Data Analytics

Historic and personal analysis of music based on Spotify.

Jaime Sastre Crespo

December, 2022

Table of content

1. Introduction: Business Use Case

How often do you listen to music? Are you a music lover and you need music in your daily life? Have you ever thought about analyzing the music you listen to, or how has the music evolved over the years? Or maybe how much time do you spend listening to music? Nowadays, it is so easy to give answers to these questions thanks to the data, but also thanks to apps like Spotify. I am assuming everyone knows Spotify but just in case: Spotify is a digital music streaming service that gives you access to millions of songs, podcasts and videos from artists all over the world, like Apple Music. It has a free version and also a premium with different plans.

Spotify is making some cool projects in order to personalize more and more and offering a whole experience, like the annual wrapper highlighting your 100 top songs and creating a playlist with that 100 songs, how much time did you spend listening to music, it is creating a video showing your top 5 artists, how is your taste in music with some cool definitions, etc. A really cool stuff. Nowadays, it is not enough having a great product, you have to create an experience to make the client feel special and Spotify is doing it great.

As you can imagine, I spend tons of time listening to music since I am addicted to music and also a guitar player. I think the first artist I listened to (and I remember) was Eminem when I was just 6 or 7 years old (eternally grateful to my sister). Furthermore, my whole family is music lovers and I had really good influences thanks to them. Maybe that's why I love almost all types of music.

That being said and now that you know a little about this part of me, it makes total sense that this project is about music, approaching different topics taking as base Spotify. Some of the goals of this project are: identify how the music evolved through history, how it has changed to have a better understanding about the music. Also, I would like to know about my taste in music, how many hours I spent in my day after day listening to music last year, I would like to create a playlist based on some recommendations.

To achieve the goals of the project, I am going to go through the different steps that I faced during my project such as finding the proper data and data sources , collecting the data, cleaning and exploratory data analysis (EDA in advance), having some insights, among others.

2. Data and data sources

First of all, it is necessary to mention that Spotify has an amazing API, friendly user -you can find the documentation here https://developer.spotify.com/documentation/web-api/ a and also a python library (spotipy). My first intention was to collect data as much as possible from the API but I was not able to scrape everything that I need for my analysis, so I mixed different data sources depending on the goal.

To study how music evolved during history, I needed at least 50 years of music with some features and I found a dataset in Kaggle that covers this. This dataset contains tracks in Spotify from 1920 to 2020 with several features and a bunch of songs, almost 600.000 songs. These features are going to give us a lot of information about how music changes and we will see clear differences between decades of music. Following you can find the features explanation :

- id: (Id of the song in Spotify)
- 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.
  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.
  duration_ms: Duration of the song in milliseconds
  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.
- valence: Measures the "musical positivity". Songs with high valence tend to be happier and more euphoric (0 to 1).
  popularity: Measures the popularity of the song (0 to 100).
  tempo: Measures the tempo in beats per minute (BPM).
  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 strong likelihood that the track is live.

- loudness: Measures the loudness of a song in decibels.

- speechiness: Detects the presence of spoken speech in the song (0 to 1).

- 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.

- explicit: Indicates if it contains aggressive language (bad words, drug talk, etc) (0 = Does not have, 1 = Has).

- 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.

- timesignature: Indicates the time signature of the song.

- artists: List of artists

- id_artists: Ids artists

- release_date: Release date of the song

- name: Name of the song

Once I have this 'objective' data, in the sense that I have no subjective songs with personal taste that can cause some bias in the features, I wanted to see how it is my taste and study how it is my behavior listening to music. After investigating some sources, I found that Spotify has the option to request your own historical data with a lot of information in it, such as every single song you played (date and time), for how long you play that song, name of the song and the artist, etc for the last year (not a natural year, i.e. from December 21 to December 22). Spotify can take up to 30 days to send you this information but usually it takes around 4 days. From this data I can make some cool visualizations to see how my music trend of this year was and more.

The columns for this data are very explainable by themselves:

- endTime: date and time of the song played

- artistName
- trackName
- msPlayed: milliseconds played

Last but not least, I used the Spotify API to scrape some playlist and my current loved songs. The playlists that I scraped are from 2018,2019,2020 and 2021. All of them are a playlist that Spotify creates for every year with your 100 top songs of that particular year. I thought this is useful in order to see some insights as to my favorite artist through the years, if I have some special song in all of those years, etc. Furthermore, I scraped my current favorite songs. Spotify allows you to do that for the periods defined as ‘short’, ‘mid’ and ‘long term’, taking 50 songs for each period mentioned. That way, I built a dataset with my top songs from the last 5 years. I used this last part of the data (the short-mid-long term songs) to build a playlist of recommendations, taking into account that we are going to have a playlist that matches perfectly with your current taste of music.

The columns that I have:

- Track_name
- Album
- Track_id: spotify gives a specific Id for every single song upload
- Artist
- Duration: duration of the song in milliseconds
- Popularity: explained above.
- Danceability: explained above.
- Energy: explained above.
- Key: explained above.
- Loudness: explained above.
- Mode: explained above.
- acousticness: explained above.
- Speechiness: explained above.
- Instrumentalness: explained above.
- Liveness: explained above.

- Valence: explained above.
- Tempo: explained above.
- Year_playlist: year of the playlist (2018 to 2022)
- Timeframe (short/mid/long term)

3. Data collection

As I mentioned before, my ‘historical’ data of tracks from 1920 to 2020 in Spotify is from Kaggle. So, I just downloaded the data and saved it on my computer. The second data, my ‘streaming’ data of one year from Spotify, I requested from Spotify to send it to me. Finally, I used the Spotify API to scrape some playlists and the current songs that I listen to.
By far, I needed to be familiar with the Spotify API documentation to be able to authenticate my account, creating a Dashboard in the web-developer of spotify section to be able to use the API… a lot of steps which I am going to try to summarize:

- Step 1: Log In to Spotify Developer
- Step 2: Create a Client ID
- Step 3: Retrieve Client ID and Client Secret
- Step 4:Explore: Spotify’s Web API Documentation
- Step 5: Install Spotipy (pip install)
- Step 6: Import and Set Up Spotipy(auth)
- Step 7: Retrieve Data from Spotify Web API (different ways of doing this)

Important: Do not share your Client ID and your Client Secret ID in GitHub. You can create a file .txt and import in your script with the following command:

```
with open(path / "client_s.txt") as f:
   content = f.readlines()
content = [x.strip() for x in content]
```

Where:

```
path = Path("/Users/jaimesastrecrespo/DAFT_1022/Final_project/spotify_App")
```

And you have your ‘dedicated’ data like this :

```
client_id = content[0]
client_secret = content[1]
```

```
redirect_uri = 'http://127.0.0.1:8080/yep'
username = content[3]
```

4. Data cleaning and Exploratory data analysis(EDA).

I worked on three different scripts .ipynb for cleaning and EDA of my three different data sets.

