Spotify Track Popularity

September 6, 2024

1 Project Overview

The main question we are trying to answer is: **What factors influence the popularity of a song on Spotify?** For this, we will use Spotify's API, which contains a popularity index for each song, as well as attributes such as like valence, loudness, energy, lyrics etc. We will also have access to information about artists and music charts to explore our question.

2 Analysis of Dataset

To answer our central questions about Spotify popularity and its driving factors, we have constituted our own database on BigQuery using Kaggle Spotify datasets 'Tracks", "Artists", "Charts", and "Lyrics" each containing information of a group of songs gleaned from the Spotify API.

Tables

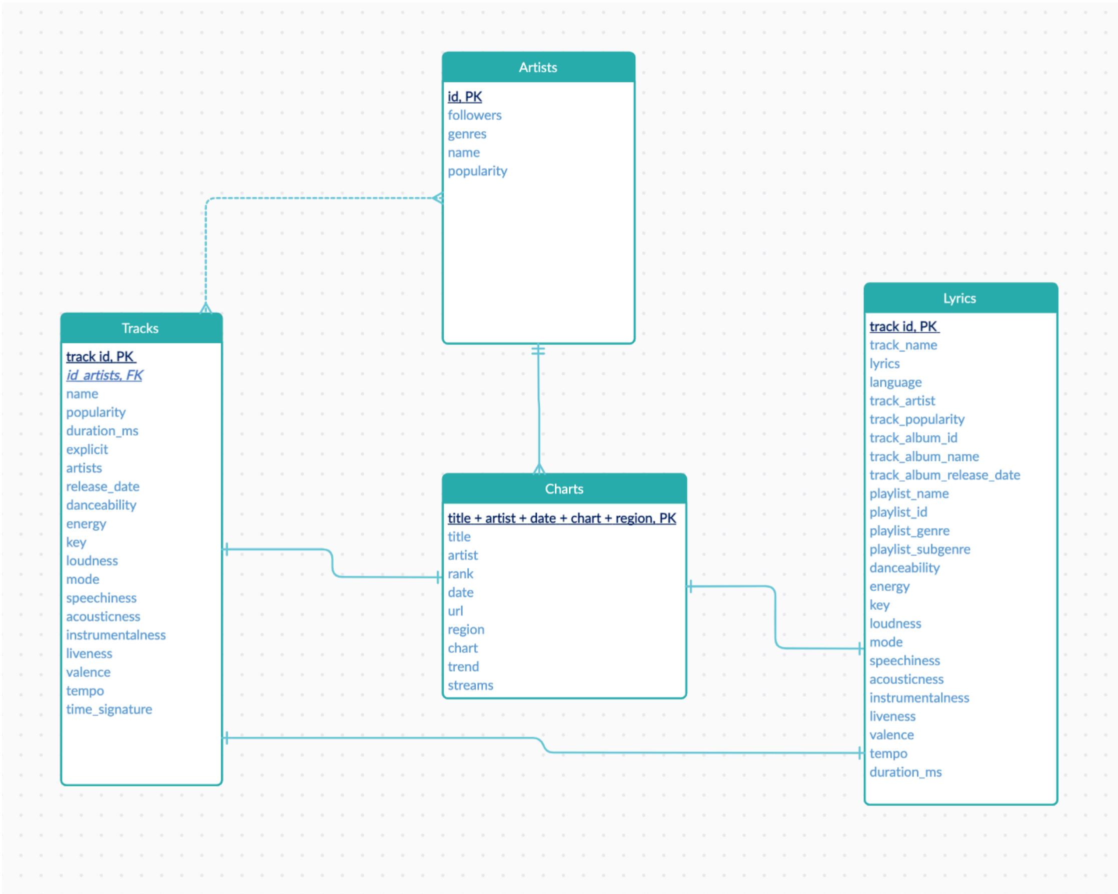
- **Tracks** : This dataset contains information over more than 500,000 Spotify tracks, including, artist, album, audio features (e.g. loudness), and popularity.
- **Artists** : This dataset describes contains the list of artists along with their popularity, genres, and number of followers in 2020.
- **Charts** : This is a complete dataset of all the "Top 200" and "Viral 50" charts published globally by Spotify. Spotify publishes a new chart every 2-3 days. This is its entire collection since January 1, 2017
- **Lyrics** : This dataset contains various types of information over more than 18,000 Spotify songs. The Lyrics dataset complements the Tracks dataset, by adding new information that will be used in our exploration and ML prediction, such as the lyrics, the genre/subgenre, and information about playlists.

2.0.1 Relational schema

We drew a relational schema using the IDEF1X data modeling language to clearly represent our database and the relationship among the tables. The relational schema summarizes the Primary Keys, the Foreign Keys, and consequently the possible JOINS among tables.

[167]:
```
from IPython.display import Image
Image('/content/drive/MyDrive/CS145/relational_schema.png',width=1300,␣
↪height=980)
```

[167]:



- **Tracks**

The Primary Key is track_id, and the Foreign Key is id_artists, which can be found in the Artists dataset.

1. *One-to-one relationship with Charts* : Logically, a certain chart can contain the same track multiple times (on different dates), and a track can be contained in multiple charts. However, the PK of charts is an aggregate key composed of title + artist + date + chart + region. There is no separate table only for charts. Thus, the relationship between Tracks and Charts is one-to-one rather than many-to-many.

2. *Many-to-Many relationship with Artists* : A track can be composed by many artists, and an artist can compose many tracks.

3. *One-to-One relationship with Lyrics* : Both tables have the same primary key, although they contain different information.

- **Artists**

The Primary Key is id, which is the Spotify ID of the artist.

1. *One-to-Many relationship with Charts* : A chart for a song on a certain day can contain multiple artists, buy an artist can't figure multiple times in a chart for a certain song on a

certain day. Again, this is because the PK of charts is an aggregate key composed of title + artist + date + chart.

2. *Many-to-Many relationship with Tracks* : A track can be composed by many artists, and an artist can compose many tracks.

3. Note that there is relationship with the Lyrics dataset, since the Lyrics dataset does not contain the artist_id, but only the artist name, which might not be unique.

- **Charts**

The Primary Key is an aggregation of track_id + artist_id + date + chart + region. This represents the fact that a song written by 1 or more artists appeared in a chart on a specific date in a specific region. Note that there is no separate table for charts.

1. *One-to-one relationship with Tracks* : Logically, a certain chart can contain the same track multiple times (on different dates), and a track can be contained in multiple charts. However, the PK of charts is an aggregate key composed of title + artist + date + chart + region. There is no separate table only for charts. Thus, the relationship between Tracks and Charts is one-to-one rather than many-to-many.

2. Note that there is relationship with the Artists dataset, since the Lyrics dataset does not contain the artist_id, but only the artist name, which might not be unique.

3. *One-to-One relationship with Lyrics* : Logically, a certain chart can contain the same song multiple times (on different dates), and a song can be contained in multiple charts. However, the PK of charts is an aggregate key composed of title + artist + date + charts. There is no separate table only for charts. Thus, the relationship between Lyrics (which are essentially songs) and Charts is one-to-one rather than many-to-many.

- **Lyrics**

The Primary Key is track_id. This table is not directly linked to the Artists table, because it does not contain an artist_id variable. Rather, it contains the artist name, which is not unique (and thus, is not a foreign key).

1. *One-to-One relationship with Charts* : Logically, a certain chart can contain the same song multiple times (on different dates), and a song can be contained in multiple charts. However, the PK of charts is an aggregate key composed of title + artist + date + chart + region. There is no separate table only for charts. Thus, the relationship between Lyrics (which are essentially songs) and Charts is one-to-one rather than many-to-many.

2. *One-to-One relationship with Tracks* : Both tables have the same primary key, although they contain different information.

2.0.2 Tables

We will now describe the tables in detail. Following is a data dictionary with the description of each variable, along with their type.

2.0.3 Tracks table (133.05 MB)

The Tracks table contains information over more than 500,000 Spotify tracks, including, artist, album, audio features (e.g. loudness), and popularity.

It has 586,672 rows and 20 columns :

- **acousticness** *(float)* : A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **analysis_url** *(string)* : A URL to access the full audio analysis of this track. An access token is required to access this data.
- **danceability** *(float)* : 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** *(int)* : The duration of the track in milliseconds.
- **energy** *(float)* : 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.
- **id** *(string)* : The Spotify ID for the track.
- **instrumentalness** *(float)* : 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** *(int)* : 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.
- **liveness** *(float)* : 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** *(float)* : 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 correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
- **mode** *(int)* : 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.
- **speechiness** *(float)* : 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** *(float)* : 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.

- **time_signature** *(int)* : 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".
- **track_href** *(string)* : A link to the Web API endpoint providing full details of the track.
- **type** *(string)* : The object type.
- **uri** *(string)* : The Spotify URI for the track.
- **valence** *(float)* : 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).

2.0.4 Artists Dataset (73.1 MB)

The artists dataset contains the list of artists along with their popularity, genres, and number of followers in 2020.

- **id** *(string)* : The Spotify ID for the artist.
- **followers** *(float)* : A URL to access the full audio analysis of this track. An access token is required to access this data.
- **genres** *(string)* : 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.
- **name** *(string)* : The duration of the track in milliseconds.
- **popularity** *(int)* : 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.

2.0.5 Charts Dataset (3.46 GB)

- **title** *(string)* : The title of the track.
- **rank** *(int)* : The rank in the chart.
- **date** *(date)* : The date when the song figured in the chart.
- **artist** *(string)* : The name of the artist.
- **url** *(string)* : The Spotify url of the song.
- **region** *(string)* : The region where the song reached a certain rank on a specific date.
- **chart** *(string)* : Charts that tabulate the relative weekly popularity of songs
- **trend** *(string)* : A binary value to represent whether the song's rank moved up or down in a chart, in a specific region.
- **streams** *(int)* : The number of streams accumulated for a song.

2.0.6 Lyrics Dataset (42.47 MB)

- **track id** *(string)* : The Spotify ID of the track.
- **track name** *(string)* : A URL to access the full audio analysis of this track. An access token is required to access this data.
- **track artist** *(string)* : The artist of the track. Note that there can be only one in this dataset.
- **lyrics** *(string)* : The lyrics of the song.
- **track_popularity** *(float)* : The popularity of the track, which is a score between 0 and 1.
- **track_album_id** *(string)* : A unique identifier for the track album.
- **track_album_name** *(string)* : The name of the track album.
- **track_album_release_date** *(date)* : The release date of the track album.
- **playlist_name** *(string)* : The name of the playlist.
- **playlist_id** *(string)* : A unique identifier for the playlist.
- **playlist_genre** *(string)* : The genre of the playlist.
- **playlist_subgenre** *(string)* : The subgenre of the playlist.
- **acousticness** *(float)* : 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** *(float)* : 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** *(float)* : 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.
- **instrumentalness** *(float)* : 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** *(int)* : 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.
- **liveness** *(float)* : 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** *(float)* : 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 correlate of physical strength (amplitude). Values typically range between -60 and 0 db.

- **mode** *(int)* : 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.
- **speechiness** *(float)* : 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.
- **valence** *(float)* : 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).
- **tempo** *(float)* : 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.
- **duration_ms** *(int)* : The duration of the track in milliseconds.
- **language** *(string)* : Language of the song.

3 Data Exploration

First, we authenticate and import libraries.

[87]:
```
# Run this cell to authenticate yourself to BigQuery
# Anhtony : cs145-project-1-365108
# Othman : cs145-365221
from google.colab import auth
auth.authenticate_user()
project_id = 'cs145-365221'
#project_id = 'cs145-project-1-365108' # Anthony's project_id
```

[88]:
```
# Initialize BiqQuery client
from google.cloud import bigquery
client = bigquery.Client(project = project_id)
```

[89]:
```
import matplotlib.pyplot as plt
from plotnine import *
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
import random
import seaborn as sns
import pandas as pd
import numpy as np
```

```
%matplotlib inline
```

3.1 Data preparation and feature engineering

3.1.1 We join the lyrics dataset with the tracks dataset, to have access to the full set of variables concerning tracks. We need to do this because the lyrics dataset does not contain some variables such as 'explicit' or 'release date', and the tracks dataset does not contain variables such as "lyrics" or "playlist genre'.

