abc
broadcas... 248
8:55
9:00
9:14
9:20
9:25
9:30
9:40
9:45
9:50
9:54
9:55
Unknown

Abc
broadcast timezone 2
Asia/Tokyo
Unknown

Figure 5: Amount of unknown values in broadcast time and broadcast timezone

I did still want to see if broadcasting time affected anything so I decided to use ‘broadcast day’ since although there are still some unknowns, it has slightly more information then ‘broadcast time’ and ‘broadcast timezone’. My speculation as to why there are so many unknowns in the data is because MAL is volunteer run, and this data is not required to be entered so it may have been forgotten. You can see the amount of ‘Unknown’ compared to the values of ‘broadcast day’ in Figure 6.

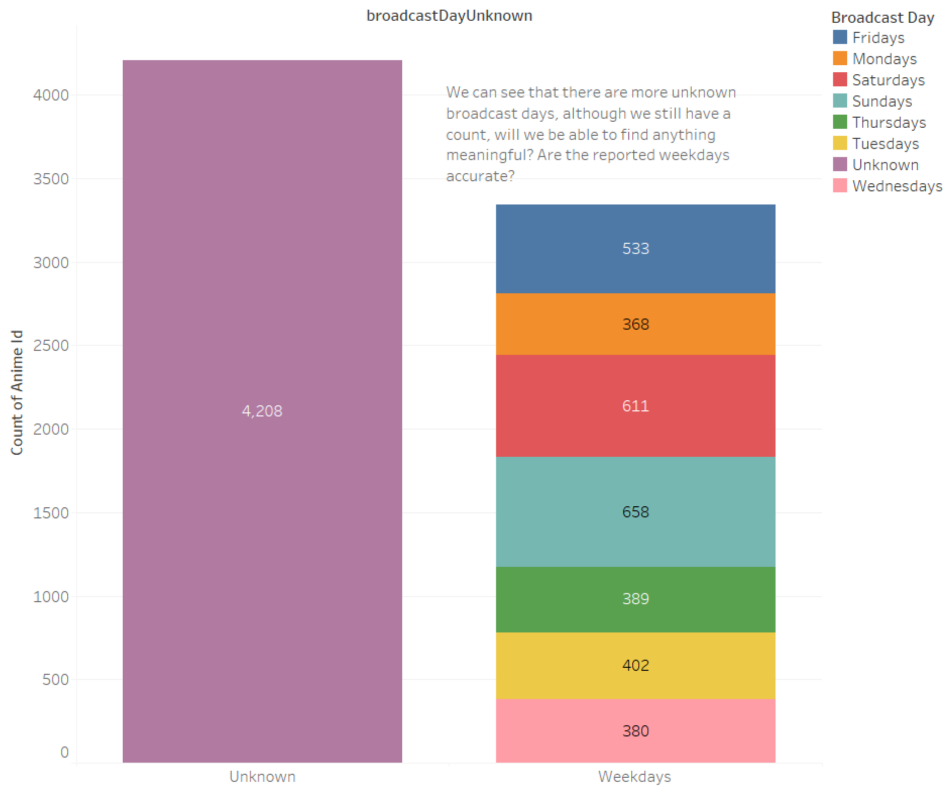
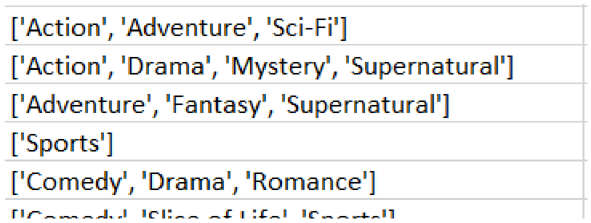


Figure 6: Amount of Unknown broadcast days

My interest in the broadcasting time stems from the fact that some anime streaming services, such as Crunchyroll, release currently airing anime on their services within hours of the initial broadcast in Japan. For example, the anime, Jujustsu Kaisen, is currently airing and other regions around the world can watch newly released episodes around 2 hours after the initial broadcast date, which was mentioned in a Dexerto article, which can be found here. It would be interesting to see if there is any data hinting at this.

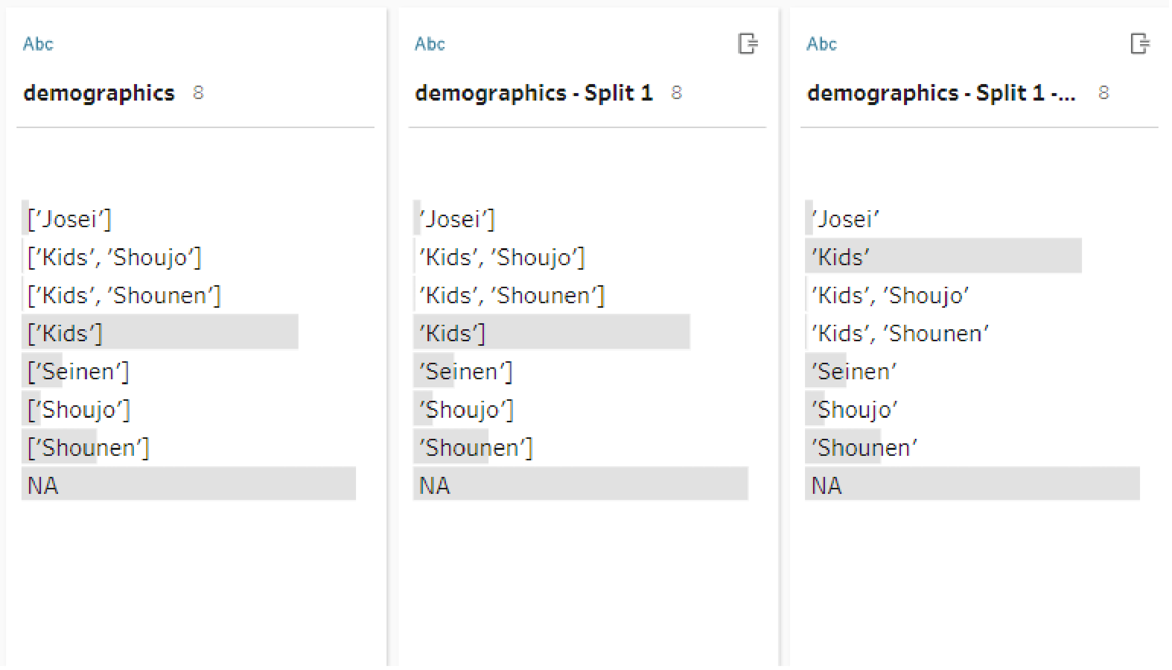
The following measures, 'producers', 'studios', 'genres', 'themes', and 'demographics', contained groups of values that included 'mal_id', 'name', 'type', and 'url'. I decided that the name data is the only data that I would need for my purposes, so I collected only the names and put them into an array for each anime. For 'producers' and 'studios', if they were blank, I decided to put 'Unknown' as the value. For 'genres', 'themes', and 'demographics', I decided to put 'NA' as the value as opposed to 'Unknown' as not all animes have to a specific genre, theme, or demographic.

I ended up cleaning up the brackets by splitting them. I figured that I might have some issues in the future if I left it in a list format, as seen in Figure 7., so I decided to clean it up a bit more. Figure 8. Shows how I got rid of the encasing brackets in Tableau Prep using splits. For the single quotations, I got rid of them in Tableau Prep by replacing them all with a space which you can see in Figure 9.