- For my historical data of Kaggle:

First of all, I imported the libraries that I usually use to clean my data: Pandas, Numpy, Datetime… and also some libraries for my visualizations: Matplotlib, Seaborn, Altair…

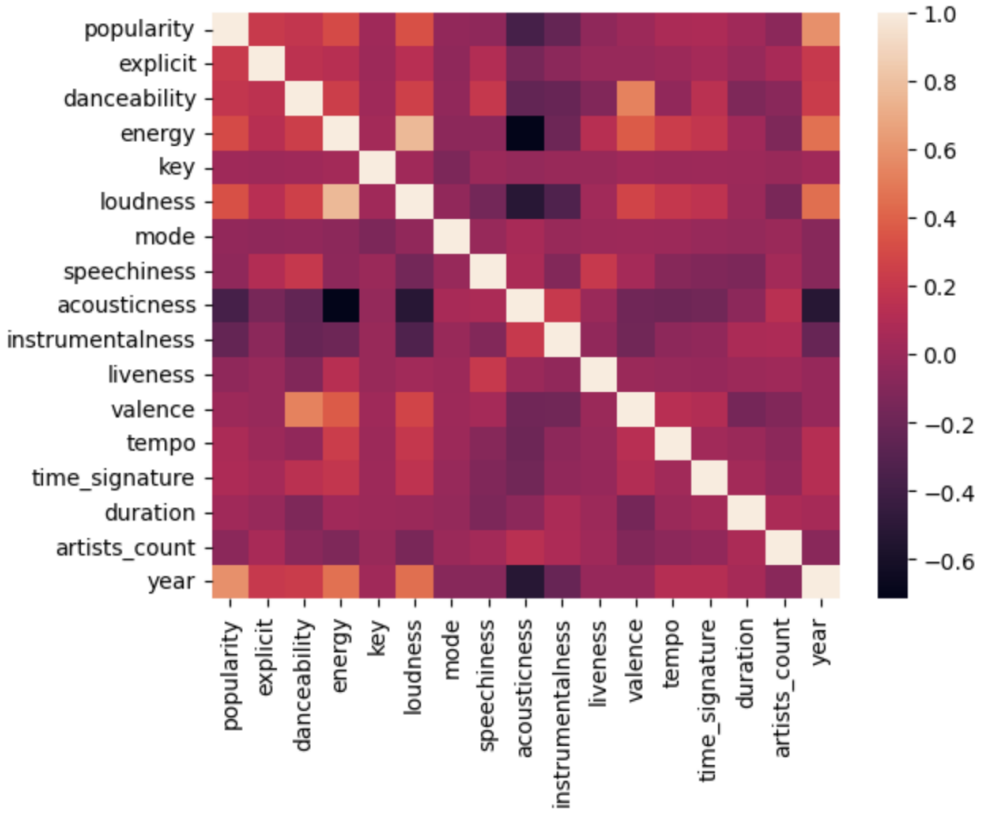
Secondly, I imported my data file (in .csv) and I started to look at some easy commands but necessary ones: .head() , .describe, isna() , etc. Looking for missing values, duplicated and dealing with them. I only had 71 missing values, so I decided to drop them because I had almost 600.000 rows so it was nothing. At the very beginning,

looking for duplicates I had as a result 0 duplicates, but I thought that was not possible. Why? Because I know Spotify (it is mandatory to know the project's domain and in this case, I proved it). Spotify usually has more than one version of songs. I am not speaking about different songs (i.e. a rock song recorded in acoustic format), I am speaking about different names of songs but the audio is exactly the same or close. Some people uploaded the same song, basically. So, what I did is to go further in the duplicates searching and looking for the same value in the columns name (of song) and the artists. If that matches, drop those rows keeping the first one. I found around 60.000 duplicates so I just dropped them.

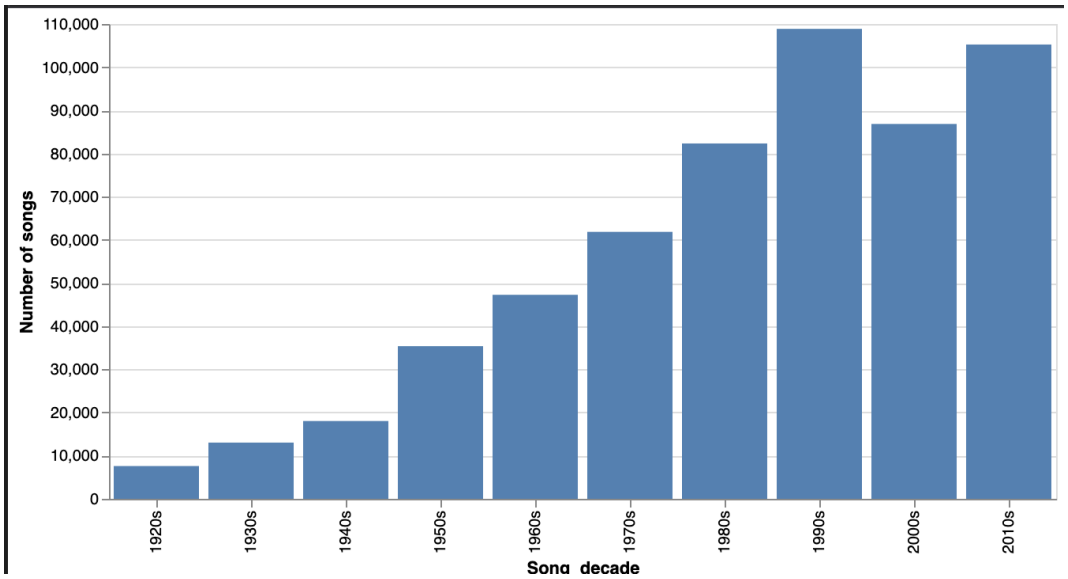
Once I have my clean data, I plotted some graphs:

- some correlations heatmaps to see if I have high correlation between my variables,
- A mark line chart where we can observe the trend of four main features ("acousticness","danceability","energy","valence"),
- A bar chart to see how many songs they were published per decade,
- Among others vizualisations.

- Correlation heat map



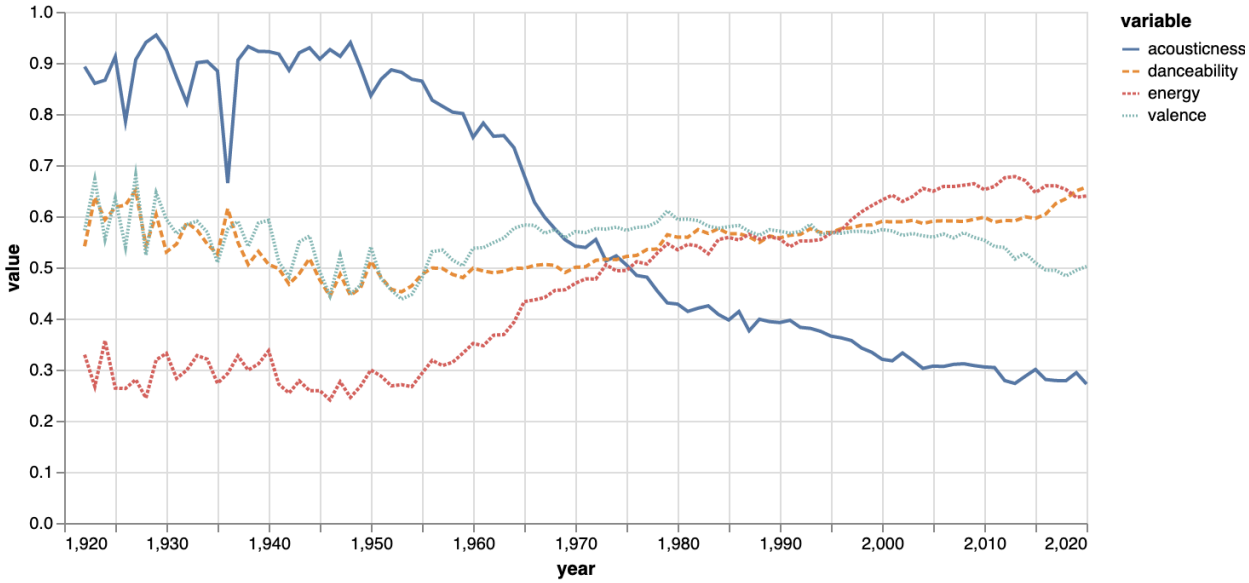
As the graph is showing, we don't have highly correlated variables. So we are good in this aspect, because a highly correlated variable could bring some noise to our analysis.