[90]:
```
%%bigquery prepare --project $project_id

# Join the lyrics dataset to the tracks dataset
SELECT *
FROM `cs145-365221.spotify_database.lyrics` lyrics
JOIN (SELECT explicit, release_date,time_signature,id FROM `cs145-365221.
 ↪spotify_database.tracks` ) tracks
ON lyrics.track_id = tracks.id
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[91]:
```
prepare
```

[91]:
```
                    track_id                            track_name  \
0     1GMDpf82TUwTVBPYiu0dmR                          Switch Lanes
1     2BJSMv0GABRxokHKBOOI8i          Shoota (feat. Lil Uzi Vert)
2     3keUgTGEoZJt0QkzTB6kHg                        Truffle Butter
3     3m8CQnnfJJp4eQMWWl3zay                       Drank in My Cup
4     3uulVrxiI7iLTjOBZsaiF8                          Donald Trump
…                        …                                     …
6541  4Km5HrUvYTaSUfiSGPJeQR  Bad and Boujee (feat. Lil Uzi Vert)
6542  5274I4mUMnYczyeXkGDWZN                            Fine China
6543  3zf852pgVUpYqQD1FTLa69                  Booyah - Original Mix
6544  6dMHdkQmWuDuDltWjBLJBd                                Karate
6545  6qqd7DGn2VXzxsR4k3Ycun                  Fantasias - Unplugged

        track_artist                                             lyrics  \
0               Tyga  Uhh, when I switch lanes, Phantom doors swing …
1       Playboi Carti  Yeah Now Now is my time Now is my time(That-th…
2         Nicki Minaj  You know Touchin' Yeah Night of You know Touch…
3         Kirko Bangz  NA I done came down, hold up Grip the grain, r…
4          Mac Miller  Hey Ayo, Sap! What's good, bruh? This man is k…
…                …                                                …
6541           Migos  You know, young rich niggas You know somethin'…
6542          Future  The world on drugs Ten (Yeah) thousand dollar …
```

```
6543         Showtek  Yes son, all we care about Is then party keepi…
6544           R3HAB  Energy, give me energy Energy, give me energy …
6545  Rauw Alejandro  NA (Yeah)¡Gangalee! (Uh-uh-uh) Ra'-Rauw ¿Cómo …

      track_popularity          track_album_id  \
0                   64  5PKYeoSKEVQd7ZTnwnWRn7
1                   77  7dAm8ShwJLFm9SaJ6Yc58O
2                   64  0cg0JTyl731GnvVS1MyYjj
3                   62  7tivRA9WDD0rWVazWm2pFS
4                   68  6eFkuEfykAUpthUiUeu3zw
…                …                     …
6541                77  2AvupjUeMnSffKEV05x222
6542                80  6P9PZjWXoCRF5b66BafPKY
6543                41  7iQyAbpQ9istpcWKdTQDIZ
6544                57  2d08mANNHmeIsJLnbqE6NU
6545                77  2NQINd10CuEMzd7wBMZc7G

                                track_album_name track_album_release_date  \
0                       Hotel California (Deluxe)               2013-01-01
1                                        Die Lit               2018-05-11
2                                 Truffle Butter               2015-01-23
3                                 Drank In My Cup               2011-09-16
4                           Donald Trump - Single               2011-05-17
…                                             …                      …
6541                                     Culture               2017-01-27
6542  Future & Juice WRLD Present… WRLD ON DRUGS             2018-10-19
6543                         Booyah (The Remixes)               2013-12-20
6544                                      Karate               2014-12-29
6545                        Fantasias (Unplugged)               2019-11-05

         playlist_name             playlist_id  … instrumentalness liveness  \
0        Hip-Hop 'n RnB  0275i1VNfBnsNbPl0QIBpG  …         0.000545   0.1040
1        Hip-Hop 'n RnB  0275i1VNfBnsNbPl0QIBpG  …         0.000000   0.1220
2        Hip-Hop 'n RnB  0275i1VNfBnsNbPl0QIBpG  …         0.000041   0.1240
3        Hip-Hop 'n RnB  0275i1VNfBnsNbPl0QIBpG  …         0.000000   0.1980
4        Hip-Hop 'n RnB  0275i1VNfBnsNbPl0QIBpG  …         0.000000   0.3910
…                 …                     …  …              …        …
6541   Trap Americana  7tkgK1tm9hYkWp7EFyOcAr  …         0.000000   0.1230
6542   Trap Americana  7tkgK1tm9hYkWp7EFyOcAr  …         0.000000   0.1260
6543   Big Room House  7vJOXFe40axY7qS39vGDyH  …         0.011100   0.0622
6544   Big Room House  7vJOXFe40axY7qS39vGDyH  …         0.002400   0.6840
6545  Reggaeton 2020   7xWuNevFBmwnFEg6wzdCc7  …        0.000000   0.1200

      valence    tempo  duration_ms  language  explicit  release_date  \
0       0.463   92.486       221493        en         1    2013-01-01
1       0.480  153.069       153800        en         1    2018-05-11
2       0.491  105.113       219227        en         1    2015-01-23
```

```
3       0.234  132.890       232160        en          1     2011-09-16
4       0.836  162.994       165908        en          1     2011-05-17
…       …      …          …      …      …          …
6541    0.175  127.076       343150        en          1     2017-01-27
6542    0.551  166.111       141587        en          1     2018-10-19
6543    0.499  127.997       311080        en          0     2013-12-20
6544    0.668  128.010       210000        en          0     2014-12-29
6545    0.538   91.952       200594        es          0     2019-11-05

      time_signature                      id
0                  4  1GMDpf82TUwTVBPYiu0dmR
1                  4  2BJSMvOGABRxokHKBOOI8i
2                  4  3keUgTGEoZJt0QkzTB6kHg
3                  4  3m8CQnnfJJp4eQMWWl3zay
4                  4  3uulVrxiI7iLTjOBZsaiF8
…               …                      …
6541               4  4Km5HrUvYTaSUfiSGPJeQR
6542               4  5274I4mUMnYczyeXkGDWZN
6543               4  3zf852pgVUpYqQD1FTLa69
6544               4  6dMHdkQmWuDuDltWjBLJBd
6545               4  6qqd7DGn2VXzxsR4k3Ycun

[6546 rows x 29 columns]
```

3.1.2 Nickolay Lamm, a Pittsburgh-based digital artist, has a recent project called "History of Love" in which he collected the data of the songs on Billboard's Year-End Hot 100 list since 1960. He lists the most popular words in these songs.

The list of these words : - Baby - Girls - Boys - Home - Love - Money - Foul - Body - Sex

We will create dummy variables that account for the presence of these words in our songs.

[92]:
```
%%bigquery dummy --project $project_id

SELECT *,
case when LOWER(lyrics) like '%baby%' THEN 1 ELSE 0 END AS  baby,
case when LOWER(lyrics) like '%girl%' THEN 1 ELSE 0 END AS  girl,
case when LOWER(lyrics) like '%boy%' THEN 1 ELSE 0 END AS  boy,
case when LOWER(lyrics) like '%home%' THEN 1 ELSE 0 END AS  home,
case when LOWER(lyrics) like '%love%' THEN 1 ELSE 0 END AS  love,
case when LOWER(lyrics) like '%money%' THEN 1 ELSE 0 END AS  money,
case when LOWER(lyrics) like '%foul%' THEN 1 ELSE 0 END AS  foul,
case when LOWER(lyrics) like '%body%' THEN 1 ELSE 0 END AS  body,
case when LOWER(lyrics) like '%sex%' THEN 1 ELSE 0 END AS  sex,
FROM
(SELECT *
FROM `cs145-365221.spotify_database.lyrics` lyrics
```

```
JOIN (SELECT explicit, release_date,time_signature,id FROM `cs145-365221.
↪spotify_database.tracks` ) tracks

ON lyrics.track_id = tracks.id)
ORDER BY RAND()
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[93]:
```
dummy
```

[93]:
```
                    track_id      track_name      track_artist  \
0     0TiC3GtlMCskf2hIUIBcDV       Crew Love             Drake
1     5RsUlxLto4NZbhJpqJbHfN  Jessie's Girl  Rick Springfield
2     0ntQJM78wzOLVeCUAW7Y45     Sex on Fire     Kings of Leon
3     3JyvSSU0VnlMUsQckyEVfX        Darkside          grandson
4     5LN1B9uVAVleCZ2euGarvi             MVP             Big L
…                        …             …               …
6541  31GBvPUgO7MJltUnBlOpe9     Mass Appeal       Gang Starr
6542  4h0zU3O9R5xzuTmNO7dNDU        Lost Boy          Ruth B.
6543  5rwdhliMmo0aAQ08vU0AOZ            Maps          Maroon 5
6544  75JFxkI2RXiU7L9VXzMkle  The Scientist          Coldplay
6545  21p8xjq0WTm3HZKHuDEweg         Tell Me     Groove Theory

                                                 lyrics  track_popularity  \
0     Take your nose off my keyboard What you bother…                51
1     Jessie is a friend Yeah, I know, he's been a g…                70
2     Lay where you're layin' Don't make a sound I k…                79
3     The kid has got a dark side Best believe it, p…                60
4     Ayo, spark up the phillies and pass the stout …                53
…                                                   …               …
6541  NA "Money's growin' like grass with the mass a…                61
6542  There was a time when I was alone Nowhere to g…                76
6543  I miss the taste of a sweeter life I miss the …                60
6544  Come up to meet you, tell you I'm sorry You do…                83
6545  I've been doing my own thing Love has always h…                63

                track_album_id                  track_album_name  \
0       63WdJvk8G9hxJn8u5rswNh               Take Care (Deluxe)
1       4KKFWTePKtgb6mOwFDqxYa                Working Class Dog
2       5CZR6ljD0x9fTiS4mh9wMp                Only By The Night
3       6puy3Q1mjuizTB4i91Xorq           a modern tragedy vol. 2
4       7xvBUHu5jJ7X0wdRHudLFD  Lifestylez Ov Da Poor & Dangerous
…                        …                               …
6541    67kl5m0df6Bn0aSe3g5Ea7                      Hard To Earn
6542    7drYNu2imHk188vP81icR3                          Lost Boy
6543    4KXLjIEas8MTwwX3xpmAdC                        V (Deluxe)
```

```
6544  0RHX9XECH8IVI3LNgWDpmQ        A Rush of Blood to the Head
6545  0VVegiri01eyyf0KrLmxtc                     Groove Theory

     track_album_release_date                                    playlist_name  \
0                  2011-11-15                              Urban Contemporary
1                        1981                            The Sound of Album Rock
2                  2008-09-23                                  Permanent wave
3                  2019-02-22                            2019 in Indie Poptimism
4                  1995-03-28                  90's Hip Hop Ultimate Collection
…                         …                                                 …
6541               1994-03-08                  90's Hip Hop Ultimate Collection
6542               2015-08-21                              urban contemporary
6543               2015-05-18                         Today's Hits 2000-Present
6544               2002-08-08  Mix ElectroPop//ElectroHouse// DeepHouse 2020
6545               1995-07-25       New Jack Swing - 90s R&B fused w Hip Hop

                 playlist_id  …                      id baby  girl  boy  \
0     4Pbs84EQbuAblxlp6Chz0d  …  0TiC3GtlMCskf2hIUIBcDV    0     1    0
1     3yj9YnQGTdnFuKbDyXGDi6  …  5RsUlxLto4NZbhJpqJbHfN    0     1    0
2     0t0y7ZY4E2PadXIyj8zU43  …  0ntQJM78wz0LVeCUAW7Y45    0     0    0
3     16RNbqnNCCLlBJti7JU5nc  …  3JyvSSU0VnlMUsQckyEVfX    0     0    0
4     4IG024zoaGMurhTFBkMAv9  …  5LN1B9uVAVleCZ2euGarvi    0     1    0
…                          …   …                      …    …     …    …
6541  4IG024zoaGMurhTFBkMAv9  …  31GBvPUg07MJltUnBl0pe9    0     0    1
6542  4WiB26kw0INKwbzfb5M6Tv  …  4h0zU309R5xzuTmN07dNDU    0     0    1
6543  6a66cg3HcsjYkisYyQcov6  …  5rwdhliMmo0aAQ08vU0A0Z    1     0    0
6544  23swqzp0ZwW1NhPiZ7iyFI  …  75JFxkI2RXiU7L9VXzMkle    0     0    0
6545  79xd4wnVuKZK4rJMsL2wPa  …  2lp8xjq0WTm3HZKHuDEweg    1     1    1

      home  love  money  foul  body  sex
0        0     0      1     0     0    0
1        0     1      0     0     1    0
2        0     0      0     0     0    1
3        0     0      0     0     1    0
4        0     1      1     0     0    0
…     …    …     …    …     …    …
6541     0     0      1     0     0    0
6542     1     1      0     0     0    0
6543     0     0      0     0     0    0
6544     0     1      0     0     1    0
6545     0     1      0     0     0    0

[6546 rows x 38 columns]
```

3.1.3 We randomly shuffle the dataset and add it to BigQuery. The reason we shuffle it is because in the Machine Learning part, we are going to take the first 80% as training, the next 10% as validation, and the last 10% as testing. We would like this split to be random so we shuffle the dataset now.

