**Matthew Spitulnik**

**IST 769: Advanced Big Data Management**

**Top Songs Comparison**

**Introduction**

This project aims to utilize Spotify's API and the spotipy python library to obtain music-based data from Spotify for analysis. This will be accomplished with big data tools like Apache Spark and Neo4j to build a pipeline to extract, transform, and load the data into formats suitable for analysis. Neo4j will be specifically used to better observe and visualize the relationships between different songs (tracks), artists, playlists, and musical genres.

**Data Collection**

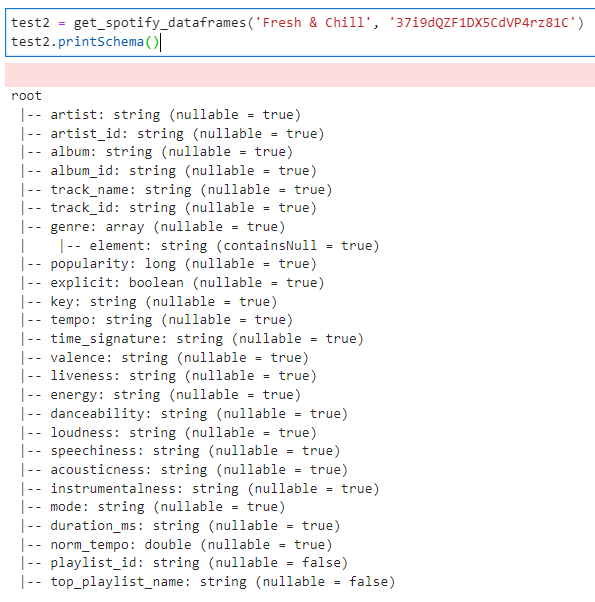
To collect the data for this project, 4 original functions were defined that utilized different built-in spotipy functionalities to bring the data in, do a little bit of cleanup, then put it all together so it could then be transformed as necessary by different tools. The 4 original functions were:

*get\_top\_playlists:* This function took in a list of country codes, then entered them each into the *“sp.featured\_playlists”* function, producing the top featured playlist for each inputted country. The following demonstrates the function being used and the resulting output:

*get\_playlist\_tracks:* This function takes in a playlist ID and then uses the *“sp.playlist\_tracks”* function to look up all of the tracks for that playlist. The resulting output contained a lot of information (much of which was not needed), but the next original function helped to further parse this data.



*get\_spotify\_dataframes:* This function takes in the name of a playlist and its playlist ID, then inputs it into the *get\_playlist\_tracks* function to get all of the tracks for the playlist. The required information for each track is collected and constructed into a spark data frame. *get\_spotify\_dataframes* also uses the track info with the “*sp.audio\_features”* command to obtain additional audio features.



*make\_sp\_dataset:* This function allows for all three of the previous original functions to be combined and iterated over so that large amounts of data could automatically be collected. *make\_sp\_dataset* takes in a list of country codes, which are inputted into the *get\_top\_playlists* function to collect the top playlist info for each county. That playlist info is then inputted into *get\_spotify\_dataframes,* which uses *get\_playlist\_tracks* to collect all of the individual track information. For each country code entered, the list of tracks from that country’s top featured playlist is collected and compiled into a spark data frame, which is combined with the info from the other country codes that were entered, resulting in one large dataset.



Not all countries currently have featured playlists, so a quick function was designed that could test which countries do, then add those country codes to a list for future use. The list consisted of 181 countries that information could be collected from. When trying to use that list all at once with the *make\_sp\_dataset*, it was determined that the Spotify API had limits to how much it could be accessed from a source within a certain timeframe. Once that limit was reached, the source was blocked from gaining access for a certain amount of time. Because of this, additional measures were built in for staggering out the data collection, while also making sure there wasn’t too much of a delay that could lead to the script timing out. Due to these limitations, while the functions were designed to write the data directly into Spark and could be moved entirely on a big data pipeline, in order to avoid potential loss of access to the API and maintain a single source of truth across all platforms, the Spark Dataframe was written to a json file and moved the json file to the respective platforms. Overall, the 181 countries with featured playlists were used to collect the data for 17,040 songs for analysis.