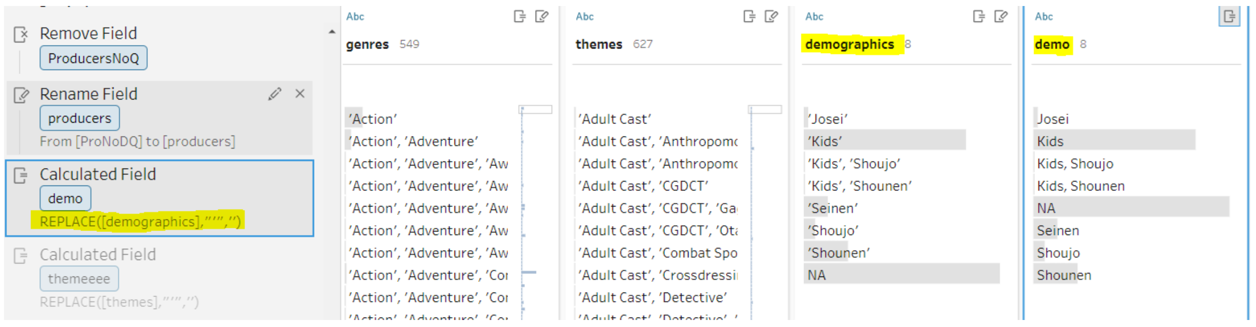
['Action', 'Adventure', 'Sci-Fi']
['Action', 'Drama', 'Mystery', 'Supernatural']
['Adventure', 'Fantasy', 'Supernatural']
['Sports']
['Comedy', 'Drama', 'Romance']

Figure 7: Example of a measure with a list for the value



demographics 8	demographics - Split 1 8	demographics - Split 1 -... 8
['Josei']	'Josei']	'Josei'
['Kids', 'Shoujo']	'Kids', 'Shoujo']	'Kids'
['Kids', 'Shounen']	'Kids', 'Shounen']	'Kids', 'Shoujo'
['Kids']	'Kids']	'Kids', 'Shounen'
['Seinen']	'Seinen']	'Seinen'
['Shoujo']	'Shoujo']	'Shoujo'
['Shounen']	'Shounen']	'Shounen'
NA	NA	NA

Figure 8: Showing the progression of splitting using '[' and ']'



Remove Field
ProducersNoQ

Rename Field
producers
From [ProNoDQ] to [producers]

Calculated Field
demo
REPLACE([demographics],"'","")

Calculated Field
themeeee
REPLACE([themes],"'","")

genres 549	themes 627	demographics 8	demo 8
'Action'	'Adult Cast'	'Josei'	Josei
'Action', 'Adventure'	'Adult Cast', 'Anthropomo	'Kids'	Kids
'Action', 'Adventure', 'Aw	'Adult Cast', 'Anthropomo	'Kids', 'Shoujo'	Kids, Shoujo
'Action', 'Adventure', 'Aw	'Adult Cast', 'CGDCT'	'Kids', 'Shounen'	Kids, Shounen
'Action', 'Adventure', 'Aw	'Adult Cast', 'CGDCT', 'Ga	'Seinen'	NA
'Action', 'Adventure', 'Aw	'Adult Cast', 'CGDCT', 'Ot	'Shoujo'	Seinen
'Action', 'Adventure', 'Aw	'Adult Cast', 'Combat Spo	'Shounen'	Shoujo
'Action', 'Adventure', 'Co	'Adult Cast', 'Crossdressi	NA	Shounen
'Action', 'Adventure', 'Co	'Adult Cast', 'Detective'		

Figure 9: Getting rid of single quotations for demographics using REPLACE in Tableau Prep

For animes that are currently airing, some measures will be empty, I forgot to clean it in the python script so I decided to clean it in Tableau Prep. It seems that the number of episodes cannot be determined until after it have finished airing so I decided to replace the null value with 'NA' for now. I may decide to update it with the current number of episodes that have been released, but as of right now I do not see a use for it yet. They also do not have a finishing airing time, so I put 'NA' as the value only if they are currently airing. If not, I put 'Unknown'.

While I was looking at the anime data, I did realize that some 'title_english' values are null, however I left it as is as I am anticipating using 'title' for most purposes, which has no null values. There were also titles that null for their ranking. I decided that this information was not useful for me as I anticipate that I will be using the rankings quite often. I decided to get rid of them. You can see this in Figure 10.

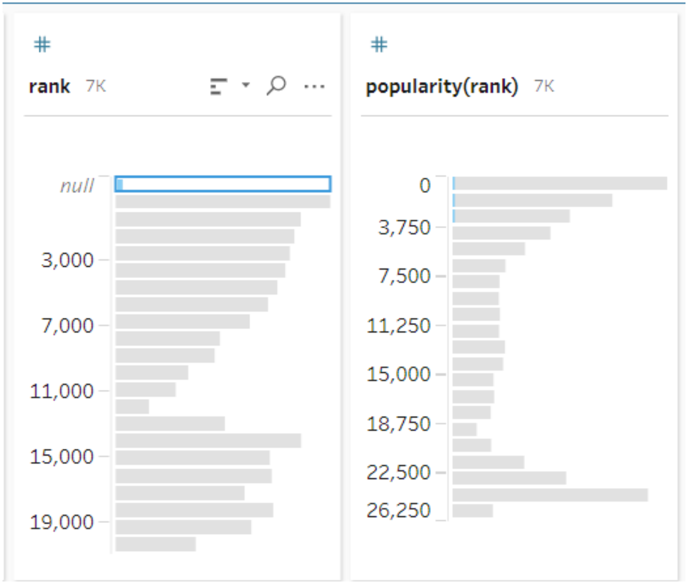
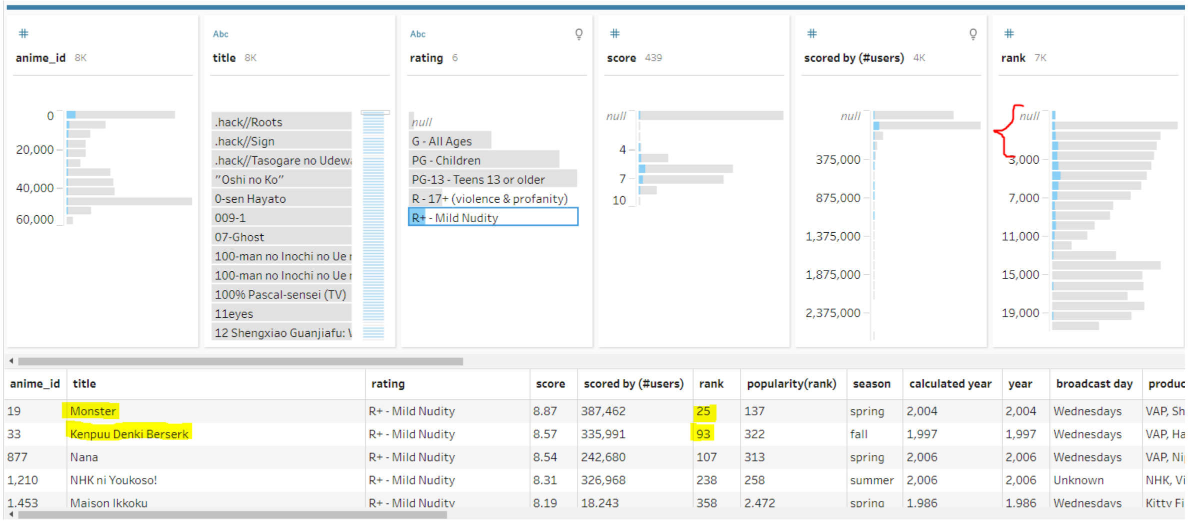


Figure 10: Showing null values in rank

In addition, I considered removing anime that had more mature themes, however I had a difficult time determining what to keep and what to remove. For instance, I attempted removing anime that had the rating ‘R+ (mild nudity)’ but found that there were some titles that were within the top 100 ranking. I decided that using just the rating was not a good enough identifier of mature themes. I also tried to look at other measures, but they all seemed like an arbitrary way of getting rid of titles.

