

Now Playing: Spotify's Most Streamed Songs

WRAPPED



TOP
ARTISTS
BY
STREAMS

TOP
SONGS
BY
STREAMS

PLATFORM
DOMINANCE

SONGS WITH
HIGHEST
DANCEABILITY
& ENERGY



Shuffle Play: "The Dataset"

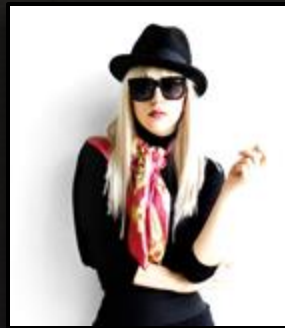
RECURRING
ARTISTS IN
CHARTS

SONGS WITH
LEAST
SPEECHINESS

TOP ARTISTS
WITH MULTIPLE
CONTRIBUTORS

RECURRING
ARTISTS IN
PLAYLISTS

Artists



Track

ly

Flowers

Escapism

Kill Bill

Blinding Lights

Heat Waves

Sumr

Streaming Metrics

arts

Streams

Spotify Charts

Deezer Playlists

Shazam Charts

Spotify Playlists

Apple Playl

Musical Attributes

om

key

mode

danceability

energy

valence

liveness

acousticness

spe

Description of the dataset

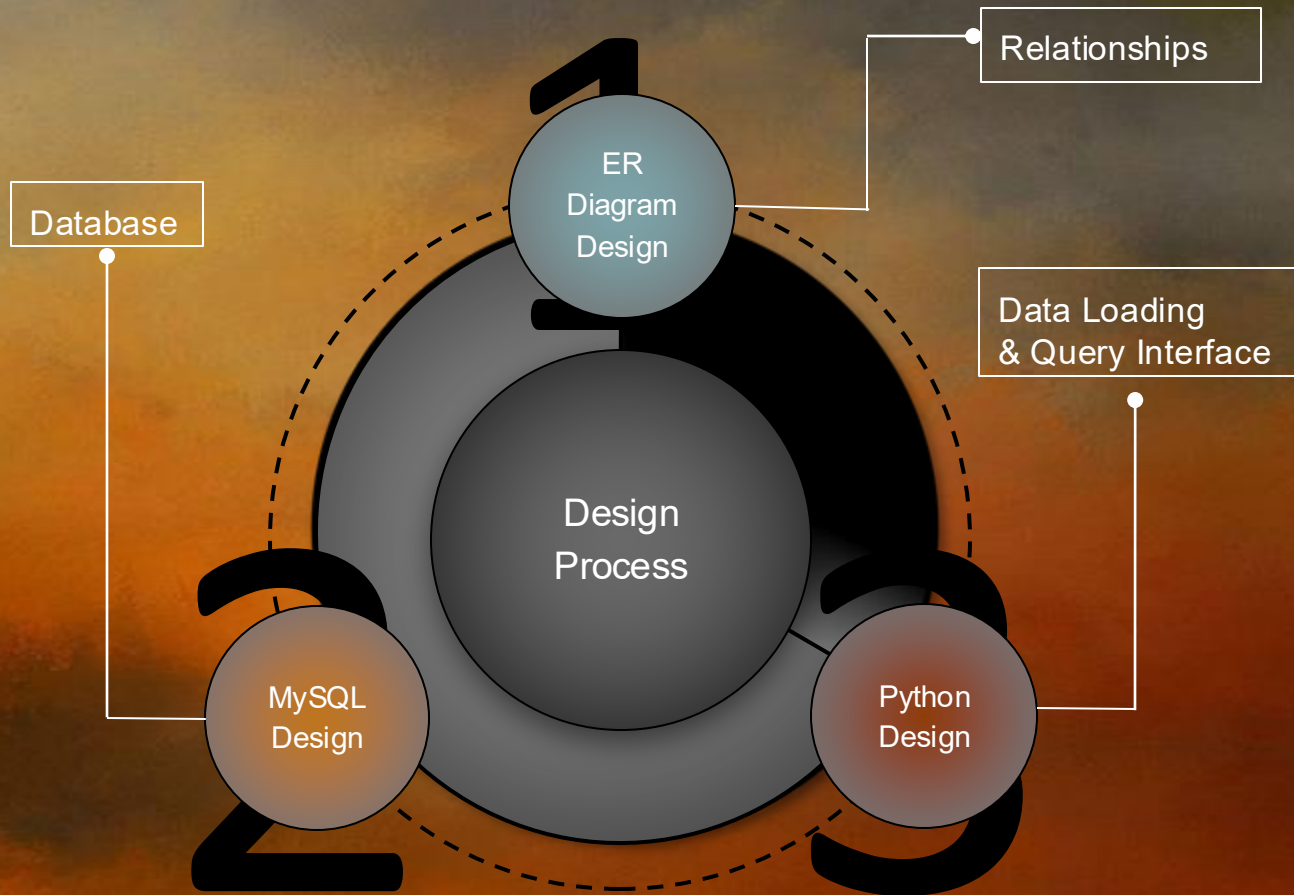
The dataset contains important information about Spotify top-streamed songs, including:

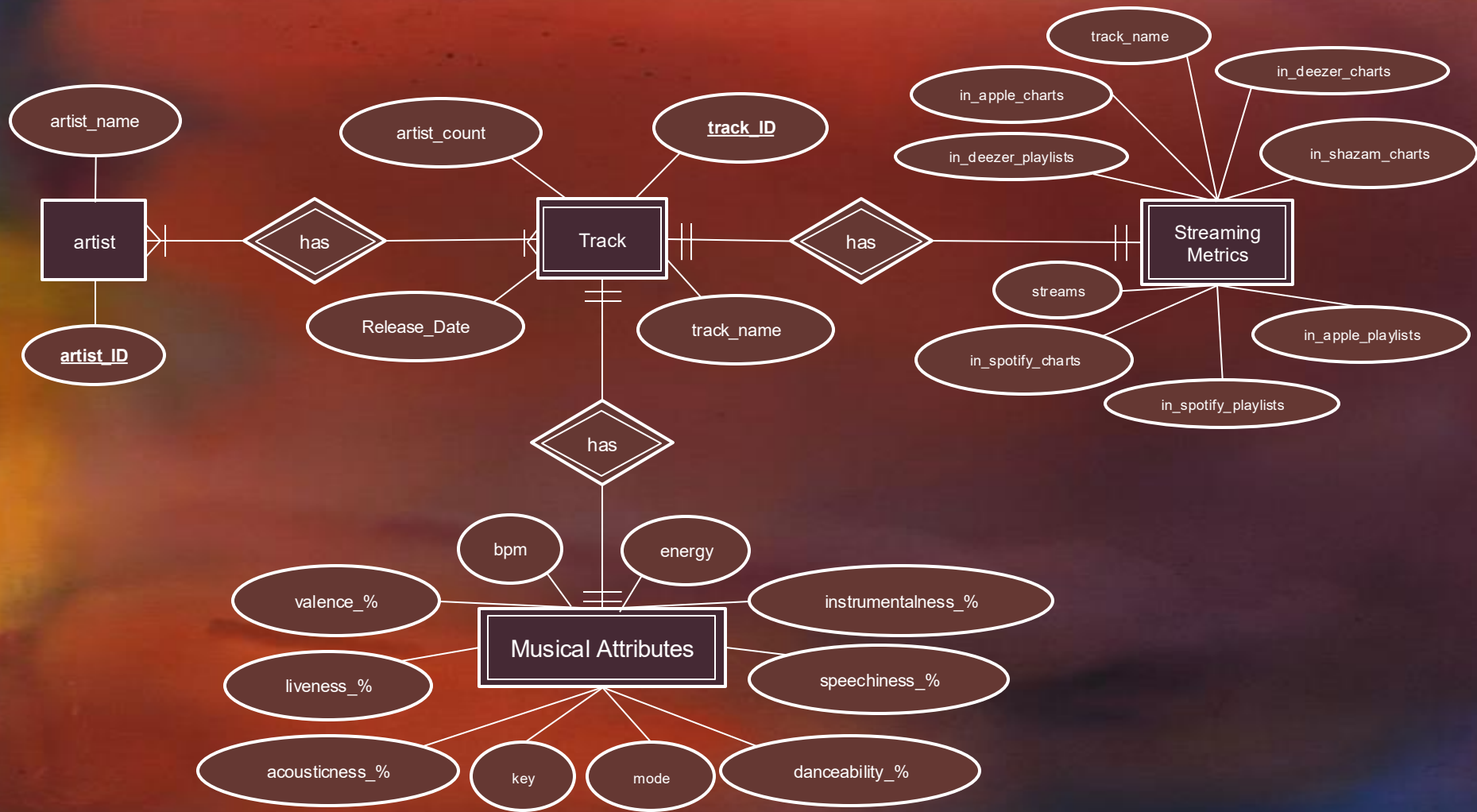
- General track details such as song name, artists, collaborators and release dates.
- Streaming metrics among Spotify, Apple Music, Deezer and Shazam, including their appearance in numerous playlists and chart rankings
- Information about musical attributes such as BPM, danceability, energy and acousticness (crucial for pattern identification)

