Music Genre Analysis

Connor Casey
Luke Milton
Matthew Matti
Lynn Soors
Dylan Mavencamp

Project 3

- Change in popularity of music genres across 2015 to 2024
- Spotify track streams
- # of total streams a genre has
- Based on the total amount of stream a song has got in its Spotify history
- Divided based on the songs release year

Introduction

link

- Number of times a song has been played on spotify
- Song must play for at least 30 seconds
- Will not count if <30 seconds
- Complex algorithm to count streams
- Tracks and removes “fraudulent streams”

What counts as a stream on Spotify?

-Process of finding the dataset

Spotify API

Kaggle Top 10000 Spotify Songs from 1950-Now

-Spotify API Problems

Could not figure out how to use Spotify's API

-Limitations with the dataset that we noticed right away

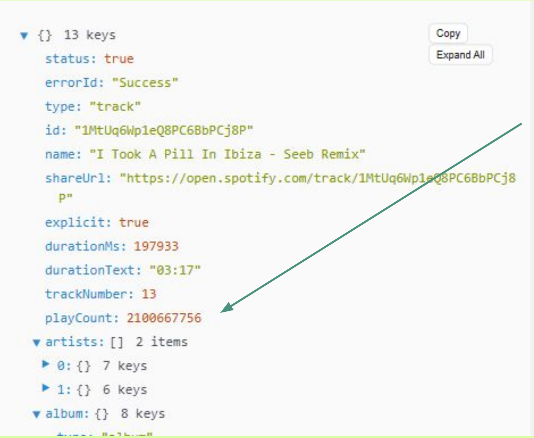
- No stream counts
- Vague popularity score
- Lot of genres
- Harder to clean

Our First Dataset

Used regex to extract the Track URI to be used in a loop to get stream counts

```
import re

df['Track_URI'] = df['Track URI'].str.extract(r'(?:spotify:track:)(\s*\w{22,})')

df
```

	Track URI	Track Name	Artist URI(s)	Artist Name(s)	Album URI	Album Artist Name(s)	Album Release Date	Track Duration (ms)	Track Preview URL	Explicit	Popularity	Artist Genres	Track_URI
0	spotify:track:0vNPJrUrBnMFdCs8b2MTNG	Fader	spotify:artist:4W48hZAnAHVOC2c8WH8pcq	The Temper Trap	spotify:album:0V59MMtgoruvEqMv18KAOH	The Temper Trap	2009	192373	https://p.scdn.co/mp3-preview/14264bd1501d2723...	False	0	indietronica,modern rock,shimmer pop	0vNPJrUrBnMFdCs8b2MTNG
1	spotify:track:0NpvdCO506uO58D4AbKzki	Sherry	spotify:artist:6mcrZQmgzFGRWf7C0SObou	Frankie Valli & The Four Seasons	spotify:album:0NUEQILaBzavnzcMEs4buZ	Frankie Valli & The Four Seasons	1/14/2003	152160	https://p.scdn.co/mp3-preview/e3f765262ebc349e...	False	54	adult standards,bubblegum pop,doo-wop,lounge,n...	0NpvdCO506uO58D4AbKzki
2	spotify:track:1MtUq6Wp1eQ8PC6BbPCj8P	I Took A Pill In Ibiza - Seeb Remix	spotify:artist:2KsP6tYLJITBvSUxnwlVWa, spotify...	Mike Posner, Seeb	spotify:album:1Tz3Ai1guEFf4hV3d9i17K	Mike Posner	5/6/2016	197933	https://p.scdn.co/mp3-preview/7bae6aac6d699135...	True	63	dance pop,pop,pop dance,pop rap,pop dance	1MtUq6Wp1eQ8PC6BbPCj8P
3	spotify:track:59Iq75uFIqzUZcgZ4CbqFG	Let Go for Tonight	spotify:artist:7qRII6DYV06u2VuRPAVqug	Foxes	spotify:album:5AQ7uKRSpAv7SNUI4j24ru	Foxes	5/12/2014	238413	https://p.scdn.co/mp3-preview/84a003d72f9f1468...	False	39	electropop,metropopolis,uk pop	59Iq75uFIqzUZcgZ4CbqFG
4	spotify:track:7KdcZQ3GJeGdserhK61kfv	The Way I Want To Touch You	spotify:artist:7BEfMxbaqx6dOpbtIEqScm	Captain & Tennille	spotify:album:3GUxesVyOehlnaxJyCTh6d	Captain & Tennille	1/1/1975	163586	https://p.scdn.co/mp3-preview/9e7a4a7b7dc56dc3...	False	35	mellow gold,soft rock,yacht rock	7KdcZQ3GJeGdserhK61kfv



```
▼ {} 13 keys
    status: true
    errorId: "Success"
    type: "track"
    id: "1MtUq6Wp1eQ8PC6BbPCj8P"
    name: "I Took A Pill In Ibiza - Seeb Remix"
    shareUrl: "https://open.spotify.com/track/1MtUq6Wp1eQ8PC6BbPCj8
      P"
    explicit: true
    durationMs: 197933
    durationText: "03:17"
    trackNumber: 13
    playCount: 2100667756
  ▼ artists: [] 2 items
    ▶ 0: {} 7 keys
    ▶ 1: {} 6 keys
  ▼ album: {} 8 keys
```