[ ]:
```
# We upload this dataframe to BigQuery because we are going to work on it for␣
↪the rest of the project.

random.seed(10)

dummy = shuffle(dummy)

dummy.to_csv('/content/sample_data/dummy.csv')

# some variables
filename = '/content/sample_data/dummy.csv' # this is the file path to your csv
dataset_id = 'spotify_database'
table_id = 'processed_tracks'

# tell the client everything it needs to know to upload our csv
dataset_ref = client.dataset(dataset_id)
table_ref = dataset_ref.table(table_id)
job_config = bigquery.LoadJobConfig()
job_config.source_format = bigquery.SourceFormat.CSV
job_config.autodetect = True

# load the csv into bigquery
with open(filename, "rb") as source_file:
    job = client.load_table_from_file(source_file, table_ref,␣
↪job_config=job_config)

job.result()  # Waits for table load to complete.

# looks like everything worked :)
print("Loaded {} rows into {}:{}.".format(job.output_rows, dataset_id,␣
↪table_id))
```

3.2 Data visualization

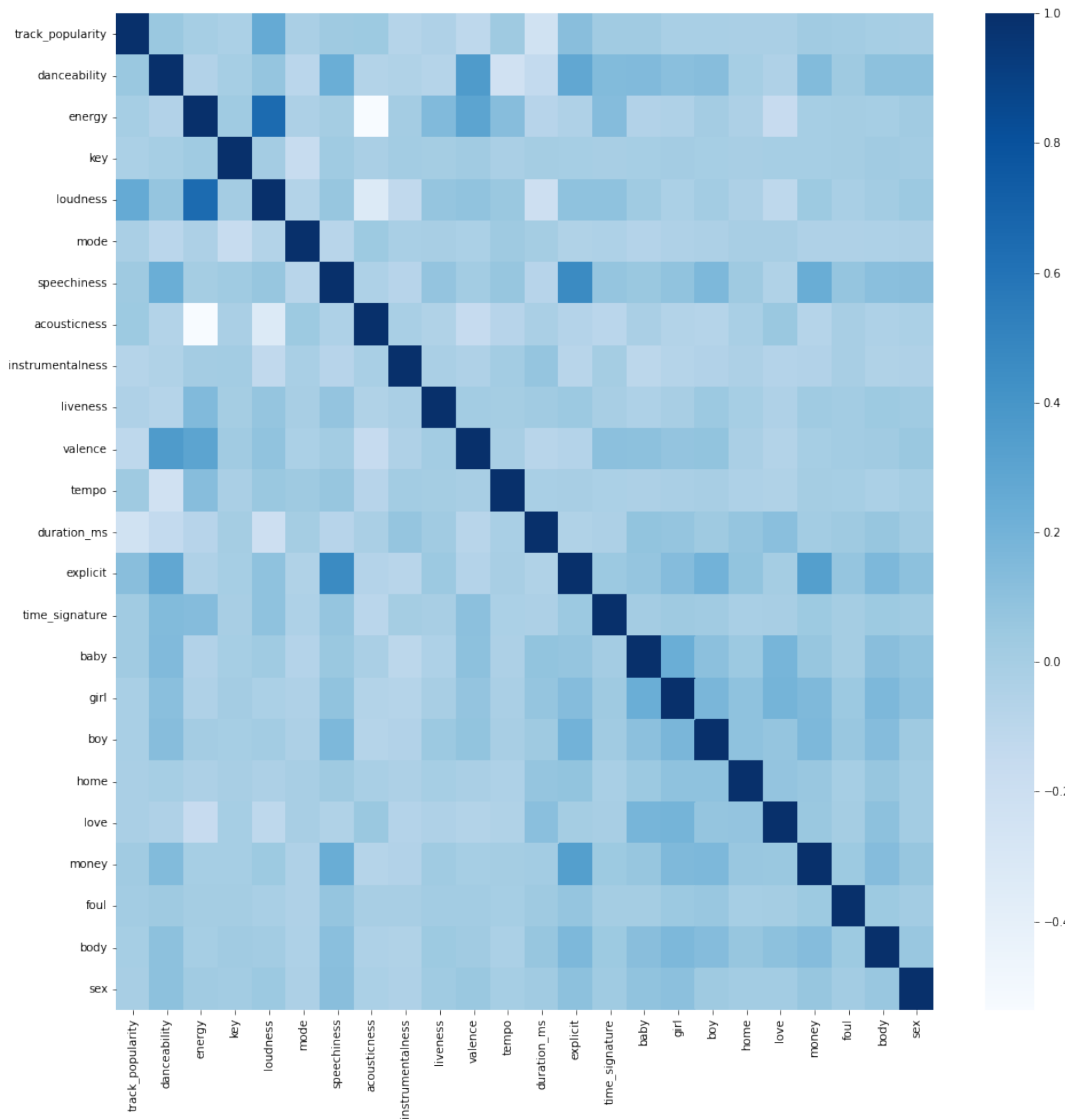
3.3 We visualize the correlation among all numeric variables

Correlation among variables

[94]:
```
# We get the numeric variables and change their type from Int64 to int64. This␣
↪is because the uppercase i is a problem for the ggplot library.
dummy_numeric = dummy._get_numeric_data()
for col in dummy_numeric:
  if dummy_numeric[col].dtype == 'Int64':
```

```
    dummy_numeric[col] = dummy_numeric[col].astype('int64')

# Heatmap
corr_mat= dummy_numeric.corr()
plt.figure(figsize=(16,16))
sns.heatmap(corr_mat,cmap="Blues")
```

[94]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa3f9fe4f70>

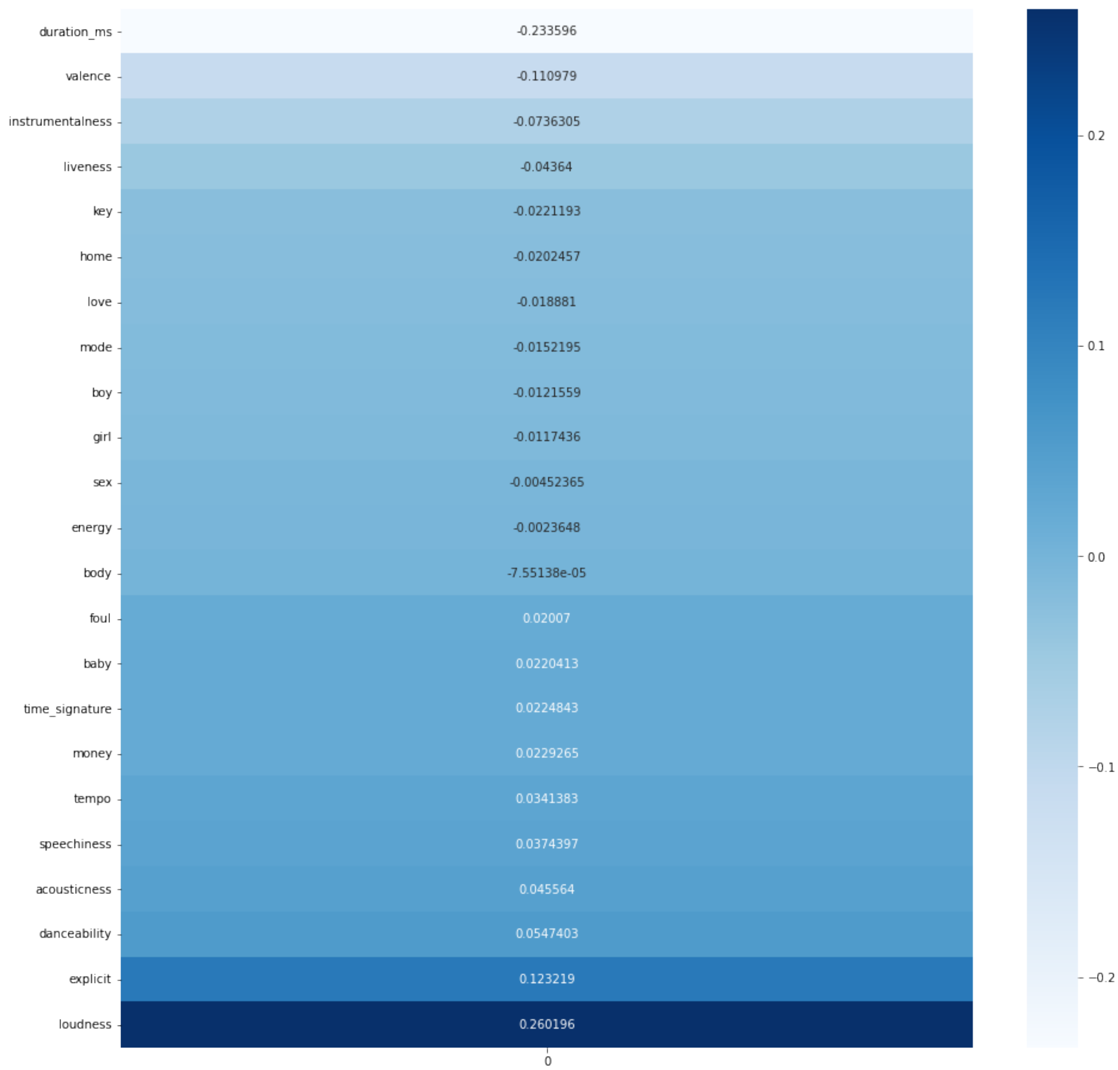


3.3.1 Most variables are not correlated, but we can see some exceptions, such as "explicit" with "speechiness", or "loudness" with "energy".

3.4 Our target variable is 'popularity'. Thus, we visualize the correlation of each variable with our target variable 'popularity'.

Correlation of variables with 'popularity'

[ ]:
```
scale = dummy_numeric.drop("track_popularity", axis=1).apply(lambda x: x.
 ↪corr(dummy_numeric.track_popularity))
plt.figure(figsize=(16,16))
sns.heatmap(pd.DataFrame(scale.sort_values()), annot=True, fmt="g",␣
 ↪cmap='Blues')

plt.show()
```



3.4.1 We can see that 'duration_ms' and 'valence' are the two most negatively correlated variables with popularity, while 'explicit' and 'loudness' are the two most positively correlated variables with popularity. Given this, we predict that these four variables will most succesfully predict popularity in our ML model.

3.4.2 Now, we analyze the distribution of popularity among all songs of our dataset.

Distribution of popularity

[ ]:
```
%%bigquery popularity_distribution --project $project_id

WITH pop AS
(SELECT
case
        when track_popularity > 0 and track_popularity <= 10  then 'Between 0␣
 ↪and 10'
        when track_popularity > 10 and track_popularity <= 20  then 'Between 10␣
 ↪and 20'
        when track_popularity > 20 and track_popularity <= 30  then 'Between 20␣
 ↪and 30'
        when track_popularity > 30 and track_popularity <= 40  then 'Between 30␣
 ↪and 40'
        when track_popularity > 40 and track_popularity <= 50  then 'Between 40␣
 ↪and 50'
        when track_popularity > 50 and track_popularity <= 60  then 'Between 50␣
 ↪and 60'
        when track_popularity > 60 and track_popularity <= 70  then 'Between 60␣
 ↪and 70'
        when track_popularity > 70 and track_popularity <= 80  then 'Between 70␣
 ↪and 80'
        when track_popularity > 80 and track_popularity <= 90  then 'Between 80␣
 ↪and 90'
        when track_popularity > 90 then 'Greater than 90'
    end as pop_count,
    FROM `cs145-365221.spotify_database.processed_tracks`)
SELECT pop_count,COUNT(pop_count) AS count_pop FROM pop GROUP BY pop_count
```

```
Query is running:   0%|          |
```

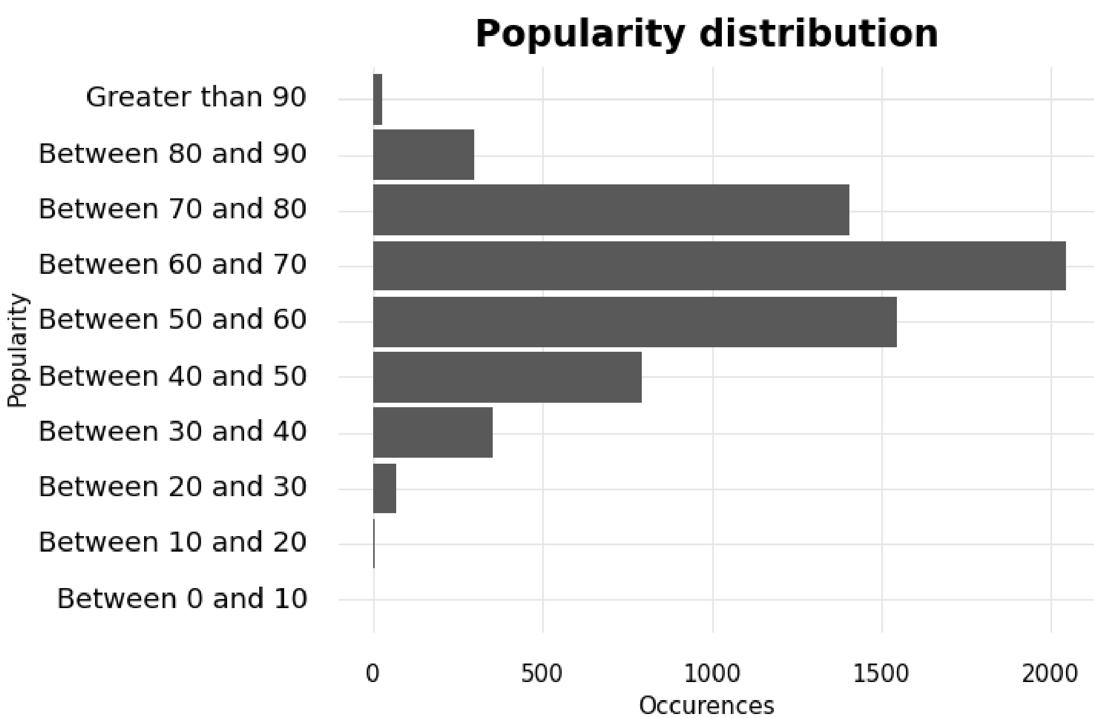
```
Downloading:   0%|          |
```

[ ]:
```
popularity_distribution
```

[ ]:
```
           pop_count  count_pop
0  Between 30 and 40        354
1  Between 60 and 70       2042
2  Between 40 and 50        795
3  Between 70 and 80       1403
```

4	Between 50 and 60	1546
5	Between 80 and 90	301
6	Greater than 90	28
7	Between 20 and 30	69
8	Between 10 and 20	7
9	Between 0 and 10	1

[ ]:
```
popularity_distribution['count_pop'] = popularity_distribution['count_pop'].astype(np.int64)
ggplot(popularity_distribution, aes(x = 'pop_count', y = 'count_pop')) + geom_bar(stat = "identity") + labs(x = "Popularity", y = "Occurences", title = "Popularity distribution") + theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11, family='Arial',color="black"),axis_title_y = element_text(size= 11, family='Arial',color="black"),panel_grid_minor = element_blank()) + coord_flip()
```