Project overview

In this project, we're exploring the trends and characteristics of tracks and artists among popular streaming platforms, such as Spotify, Apple Music, Shazam and Deezer.

Our primary goal is designing and implementing a relational database to store and analyze the Spotify Most Streamed Songs dataset.

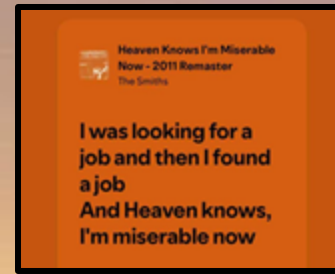




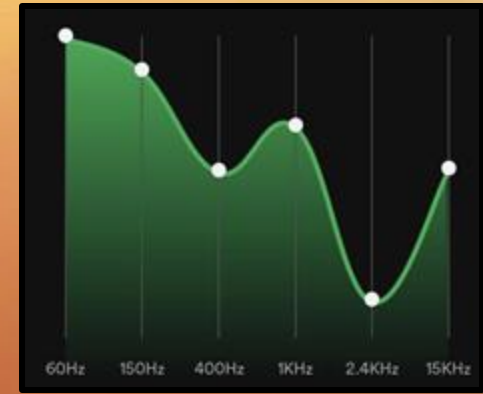
Database Creation

After creating our ER Diagram, we begin creating tables in MySQL with respect to the entities and relationships in the diagram.

Additionally, we include foreign keys and IDs in the tables that will be tools for us to call data with later in our queries.



billboard	
HOT 100	
SONG	ARTIST
1 Savage	Megan Thee Stallion ft. Beyoncé
2 Say So	Doja Cat
3 Blinding Lights	The Weeknd
4 Rockstar	DaBaby ft. Roddy Ricch
5 Toosie Slide	Drake
6 Life Is Good	Future ft. Drake
7 The Box	Roddy Ricch
8 Don't Start Now	Dua Lipa
9 Intentions	Justin Bieber ft. Quavo
10 Circles	Post Malone



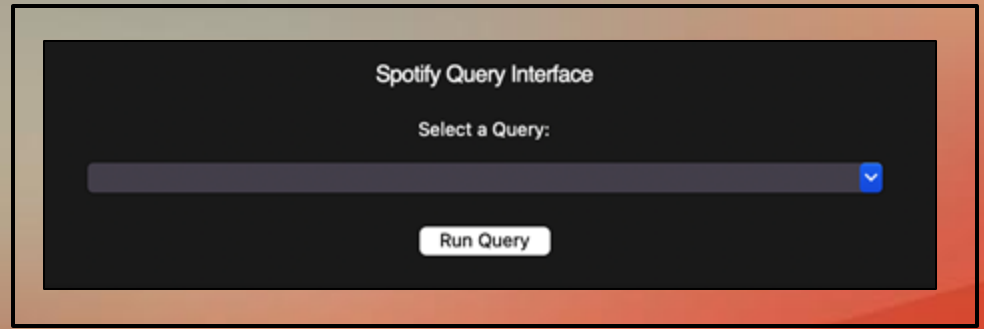


DATA LOADING

After designing and creating the database in MySQL, the next step involves populating it with clean, structured data. Here's how we accomplished this in VSCode with Python:

- *Data Cleaning and Preparation:*
We implemented a data cleaning function to handle inconsistencies, null values, and incorrect formatting in the raw CSV dataset. This ensured the data was ready for seamless insertion without causing errors or compromising results.
- *Efficient Data Insertion with the “**dataloading**” Function:*
The “**dataloading**” function connects to the MySQL database and systematically inserts the cleaned data into normalized tables. It uses:
 - Caching: For artist and track IDs to optimize performance and avoid redundant database queries.
 - Iterative Logic: To map multiple artists to tracks and establish relationships across tables like Artist_Track.
- *Structured Attribute Mapping:*
We aligned dataset columns with our table attributes, ensuring every entry (e.g., streams, playlists, musical properties) was correctly mapped into its respective table.

Query Interface



Database Connection

After loading the data, we create a query interface and connect it to the query interface.



Query Implementation

After creating and testing our queries in MySQL, we add them into the interface.



User Interface

We set up the user interface to be user friendly and efficient with our queries.

Adding Query Functionality

This step adds SQL queries to the queries dictionary for use in the interface. Each query is linked to a clear title for easy selection from the dropdown menu.

```
# Queries for the Spotify database
queries = {
    '-- Top Artist by Stream Count': ''
        SELECT
            a.artist_name,
            SUM(sm.streams) AS total_streams
        FROM
            Artist a
        JOIN
            Artist_Track at ON a.artist_id = at.artist_id
        JOIN
            Track t ON at.track_id = t.track_id
        JOIN
            StreamingMetrics sm ON t.track_id = sm.track_id
        GROUP BY
            a.artist_name
        ORDER BY
            total_streams DESC
        LIMIT 10;
    ...
}
```

Executing and Displaying Queries

This part manages the execution of SQL queries and displays the results in the interface.

It

connects to the database, retrieves query results, and dynamically updates the Treeview to show the data in a user-friendly format.

Errors are handled gracefully, ensuring a smooth user experience.

```
# Execute a query and return the result as a DataFrame
def execute_query(query):
    """
    Executes the given SQL query and retrieves the results in a pandas DataFrame.

    Args:
        query (str): The SQL query to execute.

    Returns:
        DataFrame: A pandas DataFrame containing the query results.
    """
    try:
        conn = pymysql.connect(**db_config)
        cursor = conn.cursor()
        # Execute the query and fetch all the rows
        cursor.execute(query)
        rows = cursor.fetchall()
        # Extract column names for the DataFrame
        columns = [desc[0] for desc in cursor.description]
        # Return the results as a pandas DataFrame
        return pd.DataFrame(rows, columns=columns)
    except Exception as e:
        # Show an error message if the query fails
        messagebox.showerror("Error", f"Failed to execute query: {e}")
        return None
    finally:
        if conn:
            conn.close()
```

```
# Run the selected query and display the results in the UI
def run_query():
    """
    Handles the execution of the query selected by the user
    and displays the results in the Treeview widget.
    """
    # Get the selected query from the dropdown
    selected_query = query_combobox.get()
    if not selected_query:
        messagebox.showwarning("Warning", "Please select a query first.")
        return
    # Fetch the query from the dictionary and execute it
    query = queries[selected_query]
    result = execute_query(query)
    if result is not None:
        # Clear any previous results in the Treeview
        for item in tree.get_children():
            tree.delete(item)
        # Set up the Treeview columns to match the query results
        tree["columns"] = list(result.columns)
        tree["show"] = "headings"
        for col in result.columns:
            tree.heading(col, text=col) # Set column header text
            tree.column(col, width=150, anchor="center") # Adjust column width
        # Insert the rows into the Treeview
        for _, row in result.iterrows():
            tree.insert("", "end", values=list(row))
```