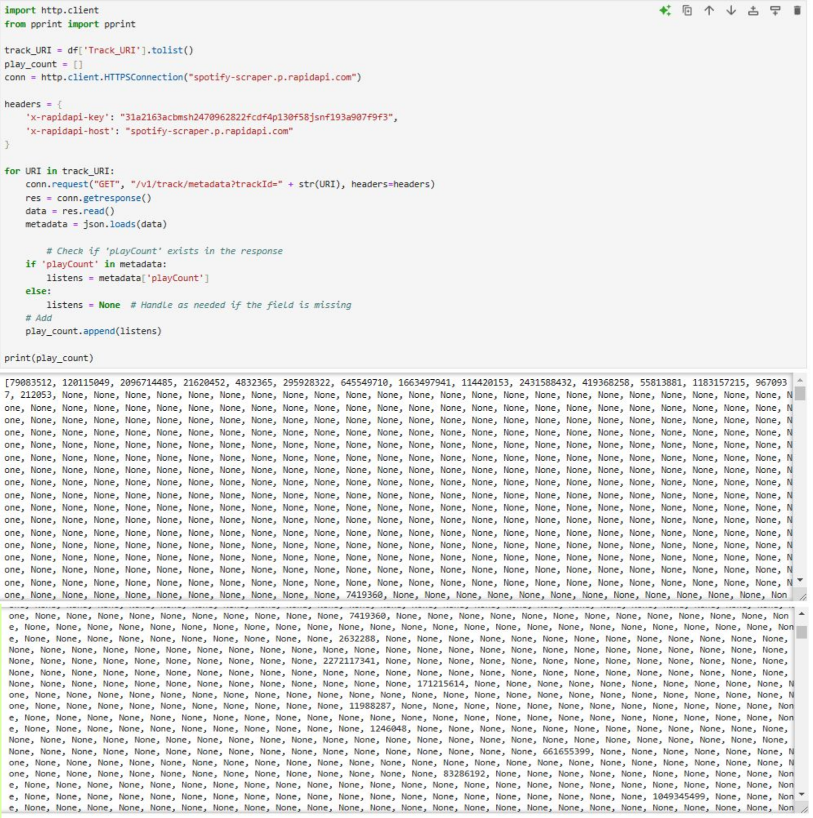
Found a Spotify Scraper from RapidAPI to use in order to create a loop that will find stream counts of each song.

The first 15 calls worked perfectly and then it was sporadic. Only 63 out of the 10,000 songs were able to be called.

Ran out of free API Calls.

```
import http.client
from pprint import pprint

conn = http.client.HTTPSConnection("spotify-scraper.p.rapidapi.com")

headers = {
    'x-rapidapi-key': "31a2163acbmsh2470962822fcdf4p130f58jsnf193a907f9f3",
    'x-rapidapi-host': "spotify-scraper.p.rapidapi.com"
}



conn.request("GET", "/v1/track/metadata?trackId=197s0DnYmVVQkkpBP6Vjb2", headers=headers) #Used a track ID from the csv to test

res = conn.getresponse()
data = res.read()

pprint(data.decode("utf-8"))
```

```
('{"message":"You have exceeded the MONTHLY quota for Requests on your current '
 'plan, BASIC. Upgrade your plan at '
 'https:\\/\\/rapidapi.com\\/DataFanatic\\/api\\/spotify-scraper"}')
```



```
import http.client
from pprint import pprint

track_URI = df['Track_URI'].tolist()
play_count = []
conn = http.client.HTTPSConnection("spotify-scraper.p.rapidapi.com")

headers = {
    'x-rapidapi-key': "31a2163acbmsh2470962822fcdf4p130f58jsnf193a907f9f3",
    'x-rapidapi-host': "spotify-scraper.p.rapidapi.com"
}

for URI in track_URI:
    conn.request("GET", "/v1/track/metadata?trackId=" + str(URI), headers=headers)
    res = conn.getresponse()
    data = res.read()
    metadata = json.loads(data)

        # Check if 'playCount' exists in the response
    if 'playCount' in metadata:
        listens = metadata['playCount']
    else:
        listens = None  # Handle as needed if the field is missing
    # Add
    play_count.append(listens)

print(play_count)
```

```
[79083512, 120115049, 2096714485, 21620452, 4832365, 295928322, 645549710, 1663497941, 114420153, 2431588432, 419368258, 55813881, 1183157215, 967093
7, 212053, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, 7419360, None, None, None, None, None, None, None, None, None, None, None, None, Non
```

```
one, None, None, None, None, None, None, None, None, None, None, 7419360, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, 2632288, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, 2272117341, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, 171215614, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, 11988287, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, 1246048, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, 661655399, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None, None, 83286192, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, 1049345499, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
```

While looking for data to use. Soundcharts was one that we came across.

-Pros and cons of using Soundcharts

Pros: A very very large amount and wide ranging data, and very helpful support.

Cons: It is a paid service

After reaching out to Soundcharts explaining what our group was doing, we were given access for 5 days to the site.

Songs

Sort by: Total | Growth | % Growth

Growth period: 7D | 28D | 90D

Artist Country : All | Song Genres : All | Streams : 1K - 4.7B | + Add filters

60,700,696 songs - Updated today - 03:20 am

#	SONG	STREAMS	POPULARITY	VIEWS	VIDEOS	PLAYS	COUNTS	PLAYS	STREAMS	PLAYS	FAVORITES
1	Die With A Smile Pop Bruno Mars, Lady Gaga	1.73B +0 0%	100 +0 0%	720.55M +417.9M ↑138.11%	2.4M +700K ↑41.18%	318.35K +135.8K ↑74.36%	9.23M +0 0%		611.04K +310K ↑102.95%		4.98K +4.5K ↑996.48%
2	DtMF Latin Bad Bunny	284.6M +0 0%	98 +0 0%			6.45K +0 0%	1.23M +0 0%	1.44M +0 0%			
3	BIRDS OF A FEATHER Alternative Billie Eilish	1.98B +581.6M ↑41.74%	96 -1 ↓-1.03%	351.53M +172.2M ↑46.85%	1.4M +200K ↑16.67%	551.06K +245.4K ↑80.28%	7.38M +2.3M ↑46.69%	8.9M +2.3M ↑35.25%	369.78K +127.1K ↑58.92%		657 +402 ↑157.65%
4	That's So True Pop Gracie Abrams	606.7M +551.6M ↑1K%	96 +9 ↑10.34%	10.52M +9.2M ↑701.58%	541.6K +478.2K ↑728.13%		2.4M +2.3M ↑1.85K%		571.87K +0 0%		
5	APT. Pop, Others Bruno Mars, ROSÉ	1.06B +0 0%	95 -1 ↓-1.04%	1.17B +874.4M ↑300.08%	4.54M +2.8M ↑164.2%	991 +788 ↑388.18%	6.64M +5.9M ↑828.47%		12.81M +11.6M ↑993.82%		5.23K +5K ↑1.76K%
6	BAILE INoLVIDABLE Latin Bad Bunny	189.83M +0 0%	95 +0 0%			2.27K +0 0%	423.47K +0 0%	873.76K +0 0%			
7	NUEVAYoL Latin Bad Bunny	175.4M +0 0%	94 +0 0%			4.39K +0 0%	538.14K +0 0%	789.12K +0 0%			
8	Sailor Song Pop, Rock Gigi Perez	664.27M +392.3M ↑144.24%	93 +1 ↑1.09%	7.27M +4.9M ↑206.27%	531.4K +162.7K ↑44.13%	59.49K +53.6K ↑909.09%	1.91M +1.3M ↑189.01%				
9	Timeless R&B Playboi Carti, The Weeknd	513.98M +0 0%	93 +1 ↑1.09%	15.64M +9.2M ↑141.01%	80.49K +37.6K ↑87.53%	1.22K +436 ↑55.9%	2.84M +2.2M ↑355.57%	7.65M +4.7M ↑157.34%	80.81K +34K ↑72.48%		607 +518 ↑582.02%
10	VOY A LLeVARTE PA PR Latin Bad Bunny	145.19M +0 0%	93 +0 0%			995 +0 0%	283.58K +0 0%	679.26K +0 0%			