[ ]: <ggplot: (8765659758084)>

3.4.3 It seems that popularity is normally distributed across all songs of our dataset, with mean between 60 and 70.

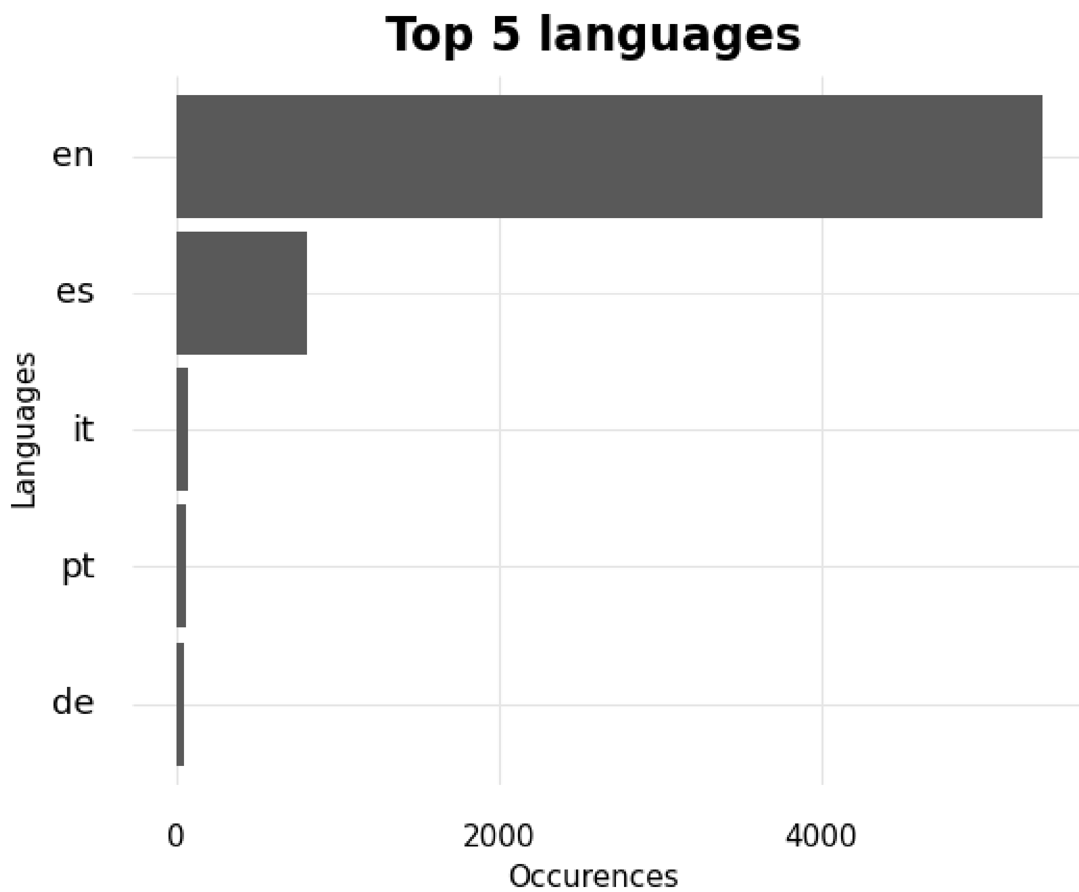
3.4.4 Songs can be written in different languages. We know visualize the top 5 languages.

Top 5 languages

[ ]:
```
%%bigquery langs --project $project_id
SELECT language, COUNT(language) AS occurences
FROM `cs145-365221.spotify_database.processed_tracks`
WHERE language != "NA"
GROUP BY language
ORDER BY occurences DESC
LIMIT 5
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[ ]:
```
langs['occurences'] = (langs['occurences']).astype(np.int64)
ggplot(langs, aes(x = 'reorder(language,occurences)', y = 'occurences')) +␣
 ↪geom_bar(stat = "identity") + labs(x = "Languages", y = "Occurences", title␣
 ↪= "Top 5 languages") + theme_minimal()  +␣
 ↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
 ↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
 ↪family='Arial',color="black"),panel_grid_minor = element_blank()) +␣
 ↪coord_flip()
```



[ ]: <ggplot: (8765659738146)>

3.4.5 English is by far the most used language, followed by spanish.

3.4.6 Now, we would like to study whether language affects a song's popularity.

Is Language a Possible Factor in Popularity?

[95]:
```
%%bigquery lang --project $project_id
SELECT language, AVG(track_popularity) AS avg_pop
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE language != "NA"
GROUP BY language
HAVING COUNT(language) > 10
ORDER BY avg_pop DESC
```

```
Query is running:   0%|          |
```

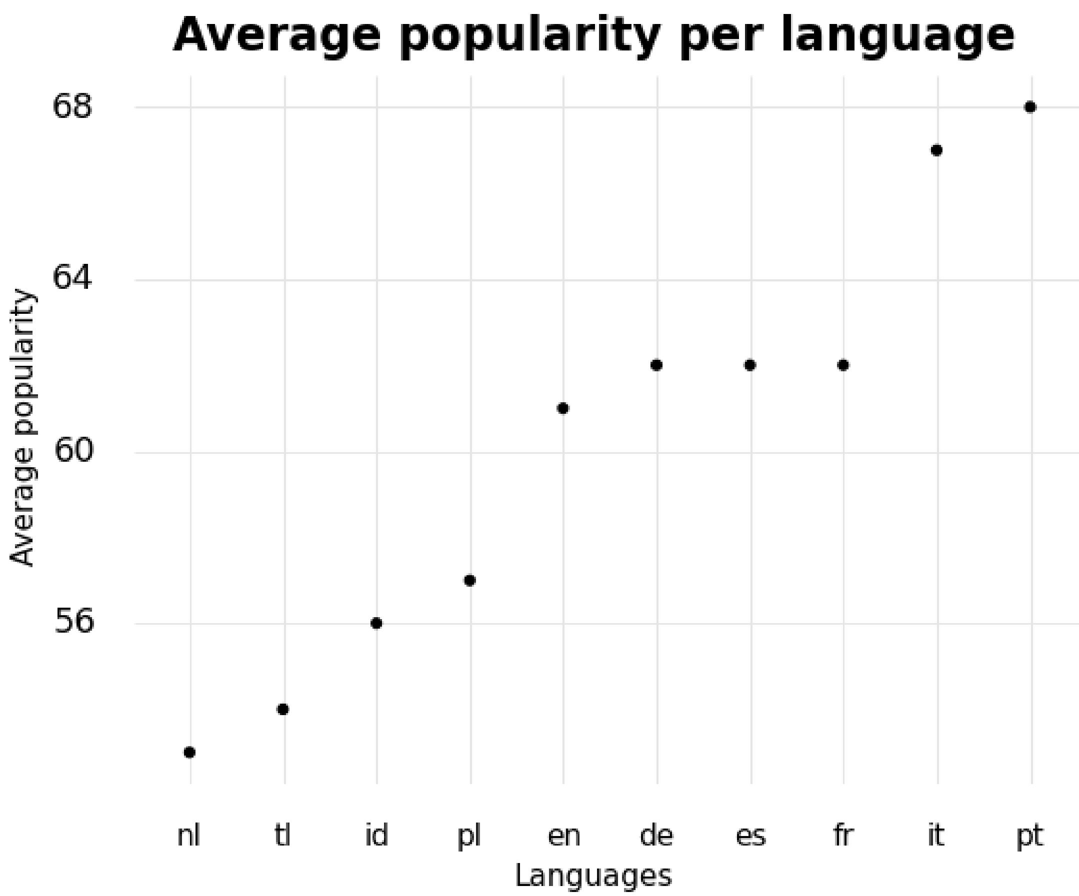
```
Downloading:   0%|          |
```

[96]:
```
lang
```

[96]:

	language	avg_pop
0	pt	68.303571
1	it	67.300000
2	de	62.954545
3	es	62.932015
4	fr	62.176471
5	en	61.695045
6	pl	57.791667
7	id	56.933333
8	tl	54.950000
9	nl	53.650000

[99]:
```
lang['avg_pop'] = (lang['avg_pop']).astype(np.int64)
ggplot(lang, aes(x = 'reorder(language,avg_pop)', y = 'avg_pop')) +␣
 ↪geom_point() + labs(x = "Languages", y = "Average popularity", title =␣
 ↪"Average popularity per language") + theme_minimal()  +␣
 ↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
 ↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
 ↪family='Arial',color="black"),panel_grid_minor = element_blank())
```



[99]: <ggplot: (8771390417917)>

3.4.7 Although most songs of the dataset are in english, the latter is not the most popular language. Portuguese and Italian are on average more popular than all other languages, followed by french, spanish, deutsch, and finally english.

3.4.8 Now, we look at the relationship between popularity and the 4 variables that we identified earlier in the heatmap : loudness, duration, explicit and valence.

Loudness

[ ]:
```
%%bigquery loudness_data --project $project_id
SELECT loudness, track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
```

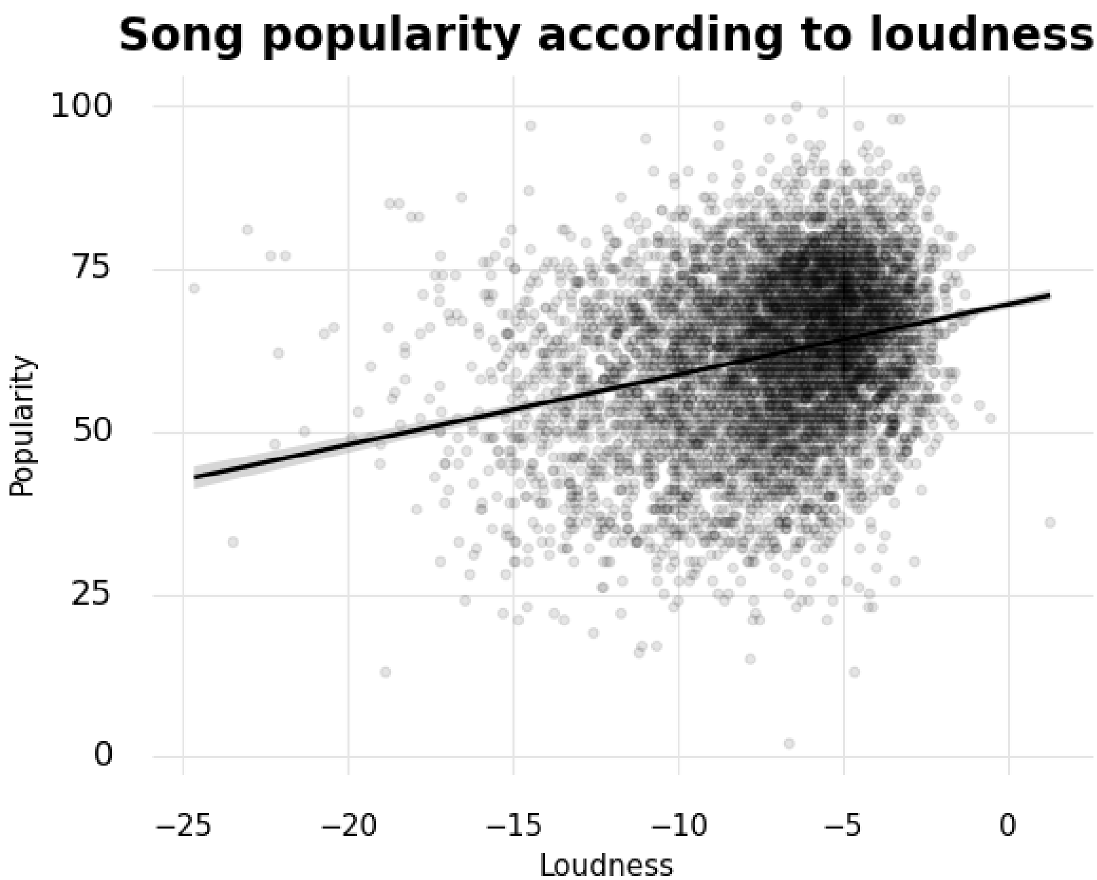
```
Query is running:   0%|          |
Downloading:   0%|          |
```

[ ]:
```
loudness_data
```

[ ]:
```
      loudness  track_popularity
0       -5.473                38
1       -5.442                65
2       -6.668                69
3       -3.754                47
4       -6.454                65
…          …                 …
6541    -7.070                53
6542    -7.865                52
6543    -5.767                62
6544    -7.358                80
6545    -4.633                57

[6546 rows x 2 columns]
```

[ ]:
```
loudness_data['track_popularity'] = loudness_data['track_popularity'].astype(np.
 ↪int64)
```

[ ]:
```
loudness_data['track_popularity'] = loudness_data['track_popularity'].astype(np.
 ↪int64)

ggplot(loudness_data, aes(x = 'loudness', y = 'track_popularity')) +␣
 ↪geom_point(alpha = 0.1) + geom_smooth(method = "lm") + labs(x = "Loudness",␣
 ↪y = "Popularity", title = "Song popularity according to loudness") +␣
 ↪theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17,␣
 ↪family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
 ↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
 ↪family='Arial',color="black"),panel_grid_minor = element_blank())
```




