Project2_Group1
Sabrina Alves
Nicklaus Austin
Lyndah Mupfunya
Yvonne Martinez

Technical Report

Topic/Inspiration

Apple Music and Spotify are two heavy music streaming services. Users are able to listen to around 82 million songs on Spotify and over 90 million on Apple Music. Both platforms service millions of users around the world and produce charts that record the most popular songs. All of our team are users of either platform, so we decided to analyze "Top 100:USA" from Apple Music and "Daily Top Songs USA" to compare overall USA's top songs. All analysis will be based on March 20, 2022 data (Apple) and March 16, 2022 (Spotify).

Sources

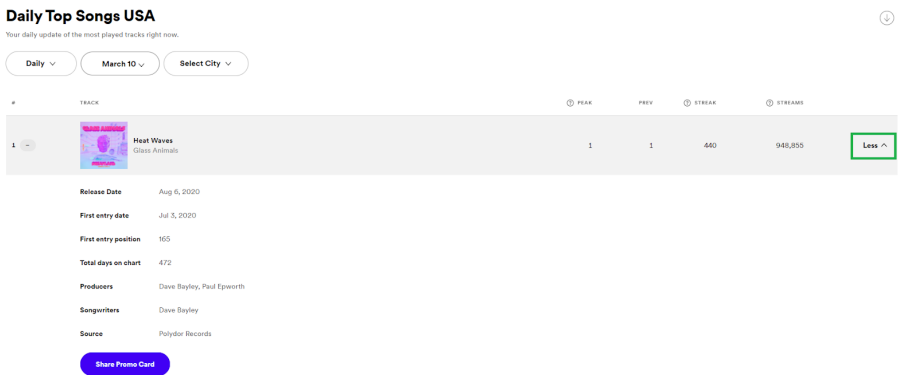
We decided to use entirely web scraping from the two links below to extract all our supporting data.
Apple:
https://music.apple.com/us/playlist/top-100-usa/pl.606afcbb70264d2eb2b51d8dbcfa6a12
Spotify:
https://charts.spotify.com/charts/view/regional-us-daily/2022-03-20

Extract

Our first step was web scraping the song list from Apple Music's "Top 100:USA". The website includes rank, song title, artist, album, and song length. We used a Python library called "BeautifulSoup" and a splinter to extract information of each song inorder to organize it into dataframes in Pandas. Once we created the data frame within our jupyter notebook we created a csv.

The same process was followed for Spotify's "Daily Top Songs USA " to extract rank, song title, artist, peak, prev, streak, and streams. There were some hurdles we had to overcome in Spotify's web scraping. Peak, prev, and streak were all elements under the same class ID . Our output continued to be only the first element (peak). After several attempts, the solution was to call out each individual element out of the list (ex: [0],[1],[2]).

An additional hurdle in web scraping Spotify was finding the drop down to extract the remaining song information. The drop down only appears with a mouse hover. We attempted using action chains with a selenium driver, but were not successful. This would be a task for in the future to solve.