Switching to Soundcharts

```
['R&B' 'Pop, Folk' 'Alternative, Rock' ... 'Rock, Country, Spoken, Pop'
 'Electro, Soundtrack, Classical, Rock' 'Pop, Country, Alternative']
```

```
# First, split the 'Song genres' column by commas and remove leading/trailing spaces
df_cleaned['Song genres'] = df_cleaned['Song genres'].str.split(',')

# Now explode the 'Song genres' column to create a new row for each genre
df_cleaned = df_cleaned.explode('Song genres')

# Remove any extra whitespace from the 'Song genres' column
df_cleaned['Song genres'] = df_cleaned['Song genres'].str.strip()

# Check the unique genres after splitting
print(df_cleaned['Song genres'].unique())
```

```
['R&B' 'Pop' 'Folk' 'Alternative' 'Rock' 'Electro' 'Hip Hop' 'Soundtrack'
 'Metal' 'Country' 'Latin' 'Blues' 'Others' 'Reggae' 'African' 'Jazz'
 'Kids' 'Spirituals' 'Classical' 'Sports' 'Asian' 'Holiday' 'Instrumental'
 'Spoken' 'European']
```

```
# This was unique given the data in Release date. Just doing 1 .fillna would not work. We had to first fill NaN values with a string version of 1776 (Just

# Fill NaN values in 'Release date' with 1776 and convert to string
df_cleaned['Year Released'] = df_cleaned['Release date'].fillna('1776').astype(str)

# Extract the year from 'Year Released' using regex
df_cleaned['Year Released'] = df_cleaned['Year Released'].str.extract(r'(\d{4})')

# Fill any remaining NaN values in 'Year Released' with 1776
df_cleaned['Year Released'] = df_cleaned['Year Released'].fillna(1776)

# Convert the 'Year Released' column to integers so further mathamatical processes can be done
df_cleaned['Year Released'] = df_cleaned['Year Released'].astype(int)
```

-Taking Matts code and manually cleaning the CSV

-Some dates had errors

-Wanted to include anything that may be of use

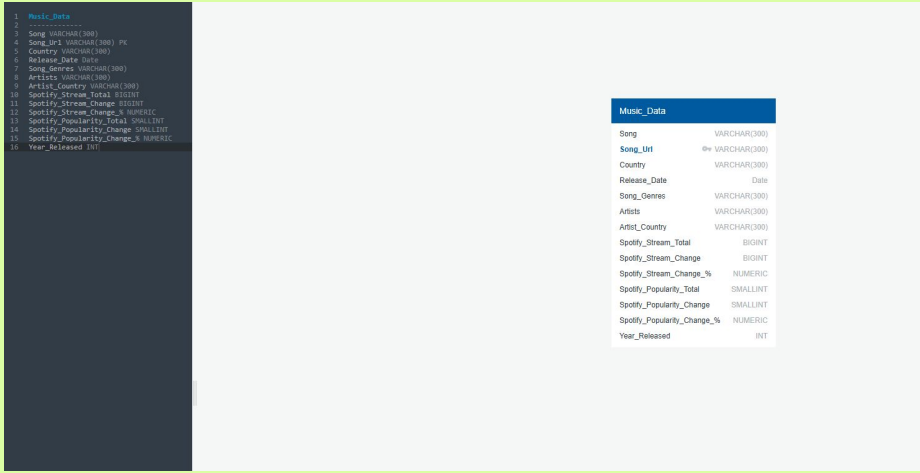
Basic CSV Cleaning

-Created a QuickDBD for the Excel file

-Using Postgres created a database

-Tested my database with SELECT queries

Setting up an SQL Database



Process of creating interactive visualizations

- Read clean csv into Pandas file
- Loop through clean csv in Pandas to create data frame with year and combined streams for genre within that year
- Convert to json file
- Write html and .js files to creative interactive visualizations run via live server
- Made graphs using Plotly

