ISTA 322 Final Report

Data

Data Sources:

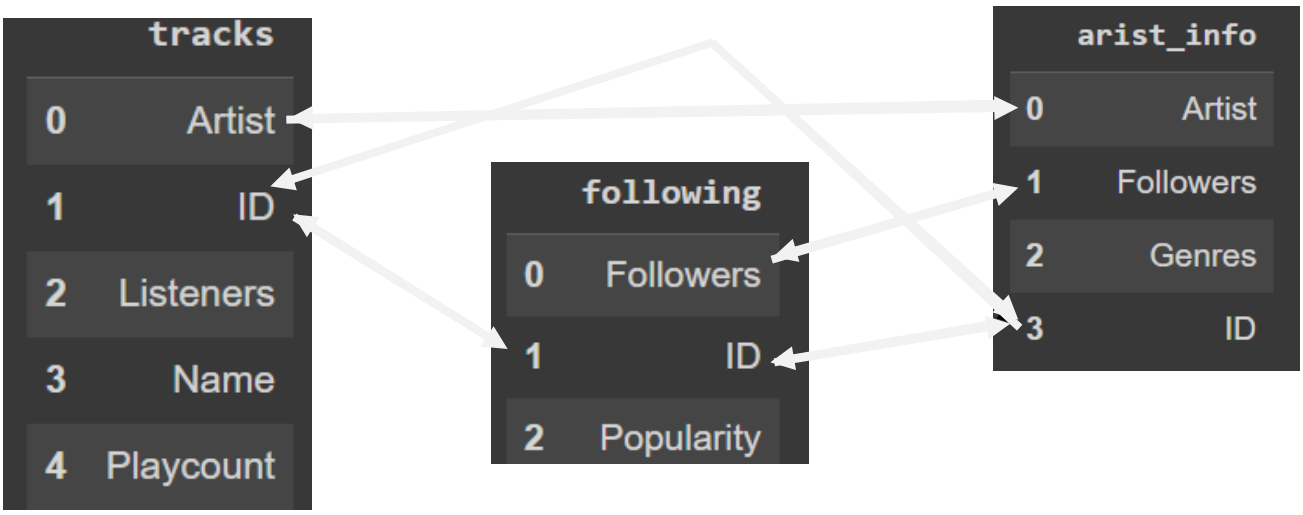
- https://developer.spotify.com/
- https://www.last.fm/api

Description:

From the Spotify dataset, the available data included an artist, an ID associated with them, their popularity, the followers on the platform, the genres of music for that artist as well as their top tracks. I selected all of these except for their top tracks to use in my data set. From the Last.fm dataset, the available data included the artist, an ID associated with them, their tracks names, the play count of those tracks, and listeners of the tracks. I selected everything from this collection except ID because I already had an ID. I made both of these data sets into panda data frames and then merged them on a common column, that being artist for both data sets. I then made this merged data set into a SQL table. From this table, I made sub tables based on the descriptors of the artist and what I thought was appropriate to make into a sub table. All of these tables are connected through the primary key of ID from the Spotify data, I just put it into every table, so they were all related. I honestly didn't run into many data cleaning issues, mostly because the data was relatively clean. I went through and did basic cleaning such as removing Nans and making the formatting between sets easily compatible with each other, however getting the data from the API's was by far the most difficult and time-consuming task for the project for me.

RDB Schema

The schema contains 4 SQL tables: rappers table, artist information table, following table, and tracks table. Here we can see the sub tables and their respective columns visualized. Here are the keys between them. I chose not to visualize the main one due to the large amount of arrows I would be drawing to each table.



As we can see every single sub table can relate all of their columns to the main rapper's table. The other tables can relate ID between every data table, my following table relates to my artist info table by the followers, and tracks and artist info relate to each other by artist.

Queries and Plots

I'm going to provide some queries and some data tables just to show what kind of relationships can be derived from the data set I created, the plots and queries are relatively self explanatory.

```
### First just wanna see what genres are most prevelant amoung the data set
genre_count = """SELECT Genres, COUNT(*) AS GenreCount
FROM artist_information_table
GROUP BY Genres
ORDER BY GenreCount DESC;"""
genre_count = run_query(genre_count)
print(genre_count)
```