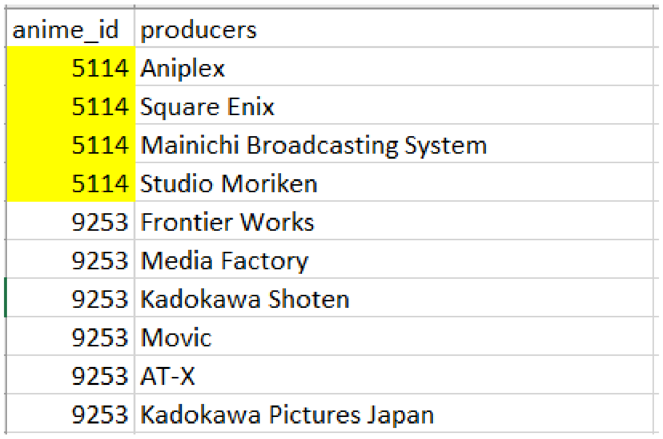


anime_id	title	rating	score	scored by (#users)	rank	popularity(rank)	season	calculated year	year	broadcast day	produc
19	Monster	R+ - Mild Nudity	8.87	387,462	25	137	spring	2,004	2,004	Wednesdays	VAP, Sh
33	Kenpuu Denki Berserk	R+ - Mild Nudity	8.57	335,991	93	322	fall	1,997	1,997	Wednesdays	VAP, Ha
877	Nana	R+ - Mild Nudity	8.54	242,680	107	313	spring	2,006	2,006	Wednesdays	VAP, Ni
1,210	NHK ni Youkoso!	R+ - Mild Nudity	8.31	326,968	238	258	summer	2,006	2,006	Unknown	NHK, Vi
1,453	Maison Ikkoku	R+ - Mild Nudity	8.19	18,243	358	2,472	spring	1,986	1,986	Wednesdays	Kitty Fi

Figure 11: Showing anime within the top 100 ranked anime that have a 'R+ Mild Nudity' rating

Sequentially, this part did not come after the previous step, but during my light analysis portion. I decided to put it in this section since I think it will clutter up the light analysis portion later. I ended up deciding to create multiple csv files containing relational data for all the measures that had a list as its value. What I decided to do was to split up the strings in the list and then link them to the ‘anime_id’ they were from, which can be seen in Figure 12.

I did this with ‘producers’, ‘studios’, ‘theme’, ‘genre’, and ‘demographic. I still kept the list in the original animeData, however I found using separate csv files for these relations made it easier to analysis in Tableau. As of right now, I am unsure if I will use Tableau solely, but I suspect that I will have to use python for some of the analysis and having all the values as a list seems more convenient.



anime_id	producers
5114	Aniplex
5114	Square Enix
5114	Mainichi Broadcasting System
5114	Studio Moriken
9253	Frontier Works
9253	Media Factory
9253	Kadokawa Shoten
9253	Movic
9253	AT-X
9253	Kadokawa Pictures Japan

Figure 12: Showing the structure for relational data

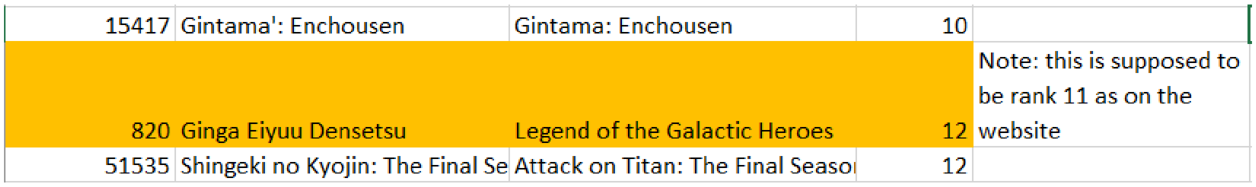
While cleaning my data, I did find more information on how scores and rankings are ordered on MAL which I think will be useful in the future for analysis when I have to consider scores and rankings.

*Weighted Score = (v / (v + m)) * S + (m / (v + m)) * C*
S = Average score for the anime/manga
v = Number users giving a score for the anime/manga †
m = Minimum number of scored users required to get a calculated score
C = The mean score across the entire Anime/Manga database

The "Top Upcoming" and "Most Popular" rankings are ordered by the number of users who have added the entry to their list. All other Top Anime and Top Manga rankings are ordered by weighted score, as calculated above.

Cleaning data from topAnime.json

This file contains the top rankings of anime which I collected the 'anime_id', 'title', 'title_english', and 'rank' from and put it into a csv file. I did a basic cleaning by looking at the csv file and saw some duplicates that I manually changed after double checking online.



15417	Gintama': Enchousen	Gintama: Enchousen	10	
820	Ginga Eiyuu Densetsu	Legend of the Galactic Heroes	12	Note: this is supposed to be rank 11 as on the website
51535	Shingeki no Kyojin: The Final Se	Attack on Titan: The Final Seaso	12	

Figure 13: Highlighted anime has the same rank as another anime

Something that I noted was that the ranking and the scores can change over time. As my data was from the month of October, there are some discrepancies from the website. Since that is the case, I only updated information according to the website if it was going to cause an issue during analysis like the above example.

I also noticed that there were many anime titles at the bottom of the csv file where there was no rank. I decided to get rid of them as this csv file was to see the anime rankings. If the 'anime_id' does not appear in the csv, I will know that there is no ranking for that anime.

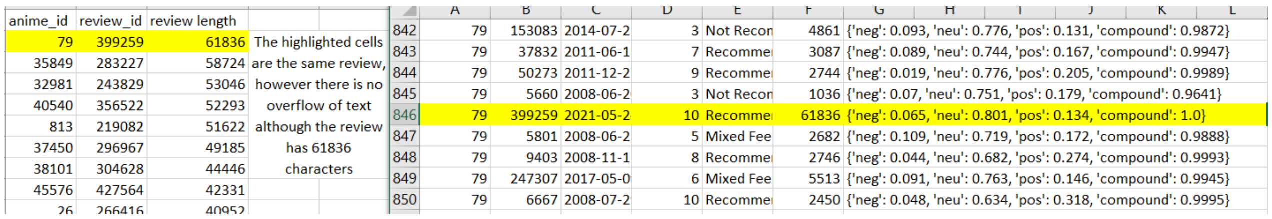
This is the point where I felt a bit silly as I had spent some time cleaning and fixing somethings. However, I realized that this information is all in my animeData already and I did not need this file that I was working on. I would have also had to join the or link the two files later on for analysis. At this point, I stopped working on the file and use my animeData file for the rankings.

Cleaning data from animeReview.json

I initially gathered the data is that I made a simple script to gather all the anime ids and put it into a singular csv file. What I did not realize is that the 'mal_id' value actually refers to the ID of the review and not the anime id, so I did not know what anime the reviews were for. I ended up having to re-request the reviews and create a json file for each 'anime_id'. Since I had filtered out some anime titles already in my animeData, I created a script that would collect all the 'anime_id's and put them into a text file. I only requested reviews from the anime ids that were in the text file. I also noticed that some of the json files were only 1KB in size so I decided to see the contents. Many titles did not have reviews, so I adjusted my python to skip over the json files that did not have any reviews.