Select Year

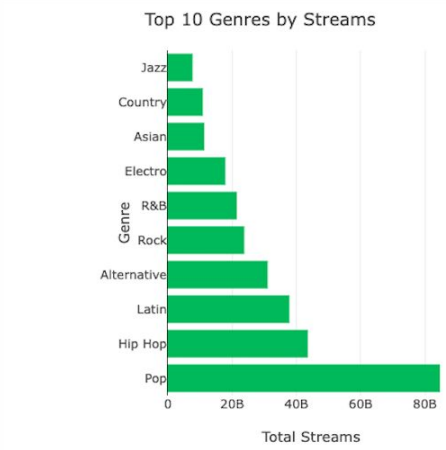
2024

Statistics

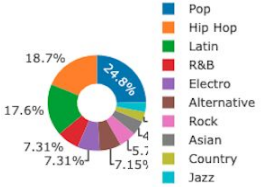
Total Songs: 1,026

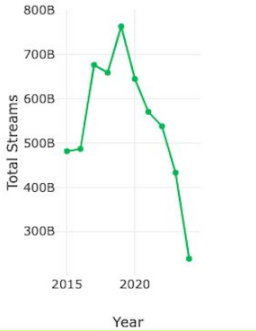
Total Streams: 238,699,508,210

Unique Genres: 101



interactive link





```python
# Connect to SQLite3 database
conn = sqlite3.connect('songs.db')

start_year = 2015
end_year = 2024
# SQL query for "Rock" genre
query_rock = f"""
SELECT "Song", "Spotify stream Total", "Year Released"
  FROM songs_data
  WHERE "Year Released" BETWEEN {start_year} AND {end_year} AND "Song Genres" LIKE "%Rock%"
  GROUP BY "Year Released"
  ORDER BY "Spotify stream Total" DESC;
"""

# SQL query for "Pop" genre
query_pop = f"""
SELECT "Song", "Spotify stream Total", "Year Released"
  FROM songs_data
  WHERE "Year Released" BETWEEN {start_year} AND {end_year} AND "Song genres" LIKE "%Pop%"
  GROUP BY "Year Released"
  ORDER BY "Spotify stream Total" DESC;
"""

# SQL query for "Hip Hop" genre
query_hip_hop = f"""
SELECT "Song", "Spotify stream Total", "Year Released"
  FROM songs_data
  WHERE "Year Released" BETWEEN {start_year} AND {end_year} AND  "Song Genres" LIKE "%Hip Hop%"
  GROUP BY "Year Released"
  ORDER BY "Spotify stream Total" DESC;
"""

# SQL query for "Alternative" genre
query_alt = f"""
SELECT "Song", "Spotify stream Total", "Year Released"
  FROM songs_data
  WHERE "Year Released" BETWEEN {start_year} AND {end_year} AND "Song Genres" LIKE "%Alternative%"
  GROUP BY "Year Released"
  ORDER BY "Spotify stream Total" DESC;
"""

# SQL query for "Country" genre
query_country = f"""
SELECT "Song", "Spotify stream Total", "Year Released"
  FROM songs_data
  WHERE "Year Released" BETWEEN {start_year} AND {end_year} AND "Song Genres" LIKE "%Country%"
  GROUP BY "Year Released"
```

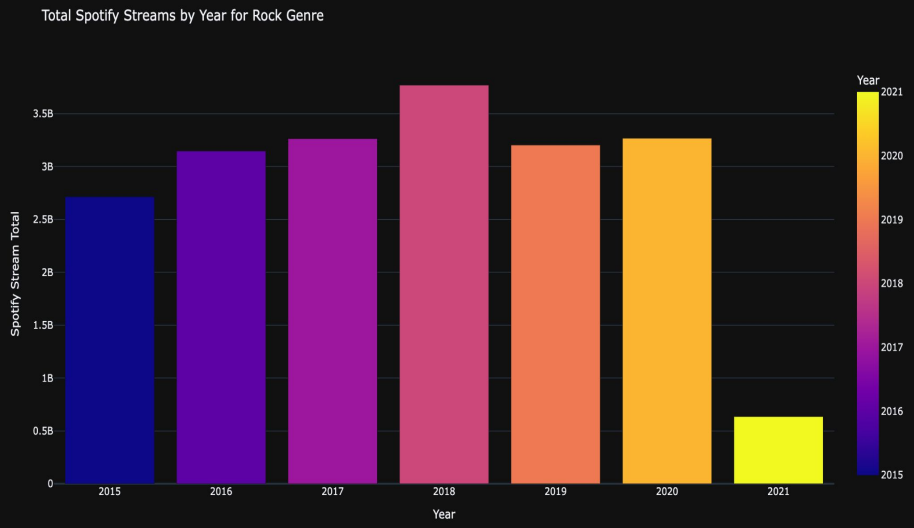
Used SQLiteStudio to test queries on the songs.db

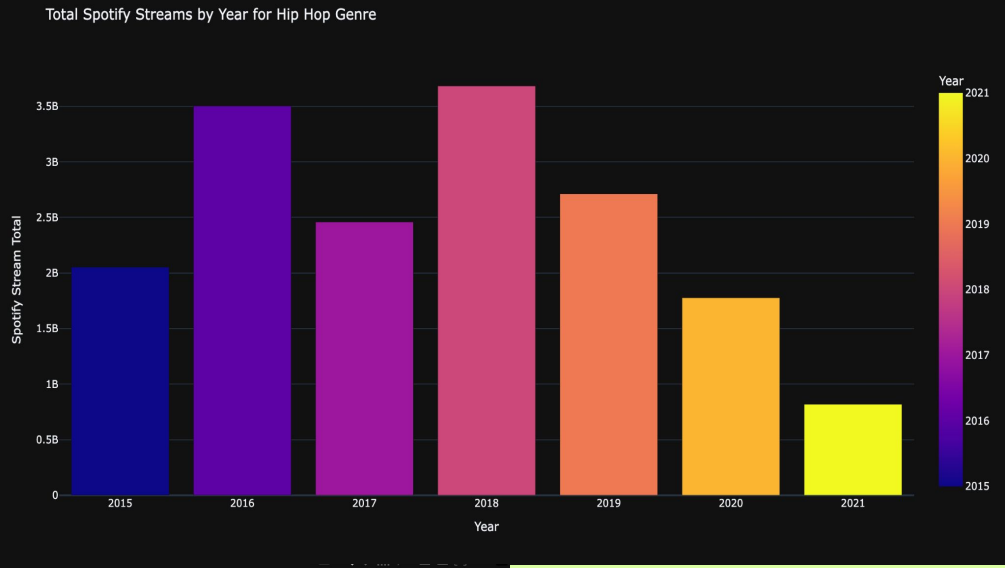
```
# Read the data into pandas DataFrames
df_query1 = pd.read_sql_query(query_rock, conn)  # Rock data
df_query2 = pd.read_sql_query(query_pop, conn)   # Pop data
df_query3 = pd.read_sql_query(query_hip_hop, conn)   # Hip Hop
data
```