```
[ ]: <ggplot: (8765657712086)>
```

3.4.9 According to this chart, the louder a song, the more popular it is, even though there is no perfect correlation.

3.4.10 We conduct the same analysis with duration.

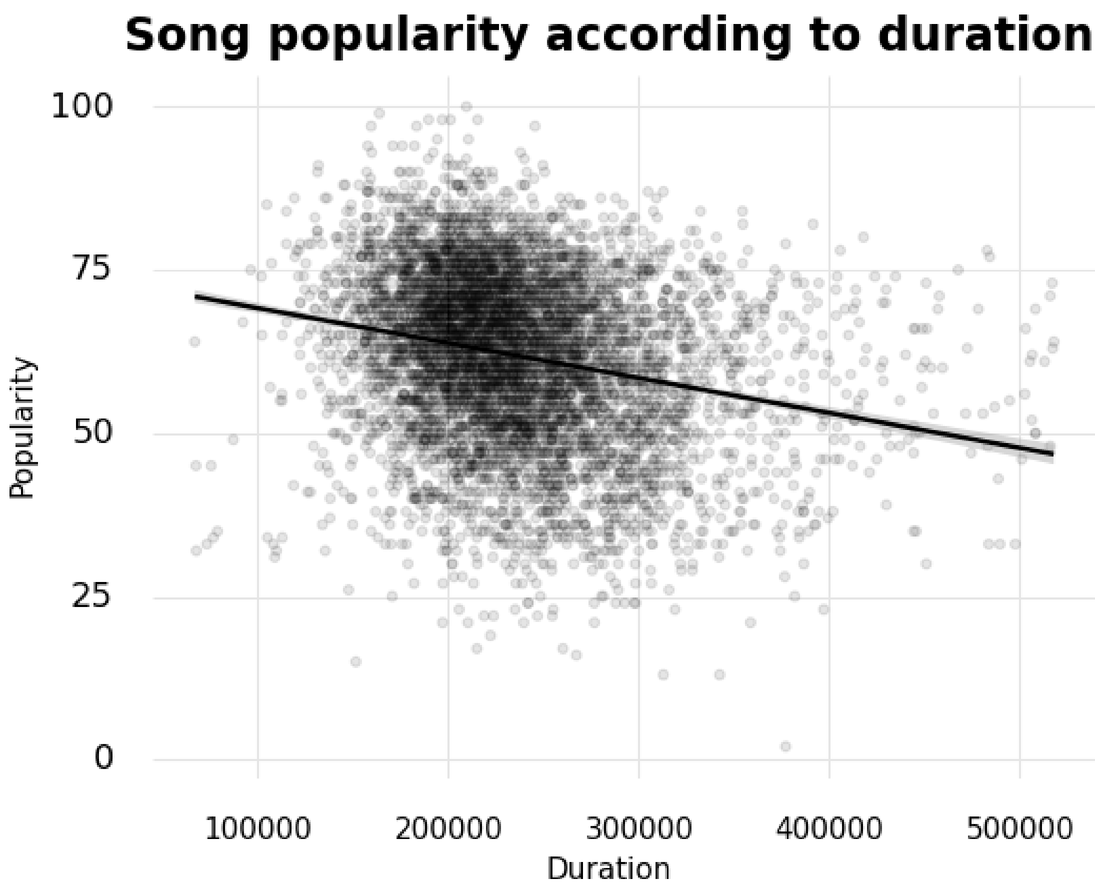
Duration

```
[ ]: %%bigquery duration --project $project_id
SELECT duration_ms, track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

```
[ ]: duration['track_popularity'] = duration['track_popularity'].astype(np.int64)
duration['duration_ms'] = duration['duration_ms'].astype(np.int64)
```

```
ggplot(duration, aes(x = 'duration_ms', y = 'track_popularity')) +␣
↪geom_point(alpha = 0.1) + geom_smooth(method = "lm") + labs(x = "Duration",␣
↪y = "Popularity", title = "Song popularity according to duration") +␣
↪theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17,␣
↪family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank())
```




```
[ ]: <ggplot: (8765656834262)>
```

3.4.11 The longer a song, the less popular it is. This might be because people like catchy, quick songs, and because radios do not necessarily play long songs.

3.4.12 We conduct the same analysis with valence.

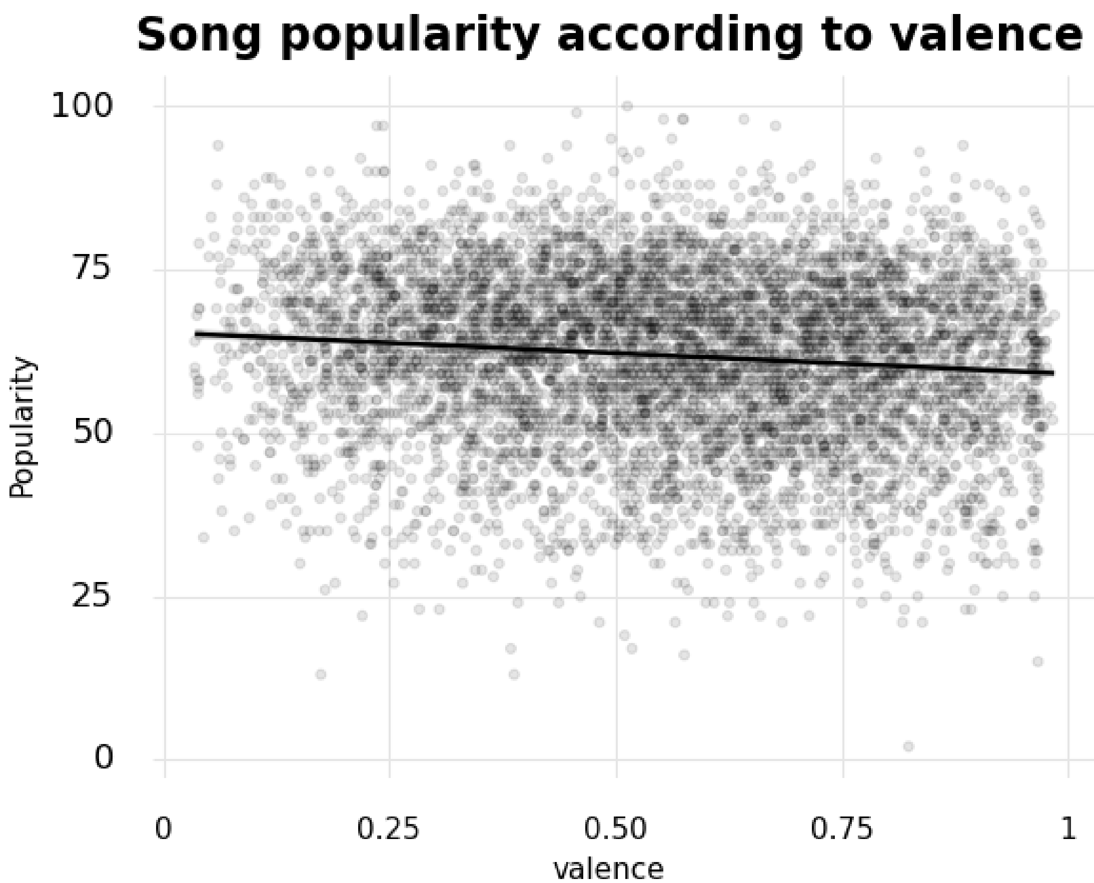
Valence

[4]:
```
%%bigquery val --project $project_id
SELECT valence, track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
```

```
Query is running:   0%|          |
```

```
Downloading:   0%|          |
```

[7]:
```
val['track_popularity'] = val['track_popularity'].astype(np.float64)
val['valence'] = val['valence'].astype(np.float64)

ggplot(val, aes(x = 'valence', y = 'track_popularity')) + geom_point(alpha = 0.
 ↪1) + geom_smooth(method = "lm") + labs(x = "valence", y = "Popularity",␣
 ↪title = "Song popularity according to valence") + theme_minimal()  +␣
 ↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
 ↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
 ↪family='Arial',color="black"),panel_grid_minor = element_blank())
```



[7]: <ggplot: (8771397815757)>

3.4.13 The relationship between valence and popularity seems to be rather flat, which means that valence might not be the best predictor of popularity.

Explicit

[157]:
```
%%bigquery explicit --project $project_id
SELECT explicit, avg(track_popularity) AS avg_track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`
GROUP BY explicit
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
```

```
Query is running:   0%|          |
```

```
Downloading:   0%|          |
```

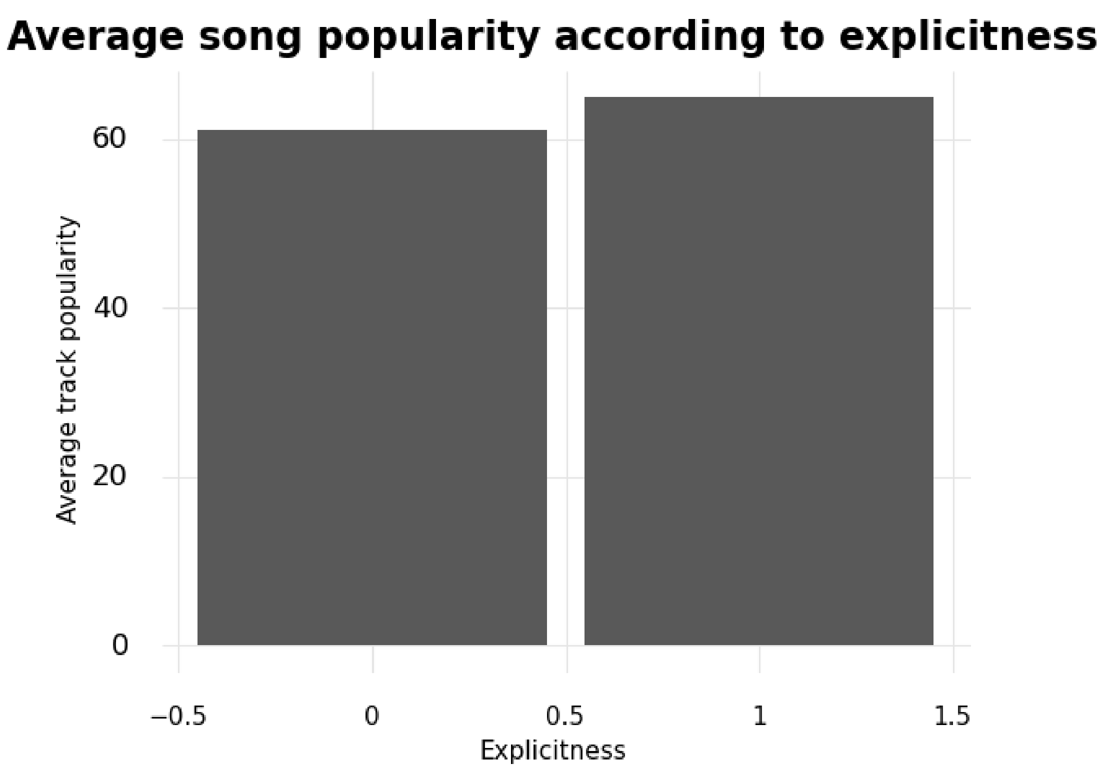
[162]:
```
explicit
```

[162]:

	explicit	avg_track_popularity
0	1	64.913328
1	0	61.027644

[164]:

```
explicit['explicit'] = explicit['explicit'].astype(np.float64)

ggplot(explicit, aes(x = 'explicit', y = 'avg_track_popularity')) +␣
↪geom_bar(stat = 'identity') + labs(x = "Explicitness", y = "Average track␣
↪popularity", title = "Average song popularity according to explicitness") +␣
↪theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17,␣
↪family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank())
```



[164]: <ggplot: (8771390118858)>

3.4.14 Explicit songs are on average more popular.

3.5 Genres / Sub-genres

3.5.1 We now analyze the distribution of genres in our dataset.

3.5.2 Distribution of genres

[40]:
```
%%bigquery genre_distrib --project $project_id
SELECT count(playlist_genre) AS count_playlist_genre, playlist_genre
FROM `cs145-365221.spotify_database.processed_tracks`
GROUP BY playlist_genre
ORDER BY count_playlist_genre DESC
```

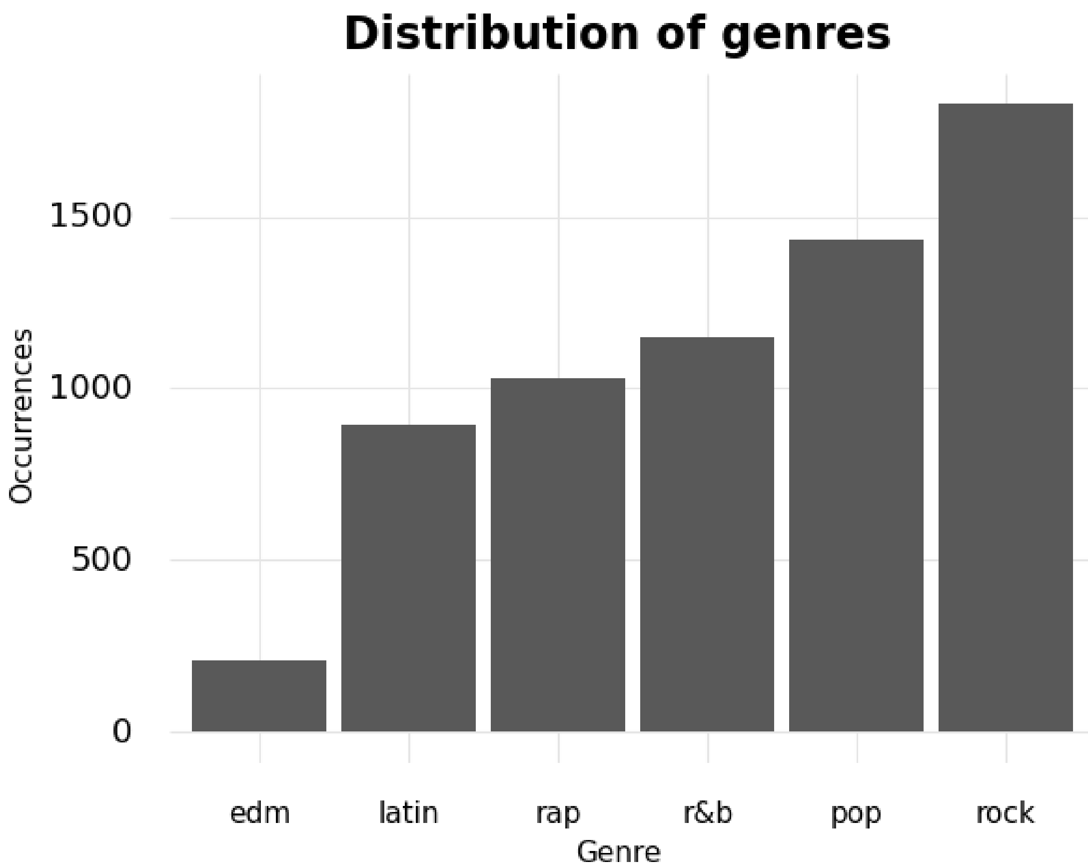
```
Query is running:   0%|          |
Downloading:   0%|          |
```