UI Layout Setup for Spotify Query Interface

- **Title Label:** Displays the application title prominently at the top.
- **Dropdown Menu:** Enables users to select a query from a list of predefined options.
- **Run Query Button:** Executes the selected query and fetches data.
- **Treeview Frame:** Dynamically displays query results in a table-like structure.
- **Global Widgets:** Ensures accessibility for critical UI components across the application.

```
# Set up the UI layout
def setup_ui(root):
    """
    Sets up the graphical interface for the application, including labels, buttons, and the Treeview.

    Args:
        root (Tk): The root Tkinter window.
    """
    # Add a title label at the top of the window
    tk.Label(root, text="Spotify Query Interface", font=("Helvetica", 16)).pack(pady=10)
    # Add a label and dropdown menu for selecting queries
    tk.Label(root, text="Select a Query:").pack(pady=5)
    global query_combobox
    query_combobox = ttk.Combobox(root, values=list(queries.keys()), state="readonly", width=60)
    query_combobox.pack(pady=5)
    # Add a button to run the selected query
    tk.Button(root, text="Run Query", command=run_query).pack(pady=10)
    # Add a frame to hold the Treeview widget
    tree_frame = tk.Frame(root)
    tree_frame.pack(fill="both", expand=True, padx=10, pady=10)
    # Add the Treeview to display query results
    global tree
    tree = ttk.Treeview(tree_frame)
    tree.pack(fill="both", expand=True)
```


Lists artists by the highest number of streams.

TOP
ARTISTS
BY
STREAMS

Lists songs by the highest percentage of danceability and energy.

SONGS WITH
HIGHEST
DANCEABILITY
& ENERGY

Lists songs by their appearance count on different playlists

RECURRING
ARTISTS
IN
PLAYLISTS

Lists artists by the count of their collaborations

TOP ARTISTS
WITH MULTIPLE
CONTRIBUTORS

SONGS WITH
LEAST
SPEECHINESS

Lists songs by the lowest percentage of speechiness.

TOP
SONGS
BY
STREAMS

Lists songs by the highest number of streams.

PLATFORM
DOMINANCE

Lists songs by the highest number of streams.

RECURRING
ARTISTS
IN
CHARTS

Lists songs by their appearance count on different charts

Query analysis

In frame of this project, our group developed the following queries:

- Top Songs by Stream Count
- Top Artists by Stream Count
- Artist Appearing Most in Playlists
- Most Collaborated Artists
- Minimum Speechiness
- Artists Appearing Most in Charts
- Average Danceability & Energy
- Popularity Score of Artists (Normalized)
- Top Acoustic tracks by acousticness and streams
- Most streamed track per year

Spotify Query Interface

Select a Query:
-- Top Artist by Stream Count

Run Query

artist_name	total_streams
Bad Bunny	23813527270
The Weeknd	22397604621
Ed Sheeran	15316587718
Taylor Swift	14630378183
Harry Styles	11608645649
Eminem	10193727260
Dua Lipa	9980020481
Justin Bieber	8243081039
Drake	8043031261
BTS	7780428159

Top Artist by Stream Count:

Outputs the Top 10 artists with the highest number of total streams by gathering the artist name by artist ID. Total streams is calculated by summing up the stream counts for artists across the dataset.

Top Songs by Stream Count:

Outputs the Top 10 songs with the highest number of streams by gathering track names from track ID's and their streams. It is ordered by descending order.