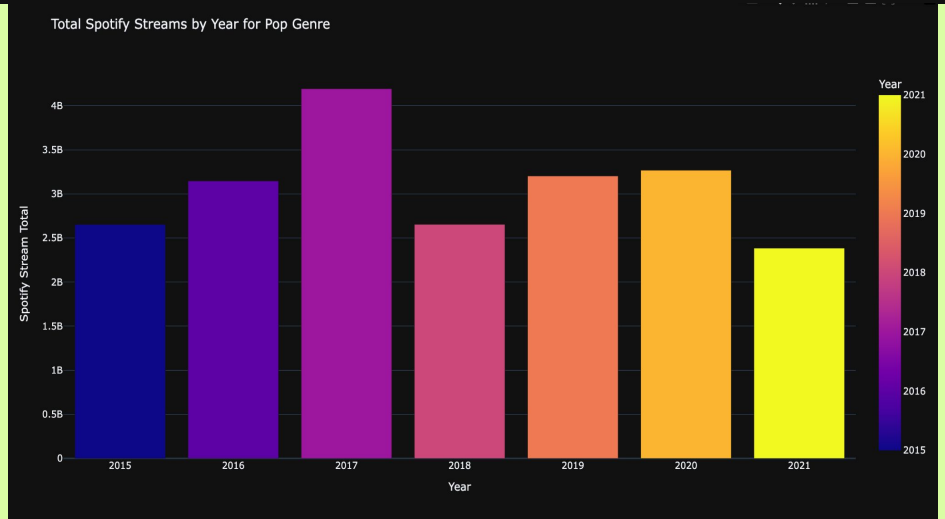
Reading the queries.

Using plotly to create the chart

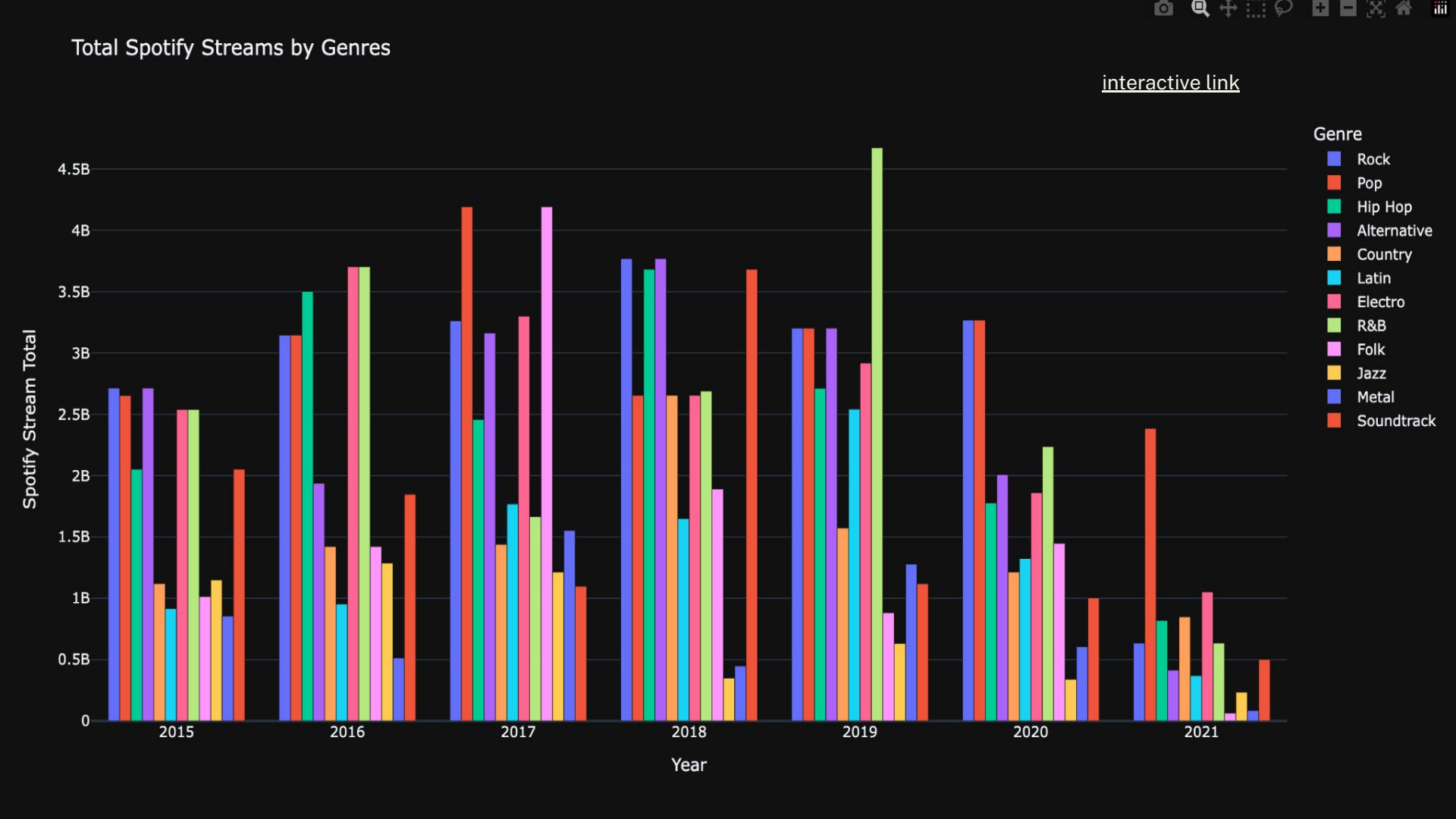
```
mport plotly.express as px

# Create the bar chart for Rock genre
fig_rock = px.bar(df_query1,
                  x='Year Released',
                  y='Spotify stream Total',
                  title="Total Spotify Streams by Year for Rock Genre",
                  labels={'Year Released': 'Year', 'Spotify stream Total': 'Spotify
Stream Total'},
                  color='Year Released',  # Optional: color by year
                  template='plotly_dark')  # Optional: dark theme for better visibility

# Save the chart as an interactive HTML file
fig_rock.write_html("html/spotify_rock_streams_by_year.html")

# Show the plot
fig_rock.show()
```


Total Spotify Streams by Year for Rock Genre
Spotify Stream Total
0
0.5B
1B
1.5B
2B
2.5B
3B
3.5B
2015
2016
2017
2018
2019
2020
2021
Year
Year
2021
2020
2019
2018
2017
2016
2015