We see a gradual progression in the number of songs from 1920 to 1980. This is easily explained, because as time goes on, technology makes it more accessible to create and listen to music, so it seems normal this gradual progression. In the 90's there was a growth out of the ordinary, in fact, to this day is the decade in which more music has been produced. But why in the first decade of the 90's? The answer is simple, variety. The 90's is considered the golden decade of music due to the wide range of musical styles that emerged and had their place in society. Grunge, captured by Kurt Cobain with Nirvana, Oasis, Blink-182 or Green Day, rap with Eminem, Madonna and the great Michael Jackson, The Backstreet Boys, Extremoduro… are just some of the examples that show the great variety of musical genres that were represented by artists of an exceptional level, which even today, are still considered superstars.

In addition, in the 90's a fundamental element was added to the music, the video. Today we no longer conceive music only for the sound, but it is much more and video clips have become an indispensable part of the sector. For all these reasons, the explosion from the 80's to the 90's took place and will continue with the gradual growth previously observed.

After, We came back to similar levels as in the 80s but after, in 2010, again a huge peak. In my opinion, we are again in a great moment for music, with a lot of variety and many artists showing their potential.

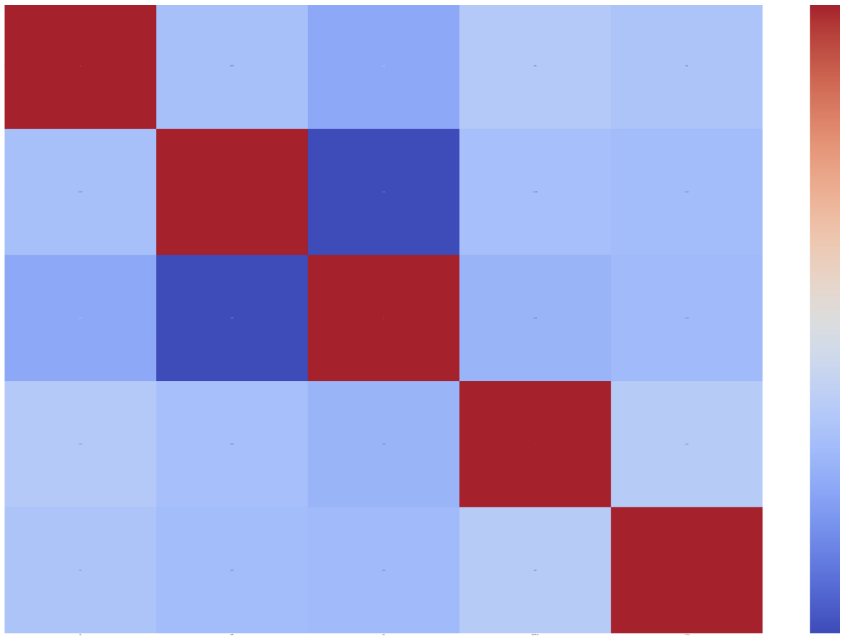


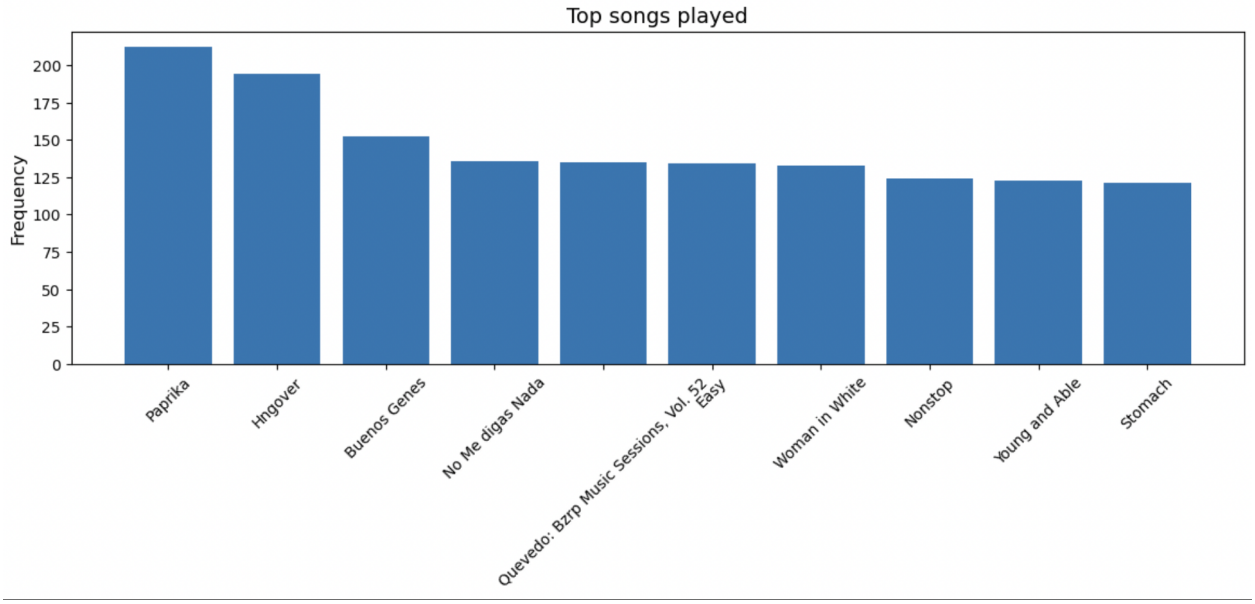
Here we can see how the accousticness of the music went down and dropped around 60%. This makes total sense because we are producing electronic music as we want energetic music, with loud sounds, mostly for parties. Actually, techno music has been in a high position the last few years. That means, music with high accusticness is being affected but still being important in the scene.

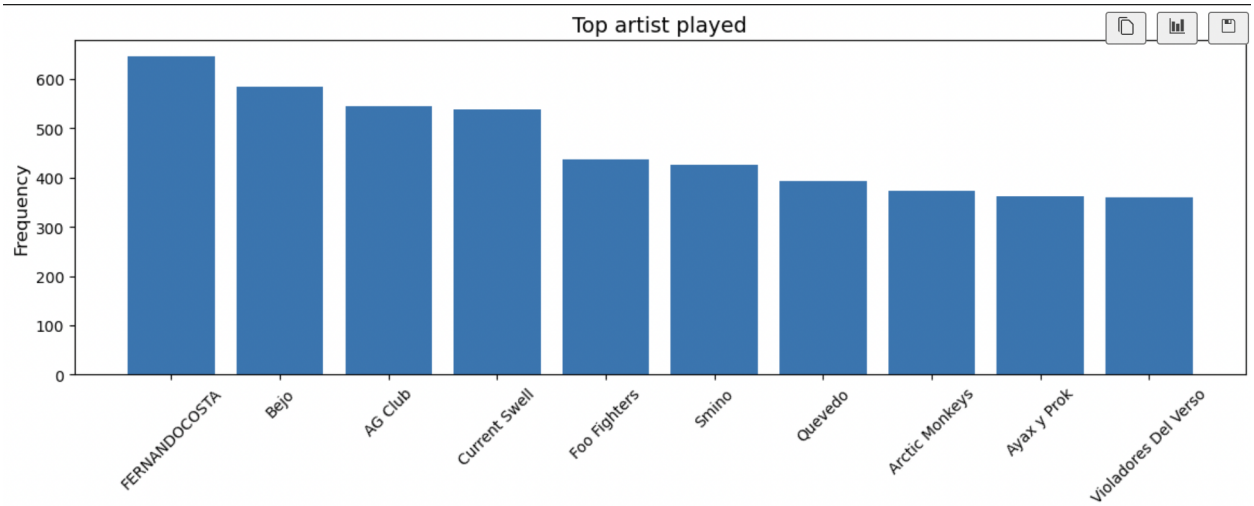
- My streaming data from Spotify:

My streaming historic data from the last year of Spotify was provided in three different CSV files as I spend a lot of time listening to music (48.000 minutes in total this 2022). I concatenate them and I transformed my column ‘Endtime’ into datetime to be able to extract valious information such as the time, the day, weekday, etc… in different columns, that I used after in some visualisations. It is okay to have duplicate songs in this kind of file as it is a record of every single song that I played, even if it was played only for 2 seconds it will appear. I also created a column as UniqueID which contains the name of the artist and the name of the track.
Once I had my date clean, I could plot some insightful results:

- Correlation heat map (no high corr columns);
- Top ten songs (in terms of numbers of times played),
- Top ten artists (in terms of numbers of times I played that artist).