[41]:
```
genre_distrib
```

[41]:

	count_playlist_genre	playlist_genre
0	1827	rock
1	1433	pop
2	1152	r&b
3	1031	rap
4	896	latin
5	207	edm

[60]:
```
genre_distrib['count_playlist_genre'] = genre_distrib['count_playlist_genre'] .
 ↪astype(np.int64)

ggplot(genre_distrib, aes(x = 'reorder(playlist_genre,count_playlist_genre)', y␣
 ↪= 'count_playlist_genre')) + geom_bar(stat = 'identity') + labs(x = "Genre",␣
 ↪y = "Occurrences", title = "Distribution of genres") + theme_minimal()  +␣
 ↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
       legend_position= 'none',
       legend_title = element_blank(),
       legend_text = element_blank(),
       axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
       axis_text_x = element_text(size= 11,␣
 ↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
 ↪family='Arial',color="black"),panel_grid_minor = element_blank())
```



[60]: <ggplot: (8771394355697)>

3.5.3 Genres and popularity

3.5.4 Now we analyze the popularity of each genre.

[26]:
```
%%bigquery genres --project $project_id
SELECT AVG(track_popularity) as avg_popularity, playlist_genre
FROM `cs145-365221.spotify_database.processed_tracks`
GROUP BY playlist_genre
ORDER BY avg_popularity DESC
```

```
Query is running:   0%|          |
```

```
Downloading:   0%|          |
```

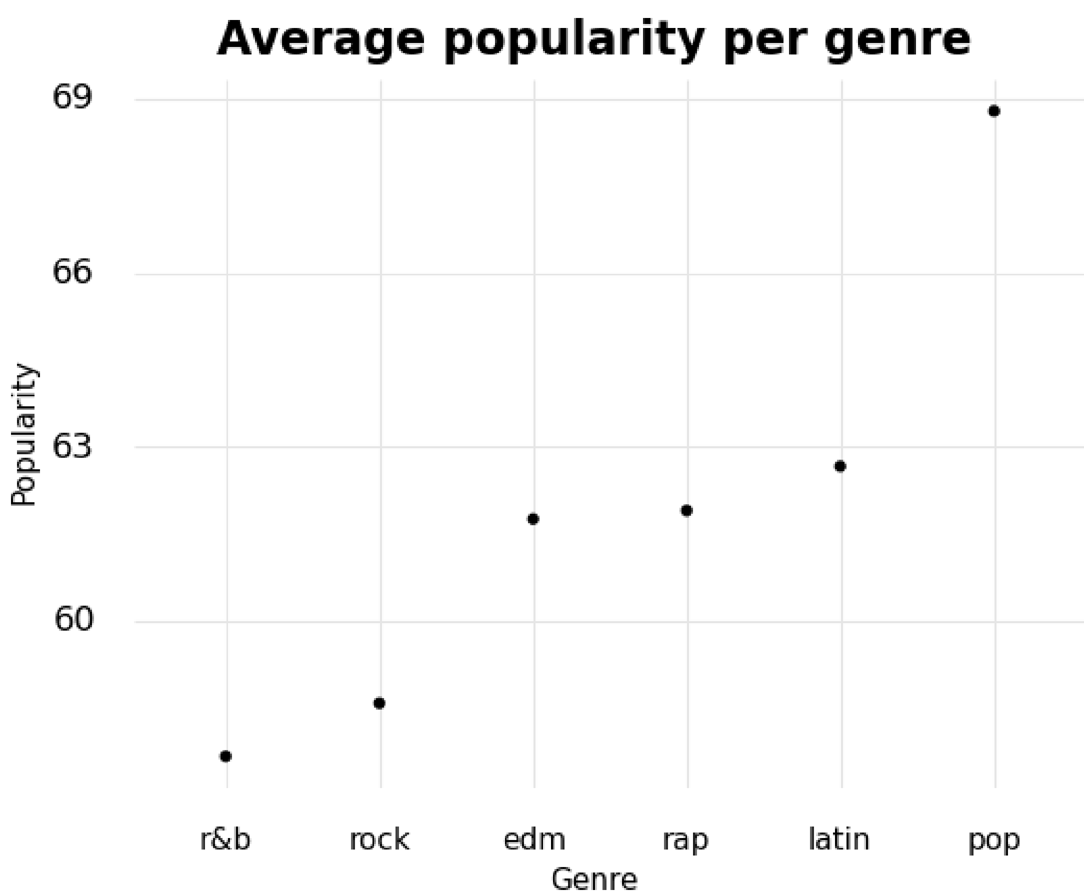
[27]:
```
genres
```

[27]:
```
   avg_popularity playlist_genre
0       68.788555            pop
```

```
1        62.668527          latin
2        61.907856            rap
3        61.763285            edm
4        58.590038           rock
5        57.674479            r&b
```

[100]:

```
ggplot(genres, aes(x = 'reorder(playlist_genre,avg_popularity)', y =␣
↪'avg_popularity')) + geom_point() + labs(x = "Genre", y = "Popularity",␣
↪title = "Average popularity per genre") + theme_minimal()  +␣
↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank())
```



[100]: <ggplot: (8771390362043)>

3.5.5 Pop music seems to be the most popular genre on average, and there is a significant difference among genres, which means that this could be a good predictor of popularity.

3.5.6 Distribution of subgenres

[57]:
```
%%bigquery subgenre_distrib --project $project_id
SELECT count(playlist_subgenre) AS count_playlist_subgenre, playlist_subgenre
FROM `cs145-365221.spotify_database.processed_tracks`
GROUP BY playlist_subgenre
ORDER BY count_playlist_subgenre DESC
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

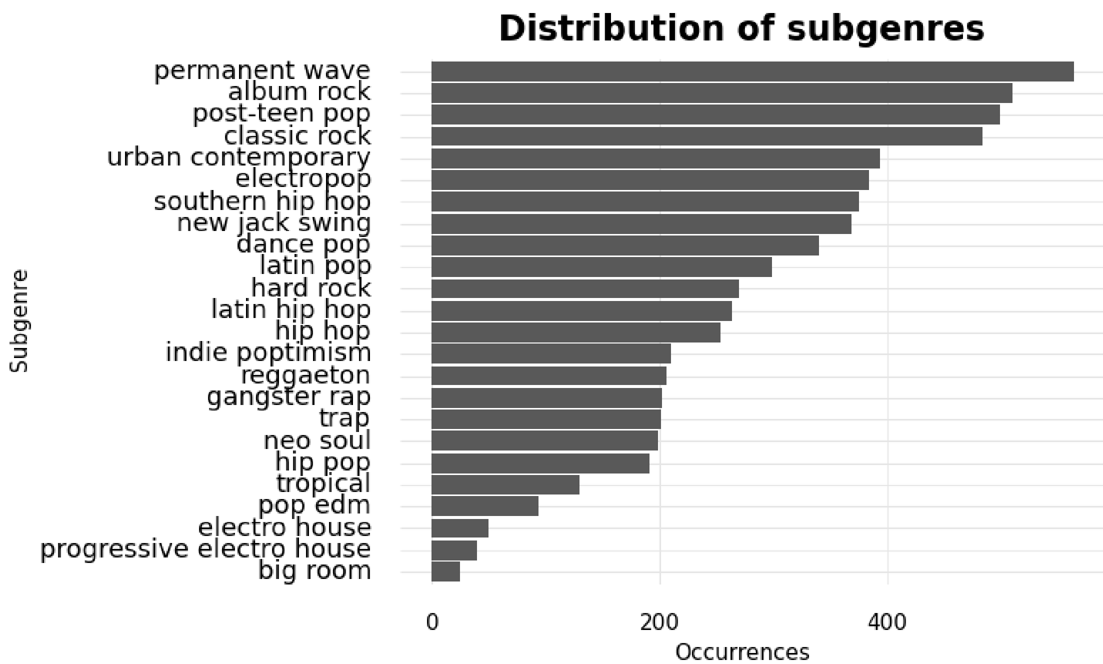
[58]:
```
subgenre_distrib
```

[58]:

	count_playlist_subgenre	playlist_subgenre
0	564	permanent wave
1	510	album rock
2	499	post-teen pop
3	483	classic rock
4	394	urban contemporary
5	384	electropop
6	375	southern hip hop
7	368	new jack swing
8	340	dance pop
9	298	latin pop
10	270	hard rock
11	263	latin hip hop
12	253	hip hop
13	210	indie poptimism
14	206	reggaeton
15	202	gangster rap
16	201	trap
17	199	neo soul
18	191	hip pop
19	129	tropical
20	93	pop edm
21	50	electro house
22	40	progressive electro house
23	24	big room

[59]:
```
subgenre_distrib['count_playlist_subgenre'] =␣
↪subgenre_distrib['count_playlist_subgenre'].astype(np.int64)
```

```
ggplot(subgenre_distrib, aes(x = 'reorder(playlist_subgenre,count_playlist_subgenre)', y = 'count_playlist_subgenre')) + geom_bar(stat = 'identity') + labs(x = "Subgenre", y = "Occurrences", title = "Distribution of subgenres") + theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11, family='Arial',color="black"),axis_title_y = element_text(size= 11, family='Arial',color="black"),panel_grid_minor = element_blank()) + coord_flip()
```



[59]: <ggplot: (8771391141798)>

3.5.7 Permanent wave and album rock are two different subgenres of rock and are the most popular subgenres.

3.5.8 Sub-genres and popularity

[102]:
```
%%bigquery subgenres --project $project_id
SELECT AVG(track_popularity) as avg_popularity, playlist_subgenre
FROM `cs145-365221.spotify_database.processed_tracks`
GROUP BY playlist_subgenre
ORDER BY avg_popularity DESC
```

```
Query is running:   0%|          |
Downloading:   0%|          |
```

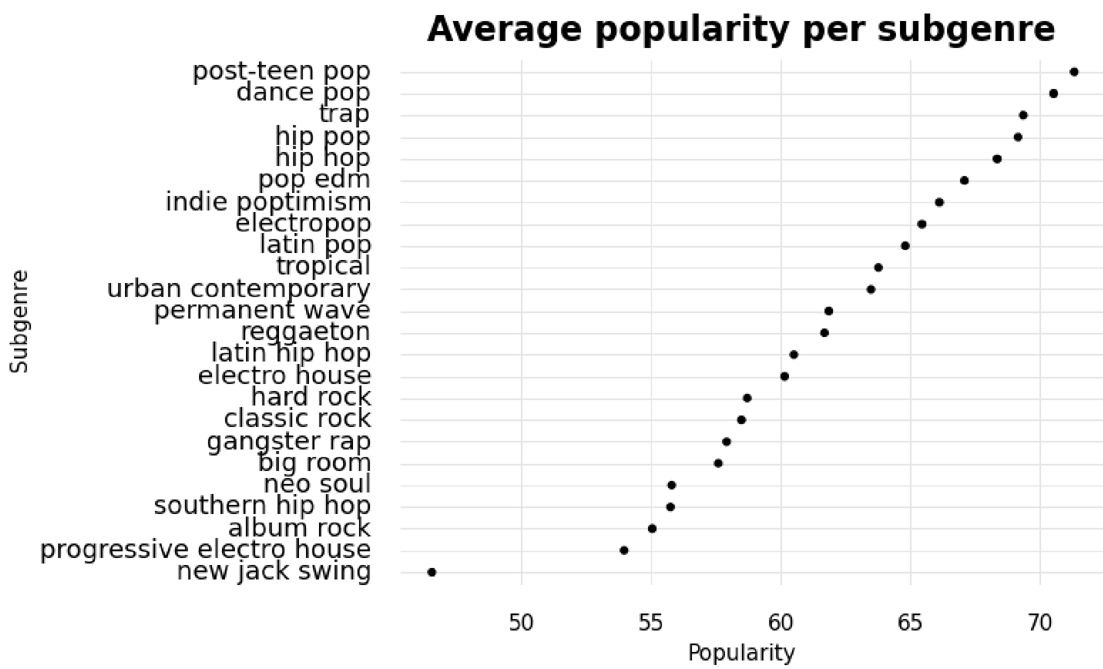
[103]:
```
subgenres
```

[103]:

	avg_popularity	playlist_subgenre
0	71.314629	post-teen pop
1	70.520588	dance pop
2	69.343284	trap
3	69.146597	hip pop
4	68.339921	hip hop
5	67.075269	pop edm
6	66.109524	indie poptimism
7	65.437500	electropop
8	64.798658	latin pop
9	63.759690	tropical
10	63.469543	urban contemporary
11	61.851064	permanent wave
12	61.674757	reggaeton
13	60.498099	latin hip hop
14	60.140000	electro house
15	58.696296	hard rock
16	58.476190	classic rock
17	57.905941	gangster rap
18	57.583333	big room
19	55.783920	neo soul
20	55.738667	southern hip hop
21	55.035294	album rock
22	53.950000	progressive electro house
23	46.538043	new jack swing

[104]:
```
ggplot(subgenres, aes(x = 'reorder(playlist_subgenre,avg_popularity)', y =␣
↪'avg_popularity')) + geom_point() + labs(x = "Subgenre", y = "Popularity",␣
↪title = "Average popularity per subgenre") + theme_minimal()  +␣
↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
```