**Data Dictionary** [**(Source)**](https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features)

Below is a list of the key audio features that were collected and used for this project:

acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

duration\_ms: The duration of the track in milliseconds.

energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

explicit: Whether the track has explicit lyrics (true = yes it does; false = no it does not OR unknown).

genre: The different genres that the artist of a song is regularly associated with. Individual tracks did not have genres associated to them, so it was assumed for the confounds of this project that a track would fall under the genres of the artist.

instrumentalness: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

key: The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1. Range: -1 - 11

liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides a strong likelihood that the track is live.

loudness: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlation of physical strength (amplitude). Values typically range between -60 and 0 db.

mode: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

norm\_tempo: Normalized version of the "tempo" feature.

popularity: The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently. Note: the popularity value may lag actual popularity by a few days: the value is not updated in real time.

speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

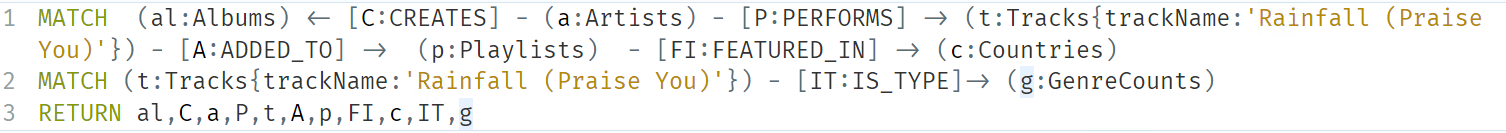
tempo: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. Example value: 118.211

time\_signature: An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

**Neo4j Analysis**

Neo4j is a very useful tool for analyzing this music data since multiple relationships exist within the data that Neo4j can help explore. Artists perform tracks, which are found on albums, which are made by the artists. Tracks are added to playlists, which are featured in certain countries. Tracks fall under a certain type of genre. The image below shows an example of the relationships involved with the song titled “Rainfall (Praise You)”. The song is performed by Tom Santa and is found on the album also titled “Rainfall (Praise You)”, which was also created by Tom Santa. The song was added to the playlists “Dance Party” and “Dance Hits”, which are featured in Azerbaijan and Argentina, respectively. “Rainfall (Praise You)” is considered to be the genre “stutter house”, and since the song is on playlists featured in Azerbaijan and Argentina, it is also noted that those countries have a connection to that genre. Tom Santa performs a song that is on two playlists, so the relationship showing how many playlists an artist can be found on is also displayed.



A diagram of different colored circles

Description automatically generated with low confidence

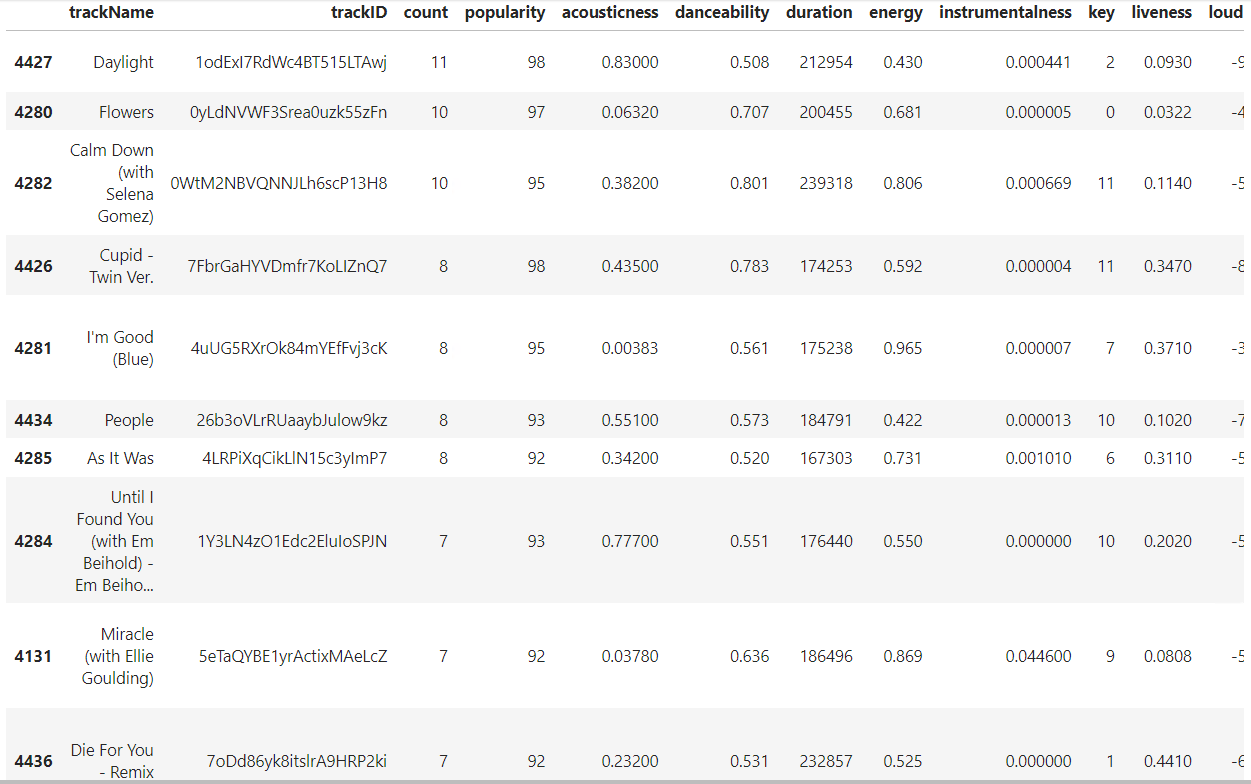
Within the collected data, certain elements also show up multiple times across different rows, with only slight differences in each row. One playlist is comprised of certain tracks that don’t change, but the playlist is featured in different countries. Artist information doesn’t change, but they create multiple albums and perform different songs. Track features don’t change, but a track is found on multiple playlists. Some songs even have the same name, but may be a completely different song or a different recording of the same song (studio vs. live, regular vs. acoustic, etc.). Neo4j provides an easy way to separate the data into related segments while consolidating it down to the unique, distinct data points, allowing for the required properties to only be accessed as they are needed. The diagram below shows how the data was broken up into different node labels within Neo4j, the properties of the nodes in each label, the relationships between each label and the relationship type (direct or merged), and how many nodes each node label contains. Although the original dataset contained 17040 rows, each row representing a track on a playlist featured in 181 countries, there were actually 6343 unique tracks, 75 unique playlists, 4384 unique artists, 5990 unique albums, and 1218 unique genres.