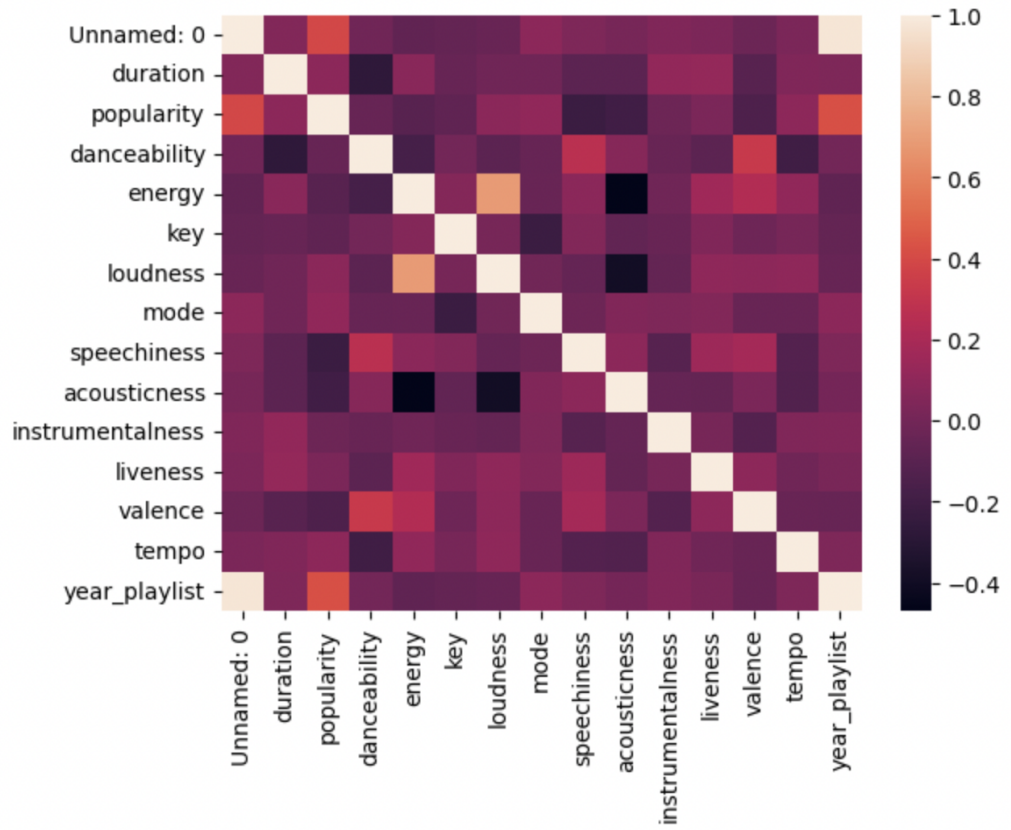
- <u>For my last dataset, my historical top songs data (2018-2022):</u>

As normally, I imported the libraries that I needed to clean and the data. As I had 5 different datasets, I added a column to indicate the year of each one and concatenate all of them. Afterwards, I searched for duplicates and I found some. I decided to drop them because they were always from the last part of my data. This means that in my collection of data with the API for the year 2022, taking into account that I was having my 50 top songs most played for short term, mid and long term, it makes sense that I have some of them duplicated. That's why I decided to drop those rows.

Also, I decided to drop the column ‘timeframe’ because the main point here was to have the year. It was not important to have that information and generate missing values in the rest of the data (for years 2018,2019,2020,2021). I also converted the column of milliseconds to seconds, because it is more understandable.

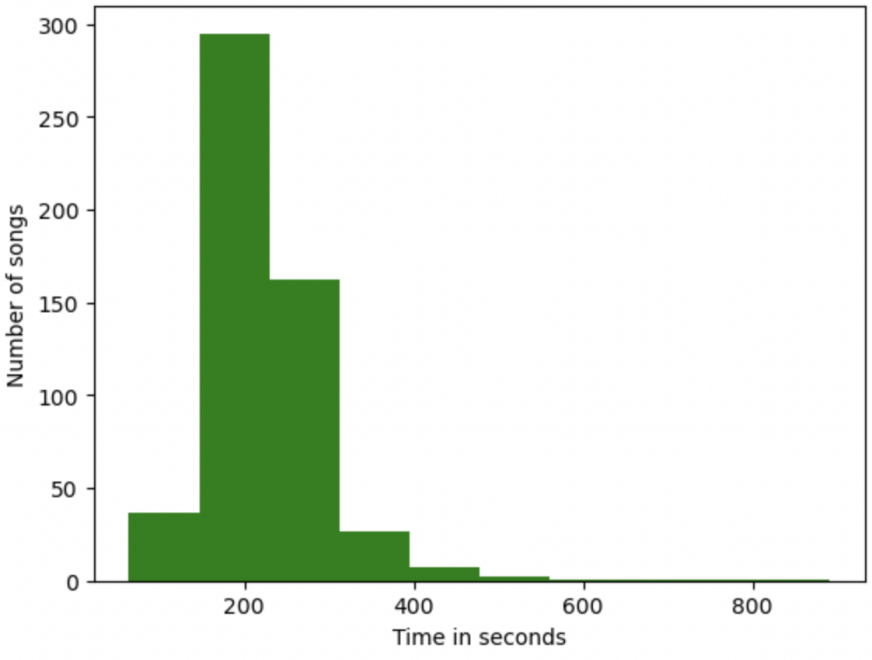
At that point, I was able to plot some charts:

- Correlation heat map;

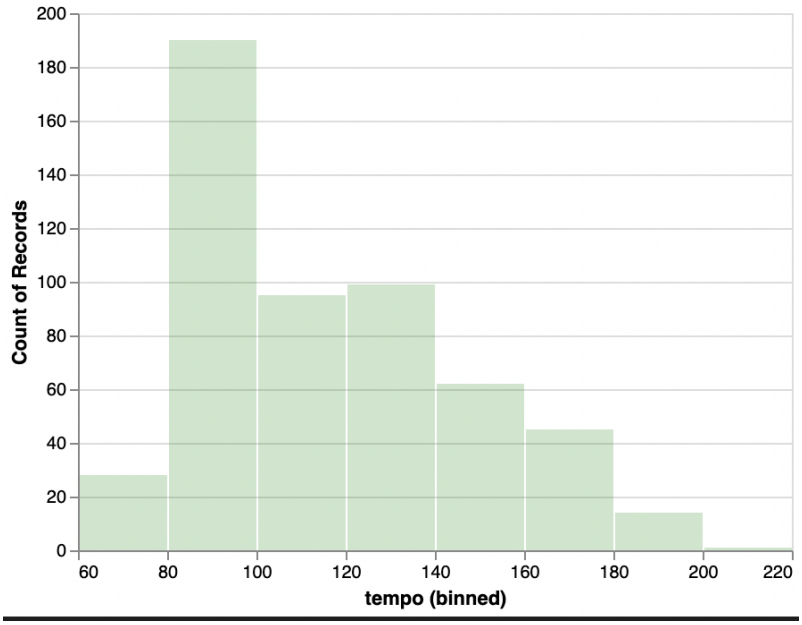


In this case, we have the column year_playlist highly correlated with Unnamed:0 (the index) and it is completely fine as the year was introduced for each data set. Therefore, our first 100 rows are for the year 2018, the next 100 for 2019 and so on.

