The goal of my project was to explore how music popularity differs across multiple platforms, Spotify, LastFM, and Genius. Each of these platforms measures popularity in a unique way. Spotify shows short term streaming trends, lastFM looks at long term listening and Genius looks at lyrical engagement by tracking how often people view song lyrics. My general question was whether the most streamed songs on Spotify were also on lastFM and mostly discussed or talked about on Genius. I also wanted to look at Genius Pageviews to see what genre was the most popular based on that information.

I collected data using several API's and libraries. Spotify was the main foundation of my dataset as well. My original plan was to use Spotify's Top 50 Global playlist, but I soon realized that Spotify's API restricted those charts, which caused a lot of trial and error to figure out why I was getting a 404 error. To work around this, I decided to use a constantly updated playlist called Top 100 Most Streamed Songs which was last updated October 24, 2025 (around the time I submitted my project). This allowed me to access the current and reliable streaming data while getting past the 404 error. Using the Spotify Web API, I was finally able to gather each song's title, artist, track ID, and popularity score.

However, Spotify's API doesn't exactly let you grab playcounts for each song, which meant I had to go back to our previous project and use that data. I went in with Selenium using Firefox to scrape the playcounts directly from the website. This wasn't as challenging as it was in the original Spotify project we did, but for some reason I was getting loads of timeouts and lots of delays which was very frustrating. I then had to research ways to slow down the website so it wouldn't overload which led me to "driver_implicitly_wait(5), which let each page load up to five seconds and for safety I added "time.sleep(0.5) to pause between requests to also avoid an overload. Once these playcounts were finally not timing out, I finally collected them, removed commas, converted the text into numbers, and stored them in a list for later.

I then gathered data from LastFM to get long term engagement. While Spotify focused on more recent trends, I also wanted historical ones as well. Using LastFM's API track.getINFO, I was able to retrieve two metrics for each song: the total number of playcounts and number of unique listeners. These values represented how long a song has been played over time and how many people listened to it. LastFM's API was pretty sensitive to names, so I use regular expressions or re.sub to clean up the titles and artist names by removing remix or live at. Having this step was pretty important because it helped match more songs to titles than it did without it.

For my third source of data, I used Genius API to measure the engagement. Genius doesn't really track listeners, but it does track pageviews. A song with millions of lyric pageviews might not be the most streamed song, but it does resonate more with people. I connected to Genius using the lyricsgenius library and API token. For each song, I searched Genius by artist and title. If nothing was found, it only looked up the title and tried to connect it to that artist. Once it was found, I extracted those pageviews using json. To make it even more accurate I wrote two functions clean_title and clean_artist. The first one (clean_title) removed parentheses, dashes, features, while the second one trimmed down anything after a comma. Having these smaller adjustments also helped a lot with matching songs to artists and prevented a lot of mismatches.

I also used time.sleep(1) because there were a few instances where it timed out between API calls. Some smaller or independent artists didn't have any Genius entries, which resulted in some missing data, but for the majority, the songs were found.

The Genius data showed something interesting. Songs with more emotional or storytelling lyrics had a larger pageview count than repetitive ones. Tracks like Bohemian Rhapsody or Believer had millions of pageviews even when their streaming numbers weren't as high as others. This showed that people turn to Genius when they want to understand or interpret a song more, and not just to listen to it. It showed an entirely different engagement to the community that was driven by meaning over popularity.

I also attempted to retrieve genre data for each artist using Spotify's artist endpoint. My main goal was just to categorize each song and compare the averages by each genre. Unfortunately, Spotify genre data is inconsistent, so many large artists don't have a genre tag. Even after cleaning the artist names and making more API calls for each ID, it still came back empty and I was really confused. I still was able to assign broader categories like Hip Hop, Rock, and Pop, which felt like enough to give the comparisons for my final results.

Once I collected all my data from each platform, I combined everything into a pandas dataframe. Each row represented a song, with columns for titles, artists, track ids, popularity, playcounts for spotify and lastFM, lastFM listeners, pageviews, and the genre. Creating this dataframe made the dataset much easier to work with and visualize. I ran around 50 sets of songs so that I could get a better understanding, missing data appeared as None or NaN.

I then created a few visualizations using matplotlib. The first charge was Average Genius Pageviews by Genre, which compared lyrical engagement across different genres I collected. Rock and Pop had the highest average Genius pageviews, showing that those audiences were more likely to look up lyrics based on those types of songs. Southern gothic and folk pop had some of the lowest, which was interesting because I feel like people would still look up those pageviews at the same quantity, but it could also be due to a lot of missing genres. My second visualization was my scatter plot comparing Spotify popularity with lastFM playcounts. This chart showed that many viral songs have higher Spotify scores but average lastFM counts, meaning they trend quickly but don't last long. Meanwhile, older songs had lower Spotify scores, but a lot higher lastFM playcounts. My third visualization focused on Top 10 Songs by Genius Pageviews, ranked from most viewed lyrics on Genius to least. The top results started with Ed Sheeran's Shape of You and the last one on that list was Love Yourself by Justin Bieber. This meant that Shape of You was more emotional and drove more engagement while Love Yourself wasn't as much. My final chart compared Average Spotify vs LastFM Playcounts by Genre, showing that Pop even in different categories like soft or folk dominate.

Several parts of my project worked pretty well. The combination of my API's were a little tricky at first but ended up sort of working. Selenium finally scraped my data. Using regex helped clean titles and artists which helped for my searches. Using pandas to structure and combine everything made my analysis easier to manage. The visualizations showed the differences in

popularity, short/long term streaming, loyalty, and curiosity across different genres. The Genius pageview portion, in particular, stood out to me as the more insightful part of the project. It proved that songs can still remain relevant even if they fall off the streaming charts because people will still listen to them and continue to look up their meaning.

There were also parts that didn't work as well. Spotify's genre data was mostly incomplete, and I couldn't use the official playlists that I wanted because of their restrictions. Genius and Spotify also timed out a lot, but I realized this was more of an internet issue rather than a debugging one. Some songs, like indie or niche ones were found and skipped. I also noticed that each platform defines their popularity on a different scale, which made it harder to compare the raw numbers. Instead I tried to focus on patterns and trends.

Throughout the entire process, I used a few resources to learn and troubleshoot. I relied a lot on chatGPT and the debugger inside jupyter notebook to understand what I was doing wrong and how I could improve on it. I also had to research different ways to make my charts look nicer and more diverse. I also referred back to my notes from class a lot which helped a lot for the main center of my code.

The results showed me that popularity in music depends on where and how you look at it. Spotify represents what people are listening to right now, lastFM captured what people listened to over time, and Genius showed what songs deepened a connection with a person. Pop songs mainly dominated the charts, while rock and hip hop had a lot longer engagement. The songs with the most Genius pageviews were always the most popular either, they were typically the ones with the most interesting lyrics.

There were some things I also couldn't finish. I wasn't able to fully retrieve genre data for every artist which was unfortunate. Genius searches failed a lot also because of timeouts or missing entries that overloaded the browser. I also couldn't use Spotify's official charts because of their restrictions which was also very unfortunate. Despite all this I was still able to overcome my challenges, even if I didn't collect the data I exactly wanted.

If I continued this project, I would make a lot of improvements. I would first add caching to save my API results locally so I wouldn't have to redownload them every time. I would also include more streaming platforms to expand on my comparison. I would also use databases like MusicBrainz to complete my genre tags.

Overall, I thought the project went pretty well considering my drawbacks and limitations, and it also taught me how to combine different types of platforms into one dataset more thoroughly. It showed me that popularity isn't just about how many streams you get, but about engagement and connection to the song as well.