Modeling Spotify Data

Maura Keith

Northeastern University, Boston, MA, USA

# Abstract

The purpose of this database project was to represent a few Spotify user accounts and explorer patterns in the data. I was able to understand some key relationships and patterns between artists, albums, songs, and users through compiling data, creating a database, and answering questions with queries.

# Introduction

Spotify is undeniably one of the popular music streaming platforms in today’s market, if not the most popular. Their platform allows for users to make themed playlists for listening and contains most every song that has ever been published; however their popularity comes from their sophisticated artificial intelligence. Based upon users’ individual listening behavior, Spotify suggests new music to try, taking into consideration the listening patterns of other users that have similar taste [1]. Since there’s tens of millions of Spotify users, there’s a massive amount of data to be mined, and Spotify’s results are fascinating.

My approach to this project was inspired by Spotify’s truly exceptional use of data. As I’m unable to access listening history for all of the tens of millions of users and don’t have the means to analyze that big data, I wanted to model my own Spotify account. I chose to add some data from other accounts as well so that I could compare data between users.

In this database that I have compiled, I’m able to look at every song in my playlists. Each song belongs to one album, and each album belongs to one artist. Playlists are public and created by one user, and they can contain many songs. I used public data provided by Spotify to include the popularity, release date, key signature, length, and many other entities that make music unique. In entering this data, my mind rushed through many questions that could be answered with queries in this database: Who out of my friends has the most generic music taste? Is there a month that tends to have the most album releases? What is the distribution of major and minor keys?

# Database Design

There are two main categories of data that I used to create my database: user data and musical data. User data represents a user such as myself and the contents of my playlists, and musical data describes the music itself.

A screenshot of a cell phone

Description automatically generated

**Figure 1:** ER diagram for Spotify database

The collection of user data was fairly simple, as I only modeled four users; since the musical data behind each user was quite hefty and needed to be hand-compiled, four users were sufficient. Playlists were each named and connected to their creator via a user ID. The contents of each playlist are represented by a separate table, playlist\_has\_song, which lists each song ID and its corresponding playlist ID.

Regarding musical data, the bulk of it came from songs themselves, which each had an ID, title, album, time signature, popularity on a 100-point scale, and so on. Songs are connected to artists through albums; each song has one and only one album, and each album has one and only one artist. For these purposes I chose to assume that every album has one artist and disregarded extra artists in songs that are “featuring Bruno Mars,” for example; I chose the primary author or group to represent these songs. Albums have popularity scores and release dates, and artists have popularity scores. I reached out to the owner of the musicaldata.com, where I collected my data, and he reported that popularity index “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.” This means that if two songs have the same number of plays, but one has been played more often recently, it will be considered more popular [4].

# Data Sources and Methods

As mentioned, there were two main data types used in the creation of this database. All user data was collected via the Spotify app or Spotify Web Player, and I used Microsoft Excel to record this [5]. This data collection applied to the user table, as I assigned a user ID to each person. Similarly, I assigned each playlist an ID, recorded its name, and recorded the user ID of its creator. I also created a spreadsheet of an ID’s and titles for songs, albums, and artists. At this step I also created spreadsheets for playlist\_has\_song, user\_follows\_user, user\_follows\_artist, user\_likes\_playlist, and user\_likes\_song, because these entities are specific to each user and found on their Spotify accounts.

The second half of data collection consisted of recording musical data; this was filling in the data columns omitted from the Spotify app such as time signature or popularity. This was also when I recorded genres and their relationships to each artist. All of these values came from musicaldata.com, which pulls its data directly from Spotify [3]. Besides learning that my 80’s playlist was mostly composed of songs from the 70’s, I learned more about my musical taste as I saw the data behind it unfold. This was definitely the most tedious step, as every song, album, and artist has hand-recorded data to describe it.