Spotify Query Interface

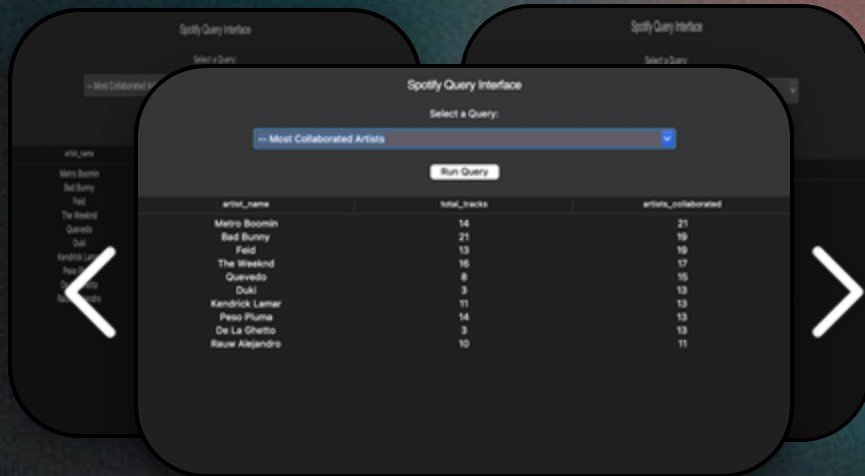
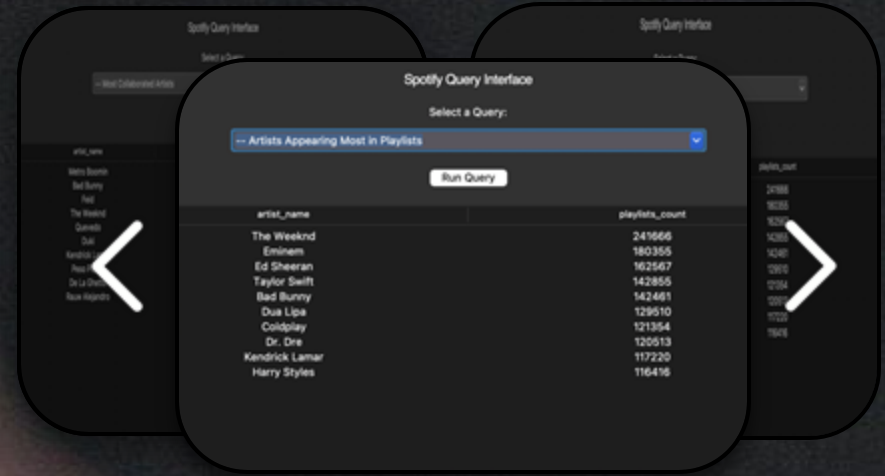
Select a Query:
-- Top Songs by Stream Count

Run Query

track_name	streams
Blinding Lights	3703895074
Shape of You	3562543890
Someone You Loved	2887241814
Dance Monkey	2884791672
Sunflower - Spider-Man: Into the Spider-Verse	2808096550
One Dance	2713922350
STAY (with Justin Bieber)	2665343922
Believer	2594040133
Closer	2591224264
Starboy	2565529693

Artists Appearing Most In Playlists:

This query examines which artists are featured most frequently across numerous playlists



Most Collaborated Artists:

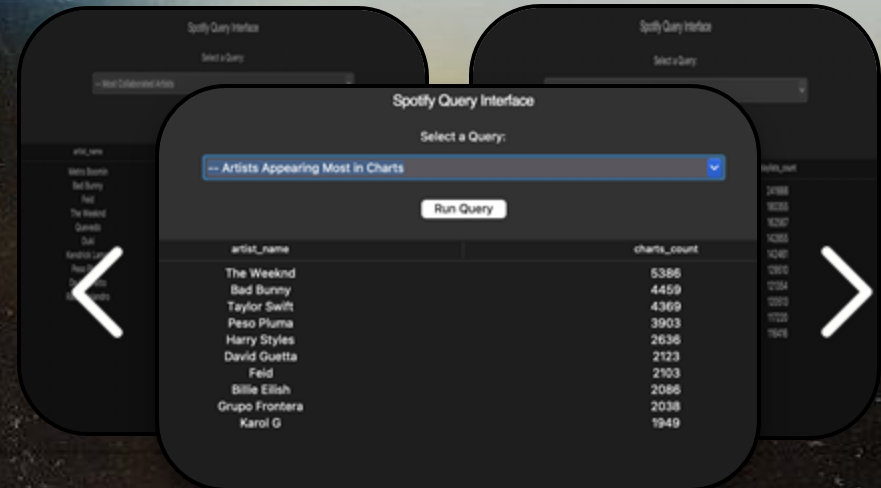
Lists the top 10 artists with the highest number of collaborations by counting unique tracks and distinct collaborators. The results are ordered by the number of collaborators in descending order.

Minimum Speechiness:

Outputs the Top 10 songs with the highest number of streams by gathering track names from track ID's and their streams. It is ordered by descending order.