Total Spotify Streams by Year for Hip Hop Genre
Spotify Stream Total
0
0.5B
1B
1.5B
2B
2.5B
3B
3.5B
2015
2016
2017
2018
2019
2020
2021
Year
Year
2021
2020
2019
2018
2017
2016
2015


Total Spotify Streams by Year for Pop Genre
Spotify Stream Total
0
0.5B
1B
1.5B
2B
2.5B
3B
3.5B
4B
2015
2016
2017
2018
2019
2020
2021
Year
Year
2021
2020
2019
2018
2017
2016
2015

```python
# Concatenate DataFrames
df_combined = pd.concat([df_query1.assign(Genre="Rock"), df_query2.assign(Genre="Pop"),
df_query3.assign(Genre="Hip Hop"), df_query4.assign(Genre="Alternative"), df_query5.assign(Genre="Country"),
df_query6.assign(Genre="Latin"), df_query7.assign(Genre="Electro"), df_query8.assign(Genre=("R&B")),
df_query9.assign(Genre="Folk"), df_query10.assign(Genre=("Jazz")), df_query11.assign(Genre=("Metal")),
df_query12.assign(Genre=("Soundtrack"))])
# Create a combined bar chart with side-by-side bars
fig_combined = px.bar(df_combined,
                      x='Year Released',
                      y='Spotify stream Total',
                      color='Genre',  # Color by genre
                      title="Total Spotify Streams by Genres",
                      labels={'Year Released': 'Year', 'Spotify stream Total': 'Spotify Stream Total'},
                      template='plotly_dark',
                      barmode='group')  # Set bars to be side by side

# Save the combined chart as an interactive HTML file
fig_combined.write_html("html/spotify_genre_streams_by_year_grouped.html")

# Show the plot
fig_combined.show()
```


Total Spotify Streams by Genres
interactive link
Genre
Rock
Pop
Hip Hop
Alternative
Country
Latin
Electro
R&B
Folk
Jazz
Metal
Soundtrack
Spotify Stream Total
4.5B
4B
3.5B
3B
2.5B
2B
1.5B
1B
0.5B
0
2015
2016
2017
2018
2019
2020
2021
Year

Results of Visualizations

- Pop is the most popular genre every year except 2018 where hip-hop surpassed it
- Total streams peaks in 2019
- The decline of rock music is notable being the 2nd most popular in 2015 and 2016 to 6th most popular in 2023
- The rise of the Latin genre from 7th most popular in 2015 to 3rd most in 2023
- 2024 Alternative having more streams than Rock
- Genres in the top 10 every year : Pop, Rock, Electro, R&B, Alternative, Latin, Hip-Hop, and Country

- Initial Data Cleaning.....
- Regex used to sort for just the numerical 4-digit year in `Project_3_test2.ipynb`
    - This was a unique case due to the data in the 'Release date'. A simple `.fillna` approach wouldn't work. We had to first fill NaN values with a string version of '1776' to avoid breaking the regex when encountering an integer instead of a string.
    - Steps:

Fill NaN values in 'Release date' with '1776' and convert to string:

```
df_cleaned['Year Released'] = df_cleaned['Release date'].fillna('1776').astype(str)
```

Extract the year from 'Year Released' using regex:

```
df_cleaned['Year Released'] = df_cleaned['Year Released'].str.extract(r'(\d{4})')
```

Fill any remaining NaN values in 'Year Released' with '1776':

```
df_cleaned['Year Released'] = df_cleaned['Year Released'].fillna('1776')
```

- Convert the 'Year Released' column to integers for further mathematical operations:

```
df_cleaned['Year Released'] = df_cleaned['Year Released'].astype(int)
```

```
from tabulate import tabulate

# Pretty print the first 20 rows of
the DataFrame using tabulate
print(tabulate(df_cleaned.head(20),
headers='keys', tablefmt='pretty',
showindex=False))
```

Using tabulate sample

```python
tabulate_df = pd.read_csv('Cleaned.csv')
print(tabulate(tabulate_df.head(15), headers='keys', tablefmt='pretty',
showindex=False))
```