- Distribution of length per track: we can see how it is left skewed. Make sense because most of the songs are between 2 mins and 3 mins and 20 seconds.



- In this bar plot, we can see how my library from the last 5 years (top songs) are right skewed in terms of tempo. This means that I have a quiet taste of music, not with high levels of tempo. Makes sense, because I like calm music more than powerful music.



5. Building a recommendation playlist.

To build a recommendation playlist in your Spotify account it is needed to base your recommendation in some seeds. In my case, as I scrapped the songs that I am currently listening to more, I wanted more songs with that bunch of features or similar to discover new songs and new artists like those that I am currently listening to.

To be able to do this, you can use the API of Spotify and the Spotipy (python library of Spotify). I made a python file (.py) where I defined custom functions (in case I want to implement this for public in the future and not only for my account) :

- Authenticate : Authenticates a user for a given spotify app.

- Create_df_top_songs: Reads in the spotipy query results for user top songs and returns a DataFrame with track_name, track_id and more columns

- top_artists_from_API: Reads in the spotipy query results for user top artists and returns a DataFrame with name, id, genres, popularity and uri.

- Create_df_recommendations : Reads in the spotipy query results for spotify recommended songs and returns a DataFrame with track_name,track_id,artist,album,duration,popularity

- Create_df_saved_songs: Reads in the spotipy query results for user saved songs and returns a DataFrame with track_name,track_id, artist,album,duration,popularity

- Create_df_playlist : Reads in the spotipy query results for a playlist and returns a DataFrame with track_name,track_id,artist,album,duration,popularity and audio_features unless specified otherwise.

- Append_audio_features: Fetches the audio features for all songs in a DataFrame and appends these as rows to the DataFrame. Requires spotipy to be set up with an auth token.

- Get_recommendations: Gives top num_recommends recommendations for a song based on a similarity_score or matrix

Once you have your seeds for your recommendation, you have to authenticate your account and use the recommendations method to get one to five songs per seed. The Spotify API only accepts a small list of seed tracks for each query, and the problem was

that what I wanted to use was the whole playlist. So I made packages of 5 seed tracks to retrieve 25 recommendations (5 per song). Here is the code:

```
recomm_dfs = []
for i in range(5,len(seed_tracks)+1,5):
   recomms = sp_m.recommendations(seed_tracks = seed_tracks[i-5:i],limit = 10)
   recomms_df = append_audio_features(create_df_recommendations(recomms),sp_m)
   recomm_dfs.append(recomms_df)
recomms_df = pd.concat(recomm_dfs)
recomms_df.reset_index(drop = True, inplace = True)
```

So, I got a playlist of 300 songs, but you only can add 100 songs to your spotify because that is the limit to add to one playlist. So I decided to add the first 100:

```
sp_m.user_playlist_add_tracks(username,
                              playlist_id="spotify:playlist:3XWyiIO6rtmYiJuW3Z91gy",
                              tracks = recomms_df["track_id"].tolist()[:100])
```

I could have solved this by creating a playlist more and adding the 100 songs in the middle and the last 100 but I think it is not necessary. For sure, the method can be improved and I will try to improve it in the next steps of my project (in the future).

I have to say that, in the first playlist of 100 songs that I created, I only had one song that I had in my loved song playlists (saved one) and in the next tries between 4 and 7. So, I can definitely say that it is a good algorithm of recommendation and a good approach to implement when you want to discover more music.

6. Databases type comparison

The next step was creating my database. I decided to use SQL because this type of database allows me to use my cleaned and structured data. Furthermore, I can perform queries on a small level and link the different tables (for example my 2018,2019… tables of my top100 songs) with each other using primary and foreign keys.

Why SQL and no NoSQL? 5 main reasons:

- SQL databases are relational, NoSQL databases are non-relational.
- SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.

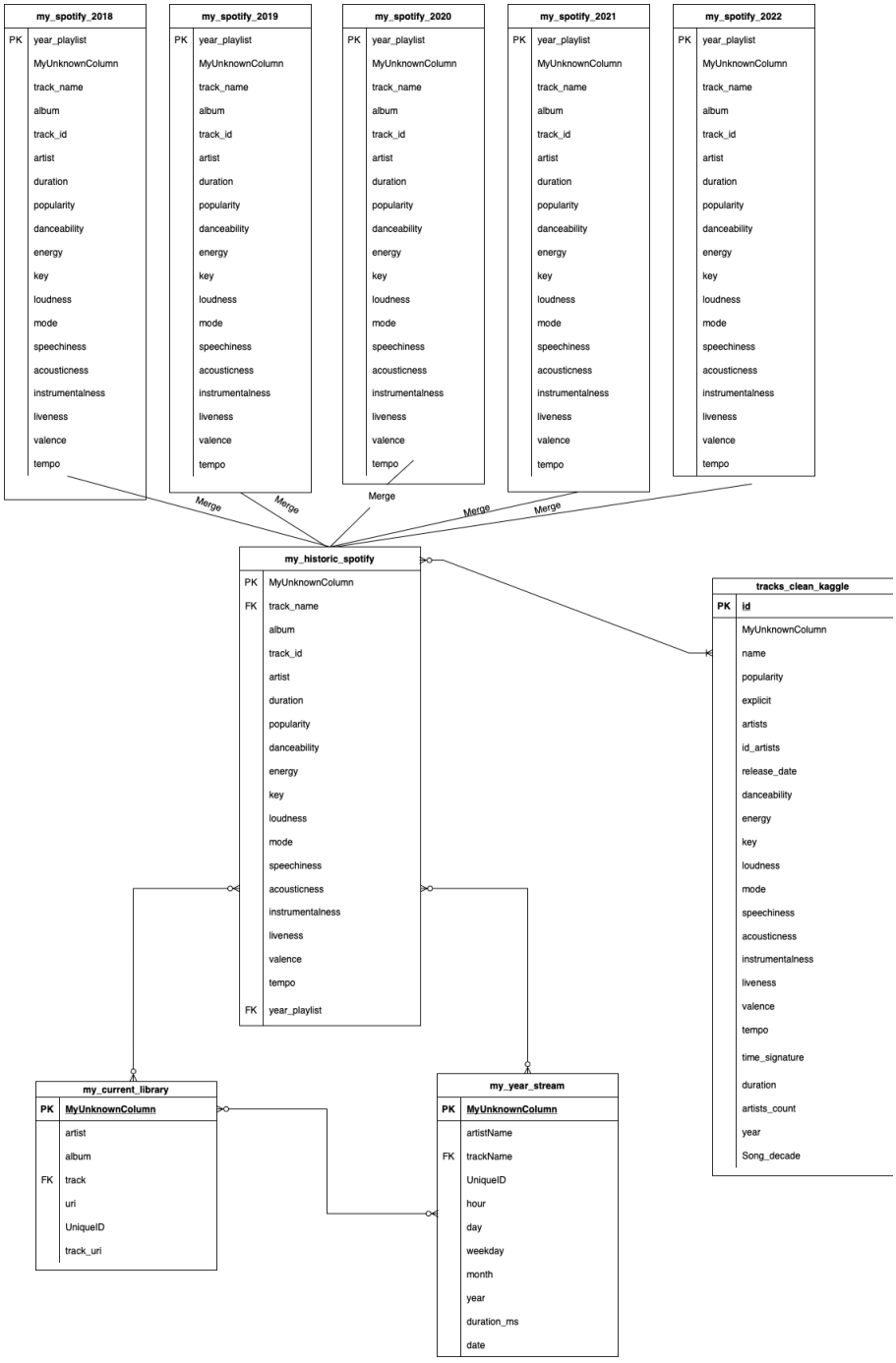
- SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
- SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
- SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documentsor JSON.