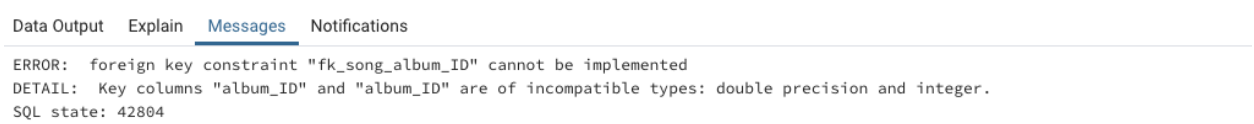
Transform

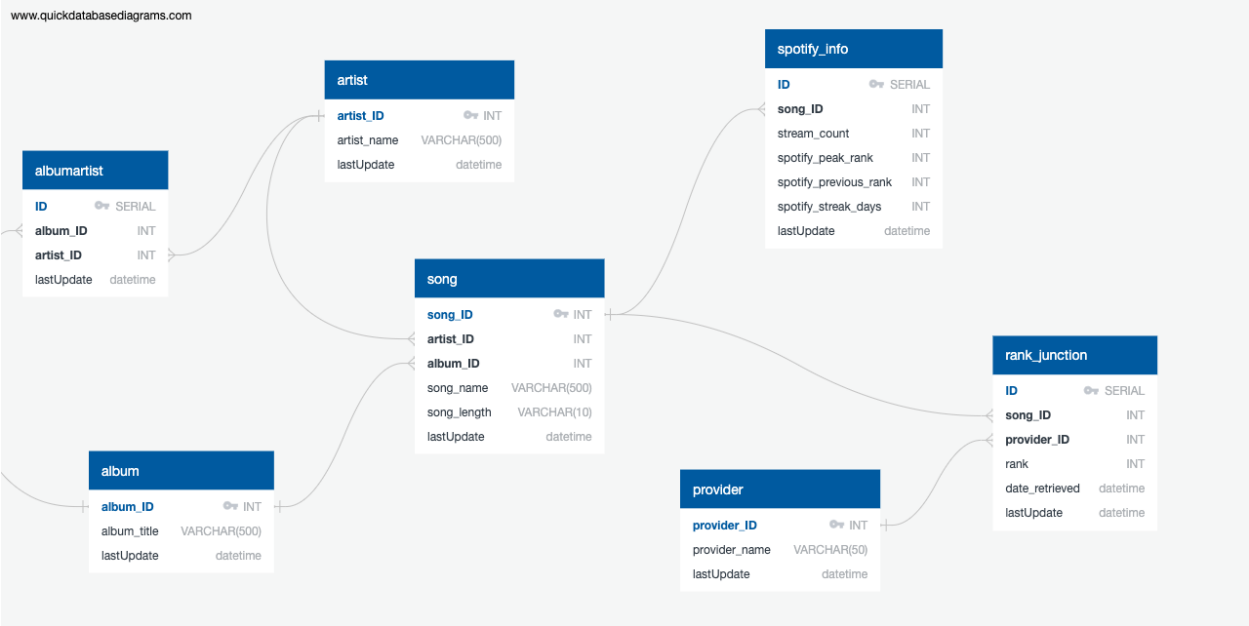
After completing the extracting and creating csv from both sources, we joined both tables by using an outer join to make our main table in our transformation notebook**.** Because we received slightly different information from each site, with our merge we made sure to appropriately rename our columns. For example Apple had different ratings than Spotify so we made sure to include both but name the columns differently ex: spotify_rank & apple_rank. All of our columns were unique so we did not separate original columns. Once our main table was finalized, we broke it into smaller tables: artist, provider, album, song, rank_junction, and spotify_info.

Load

The data web scrapped was cleaned and transformed and uploaded to PostgreSQL. We used https://app.quickdatabasediagrams.com/#/ to establish our foreign and primary keys. One thing to point out is: Our group did go a different route by already creating our own IDs while creating our tables in pandas for artist_ID, album _ID, song_ID, provider_ID.

We did have a couple issues when importing our data tables into PostgreSQL. We noticed our IDs were different data types after receiving an error and had to go back and fix our columns to match.





Queries:

After successfully loading our tables into PostgreSQL, we ran 5 queries to analyze and make conclusions from Apple and Spotify top charts. We had to work around an error of using capital letters in our column names. You will notice the use of quotes around certain columns.

First query:

```
query = """
        SELECT
            song.song_name,
            artist."artist_name"
        FROM
            song
        JOIN artist
            on artist."artist_ID" = song."artist_ID";
            """

conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	song_name	artist_name
0	'Til You Can't	Cody Johnson
1	'Till I Collapse	Eminem, Nate Dogg
2	20 Min	Lil Uzi Vert
3	2055	Sleepy Hallow
4	505	Arctic Monkeys

This first query is joining our song and artist table using the column artist_ID.

Second query:

```
query = """
        SELECT
            song."song_name",
            spotify_info."stream_count"
        FROM
            song
        JOIN spotify_info
            on spotify_info."song_ID" = song."song_ID"
        ORDER BY
            spotify_info."stream_count" desc;
                """


conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	song_name	stream_count
0	Heat Waves	934726
1	Super Gremlin	733927
2	We Don't Talk About Bruno	726391
3	Something In The Way	723210
4	STAY (with Justin Bieber)	679258

This second query is joining our song and spotify_info table using the column song_ID. We additionally ordered the result by largest count at the top.