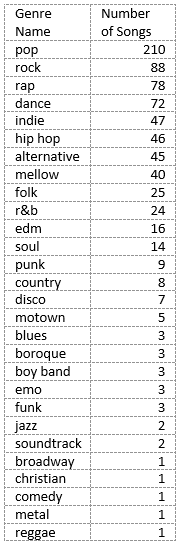
It’s worth noting weaknesses in the musical data that was collected. Spotify publishes data for all of their music, so I had access to everything I needed; there was no such thing as an obscure song that was missing from the data. However, given the vast number of songs, albums, and artists that are represented on Spotify, the data provided was auto-generated and therefore not always accurate. Specifically, the key signature and mode (major or minor), tempo, and time signature were identified by Spotify as potentially inaccurate. These inaccuracies can be caused by key or tempo changes within a song, in which case these values are estimated based on how much of the song is in each key or tempo. Also, the algorithm used by Spotify will simply compute incorrect values sometimes. Key signatures were reported in a sharp-only format as well; recall that the note C# is the same as D♭, for example. Since sharps and flats are nearly interchangeable in this sense, everything was reported as a sharp. As well, the exact release date was not always available for every album. This occurred more often for older albums, where the year was specified but not the month and day; in these cases the date defaulted to January 1st [2]. These weaknesses don’t necessarily cause any issues, however it is important to be aware of and address them when applicable, in queries for example.

In creating the database, I simply ran my setup script to create all of the needed tables and imported the data I recorded to fill them using the Table Data Import Wizard. All Excel files first needed to be converted to .csv files before being imported into MySQL Workbench. Special characters exist in the song, album, and artist tables and can’t be imported using the standard utf-8 encoding. This issue is worked around by importing with latin1 (iso8859-1) instead. All other tables don’t include special characters and may be imported in either utf-8 or latin1. Due to foreign key constraints, I imported data in the order that tables were declared: user, genre, artist, playlist, album, song, user\_follows\_user, user\_follows\_artist, user\_likes\_song, user\_likes\_playlist, artist\_has\_genre, and playlist\_has\_song. It’s not required to import data in this order so long as tables with foreign key constraints are respected. For example, album data needed to be imported after artist data, because the album table references artist ID as its foreign key.

# User Cases

Undoubtedly the most exciting part of creating this database was asking it questions via queries. I answered many questions in my query file, but the most significant are detailed below. It’s worth noting that the answers to all of these questions are specific to the music in my dataset, and they don’t necessarily reflect the patterns and behaviors of all music that exists in the Spotify platform itself. However, because this dataset it made up of my own music, along with the music of a few others, the results are specific to our own musical tastes which produces fascinating results.

I first asked some simple questions to get to know my dataset better. Such as, what is the most common genre?



select genre\_name as 'Genre Name',

count(genre\_id) as 'Number of Songs'

from artist\_has\_genre join genre using (genre\_id)

group by genre\_id

order by count(genre\_id) desc;

**Figure 2:** Vizualization of genre distribution

**Figure 3:** Genre distribution values

I wondered who, out of the four people I modeled data for, has the most generic music taste? In other words, who has the highest average popularity index of all the songs in their playlists? Unsurprisingly, my dad’s large collection of 80’s music yielded a lower average popularity than the rest of us, giving him the most obscure and unique music taste.

select user\_name as 'User', round(avg(popularity)) as 'Average Song

Popularity'

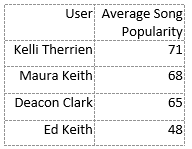
from user join playlist using (user\_id)

join playlist\_has\_song using (playlist\_id)

join song using (song\_id)

group by user\_id

order by avg(popularity) desc;



**Figure 5:** Average song popularity

values by user

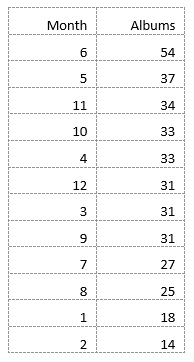
**Figure 4:** Vizualization of average song popularity by user