```
    legend_title = element_blank(),
    legend_text = element_blank(),
    axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
    axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank()) +␣
↪coord_flip()
```



[104]: <ggplot: (8771390328104)>

3.5.9 Post-teen pop and dance pop are two subgenres of pop and are the two most popular subgenres. This was expected given the young demographic of Spotify users.

3.5.10 Incorporation of charts into popularity distribution

Which countries have the highest average popularity across their number #1 charting songs in 2020 in the Top200 chart?

[106]:
```
%%bigquery top_countries_charting1 --project $project_id
WITH d AS(
SELECT region, title, artist
FROM `cs145-365221.spotify_database.charts` c
#FROM `cs145-project-1-365108.spotify_database.charts` c
WHERE EXTRACT(YEAR FROM c.date) = 2020 AND rank = 1 AND chart = 'top200'
GROUP BY region, title, artist)
```

```
SELECT region, AVG(track_popularity) AS avg_pop # title, artist
FROM d
JOIN `cs145-365221.spotify_database.processed_tracks` t
#JOIN `cs145-project-1-365108.spotify_database.processed_tracks` t
ON t.track_name = d.title
GROUP BY region #, title, artist
ORDER BY avg_pop DESC
LIMIT 20
```

```
Query is running:   0%|          |
Downloading:   0%|          |
```

[107]:

```
top_countries_charting1
```

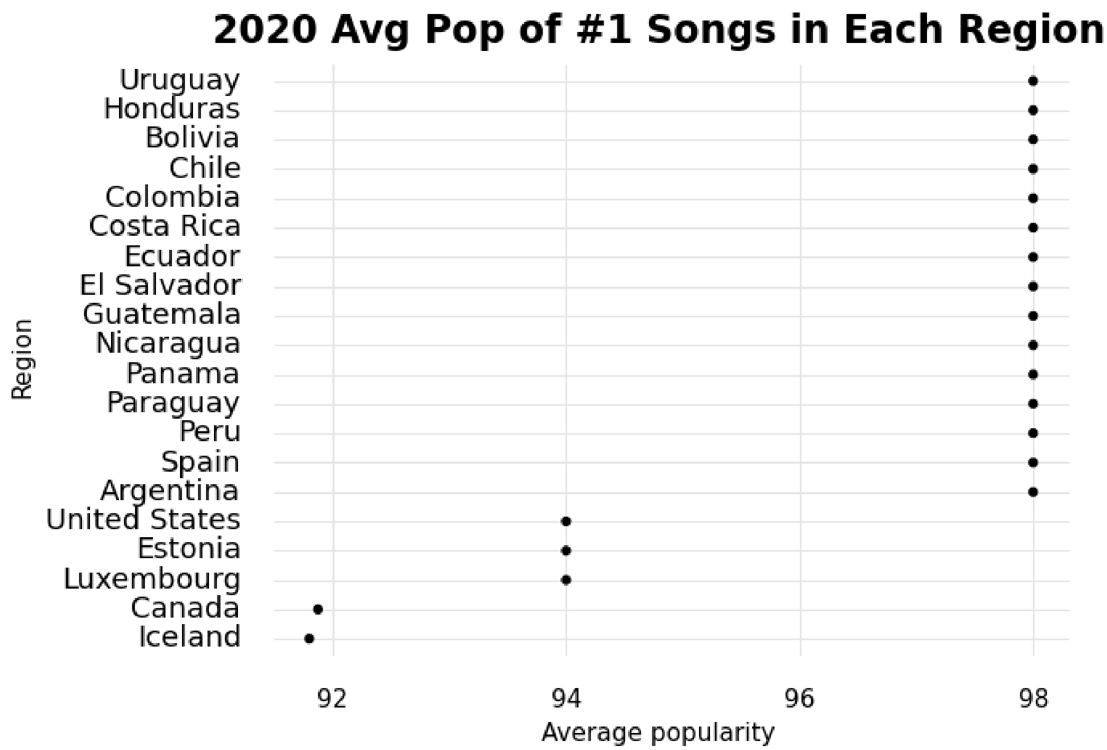
[107]:

	region	avg_pop
0	Uruguay	98.000
1	Colombia	98.000
2	Panama	98.000
3	Ecuador	98.000
4	Bolivia	98.000
5	Nicaragua	98.000
6	Peru	98.000
7	Honduras	98.000
8	Guatemala	98.000
9	Costa Rica	98.000
10	Argentina	98.000
11	El Salvador	98.000
12	Chile	98.000
13	Spain	98.000
14	Paraguay	98.000
15	Luxembourg	94.000
16	United States	94.000
17	Estonia	94.000
18	Canada	91.875
19	Iceland	91.800

[108]:

```
ggplot(top_countries_charting1, aes(x = 'reorder(region,avg_pop)', y =␣
↪'avg_pop')) + geom_point() + labs(x = "Region", y = "Average popularity",␣
↪title = "2020 Avg Pop of #1 Songs in Each Region") + theme_minimal()  +␣
↪theme(plot_title=element_text(face= "bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
```

```
    axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank()) +␣
↪coord_flip()
```



[108]: <ggplot: (8771390353393)>

3.5.11 There is a certain confluence around 98 for the highest average popularity regions for number 1 songs.

Which countries have the highest average popularity across their number #1 charting songs in 2020 in the Viral50 chart?

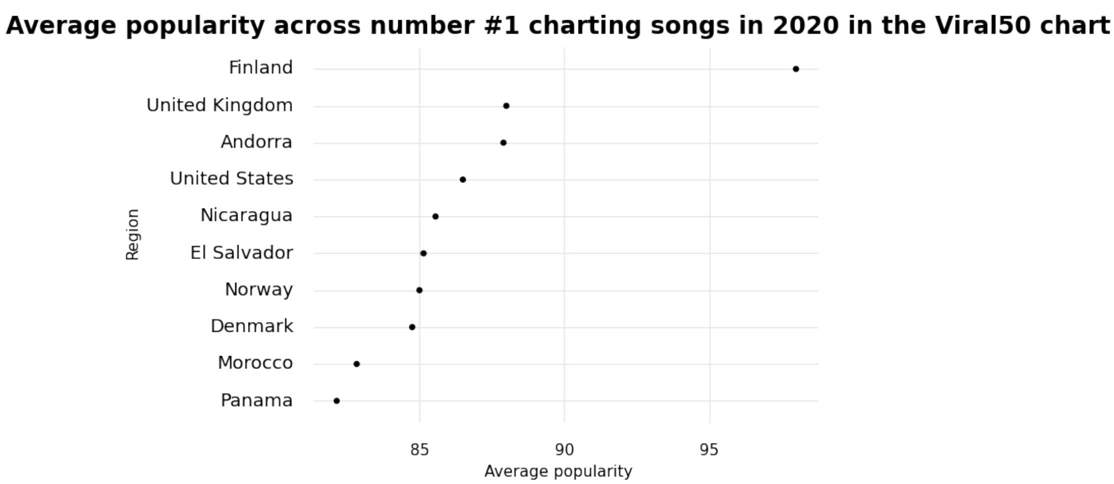
[109]:
```
%%bigquery top_countries_charting1_viral50 --project $project_id
WITH d AS(
SELECT region, title, artist
FROM `cs145-365221.spotify_database.charts` c
#FROM `cs145-project-1-365108.spotify_database.charts` c
WHERE EXTRACT(YEAR FROM c.date) = 2020 AND rank = 1 AND chart = 'viral50'
GROUP BY region, title, artist)

SELECT region, AVG(track_popularity) AS avg_pop # title, artist
FROM d
```

```
#JOIN `cs145-project-1-365108.spotify_database.processed_tracks` t
JOIN `cs145-365221.spotify_database.processed_tracks` t
ON t.track_name = d.title
GROUP BY region #, title, artist
ORDER BY avg_pop DESC
LIMIT 10
```

Query is running: 0%| |

Downloading: 0%| |

[111]:
```
ggplot(top_countries_charting1_viral50, aes(x = 'reorder(region,avg_pop)', y =␣
↪'avg_pop')) + geom_point() + labs(x = "Region", y = "Average popularity",␣
↪title = "Average popularity across number #1 charting songs in 2020 in the␣
↪Viral50 chart") + theme_minimal()  + theme(plot_title=element_text(face=␣
↪"bold", size=17, family = "Arial"),
      legend_position= 'none',
      legend_title = element_blank(),
      legend_text = element_blank(),
      axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
      axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank()) +␣
↪coord_flip()
```



[111]: <ggplot: (8771394402124)>

3.5.12 Finland has particularly high average popularity in their number 1 charting songs, suggesting that the region has had succesful viral hits.

Which countries have the highest average popularity across their TOP FIVE charting songs in 2020?

[112]:
```
%%bigquery top_countries_charting1to5 --project $project_id

WITH d AS(
SELECT region, title, artist
FROM `cs145-365221.spotify_database.charts` c
#FROM `cs145-project-1-365108.spotify_database.charts` c
WHERE EXTRACT(YEAR FROM c.date) = 2020 AND
      rank BETWEEN 1 AND 5
      AND chart = 'top200'
GROUP BY region, title, artist)

SELECT region, AVG(track_popularity) AS avg_pop # title, artist
FROM d
JOIN `cs145-365221.spotify_database.processed_tracks` t
#JOIN `cs145-project-1-365108.spotify_database.processed_tracks` t
ON t.track_name = d.title
GROUP BY region #, title, artist
ORDER BY avg_pop DESC
LIMIT 10
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

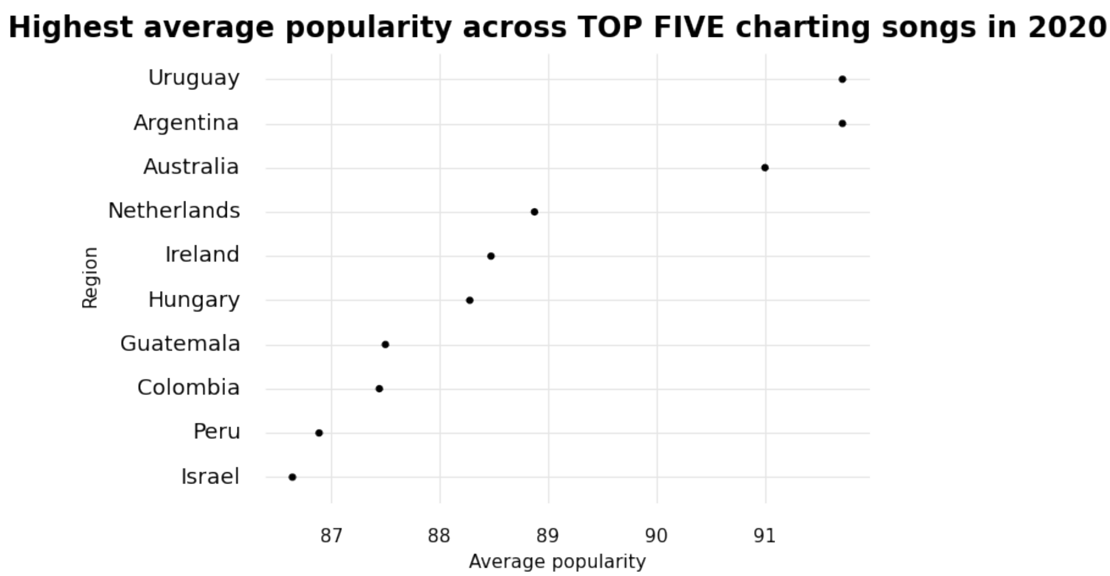
[113]:
```
top_countries_charting1to5
```

[113]:

	region	avg_pop
0	Argentina	91.714286
1	Uruguay	91.714286
2	Australia	91.000000
3	Netherlands	88.875000
4	Ireland	88.473684
5	Hungary	88.277778
6	Guatemala	87.500000
7	Colombia	87.444444
8	Peru	86.888889
9	Israel	86.642857

[115]:
```
ggplot(top_countries_charting1to5, aes(x = 'reorder(region,avg_pop)', y =␣
 ↪'avg_pop')) + geom_point() + labs(x = "Region", y = "Average popularity",␣
 ↪title = "Highest average popularity across TOP FIVE charting songs in 2020")␣
 ↪+ theme_minimal()  + theme(plot_title=element_text(face= "bold", size=17,␣
 ↪family = "Arial"),
```

```
    legend_position= 'none',
    legend_title = element_blank(),
    legend_text = element_blank(),
    axis_text_y = element_text(size=13 , family = "Arial", color = "black"),
    axis_text_x = element_text(size= 11,␣
↪family='Arial',color="black"),axis_title_y = element_text(size= 11,␣
↪family='Arial',color="black"),panel_grid_minor = element_blank()) +␣
↪coord_flip()
```



[115]: <ggplot: (8771394438470)>

3.5.13 Uruguay, Argentina and Australia are topping the top 200 charts and have the highest popularity among their respective top 5 songs.