Third query:

```
query = """
    SELECT
        song.song_name,
        spotify_info.stream_count as "Streams on Spotify",
        rank_junction.rank,
        spotify_info.spotify_streak_days as "Days on List"
    FROM
        song
    JOIN spotify_info
        on spotify_info."song_ID" = song."song_ID"
    JOIN rank_junction
        on rank_junction."song_ID" = song."song_ID"
    WHERE
        rank_junction."provider_ID"= 1
    ORDER BY
        spotify_info."stream_count" desc;
                """


conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	song_name	Streams on Spotify	rank	Days on List
0	Heat Waves	934726	1.0	446
1	Super Gremlin	733927	2.0	108
2	We Don't Talk About Bruno	726391	3.0	81
3	Something In The Way	723210	4.0	11
4	STAY (with Justin Bieber)	679258	5.0	251

This third query is reviewing rank (for spotify) and days on (spotify) list relative to stream count for each of the songs on the top 200 for Spotify.

Fourth query:

```
query = """
        SELECT
            artist.artist_name,
            count(song."artist_ID") as "Count of Artist"
        FROM
            artist
        JOIN song
            on artist."artist_ID" = song."artist_ID"
        GROUP BY
            artist.artist_name
        ORDER BY
             "Count of Artist" desc;
                """


conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	artist_name	Count of Artist
0	Doja Cat	5
1	Olivia Rodrigo	5
2	Kanye West	4
3	Morgan Wallen	4
4	J. Cole	4

This fourth query is counting the amount of times an artist appears in the top charts. We do want to note that if two artists were on a single song, our query will count it as a unique artist ID and name.

Fifth query:

```
query = """
    SELECT
        song.song_name,
        artist.artist_name,
        rank_junction.rank,
        provider.provider_name
    FROM
        artist
    JOIN song
        on artist."artist_ID" = song."artist_ID"
    JOIN rank_junction
        on rank_junction."song_ID" = song."song_ID"
    JOIN provider
        on rank_junction."provider_ID" = provider."provider_ID"
    WHERE
        rank_junction."provider_ID"= 1
    ORDER BY
        rank_junction.rank asc;
                """


conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	song_name	artist_name	rank	provider_name
0	Heat Waves	Glass Animals	1.0	spotify
1	Super Gremlin	Kodak Black	2.0	spotify
2	We Don't Talk About Bruno	Carolina Gaitán - La Gaita, Mauro Castillo, Ad...	3.0	spotify
3	Something In The Way	Nirvana	4.0	spotify
4	STAY (with Justin Bieber)	The Kid LAROI, Justin Bieber	5.0	spotify

```
query = """
    SELECT
        song.song_name,
        artist.artist_name,
        rank_junction.rank,
        provider.provider_name
    FROM
        artist
    JOIN song
        on artist."artist_ID" = song."artist_ID"
    JOIN rank_junction
        on rank_junction."song_ID" = song."song_ID"
    JOIN provider
        on rank_junction."provider_ID" = provider."provider_ID"
    WHERE
        rank_junction."provider_ID"= 2
    ORDER BY
        rank_junction.rank asc;
                """


conn = engine.connect()
df = pd.read_sql(query, conn)
df.head()
```

	song_name	artist_name	rank	provider_name
0	What Happened To Virgil (feat. Gunna)	Lil Durk, Gunna	1.0	apple
1	AHHH HA	Lil Durk	2.0	apple
2	Super Gremlin	Kodak Black	3.0	apple
3	Petty Too (feat. Future)	Lil Durk, Future	5.0	apple
4	pushin P (feat. Young Thug)	Gunna, Future, Young Thug	7.0	apple

These queries are recreating the rank list that we found on Apple Music and Spotify Charts.

Future work/Limitations/Conclusions

Due to the time constraint, in the future we would have wanted to bring additional data from Billboard. Billboard is an entertainment magazine that provides music charts similar to Spotify and Apple Music. This would have truly helped solidify USA's top tracks, as well as include a non music streaming service as a data point. Our current analysis was only in the USA , but without a time constraint, we would have additionally wanted to expand our analysis globally. We know this is possible due to both data sources having the capability of selecting global or specific countries. Additionally, because we had issues with our hover button scraping on spotify, this would be another point of future work. The hover button would have added additional info such as release date, first entry date, first entry position, total days on chart, producers, songwriters, and source.

References

https://newsroom.spotify.com/company-info/#:~:text=Discover%2C%20manage%20and%20share%20over,ad%2Dfree%20music%20listening%20experience
https://www.apple.com/apple-music/
http://allselenium.info/mouse-over-actions-using-python-selenium-webdriver/