After that, I cleaned all the json files and combined them all into one csv file. I wanted to know the sentiment of the review, so I used the nltk toolkit to get the sentiment scores for the reviews. When I looked at the scores however, I found that the neutral score was generally over 0.7, and the positive and negative scores were generally less than 0.2. What I realized is that the sentiment analysis for nltk seems to work better on shorter sentences, rather than longer reviews. I still want to explore the sentiment scores, so I decided to leave them in for now. However I feel that I may have to spend more time on the analysis and perhaps classify the data like seen in this medium article for example

While I was cleaning the data, I encountered the text from the review would spill into other cells. I was not sure why this happened. I decided to check if it was the length of the review that caused this but running a script to find the lengths of all the reviews and compare. However, the length was not the cause.



anime_id	review_id	review length	
79	399259	61836	The highlighted cells are the same review, however there is no overflow of text although the review has 61836 characters
35849	283227	58724	
32981	243829	53046	
40540	356522	52293	
813	219082	51622	
37450	296967	49185	
38101	304628	44446	
45576	427564	42331	
26	266416	40952	

	A	B	C	D	E	F	G
842	79	153083	2014-07-2	3	Not Recon	4861	{'neg': 0.093, 'neu': 0.776, 'pos': 0.131, 'compound': 0.9872}
843	79	37832	2011-06-1	7	Recomme	3087	{'neg': 0.089, 'neu': 0.744, 'pos': 0.167, 'compound': 0.9947}
844	79	50273	2011-12-2	9	Recomme	2744	{'neg': 0.019, 'neu': 0.776, 'pos': 0.205, 'compound': 0.9989}
845	79	5660	2008-06-2	3	Not Recon	1036	{'neg': 0.07, 'neu': 0.751, 'pos': 0.179, 'compound': 0.9641}
846	79	399259	2021-05-2	10	Recomme	61836	{'neg': 0.065, 'neu': 0.801, 'pos': 0.134, 'compound': 1.0}
847	79	5801	2008-06-2	5	Mixed Fee	2682	{'neg': 0.109, 'neu': 0.719, 'pos': 0.172, 'compound': 0.9888}
848	79	9403	2008-11-1	8	Recomme	2746	{'neg': 0.044, 'neu': 0.682, 'pos': 0.274, 'compound': 0.9993}
849	79	247307	2017-05-0	6	Mixed Fee	5513	{'neg': 0.091, 'neu': 0.763, 'pos': 0.146, 'compound': 0.9945}
850	79	6667	2008-07-2	10	Recomme	2450	{'neg': 0.048, 'neu': 0.634, 'pos': 0.318, 'compound': 0.9995}

Figure 14: The longest review did not overflow in the csv file indicating the issue is not related to length

You can see in the picture above that review_id 399259 has the longest review length, but it does not have any text overflow in the animeReviews csv file. In recorded the review_id of the reviews that overflowed, however I found that when I would run the cleaning script again, that different reviews would overflow. As I would only use the review text for the sentiment scores, I decided that for now I would clean the data again but without the review text. I would still be able to see the score and tags of the reviews and felt that it was enough for my purposes at the moment.

These are the important measures that I pulled out.

- review_id: the id of the review (named mal_id in the json data)
- date: the date the review was published
- score: the score for the anime that the reviewer gave
- tags: tags that the reviewer used for the anime
- sentiment scores: the sentiment scores

I loaded this into Tableau Prep to check for duplicates, which there were none. I did not see any other cleaning that I had to do for the data so I left it as is.

Initial Analysis

I decided to use Tableau for the initial analysis for my questions to see if my questions are feasible and what I need to consider for future analysis.

What is the distribution of reviews over time?

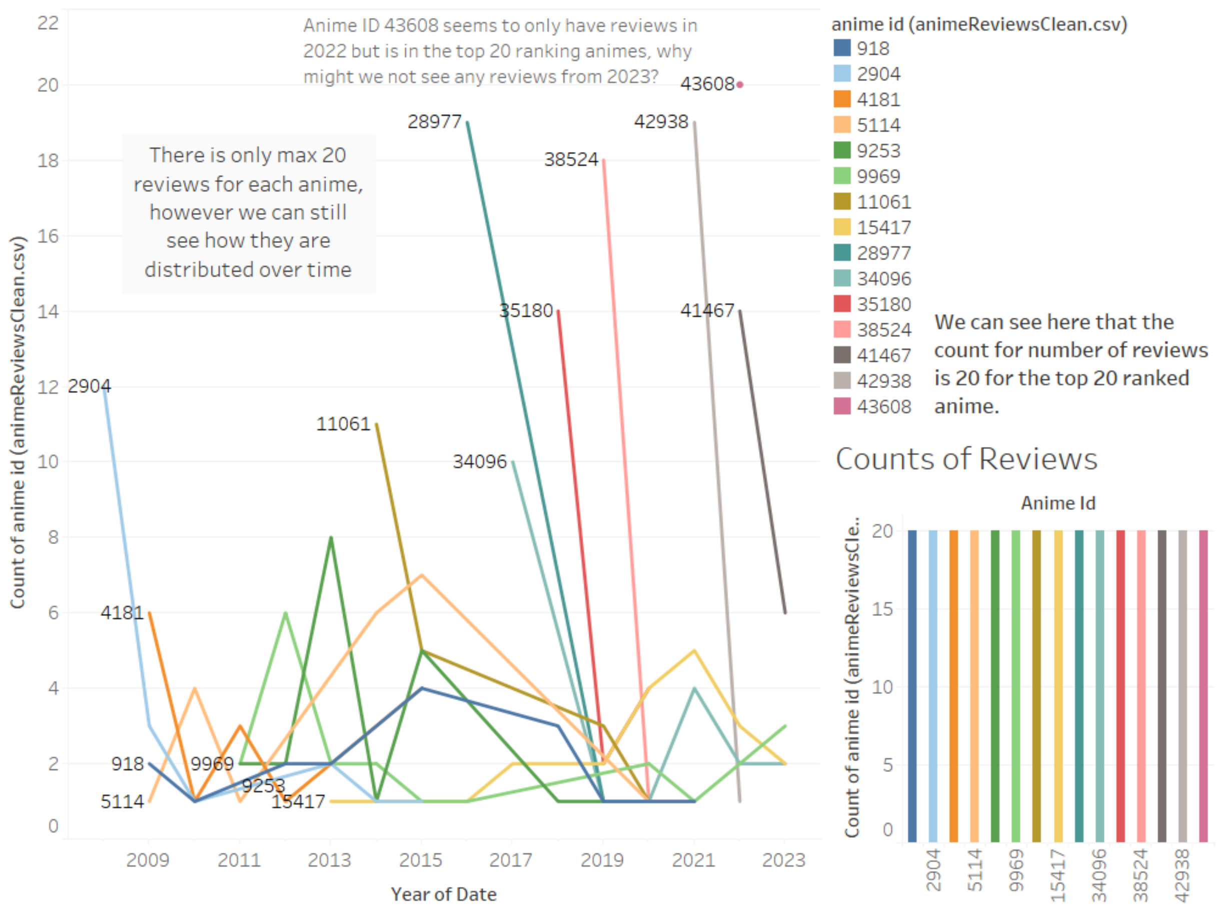


Figure 15: Visualization for distribution of reviews over time

After a simple visualization, I realized that I had only gathered the recent 20 reviews for each anime. While I would like to have all the review data for this initial analysis, I think this visualization does show we can see some distribution of reviews over time. Unfortunately, I may have to go back and gather the rest of the review data and clean it as well. However, I may be able to answer this question to a better extent.

What genres tend to have more reviews? What themes tend to have more reviews?