A picture containing text, screenshot, diagram, parallel

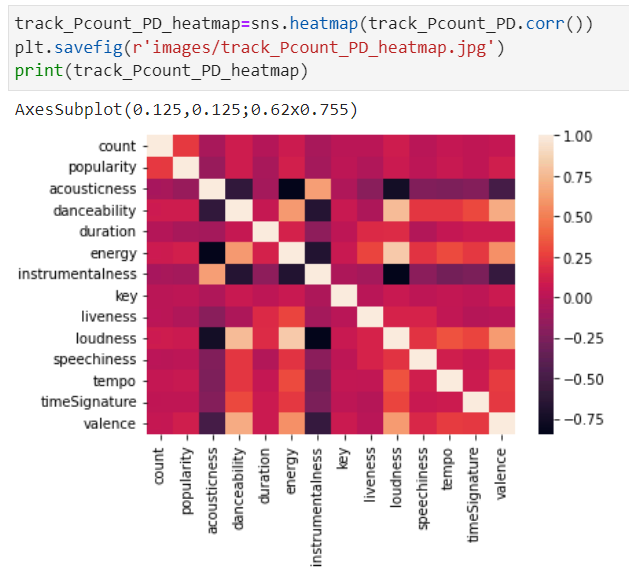
Description automatically generated

After breaking out the data in neo4j, it could then be imported back into spark to perform some analysis using the different relationships. The tracks that are found in the top ten greatest number of playlists all predictably have popularity scores in the nineties.