This allows me to quickly make queries and join different information from various tables. In my case, I would have liked to have a table with the genres per artist and make some join, but I did not have it. So, what I did is to join my top 100 songs dataframes into one and some insightful queries as I will show in advance.

The relational database I used is MySQL Workbench. In order to prepare the foundation for the creation of my database I created the following entity relationship model to clarify the different entity tables I will use and specify their relationships.

7. Entities. ERD

The entities of the diagram represent my datasets after the cleaning process. First of all, I had my five tables with the 100 top songs for each year (2018 to 2022). The primary keys are the year of the playlist. I concatenated all of them in 'my_historic_spotify' as I explained above. This one is related to my_current_library as I have songs in both ''playlists'' and also with my_year_stream as I listened to songs from those two playlists. In this case, I decided to put as primary key MyUnknownColumn which is basically the index, because the rest of the columns are not unique values, they can be repeated. That being said, the table which is bringing some neutral information is the one I took from Kaggle, which is related with my_historic_spotify as both of them are representing history, in some way. For this table, I could define id as the primary key as it is completely unique. I defined as a FK the name of the tracks.



8. Creation of the database and data importation

As I mentioned before, I used MySQL as a warehouse and as a tool to give some interesting insights. The import of the data was performed through the wizard process to

make it faster and more efficient. You can see the main queries and the main commands to create a database in MySQL in the following sentences:

- To create the database in MySQL, to use the database, to make a little visualization of my table 'my_year_stream' and finally indicating which is the primary key, once the table is already created (imported in this case).

```
1  CREATE DATABASE IF NOT EXISTS final_project;
2
3  USE final_project;
4
5  SELECT *
6  FROM my_year_stream;
7
8
9  ALTER TABLE my_current_library
10 ADD PRIMARY KEY (MyUnknownColumn);
11
```

- If I would need to create the tables I would do something like this:
    - CREATE TABLE IF NOT EXISTS my_current_library
    - (
    - MyUnknownColumn INT,
    - artist VARCHAR(35),
    - ……
    - PRIMARY KEY (MyUnknownColumn)
    -);

- You can see my 5 main queries below:

```sql
-- 1
-- To show the song and artist I played most during the year, how many times and how much time

SELECT count(artistName) as Max_artist, UniqueID, count(day), sum(duration_ms)
FROM my_year_stream
GROUP BY UniqueID
ORDER BY count(artistName) DESC ;
```

00% 13:20

Result Grid Filter Rows: Search Export: Fetch rows:

Max_artist	UniqueID	count(da...	sum(duration_ms)
136	Quevedo:No Me digas Nada	136	18944
135	Bizarrap:Quevedo: Bzrp Music Sessions, Vol. 52	135	26056
133	Current Swell:Woman in White	133	21088
123	Quevedo:Nonstop	123	24428
123	Current Swell:Young and Able	123	26201
121	Current Swell:Stomach	121	22419
116	Eminem:Lose Yourself - From "8 Mile"" Soundtr…	116	28672
113	Young Thug:Livin It Up (with Post Malone & A$A…	113	21596
109	Ayax y Prok:Guajiro	109	14554
106	AG Club:Paprika	106	16455

```sql
-- 2
-- THE SONGS I PLAYED LESS THAN 2 SECS AND THEY ARE NOT IN MY CURRENT LIBRARY, MAYBE WE SHOULD REVIEW IF WE WANT TO KEEP THEM
-- IN OUR PLAYLIST (IF THEY ARE) OR IMPROVE THE MACHINE LEARNING RECOMMENDATION OF SPOTIFY TO AVOID THIS KIND OF MUSIC


SELECT count(artistName) as Counter,  mys.UniqueID
FROM my_year_stream mys
LEFT JOIN my_current_library mcl
ON mys.artistName = mcl.artist
WHERE duration_ms < 2 and mys.artistName NOT IN (SELECT artist FROM my_current_library)
GROUP BY mys.UniqueID
ORDER BY count(artistName) ASC
;
```

)0% 19:30

Result Grid Filter Rows: Search Export:

Counter	UniqueID
1	Oques Grasses:Inevitable
1	Els Pets:No vull que t'agradi aquesta canco
1	Manel:Benvolgut
1	Bongo Botrako:Revoltosa
1	Oques Grasses:Canco de l'aire
1	Els Pets:Blue tack
1	Manel:Sabotatge
1	Manel:Ai, Dolors
1	Cesk Freixas:La Petita Rambla del Poble Sec - XL
1	Oques Grasses:Serem ocells

```sql
-- 3
-- When did I listen more songs during the week last year? Display the day and the artist.

SELECT count(artistName), weekday
FROM  my_year_stream
GROUP BY weekday
ORDER BY count(artistName) DESC
LIMIT 3;
```

00% 11:42

Result Grid | Filter Rows: Search | Export: | Fetch rows:

count(artistNa...	weekday
3669	Saturday
3653	Friday
3393	Tuesday

```sql
-- 4
-- Show if there's any song (and with some level of popularity) that I did not listen in the last year but
-- it is in my historic spotify that records my top 100 from each of the following years: 2018, 2019, 2020,
-- 2021 and 2022 (150 songs of this last year)


SELECT DISTINCT mhs.artist,  mhs.track_name, popularity
FROM my_historic_spotify mhs
LEFT JOIN my_year_stream mys
on  mhs.track_name = mys.trackName
WHERE mhs.track_name NOT IN (SELECT trackName FROM my_year_stream)
AND popularity > 40;
```

100% 17:57

Result Grid | Filter Rows: Search | Export:

artist	track_name	popularity
Funambulista	Ya Veras (with Andres Suarez)	45
DELLAFUENTE	Guerrera	61
XXXTENTACION	SAD!	83
XXXTENTACION	the remedy for a broken heart (why am I so in lo...	79
Funambulista	Me Inventare (with Dani Martin)	43
La Gossa Sorda	Esbarzers	44
XXXTENTACION	Moonlight	82
El Canto Del Loco	Zapatillas	63
Wiz Khalifa	Gin and Drugs (feat. Problem)	45
Gritando en Silencio	A la luz de una sonrisa	42

```
62    -- 5
63    -- How Was my music in terms of matching popularity in the last 5 years?
64    SELECT track_name, artist , tempo , popularity, year_playlist
65    FROM my_historic_spotify
66    WHERE popularity > 75
67    ;
```