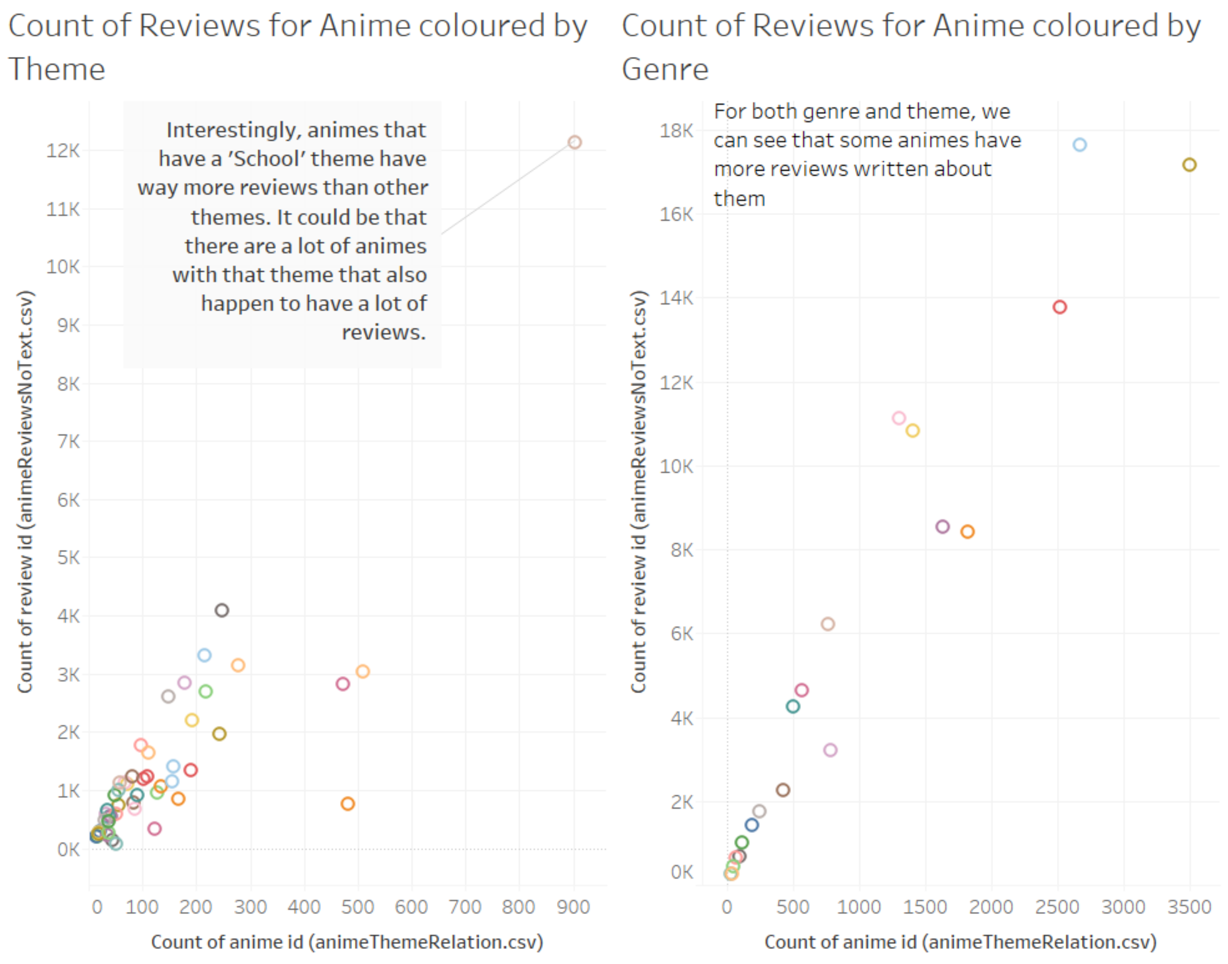


Figure 16: Visualization to identifying review counts for genres and themes

As mentioned above, I do not have all the reviews for all the anime, however I was still able to visualize which themes or genres get more reviews. While there are a lot of themes and review, I think this is a good starting point to explore this question more. Especially for themes, I may want to attempt to analyze clusters to see if there are any themes that are similar or get a similar number of reviews. If I had all the reviews, I might expect to see a distribution like the one above since although for each anime I only have a max of 20 reviews, the minimum is 0. Therefore, some themes will still have less reviews than others when we aggregate the anime data.

What animes tend to seem the most popular? Can we use reviews to see what animes are popular?

These two questions were separate at first, but I have decided to combine them as they seem to be answering a similar question.

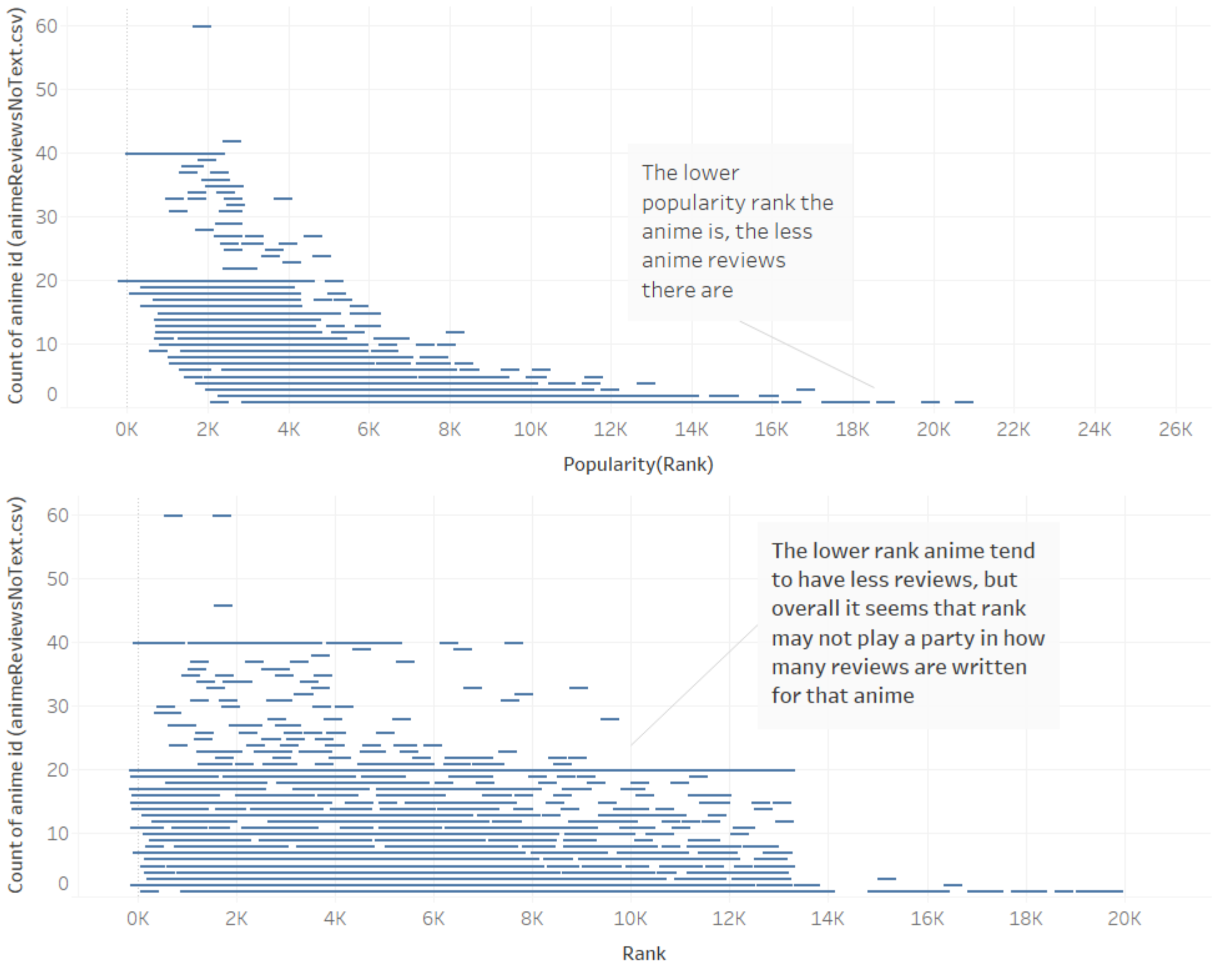
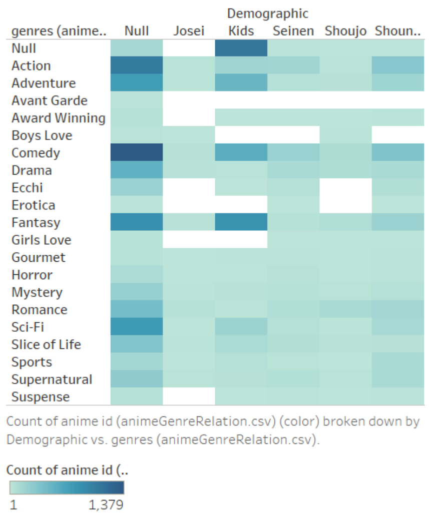


Figure 17: Visualization showing number of reviews for ranks and popularity

Note that #1 is the best rank for popularity and rank. The larger the rank or popularity number, the lower it is on the ranking. Using reviews, I find that there is no clear answer to answer this question for now. I think it does show potential analysis. I think going forward, I would like to explore if there are any measures that can determine the rank or popularity of anime. Perhaps attempting this question with more measures and classification could help me explore this question more.