Artists Appearing Most In Charts:

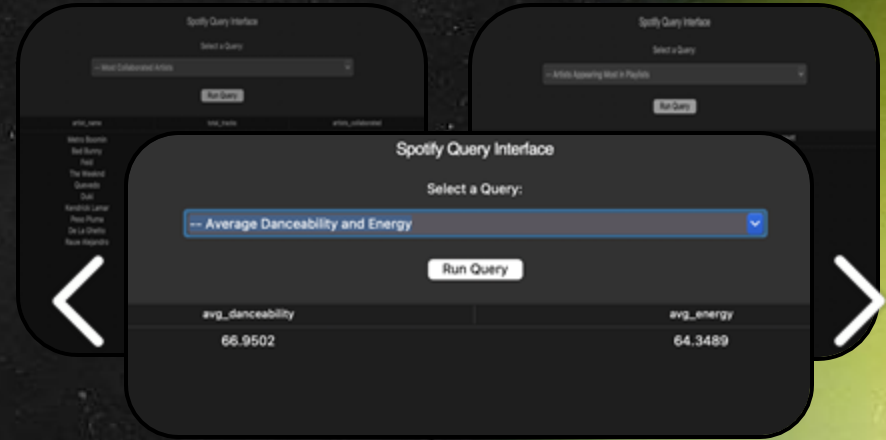
Outputs the top 10 artists who appear most frequently in charts (Spotify, Apple, Deezer, and Shazam) by aggregating chart metrics for their respective tracks. The results are ordered in descending order based on chart appearances



artist_name	charts_count
The Weeknd	5386
Bad Bunny	4459
Taylor Swift	4369
Peso Pluma	3903
Harry Styles	2636
David Guetta	2123
Feld	2103
Billie Eilish	2086
Grupo Frontera	2038
Karol G	1949

Average Danceability & Energy:

Outputs the Top 10 songs with the highest number of streams by gathering track names from track ID's and their streams. It is ordered by descending order.



The screenshot shows the Spotify Query Interface with the query 'Popularity Score of Artists (Normalized)' selected. The results are displayed in a table with three columns: artist_name, raw_popularity_score, and normalized_popularity_score.

artist_name	raw_popularity_score	normalized_popularity_score
The Weeknd	247052	0.0315
Eminem	181265	0.0231
Ed Sheeran	164292	0.0210
Taylor Swift	147224	0.0188
Bad Bunny	146920	0.0187
Dua Lipa	130823	0.0167
Dr. Dre	121621	0.0155
Coldplay	121796	0.0155
Harry Styles	119062	0.0152
Kendrick Lamar	118278	0.0151

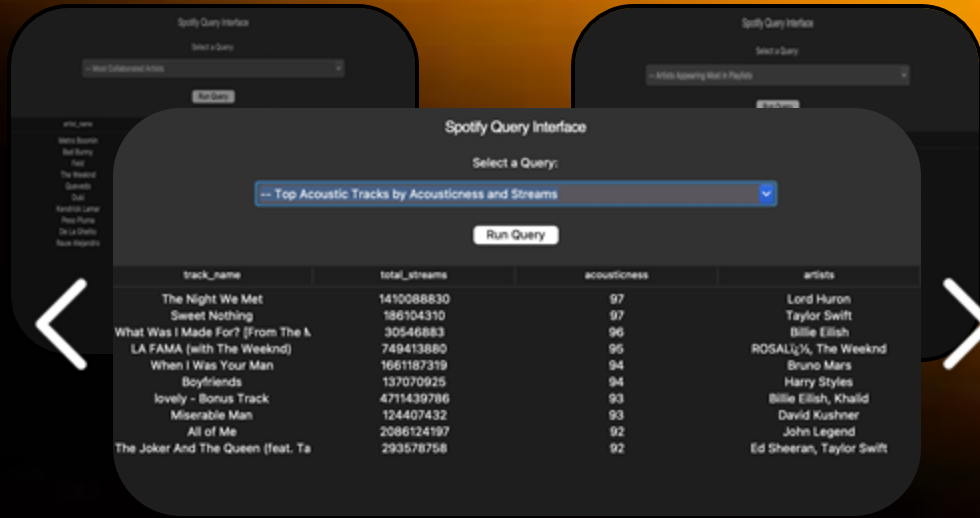
Popularity Score of Artists (Normalized):

Outputs the Top 10 artists with the most presence across all streaming platforms and charts by gathering track names from track IDs and their respective streaming metrics. Raw popularity (across all platforms) score is calculated and normalized to reflect a relative measure (fraction of total popularity).

The result is ordered in descending order.