100% 15:65

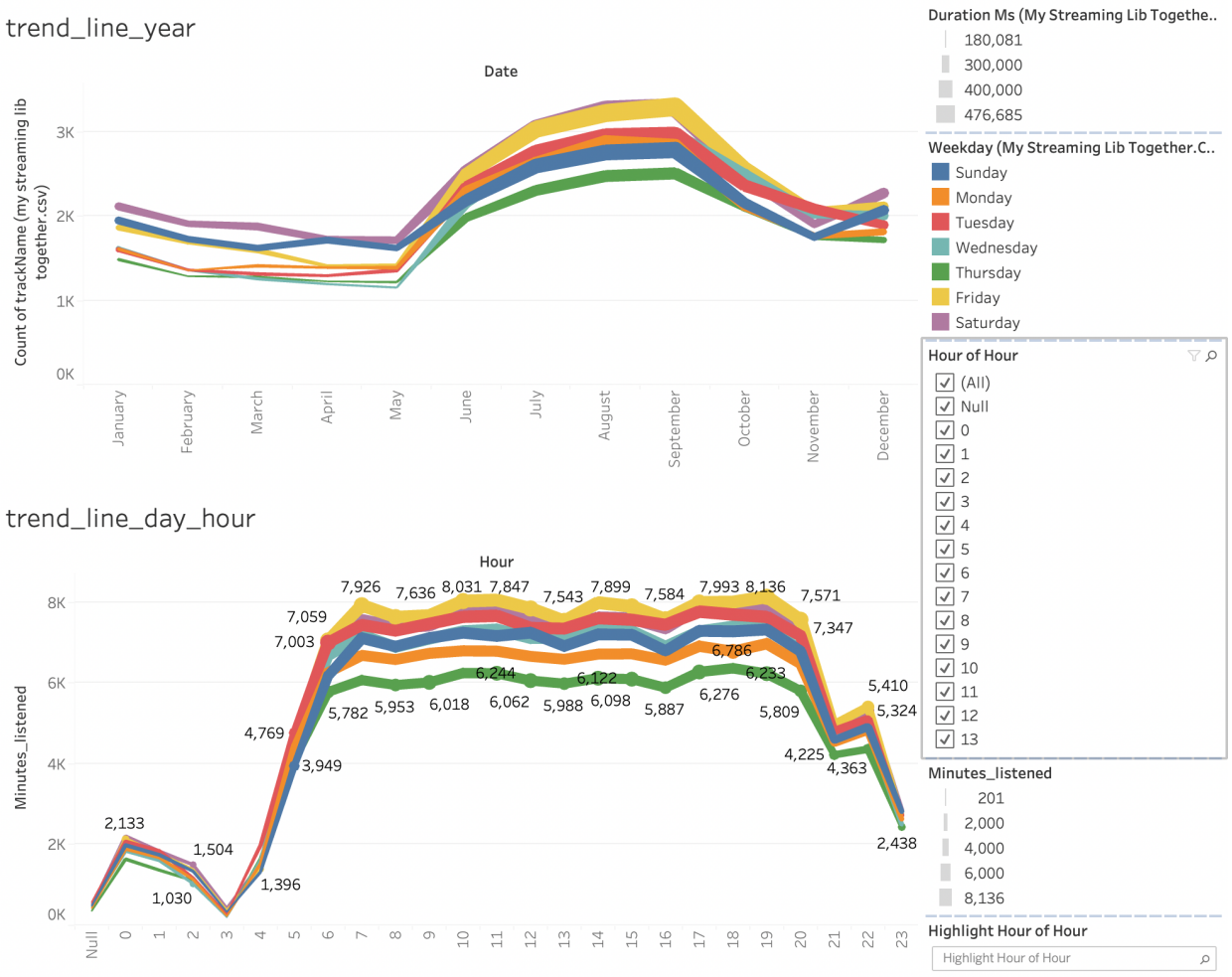
Result Grid Filter Rows: Search Export:

track_name	artist	tempo	popularity	year_playlist
SAD!	XXXTENTACION	75.023	83	2018
the remedy for a broken heart (why am I so in lo…	XXXTENTACION	119.705	79	2018
R U Mine?	Arctic Monkeys	97.094	82	2018
Moonlight	XXXTENTACION	128.009	82	2018
Do I Wanna Know?	Arctic Monkeys	85.03	87	2018
God's Plan	Drake	77.169	84	2019
Last Resort	Papa Roach	90.598	80	2019
Demons	Imagine Dragons	89.938	84	2020
Natural	Imagine Dragons	100	81	2020
Thunder	Imagine Dragons	167.997	84	2020
Take Me To Church	Hozier	128.945	83	2020
Believer	Imagine Dragons	124.949	88	2020
Self Care	Mac Miller	141.894	79	2020
Weekend (feat. Miguel)	Mac Miller	120.058	77	2020
Little Talks	Of Monsters an…	102.961	79	2020
Demasiadas Mujeres	C. Tangana	126.043	79	2021
Mon Amour - Remix	zzoilo	116.041	83	2021
WITHOUT YOU	The Kid LAROI	93.005	79	2021
Small Worlds	Mac Miller	78.267	77	2021
The Scientist	Coldplay	146.277	85	2021
Everlong	Foo Fighters	158.066	83	2022
No Me digas Nada	Quevedo	102.16	78	2022
Ahora y Siempre	Quevedo	118.964	76	2022
Learn to Fly	Foo Fighters	135.997	76	2022
Smells Like Teen Spirit	Nirvana	116.761	82	2022
The Pretender	Foo Fighters	172.984	78	2022
I Like You (A Happier Song) (with Doja Cat)	Post Malone	100.964	89	2022
Quevedo: Bzrp Music Sessions, Vol. 52	Bizarrap	128.033	97	2022