Do anime themes tend to be part of a certain demographic or genre? E.g. Is Action more associated with the Shounen demographic?



Using this heatmap we can see the count of animes that fall into different demographic and genre categories. However it does seem that a lot of animes don't seem to be categorized in terms of demographic as we have a higher count of anime when the demographic is null and when the genre contains comedy. I would want to explore if I can categorize the demographic in some way, however it may seem inaccurate if not enough of the data is categorized in the end.

Figure 18: A heatmap of demographics and genres

Does the source material for the anime affect its viewership/reviews?

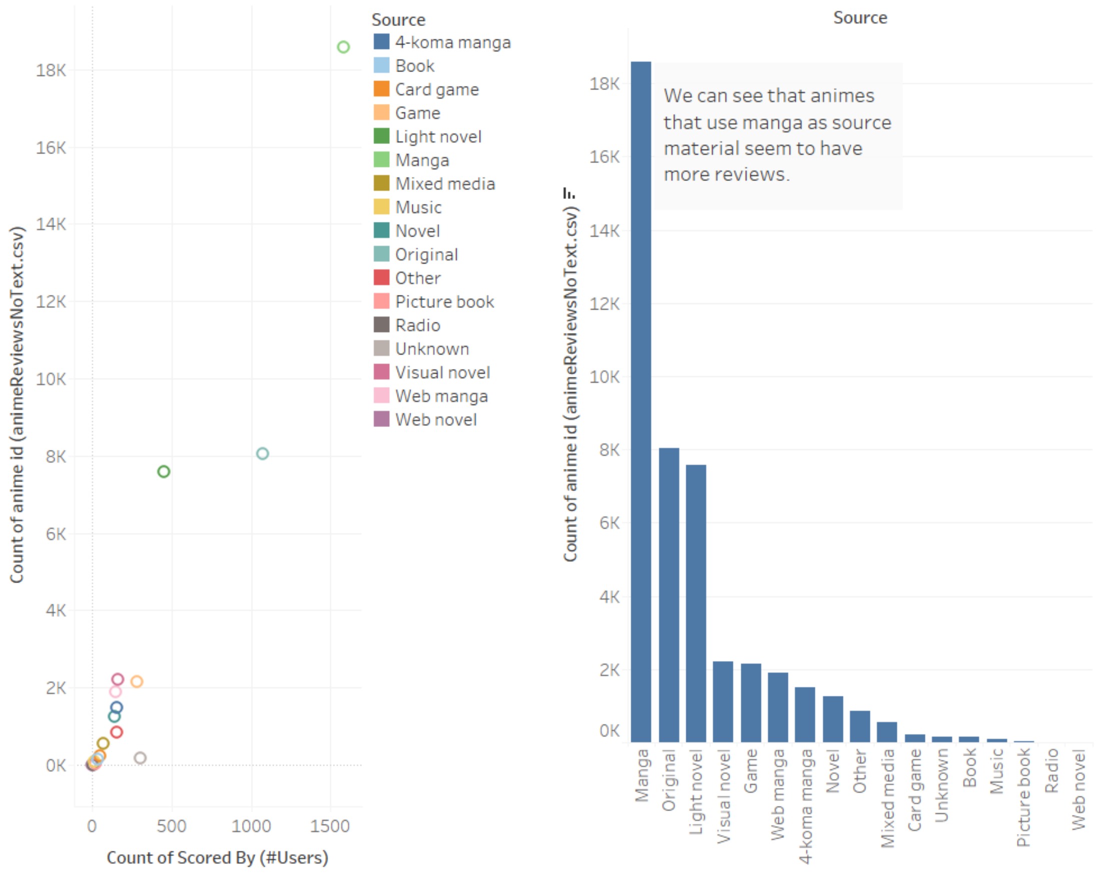


Figure 19: Visualizations showing source material, count users that scored anime, and count of reviews

Looking back on this question, I think determining viewership is difficult with my dataset. I do not have an actual metric that can count how many viewers an anime has. I used ‘Scored By (#Users)’ for now but I would not use it again as it does not actually have anything to do with viewership and it would be presumptuous of me to say that people who score anime are the only ones who watch that anime title. The bar chart does show that source material could affect how many reviews an anime has. Moving forward, I would explore how source material affects reviews, and possible rank instead of viewership.

As I have combined two of my questions, my newly proposed question is: **Are certain producers and studios associated with specific genres, themes, or demographics?**

From the visualizations in Figures 20. and 21. we can see that there is some information that producers tend to stick to certain themes or demographics. However, we can see that studios do many different themes, although we can see a bit more order from the studio and genres. My next steps for this one would to visualize genres with producers, and studios and demographics, and to also see if some producers pair up with specific studios. While I was also creating these visualizations, I did encounter a bunch of unknowns and nulls, so I would also like to explore those aspects.

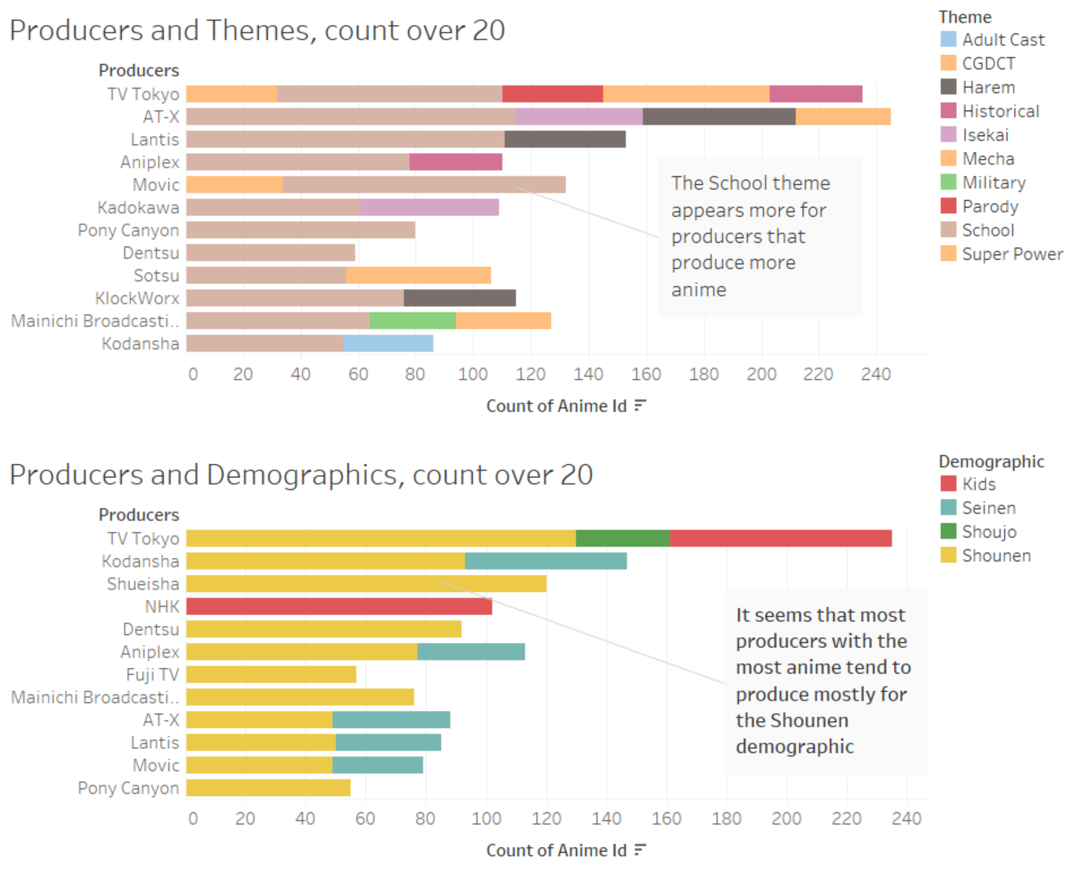
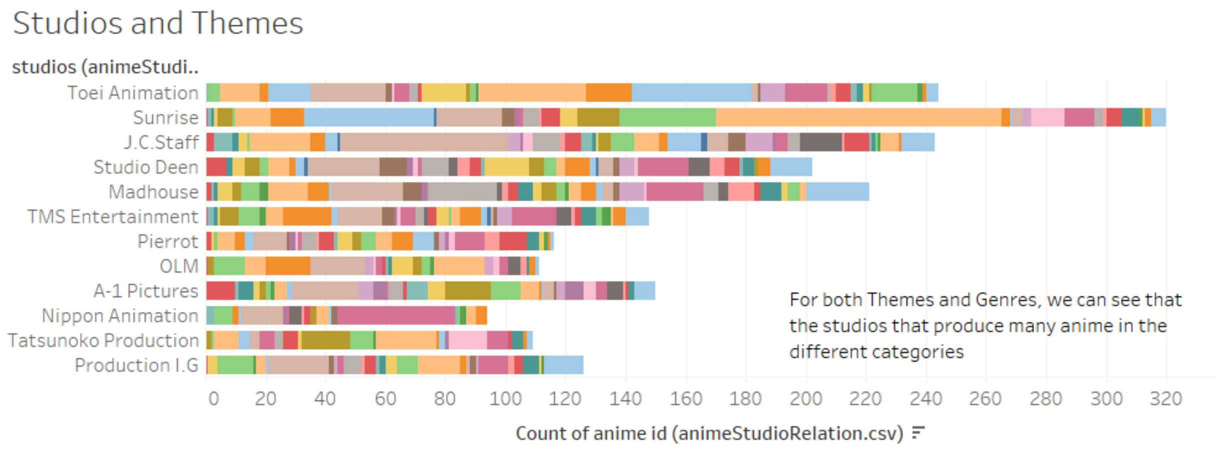