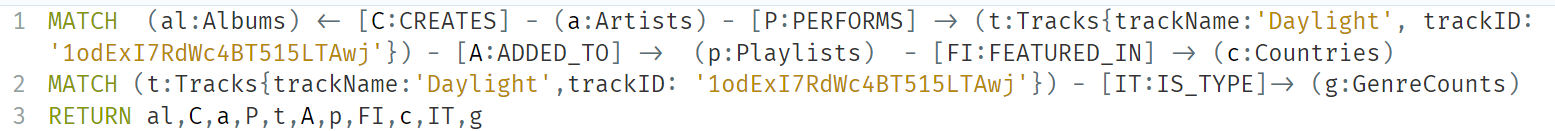
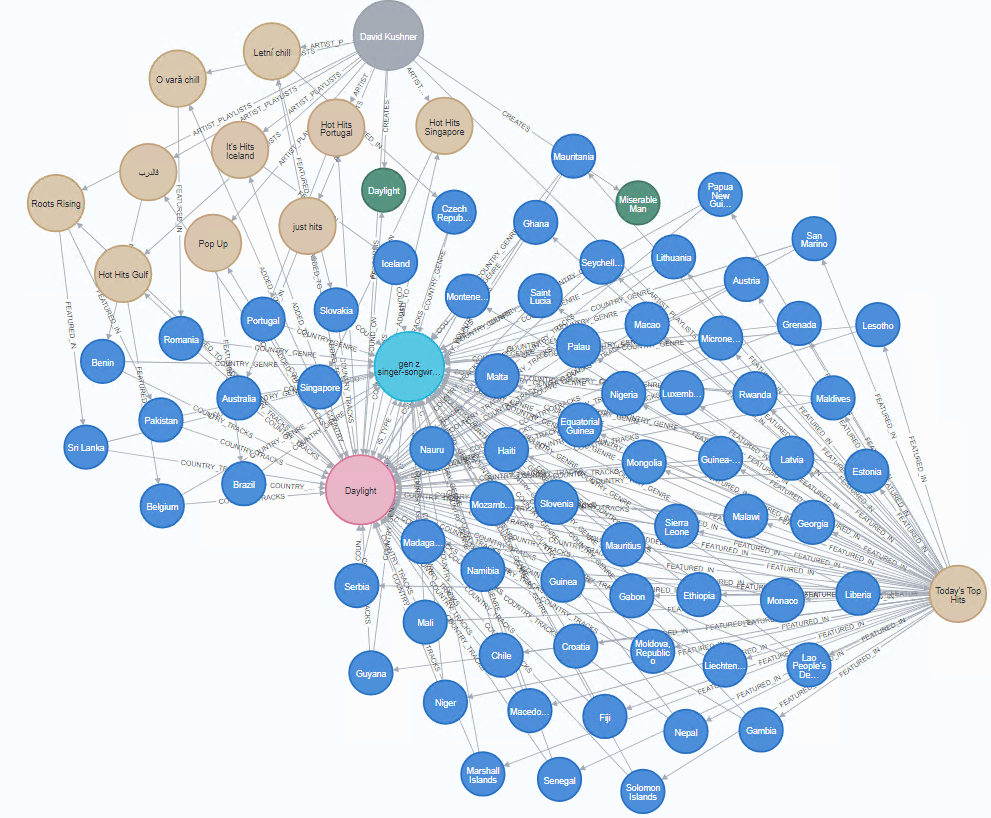




It was surprising to see however that the popularity feature was not highly correlated to the number of playlists a track was found on, although it was still more correlated than any other feature.