Top Acoustic Tracks by Acousticness and Streams:

Outputs the Top 10 most popular songs (based on stream count) with the highest percentage of acousticness by gathering track names from track IDs, their acousticness, and streams. It is ordered by descending acousticness order.



The screenshot shows the Spotify Query Interface with the query "Top Acoustic Tracks by Acousticness and Streams" selected. The results are displayed in a table with columns: track_name, total_streams, acousticness, and artists.

track_name	total_streams	acousticness	artists
The Night We Met	1410088830	97	Lord Huron
Sweet Nothing	186104310	97	Taylor Swift
What Was I Made For? (From The A	30546683	96	Billie Eilish
LA FAMA (with The Weeknd)	749413880	95	ROSALÍA, The Weeknd
When I Was Your Man	1661187319	94	Bruno Mars
Boyfriends	137070925	94	Harry Styles
lovely - Bonus Track	4711439786	93	Billie Eilish, Khalid
Miserable Man	124407432	93	David Kushner
All of Me	2086124197	92	John Legend
The Joker And The Queen (feat. Ta	293578758	92	Ed Sheeran, Taylor Swift

Spotify Query Interface

Select a Query:

-- Most Streamed Track per Year

Run Query



The screenshot shows the Spotify Query Interface with the query "Most Streamed Track per Year" selected. The results are displayed in a table with columns: release_year, track_name, artists, and streams.

release_year	track_name	artists	streams
1930	Agudo Múñiz	Stark, Theodor, ulku INC	90588517
1942	White Christmas	Bing Crosby, John Scott Trotter & I	395591396
1946	The Christmas Song (Merry Christ	Nat King Cole	388777964
1952	A Holly Jolly Christmas - Single Ver	Burl Ives	395591396
1957	Jingle Bell Rock	Bobby Helms	741301563
1958	Rockin' Around The Christmas Tree	Brenda Lee	789213520
1959	Let It Snow! Let It Snow! Let It Sno	B. Swanston Quartet, Dean Martin, I	473248298
1963	It's the Most Wonderful Time of the	Andy Williams	663832097
1968	Have You Ever Seen The Rain?	Credence Clearwater Revival	1545727611
1970	Feliz Navidad	Joni Mitchell	620034544
1971	Happy Xmas (War Is Over)	John Lennon, The Harlem Commun	460482796
1973	Dream On	Aerosmith	838586769
1975	Bohemian Rhapsody - Remastered	Queen	2197010579
1979	Wonderful Christmastime - Edited	Paul McCartney	403939467
1982	Pass The Dutchie	198918494	
1983	Every Breath You Take - Remaste	The Police	1593270737
1984	Take On Me	a-ha	1479110596
1985	Everybody Wants To Rule The Wor	Tears For Fears	1205957614
1986	Master of Puppets (Remastered)	Metallica	704174068
1987	Sweet Child O' Mine	Guns N' Roses	1533497987
1991	Smells Like Teen Spirit - Remaste	Nirvana	1690192927
1992	Creep	Radiohead	1271293243
1994	All I Want For Christmas Is You	Mariah Carey	1449779435
1995	Gangsta's Paradise	Coolio, L.V.	1357606774
1996	Single	Ed Sheeran, J Balvin	106833707
1997	Cupid 1/2/1/2/1/2 Twin Ver. (FIFTY	speed up 8282	107622516
1998	It's	The Goo Goo Dolls	1384942608

Most Streamed Track Per Year:

Outputs the most streamed tracks of their respective years. It calls the track's release year as well as its artists and streams. If multiple artists are associated with a track, their names are concatenated into a single string by GROUP_CONCAT(). The result is sorted by release year.

Thank You

Let's Rewind.

We created an ER Diagram and visualized the entities, attributes, cardinalities and relationships.

Assima
Amangeldina,
Ceyla Kaya,
Sabina
Nurseitova,
Ziyi Dong



Our Project At A Glance

In MySQL we created tables and implemented relationships and keys.

Essentially, we gathered information about artist and track trends. This output is crucial for streaming platforms like Spotify to cater to their users' preferences.

In Python we loaded the dataset into our MySQL tables and cleaned the data. After that, we created a query interface in order to make an interactive visual demonstration of our queries.