Figure 20: Bar graph of Producers and Themes, and Producers and Demographics



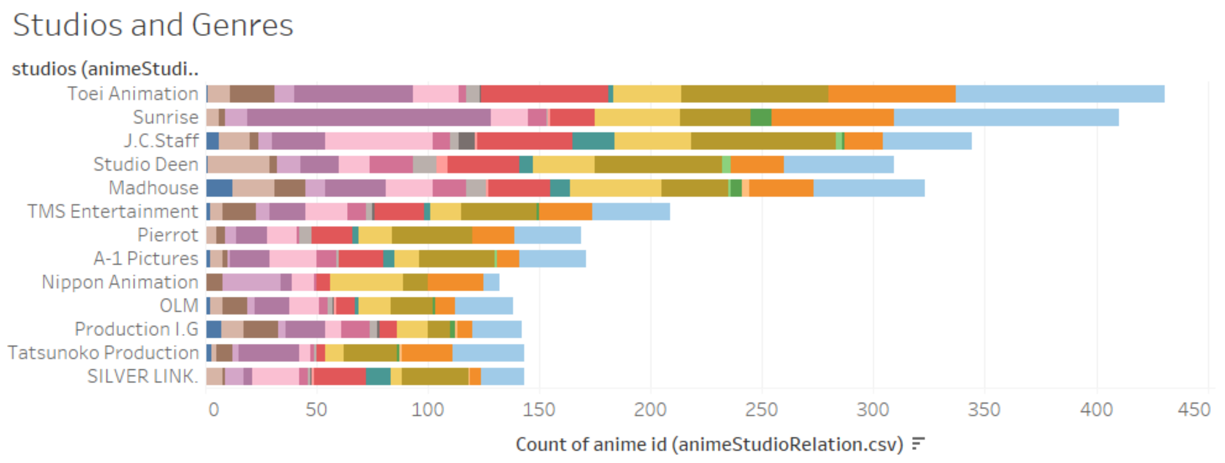


Figure 21: Bar chart of Studios and Themes, and Studios and Genres

Final Thoughts

Although I was able to produce some visualizations, I think that I will have to reconsider if I can gather more reviews if I want to more analysis regarding the reviews. For now, I will see if I am able to perform in-depth analysis on the reviews I currently have. In addition, I encountered many null and unknown values. I may see if there are other resources out there to help me find my missing information. If not, I will have to consider it as a factor when doing analysis. While my initial goal was to explore the popularity of anime the data, I realize now that my dataset might not have all my answers, however I am excited to see what information I can gather from it regardless. You can see my final analysis questions below. In addition, scripts that I have used to clean my data will be included in the Appendix.

Final Analysis Questions

- What is the distribution of reviews over time? With more reviews, does the distribution of reviews change?
- What genres tend to have more reviews? What themes tend to have more reviews? Are there similar genre or themes that get the similar amount of reviews?
- What animes tend to seem the most popular? Can we use reviews to see what animes are popular? What other measures can be used to predict the rank and popularity?
- Do anime themes tend to be part of a certain demographic or genre? E.g. Is Action more associated with the Shounen demographic? Out of the three, which measure has the biggest influence?
- Does the source material for the anime affect the amount of reviews? Does the distribution look different with more reviews?
- Are certain producers and studios associated with specific genres, themes, or demographics? What producers and studios tend to pair up together?