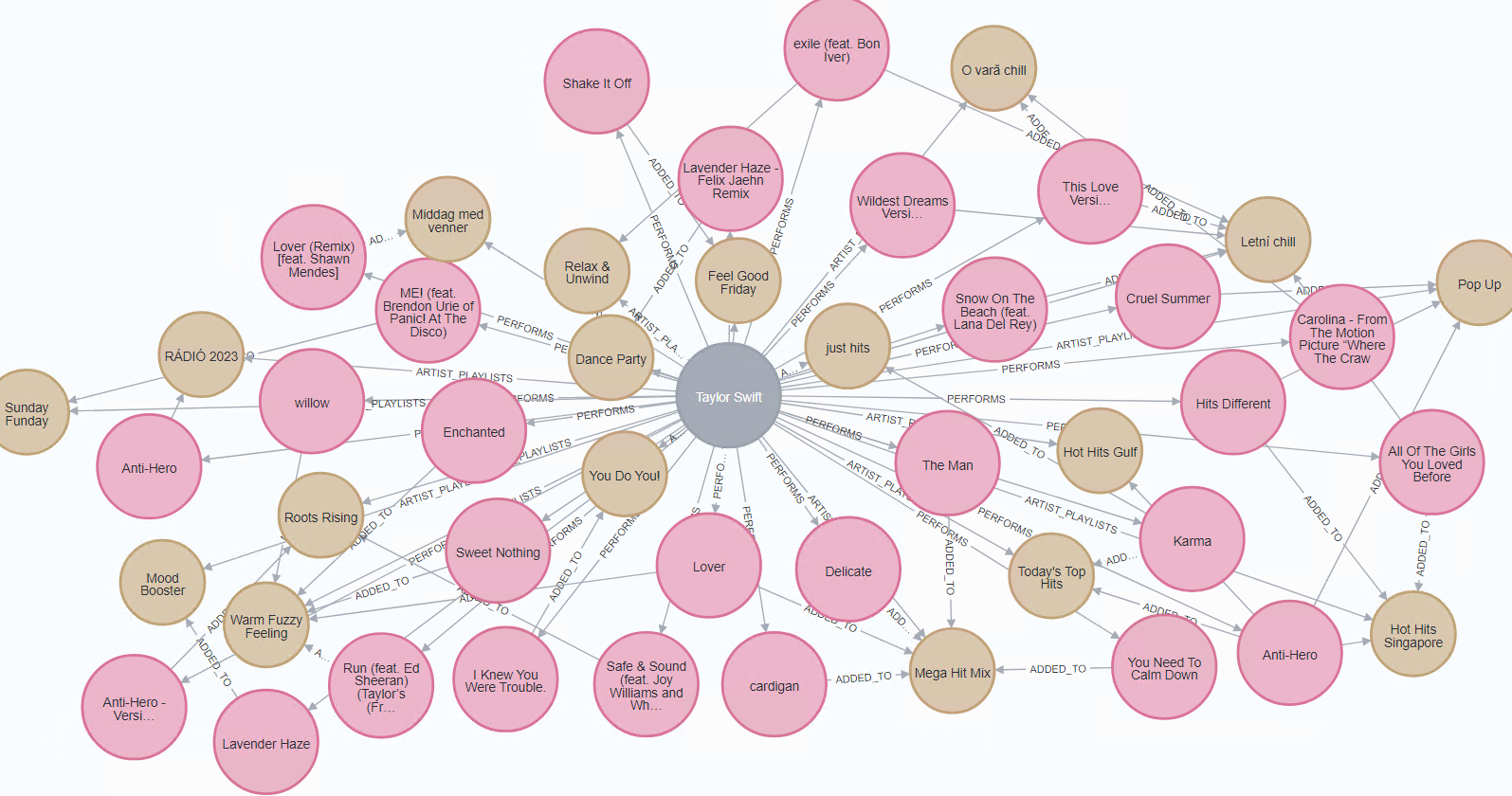


The Neo4j graph for the song that appears on most playlists, “Daylight” by David Kushner, is also much more connected to other elements than what was seen when looking at the “Rainfall (Praise You)” track. In fact, the trackID needed to be included in the query, since a different track titled “Daylight” also existed within the data.

When the data is broken down to look at things from the artist level, the popularity feature seems to play even less of a role in how many songs an artist has within the data. Taylor Swift has the most tracks across the 75 playlists with 28 in total. The below image shows the relationships between her songs and the playlists they were added to:



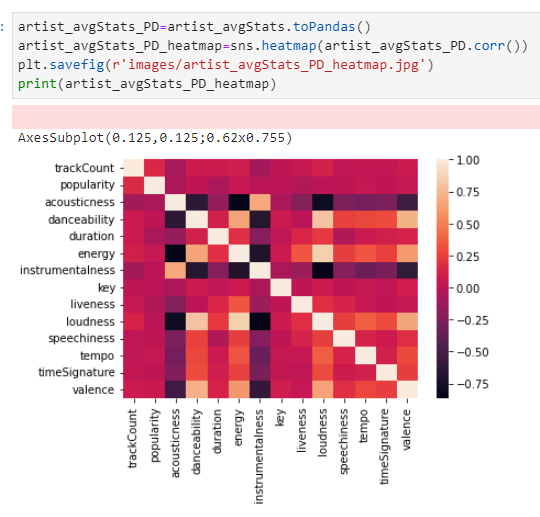


The average popularity across those 28 songs though is only ~80.89, which is still a high number, but not as high as the scores for the top ten tracks found across the playlists. In fact, none of the artists found in the top 10 for most tracks had an average track popularity score above 85. Most of the artists for the top 10 tracks have pretty recognizable names, which may show that their songs are being added to playlists because of name recognition regardless of how popular the songs are.





No audio features really seem to correlate to how many songs an artist has across the playlists or the popularity of their songs:



If the songs are broken up by the individual genres, the popularity of the songs seems to be even less relevant. When looking at the top 10 genres in terms of song count in that genre (ignoring the songs that did not have a genre associated with them), none of the genres have a popularity score over 76. There does seem to be a slight correlation, however, between the popularity of genres and the valence of the songs in that genre. This would mean genres with songs that are positive are going to be slightly more popular than genres without as much positive music.