Is there a month that tends to have more album releases? The answer to this question is more so a representation of my music than Spotify overall; I wasn’t surprised to find June in the lead with May just behind, because it’s the prime season to release upbeat summer songs.

select month(release\_date) as 'Month', count(album\_id) as 'Albums'

from song join album using (album\_id)

where not(month(release\_date) = 1 and day(release\_date) = 1)

group by Month

order by Albums desc;

**Figure 7:** Album release

values by month

**Figure 6:** Vizualization of album releases by month

The final query I found noteworthy addresses major and minor distributions. Of every song I have in my database, what percentage of them are in major keys? In honor of my father and his vast collection of songs by Duran Duran, I wondered how this specific group compared to the distribution for all songs in the database. I’ve learned that Duran Duran songs tend to be written in major keys slightly more often than other songs, meaning that their music generally has a more uplifting and happy tone to it.

select key\_mode as 'Key Mode',

round (100 \* count(song\_id)/(select count(song\_id) from song), 1) as 'Percentage of All Songs'

from song

group by key\_mode;

select key\_mode as 'Key Mode',

round (100 \* count(song\_id)/(select count(song\_id)

from song join album using (album\_id)

join artist using (artist\_id)

where artist\_name = 'Duran Duran'), 1)

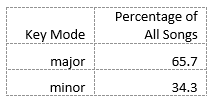
as 'Percentage of Songs'

from song join album using (album\_id)

join artist using (artist\_id)

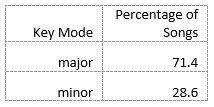
where artist\_name = 'Duran Duran'

group by key\_mode;



**Figure 9:** Major and minor

distribution for all songs



**Figure 10:** Major and minor

distribution for Duran Duran

songs

**Figure 8:** Comparison of all music with Duran Duran major and minor distribution

# Conclusions

The purpose of this database was to model the music from a few Spotify accounts and draw interesting connections from it. This was definitely fulfilled through its creation and the queries that followed. Not all queries written for the database were included here, but all were fascinating nonetheless. In compiling a collection of spreadsheets that covered many different aspects of music, I gained a stronger appreciation for what makes a song unique and the distinct patterns that we unknowingly make by simply listening to the music we enjoy. However the largest accomplishments were the queries that answered interesting questions and brought together all working aspects of this database project.

If I were to further my work with this database, I would definitely include more data. Ideally, I’d write a program that reads and enters data from Spotify and musicaldata.com into spreadsheets, so as to eliminate the manual typing aspect. Once this efficiency aspect is addressed, the Spotify database would quickly grow, making the answers to my queries more representative of music in general rather than my own taste. With more users, I would be able to answer questions such as: Who has the most liked playlists and therefore, subjectively, the best taste in music? I could study users’ average song popularity for the most followed playlists to find an “ideal” popularity index that users tend to prefer. Questions such as these are just scratching the surface of connections that can be studied within this musical database. Although I haven’t created a database even close the scale of Spotify itself, I’ve logged data from several users to gain a stronger understanding of database design and also learned to appreciate the beauty of data in our everyday world.

# References

[1] B. Marr, “The Amazing Ways Spotify Uses Big Data, AI And Machine Learning To Drive Business Success,” *Forbes*, 30-Oct-2017. [Online]. Available: http://www.forbes.com/sites/bernardmarr/2017/10/30/the-amazing-ways-spotify-uses-big-data-ai-and-machine-learning-to-drive-business-success/. [Accessed: 17-Jun-2020].

[2] “Get Audio Analysis for a Track,” *Spotify for Developers*. [Online]. Available: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-analysis/. [Accessed: 14-Jun-2020].

[3] R. Ruizendaal, *Musical Data*. [Online]. Available: http://musicaldata.com/. [Accessed: 06-Jun-2020].

[4] R. Ruizendaal, *Spotify Musical Data*, 14-Jun-2020.

[5] “Spotify Web Player,” *Spotify*. [Online]. Available: https://open.spotify.com/. [Accessed: 05-Jun-2020].