✓ 0.1s

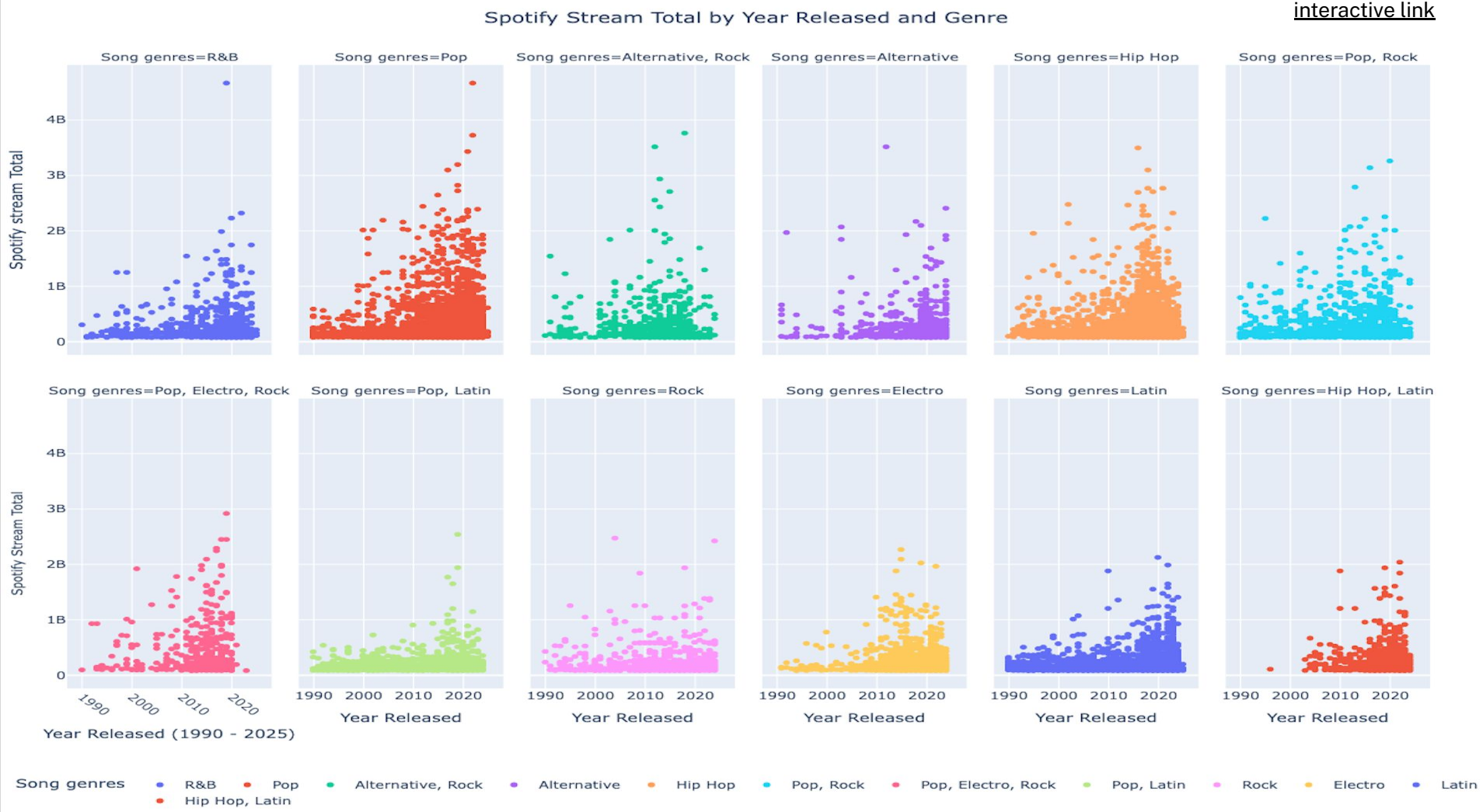
Song genres	Artist(s)	Artist Country	Spotify stream Total
R&B	The Weeknd	Canada	4673836529
Pop	Batusyex, imnotege, Lanceas, rufflws, xSyborg	Turkey, Turkey, Germany, Turkey	4673836529
Pop, Folk	Ed Sheeran	United Kingdom	4193269875
Alternative, Rock	Lewis Capaldi	United Kingdom	3770959284
Pop	Harry Styles	United Kingdom	3731855719
R&B, Electro	Daft Punk, The Weeknd	France, Canada	3704089716
R&B, Electro	Daft Punk, The Weeknd	France, Canada	3704089716
Hip Hop, Soundtrack	Post Malone, Swae Lee	United States, United States	3683693906
Alternative, Rock	The Neighbourhood	United States	3524300277
Alternative	The Neighbourhood	United States	3524300277
Hip Hop	Drake, Kyla, WizKid	Canada, United Kingdom, Nigeria	3503488880
Pop	Justin Bieber, The Kid Laroi	Canada, Australia	3439352937
Pop, Electro, Hip Hop	Justin Bieber, The Kid Laroi	Canada, Australia	3439352937
Electro, Pop	Imagine Dragons	United States	3300915632
Pop, Rock	Glass Animals	United Kingdom	3268317021