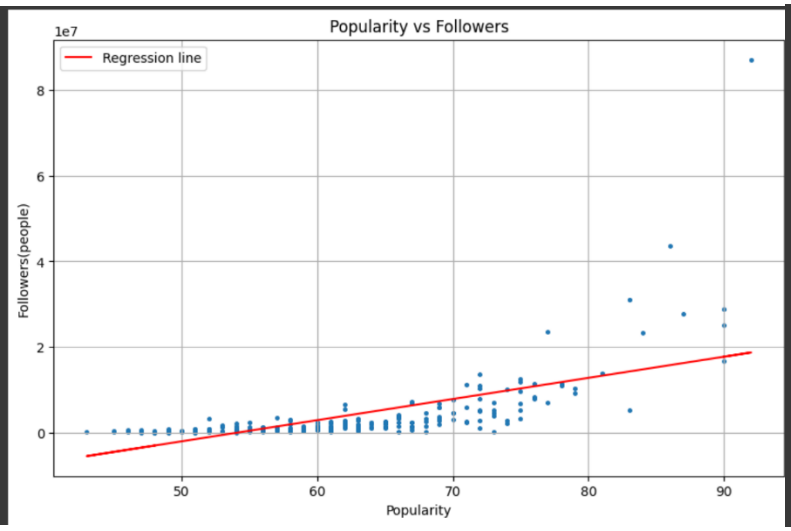
	Genres	GenreCount
0	indie pop rap	30
1	pop rap	25
2	deep underground hip hop, indie pop rap	25
3	hip hop, pop rap, rap, southern hip hop, trap	15
4	sad lo-fi, sad rap	15
..	...	...
202	battle rap, dirty south rap, southern hip hop,...	5
203	battle rap, hardcore hip hop, philly rap	5
204	crunk, dirty south rap, gangster rap, new orle...	5
205	cali rap	5
206	dirty south rap, east coast hip hop	5

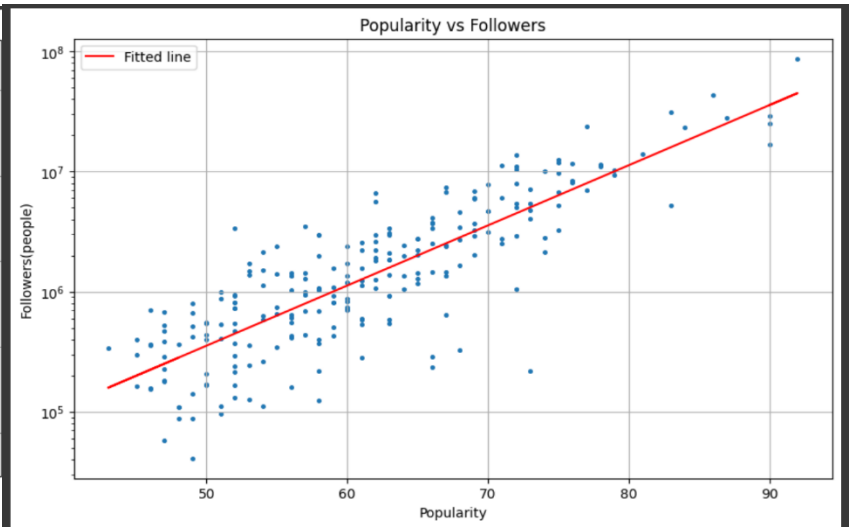
[207 rows x 2 columns]

```
query = """SELECT
    TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(genres, ',', n.digit+1), ',', -1)) AS Genre,
    SUM(Followers) AS SumFollowers
FROM
    artist_information_table
JOIN
    (SELECT 0 AS digit UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4) AS n
ON
    LENGTH(REPLACE(genres, ',' , '')) <= LENGTH(genres)-n.digit
GROUP BY
    Genre
ORDER BY
    SumFollowers DESC;

"""

### Execute the query
run_query(query)
```

	Genre	SumFollowers
0	rap	3.114363e+09
1	hip hop	2.559987e+09
2	pop rap	1.768181e+09
3	trap	7.690223e+08
4	southern hip hop	6.514304e+08
...	...	...
157	bossbeat	6.581450e+05
158	uk hip hop	6.239400e+05
159	uk drill	6.239400e+05
160	bounce	5.521450e+05
161	lo-fi rap	5.413450e+05

```
query = """
SELECT Popularity, Followers
FROM following_table;
"""

### Execute the query
data = run_query(query)
```






```
### Get playcount
query1 = """
SELECT Playcount
FROM tracks_table;
"""

### Execute the query
data1 = run_query(query1)

### Get playcount
query2 = """
SELECT Followers
FROM artist_information_table;
"""

### Execute the query
data2 = run_query(query2)
```