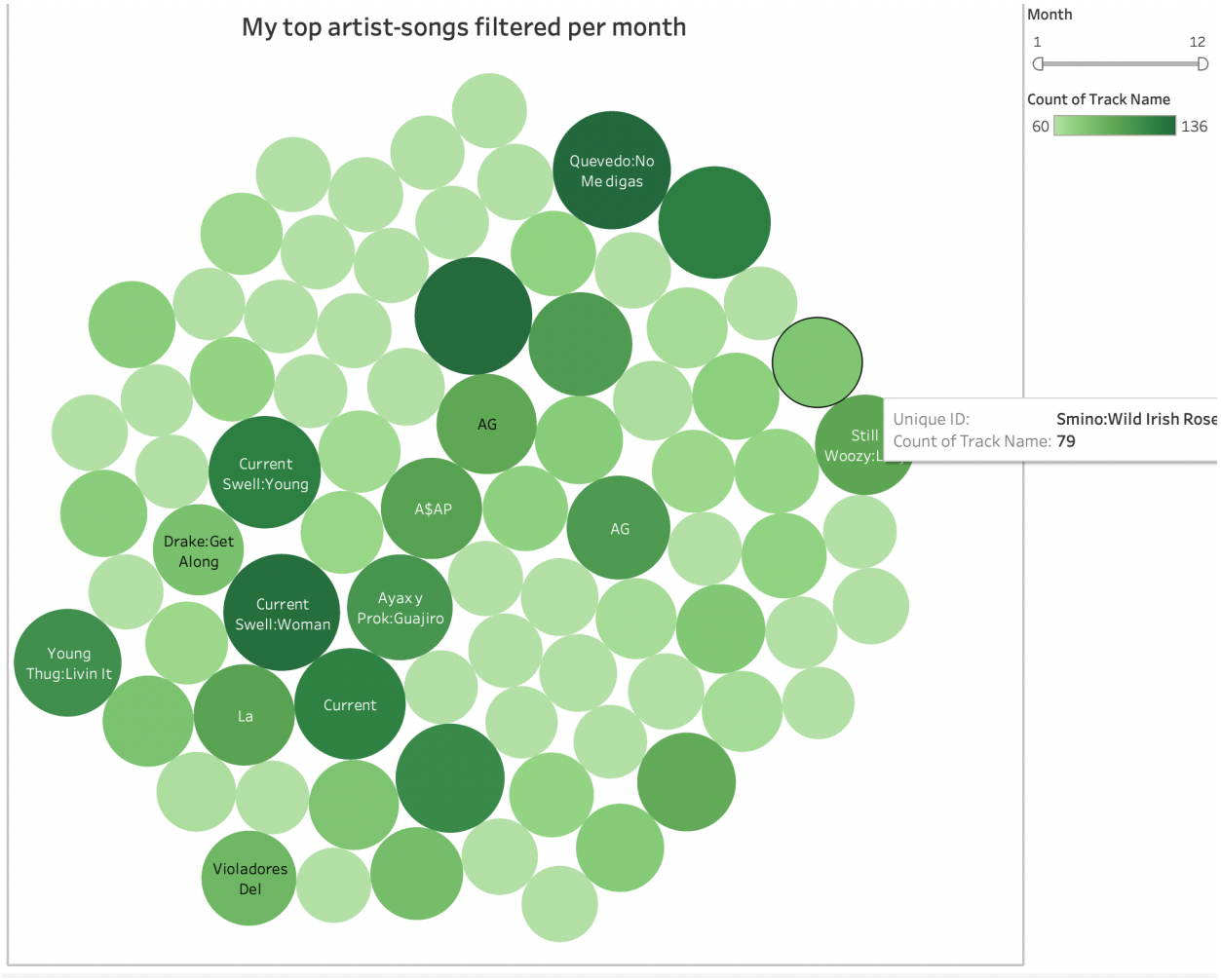
9. Some additional insights

Apart from some visualisations in the Python script, I also made some dashboards in Tableau as I consider Tableau is more interactive and more interesting to show some trends for my streaming history of Spotify. You can see below some of them:

- Trend line in terms of sum of seconds during the year. It is clear how Summer is making a difference and I have higher levels of music listened to, influenced because of my holidays and also because the days are longer in terms of sunlight and I spent more time awake or doing active stuff.
- Obviously, The lowest rate of seconds listened is during the night but it is funny how this is showing that I usually start my day very early, around 5-6-7 am.

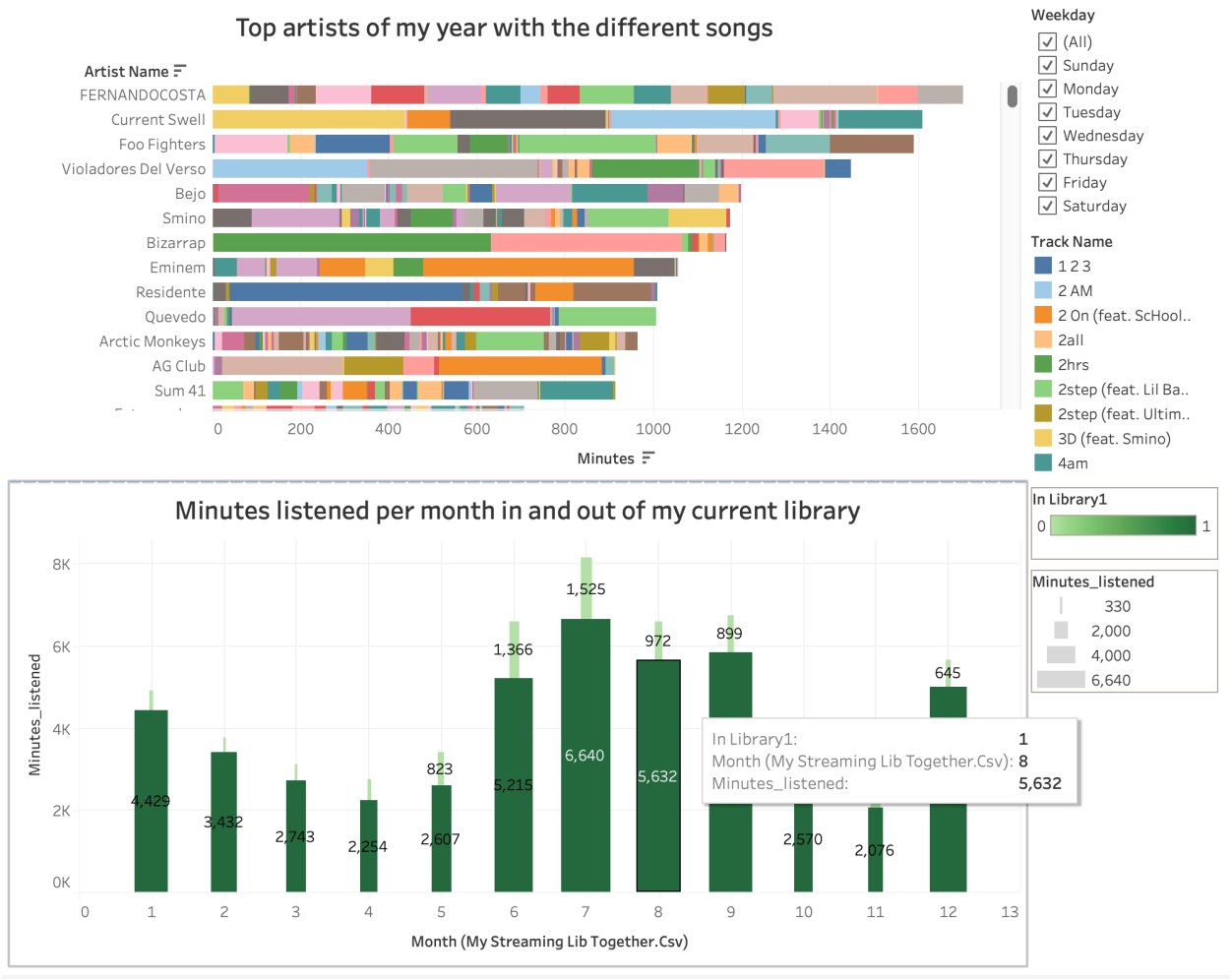


- Here we can see my top artists-songs during this year, filtered by month. (in this case I'm showing the whole year)



- In the next chart, I'm showing my top artists, differentiating the songs that I played from that specific artist. Also, I filtered by the weekday.

- Last but not least, I made a bar chart showing how many minutes (with a calculated field where I converted the seconds in minutes) I listened per month and of that total, splitting which songs are in my spotify library in which ones are not.



10. Conclusion

After this music analysis, taking into account the history of music (1920 to 2020), my last five years of top songs and my last year of streaming, all of this data based on Spotify, I can make some conclusions. First of all, the music has evolved during the years losing accoustincness and gaining energy and power. I would say that this has two main reasons: on the one hand, the means to produce music evolved to an electronic environment. On the other hand, we have more electronic artists than 'accoustics' artists, if we compare with decades ago and it is related with the prior reason.

Secondly, we had a gold decade during the 80s-90s with maybe the greatest artists ever. The decade of 2000 was less productive but it seems like during 2010 we came back to greater levels and I would say that the decade of 2020 is going to have

the same trend, in terms of number of songs produced, but I am not that sure in terms of quality.

In third place, thanks to the data that Spotify records and the API, you can have powerful insights like your top artist, top songs, how is your behavior listening to music in terms of timing, etc.

This project could be improved and make a further analysis about features, or make an app to produce playlists based on your current taste of music for every user that has a Spotify account.

11. Links

Github repository:

https://github.com/JaimeSastreCrespo/Final_project

Jira project planning:

https://daftparis.atlassian.net/jira/software/projects/SP/boards/6