2 Quick examples of using facet in the plotly library

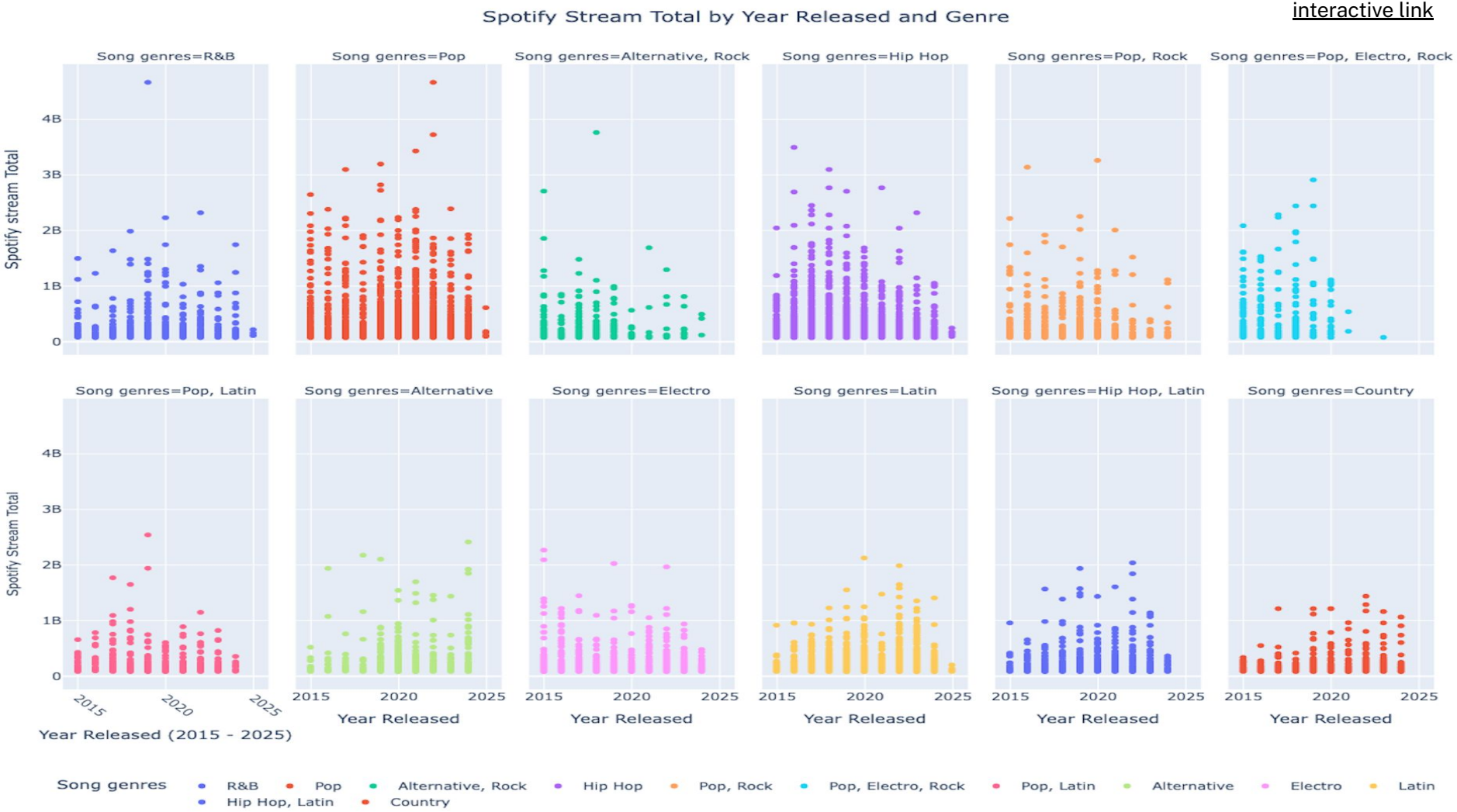
```
import plotly.express as px
```

Using Facets with Plotly examples

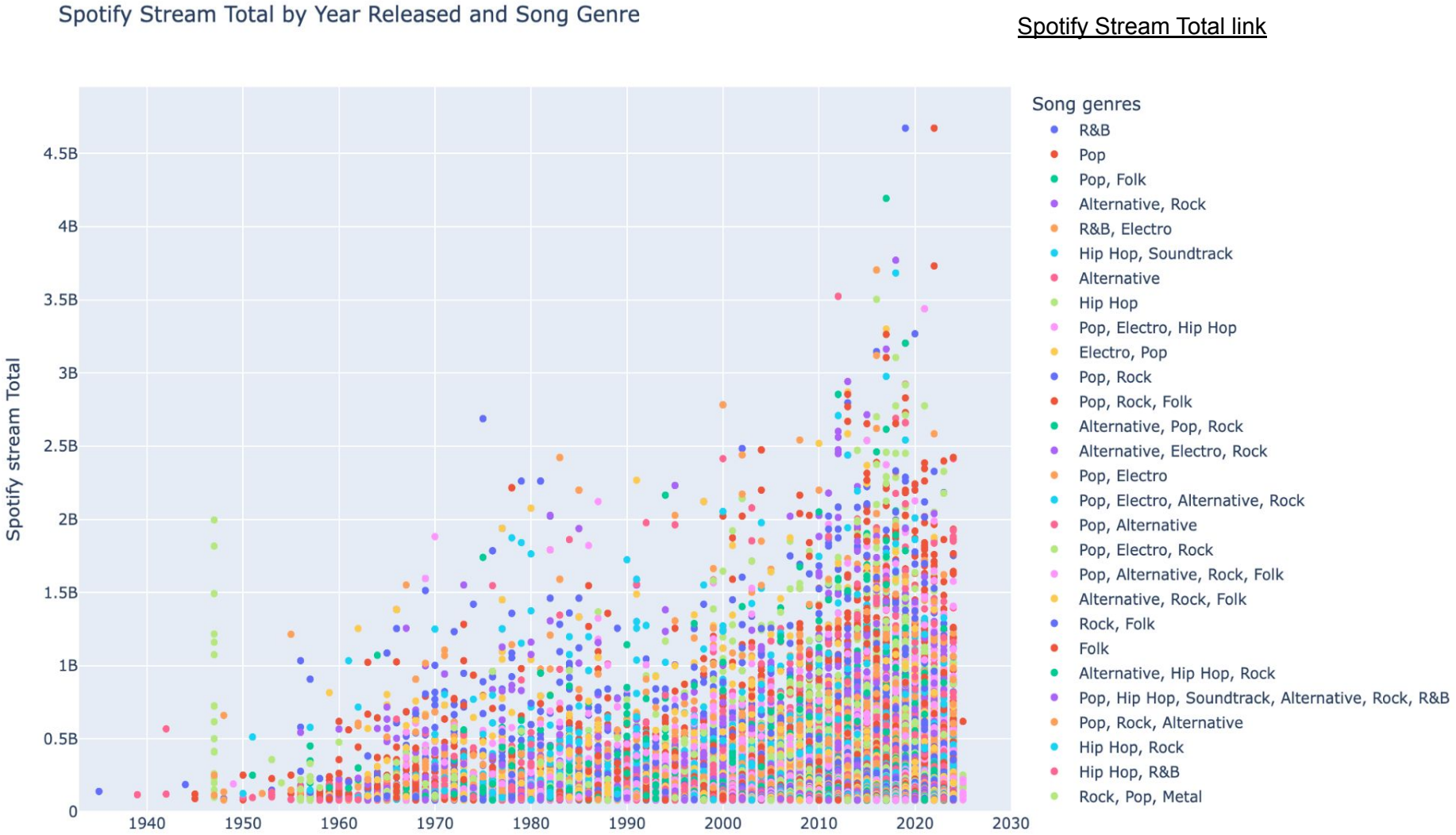
interactive link



interactive link



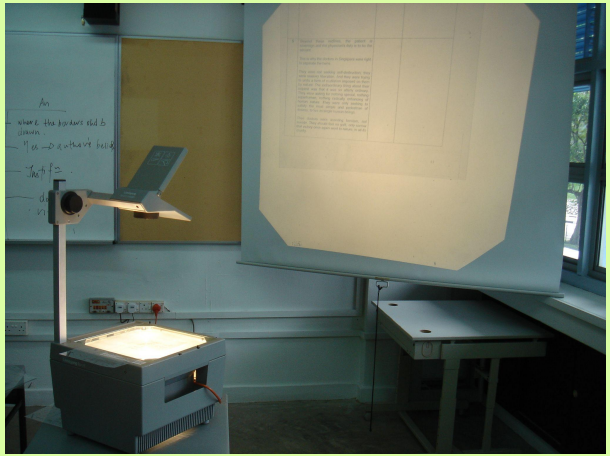
Spotify Stream Total link



Notes on Visualizations

- Interesting to note what genres often get combined when categorizing a song
- Number of streams peak between 2018 to 2020
- Any guess what those two dots at the peak of the chart are? (hint: they are the same song)

For the record.
Now this is what I think of as a slide show.
Not to be confused with the picture slide show.



Outro