Appendix

cleanAnimeData.py

Code to get the measures from animeData.json

```python
import json
import csv

# This script is to extract the necessary information animeData.json
animeArray = []
header = ['anime_id','url', 'title','title_english','source','episodes','status','aired start','aired finish','duration(minutes per episode)'
          'rating','score','scored by (#users)','rank','popularity(rank)','season','year','broadcast day','broadcast time',
          'broadcast timezone','producer', 'studios','genres','themes','demographics']

# Open the JSON file for reading with UTF-8 encoding
with open('animeData.json', 'r', encoding='utf-8') as json_file:
    # Open a text file for writing
    with open('anime_clean.csv', 'w', newline = '', encoding='utf-8') as csv_file:
        csv_writer = csv.writer(csv_file)
        csv_writer.writerow(header)
        # Iterate through each line in the JSON file
        for line in json_file:
            # Load the JSON object from the line
            json_data = json.loads(line)

            # Extract the 'data' objects from the JSON data (assuming 'data' is a list)
            data_objects = json_data.get('data', [])

            # Iterate through each 'data' object
            for data_obj in data_objects:
                # Check the type, only want 'TV'
                animeType = data_obj.get('type', None)
                animeStatus = data_obj.get('status', None)
                if animeType == 'TV' and animeStatus != 'Not yet aired':
                    animeArray.append(data_obj.get('mal_id', None))
                    animeArray.append(data_obj.get('url', None))
                    animeArray.append(data_obj.get('title', None))
                    animeArray.append(data_obj.get('title_english', None))
                    animeArray.append(data_obj.get('source', None))
                    animeArray.append(data_obj.get('episodes', None))
                    animeArray.append(animeStatus)
                    animeArray.append(data_obj.get('aired', None).get('from', None))
                    animeArray.append(data_obj.get('aired', None).get('to', None))

                    # duration is either 'Unknown' or 'X min per ep', extract only digit or 'Unknown'
                    durationString = data_obj.get('duration', None)
                    if durationString == 'Unknown':
                        animeArray.append(durationString)
                    else:
                        durationInteger = [int(word) for word in durationString.split() if word.isdigit()]
                        animeArray.append(durationInteger[0])

                    animeArray.append(data_obj.get('rating', None))
                    animeArray.append(data_obj.get('score', None))
                    animeArray.append(data_obj.get('scored_by', None))
                    animeArray.append(data_obj.get('rank', None))
                    animeArray.append(data_obj.get('popularity', None))
                    animeArray.append(data_obj.get('season', None))
                    animeArray.append(data_obj.get('year', None))

                    # get broadcast information separated
                    broadcast = data_obj.get('broadcast', None)
                    if broadcast.get('day', None) == None:
                        animeArray.append('Unknown')
                        animeArray.append('Unknown')
                        animeArray.append('Unknown')
                    else:
                        animeArray.append(broadcast.get('day', None))
                        animeArray.append(broadcast.get('time', None))
                        animeArray.append(broadcast.get('timezone', None))

                    # get the producers as a list of just the names
                    producerNames = []
                    producers = data_obj.get('producers', None)
                    for producer in producers:
                        producerName = producer.get('name', None)
                        producerNames.append(producerName)
                    if len(producerNames) == 0:
                        animeArray.append('Unknown')
                    else:
                        animeArray.append(producerNames)

                    # get the studios as a list of just the names
                    studioNames = []
                    studios = data_obj.get('studios', None)
                    for studio in studios:
                        studioName = studio.get('name', None)
                        studioNames.append(studioName)
                    if len(studioNames) == 0:
                        animeArray.append('Unknown')
                    else:
                        animeArray.append(studioNames)

                    # get the genres as a list of just the names
                    genreNames = []
                    genres = data_obj.get('genres', None)
                    for genre in genres:
                        genreName = genre.get('name', None)
                        genreNames.append(genreName)
                    if len(genreNames) == 0:
                        animeArray.append('NA')
                    else:
                        animeArray.append(genreNames)

                    # get the themes as a list of just the theme names
                    themeNames = []
                    themes = data_obj.get('themes', None)
                    for theme in themes:
                        themeName = theme.get('name', None)
                        themeNames.append(themeName)
                    if len(themeNames) == 0:
                        animeArray.append('NA')
                    else:
                        animeArray.append(themeNames)

                    # get the demographics as a list of just demographic names
                    demographicNames = []
                    demographics = data_obj.get('demographics', None)
                    for demo in demographics:
                        demoName = demo.get('name', None)
                        demographicNames.append(demoName)
                    if len(demographicNames) == 0:
                        animeArray.append('NA')
                    else:
                        animeArray.append(demographicNames)

                    # write to the csv file
                    if animeArray[0] is not None:
                        # Write the mal_id to the text file
                        csv_writer.writerow(animeArray)
                    animeArray.clear()

# Print a message to confirm that the data has been written to the text file
print("Data values have been written to anime_clean.csv")
```

getAnimeRelations.py

Code to split relational data for anime data which has been cleaned with Tableau Prep

```
import pandas as pd

# Load the CSV Data
df = pd.read_csv("anime_clean_3.csv")

# Define the columns to be split and processed
columns_to_process = [
    ('studios', 'animeStudioRelation.csv'),
    ('producers', 'animeProducerRelation.csv'),
    ('genres', 'animeGenreRelation.csv'),
    ('theme', 'animeThemeRelation.csv'),
    ('demographic', 'animeDemographicRelation.csv')
]

# Loop through the columns and process them
for column, output_file in columns_to_process:
    df_copy = df.copy()
    # Split the specified column
    df_copy[column] = df_copy[column].str.split(',')

    # Explode the Data
    df_copy = df_copy.explode(column)

    # Clean and Further Process the Data (optional)
    df_copy[column] = df_copy[column].str.strip()
    df_copy = df_copy.copy()[['anime_id',column]]

    # Save the Cleaned Data
    df_copy.to_csv(output_file, index=False)
```

getMalids.py

Code to extract animeIDs to get the reviews

```
import csv

# Replace 'your_input.csv' with the path to your CSV file
csv_file_path = 'anime_clean.csv'
mal_id_column_name = 'anime_id'  # The column name containing 'mal_id'

# List to store 'mal_id' values
mal_ids = []

# Read the CSV file and extract 'mal_id' values
with open(csv_file_path, mode='r', newline='', encoding='ISO-8859-1') as csv_file:
    reader = csv.DictReader(csv_file)

    for row in reader:
        mal_id = row.get(mal_id_column_name)
        if mal_id:
            mal_ids.append(mal_id)

# Write the 'mal_id' values to a .txt file with one value per line
with open('malidsclean.txt', 'w', encoding='utf-8') as file:
    for mal_id in mal_ids:
        file.write(str(mal_id) + '\n')
```

getAnimeReviewsRateLimited.py

Code to gather the reviews based on anime ids

```
import requests
import json
import time
from datetime import datetime

# Define the Jikan API URL
api_url = "https://api.jikan.moe/v4/anime/{}/reviews"

# Get the current date and time
current_time = datetime.now().isoformat()

# For the rate limiting
requests_per_minute = 50
seconds_per_minute = 60

# Open the file containing anime IDs
with open('malids.txt', 'r') as id_file:
    request_count = 0  # Initialize a request counter
    for line in id_file:
        anime_id = line.strip()  # Remove newline characters

        # Calculate the time to wait between requests to achieve the rate limit
        time_to_wait = seconds_per_minute / requests_per_minute

        # Make the GET request to the Jikan API
        response = requests.get(api_url.format(anime_id))

        if response.status_code == 200:
            anime_reviews = response.json()
            # Save the reviews to a separate file for each anime
            with open(f'animeReviewID{anime_id}.json', 'w', encoding='utf-8') as json_file:
                json_file.write(json.dumps(anime_reviews))
            print(f"Reviews data for anime ID {anime_id} has been saved to {anime_id}reviews.json")
        else:
            print(f"Error fetching data for anime ID {anime_id}: {response.status_code}")

        # Sleep to respect the rate limit
        time.sleep(time_to_wait)

print("All reviews data has been saved to separate JSON files.")
```

cleanAnimeReviewNoText.py

Clean the anime reviews and append them to a csv file

```
import json
import csv
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download(["vader_lexicon"])

# Define the header
header = ['anime_id', 'review_id', 'date', 'score', 'tags', 'review', 'sentiment scores']

# Define the name of your text file
text_file = 'malids.txt'
animeArray = []

# Initialize the VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Open the CSV file for writing
with open('animeReviewsClean.csv', 'w', newline='', encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(header)

    # Loop through the numbers in the text file
    with open(text_file, 'r') as file:
        for line in file:
            # Read the number from each line
            number = line.strip()

            try:
                # Open the JSON file
                with open(f'animeReviewID{number}.json', 'r', encoding='utf-8') as json_file:
                    # Iterate through each line in the JSON file
                    for json_line in json_file:

                        # Load the JSON object from the line
                        json_data = json.loads(json_line)

                        # Extract the 'data' objects from the JSON data (assuming 'data' is a list)
                        data_objects = json_data.get('data', [])
                        if data_objects != []:
                            # Iterate through each 'data' object
                            for data_obj in data_objects:
                                animeArray.append(number)
                                animeArray.append(data_obj.get('mal_id', None))
                                animeArray.append(data_obj.get('date', None))
                                animeArray.append(data_obj.get('score', None))
                                animeArray.append(data_obj.get('score', None))

                                # Check if 'tags' is not empty before accessing its first element
                                tags = data_obj.get('tags', [])
                                if tags:
                                    animeArray.append(tags[0])
                                else:
                                    animeArray.append(None)

                                # Get the review and sentiment score
                                review = data_obj.get('review', None)

                                animeArray.append(review)
                                #animeArray.append(review)

                                sentiment = sia.polarity_scores(review)
                                animeArray.append(str(sentiment))

                                # Write the row to the CSV file
                                csv_writer.writerow(animeArray)
                                print(f"Wrote {number} to animeReviewsCleaned.csv")
                                animeArray.clear()
            except FileNotFoundError:
                # Handle the case where the JSON file doesn't exist
                print(f'File {number}reviews.json not found.')

print('All data has been saved to animeReviewsCleaned.csv.')
```