3.5.14 Overall, we can see that 'loudness', 'valence', 'duration' and 'explicit' have the highest predictive power over popularity. Also, genres and subgenres seem to affect a song's popularity as well. Surprisingly, regional aggregate popularities does not favor the US, even though the most frequent language is english, but Latin American countries score well on aggregate popularity. Finally, we saw that language could also be a factor that influences popularity.

4 Data Prediction

4.1 Model 1 : We add all the 4 numeric variables that are the most correlated with popularity according to our correlation heatmap : duration,explicit,valence,loudness.

Training Data

[ ]:
```
# Create the model

model_dataset_name = 'spotify_model_linear_regression'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

[117]:
```
# We take 80% of our processed dataset, which corresponds to 5246 rows.
# Note that the data was already randomly shuffled

%%bigquery --project $project_id

CREATE OR REPLACE MODEL `spotify_database.spotify_model_linear_regression`
OPTIONS(model_type='linear_reg', input_label_cols=['track_popularity']) AS
SELECT loudness,duration_ms,valence,explicit,track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 < 5246
```

```
Query is running:   0%|          |
```

[117]:
```
Empty DataFrame
Columns: []
Index: []
```

[118]:
```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `spotify_database.spotify_model_linear_regression`)
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[118]:

	training_run	iteration	loss	eval_loss	learning_rate	duration_ms
0	0	2	53.585933	126.347206	0.4	2901
1	0	1	55.701027	126.453987	0.4	2750
2	0	0	83.355984	143.804054	0.2	2587

Evaluation data

[119]:
```
%%bigquery --project $project_id
SELECT
  *
FROM
```

```
  ML.EVALUATE(MODEL `spotify_database.spotify_model_linear_regression`, (
SELECT loudness,duration_ms,valence,explicit,track_popularity
FROM `cs145-365221.spotify_database.processed_tracks`

#FROM `cs145-project-1-365108.spotify_database.processed_tracks`

WHERE int64_field_0 > 5246 AND int64_field_0 < 5896))
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[119]:

```
   mean_absolute_error  mean_squared_error  mean_squared_log_error  \
0              9.32065           133.98352                0.041119

   median_absolute_error  r2_score  explained_variance
0               8.001451  0.132214            0.133458
```

4.1.1 These 4 variables have a certain predictive power, considering that the adjusted R-squared of this model is 0.13.

4.2 Model 2 : We add information more specific about lyrics. Does including popular words helps increase a song's popularity?

Training Data

[ ]:

```
# Create the model

model_dataset_name = 'spotify_model_linear_regression_words'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

[ ]:

```
Dataset(DatasetReference('cs145-project-1-365108',
'spotify_model_linear_regression_words'))
```

[120]:

```
# We take 80% of our processed dataset, which corresponds to 5246 rows.
# Note that the data was already randomly shuffled

%%bigquery --project $project_id

CREATE OR REPLACE MODEL `spotify_database.spotify_model_linear_regression_words`
OPTIONS(model_type='linear_reg', input_label_cols=['track_popularity']) AS
SELECT loudness,duration_ms,valence,explicit,track_popularity,baby, girl,␣
 ↪boy,home,love,money,foul,body,sex
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
```

```
WHERE int64_field_0 < 5246
```

Query is running: 0%| |

[120]: Empty DataFrame
Columns: []
Index: []

[121]:
```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `spotify_database.
  ↪spotify_model_linear_regression_words`)
```

Query is running: 0%| |

Downloading: 0%| |

[121]:

	training_run	iteration	loss	eval_loss	learning_rate	duration_ms
0	0	1	53.740950	121.831670	0.4	3167
1	0	0	81.365771	137.907932	0.2	2501

Evaluation DATA

[122]:
```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.EVALUATE(MODEL `spotify_database.spotify_model_linear_regression_words`, (
SELECT loudness,duration_ms,valence,explicit,track_popularity,baby, girl,␣
  ↪boy,home,love,money,foul,body,sex
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5246 AND int64_field_0 < 5896))
```

Query is running: 0%| |

Downloading: 0%| |

[122]:

	mean_absolute_error	mean_squared_error	mean_squared_log_error	\
0	9.328874	134.939135	0.041297	

	median_absolute_error	r2_score	explained_variance
0	8.066483	0.126024	0.128405

4.2.1 It seems like including these popular words does not help to explain a song's popularity, since the adjusted R-squared decreased.

4.3 Model 3 : We remove dummy variables and add language

Training Data

[ ]:
```
# Create the model

model_dataset_name = 'spotify_model_linear_regression_language'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

[ ]:
```
Dataset(DatasetReference('cs145-365221',
'spotify_model_linear_regression_language'))
```

[123]:
```
# We take 80% of our processed dataset, which corresponds to 5246 rows.
# Note that the data was already randomly shuffled

%%bigquery --project $project_id

CREATE OR REPLACE MODEL `spotify_database.
↪spotify_model_linear_regression_language`
OPTIONS(model_type='linear_reg', input_label_cols=['track_popularity']) AS
SELECT loudness,duration_ms,valence,explicit,track_popularity,language
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 < 5246
```

```
Query is running:   0%|          |
```

[123]:
```
Empty DataFrame
Columns: []
Index: []
```

[124]:
```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `spotify_database.
↪spotify_model_linear_regression_language`)
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[124]:
```
   training_run  iteration        loss  eval_loss  learning_rate  duration_ms
0             0          0  143.922968  145.59707            NaN         2747
```

Evaluation DATA

[125]:
```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.EVALUATE(MODEL `spotify_database.
 ↪spotify_model_linear_regression_language`, (
SELECT loudness,duration_ms,valence,explicit,track_popularity,language
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5246 AND int64_field_0 < 5896))
```

```
Query is running:   0%|          |
```

```
Downloading:   0%|          |
```

[125]:
```
   mean_absolute_error  mean_squared_error  mean_squared_log_error  \
0             9.289255           133.06795                0.040681

   median_absolute_error  r2_score  explained_variance
0                7.95389  0.138144            0.141256
```

4.3.1 Including language slightly improves the model, which means that this variable has a certain predictive capability, even though it is low.

4.4 Model 4 : Is a song popular only because of its characterics, or also because of the artist's name?

4.4.1 Here, we add the artist's name to the model.

Training Data

[ ]:
```
# Create the model

model_dataset_name = 'spotify_model_linear_regression_artist'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

[ ]:
```
Dataset(DatasetReference('cs145-project-1-365108',
'spotify_model_linear_regression_artist'))
```

[126]:
```
# We take 80% of our processed dataset, which corresponds to 5246 rows.
# Note that the data was already randomly shuffled
```

```
%%bigquery --project $project_id

CREATE OR REPLACE MODEL `spotify_database.
↪spotify_model_linear_regression_artist`
OPTIONS(model_type='linear_reg', input_label_cols=['track_popularity']) AS
SELECT␣
↪track_artist,loudness,duration_ms,valence,explicit,track_popularity,language
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 < 5246
```

```
Query is running:   0%|          |
```

[126]:

```
Empty DataFrame
Columns: []
Index: []
```

[127]:

```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `spotify_database.
↪spotify_model_linear_regression_artist`)
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[127]:

	training_run	iteration	loss	eval_loss	learning_rate	duration_ms
0	0	2	35.089206	111.869180	0.2	3183
1	0	1	47.496604	130.673800	0.2	3012
2	0	0	211.071112	232.567331	0.2	2494

Evaluation DATA

[128]:

```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.EVALUATE(MODEL `spotify_database.spotify_model_linear_regression_artist`, (
SELECT␣
↪track_artist,loudness,duration_ms,valence,explicit,track_popularity,language
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5246 AND int64_field_0 < 5896))
```

```
Query is running:   0%|          |
```

```
Downloading:   0%|          |
```

[128]:
```
   mean_absolute_error  mean_squared_error  mean_squared_log_error  \
0            10.879781          191.095408                0.061287

   median_absolute_error  r2_score  explained_variance
0               9.022353 -0.237689           -0.227528
```

4.4.2 Surprisingly, the artist's name decreases a lot the adjusted R-Squared. We would have expected the artist's name to be a great predictor of a song's popularity, but it actually worsened our model.

4.5 Model 5: Are genres and sub-genres associated with popularity?

4.5.1 We saw that genres and subgenres could have a certain predictive power, so we will add them to model 3 (without the artist's name).

Training Data

[ ]:
```
# Create the model

model_dataset_name = 'spotify_model_linear_regression_genre'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

[ ]:
```
Dataset(DatasetReference('cs145-365221',
'spotify_model_linear_regression_genre'))
```

[149]:
```
# We take 80% of our processed dataset, which corresponds to 5246 rows.
# Note that the data was already randomly shuffled

%%bigquery --project $project_id

CREATE OR REPLACE MODEL `spotify_database.spotify_model_linear_regression_genre`
OPTIONS(model_type='linear_reg', input_label_cols=['track_popularity']) AS
SELECT loudness,duration_ms,valence,explicit,track_popularity,language,␣
 ↪playlist_genre,playlist_subgenre
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 < 5246
```

```
Query is running:   0%|          |
```

[149]:
```
Empty DataFrame
Columns: []
Index: []
```

[150]:
```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `spotify_database.
↪spotify_model_linear_regression_genre`)
```

Query is running: 0%| |

Downloading: 0%| |

[150]:

	training_run	iteration	loss	eval_loss	learning_rate	duration_ms
0	0	1	60.385911	127.434462	0.4	2509
1	0	0	102.527777	147.212790	0.2	2359

Evaluation DATA

[151]:
```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.EVALUATE(MODEL `spotify_database.spotify_model_linear_regression_genre`, (
SELECT␣
↪loudness,duration_ms,valence,explicit,track_popularity,language,playlist_genre,playlist_sub
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5246 AND int64_field_0 < 5896))
```

Query is running: 0%| |

Downloading: 0%| |

[151]:

	mean_absolute_error	mean_squared_error	mean_squared_log_error	\
0	8.371443	120.153898	0.034349	

	median_absolute_error	r2_score	explained_variance
0	6.800798	0.221786	0.224338

Testing DATA

[153]:
```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.EVALUATE(MODEL `spotify_database.spotify_model_linear_regression_genre`, (
SELECT␣
↪loudness,duration_ms,valence,explicit,track_popularity,language,playlist_genre,playlist_sub
FROM `cs145-365221.spotify_database.processed_tracks`
```

```
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5896))
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[153]:

	mean_absolute_error	mean_squared_error	mean_squared_log_error
0	8.4087	112.735449	0.033542

	median_absolute_error	r2_score	explained_variance
0	7.121546	0.298476	0.304578

4.5.2 Adding genre and subgenre more than doubled the model's R-squared (from 0.13 to 0.29), which suggests that these 2 variables are indeed good predictors of popularity.

4.5.3 Finally, we make predictions on the test set with our final model (model 5).

Predictions on test set

[154]:

```
%%bigquery --project $project_id
SELECT
  *
FROM
  ML.PREDICT(MODEL `spotify_database.spotify_model_linear_regression_genre`, (
SELECT␣
 ↪loudness,duration_ms,valence,explicit,track_popularity,language,playlist_genre,playlist_sub
FROM `cs145-365221.spotify_database.processed_tracks`
#FROM `cs145-project-1-365108.spotify_database.processed_tracks`
WHERE int64_field_0 > 5896))
```

```
Query is running:   0%|          |

Downloading:   0%|          |
```

[154]:

	predicted_track_popularity	loudness	duration_ms	valence	explicit
0	57.610385	-7.108	272160	0.212	1
1	47.682009	-4.516	325467	0.697	1
2	62.114297	-6.823	295387	0.657	0
3	61.073604	-6.544	436880	0.351	0
4	61.084552	-4.755	204375	0.415	0
..	…	…	…	…	…
644	73.972776	-3.061	169733	0.396	0
645	74.040128	-4.511	203373	0.339	1
646	65.703629	-7.123	223880	0.239	0
647	57.999804	-11.209	279693	0.752	0
648	58.806872	-14.124	207307	0.517	0

```
     track_popularity language playlist_genre   playlist_subgenre
0                  64       en            rap    southern hip hop
1                  41       nl            rap    southern hip hop
2                  60       en            r&b  urban contemporary
3                  48       en            r&b  urban contemporary
4                  62       en            edm            big room
..                  …        …              …                   …
644                47       en            pop       post-teen pop
645                75       en            rap                trap
646                60       en            pop    indie poptimism
647                68       en           rock      permanent wave
648                39       en           rock      permanent wave

[649 rows x 9 columns]
```

5 Conclusion

5.0.1 Our hypothesis proved correct : the 4 attributes had a relatively strong predictive power, and language also slightly improved the model. Intuitively, we expected genres and subgenres to play an important role in a song's popularity, which proved to be true. However, we expected the artist's name to play a huge role in a song's popularity, but it happened to worsen our model considerably. This leads us to believe that just because one artist's hit increases their average popularity, it does not mean that their other songs will follow in that trend.

5.0.2 We think our results confirm normal intuitions but also show that artists should focus on aggregate features because cumulatively, the variables together constructed a stronger model than their isolated parts.

5.0.3 If we had more time, we would have studied the evolution of the popularity drivers over time. Moreover, we would have, analyzed the popularity drivers within genres and subgenres. This makes sense because for example, Indie pop excels in low energy music compared to punk rock, and an artist would be more interested in getting conclusions about their own genre. Finally, in the visualizations, we noticed clustering of a specific range of loudness, duration, and valence. It would have been interesting to focus on these clusters and construct a differences-in-differences model that treats the trends before and after these clusters as a control and treatment.