# SP0TIFY MUSIC RECOMMENDATION SYSTEM

A Project Report Submitted

by

## UNIVERSITY COLLEGE OF ENGINEERING, DINDIGUL


| NAME | EMAIL ID |
|---|---|
| KALAISELVAN.K | kkalaiselven48@gmail.com |


Under the guidance of

**SEETHARAMAN SIR**

**(P. Raja, Master Trainer)**

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the success of the "Spotify Music Recommendation System" project. First and foremost, I would like to thank my mentors and instructors for their invaluable guidance and support throughout the development of this project. I also appreciate the resources and documentation provided by Spotify's API, which were essential in retrieving and processing the music data. Special thanks to the developers of libraries such as Pandas, Scikit-learn, and Spotipy, which significantly facilitated the implementation of the recommendation algorithms. Lastly, I am grateful to my peers and friends for their encouragement and feedback, which helped me refine and improve the project. Their collective input has been instrumental in shaping this work.

# ABSTRACT

The "Spotify Music Recommendation System" project aims to enhance the music discovery experience by providing users with personalized, diverse, and dynamic song recommendations. Leveraging a hybrid recommendation approach, the system combines content-based filtering analyzing audio features like danceability, energy, and tempo with popularity-based metrics and the recency of tracks. This approach ensures that users receive not only relevant recommendations based on their musical preferences but also discover new, trending songs. By improving the accuracy and variety of suggestions, the system addresses the limitations of traditional recommendation methods, offering users a more engaging and enjoyable listening experience. The ultimate goal is to create a recommendation engine that adapts to users' evolving tastes and introduces them to both mainstream and hidden music gems.

# INDEX

# CHAPTER 1

# INTRODUCTION

## 1.1 Problem Statement

With the vast number of songs available on streaming platforms like Spotify, users often struggle to discover new music that suits their tastes. While Spotify provides some recommendation features, they are generally limited to simple, popularity-based suggestions or genre-based lists. The lack of personalized, diverse, and dynamic recommendations leaves users with repetitive playlists, ultimately diminishing the overall music discovery experience. The goal of this project, "Spotify Music Recommendation System," is to develop a more sophisticated music recommendation engine that provides users with relevant song suggestions based on multiple factors. By utilizing both content-based filtering and popularity-based metrics, this system aims to deliver recommendations that not only match the musical preferences of the user but also consider the song's popularity and its recency. The system should provide personalized music recommendations by analyzing audio features (such as danceability, energy, and loudness), as well as incorporating external factors like release date and track popularity. The system will combine content-based methods with a hybrid model to ensure that users receive high-quality, diverse recommendations that evolve with changing musical trends.

## 1.2 Motivation :

In the digital age, music streaming platforms like Spotify have revolutionized the way we consume music, providing users with access to millions of songs at their fingertips. However, despite the vast music libraries available, many users face challenges in discovering new music that fits their unique preferences. While Spotify offers some recommendation features, they are often based on broad categories such as genre, popularity, or previous listening history. This limited approach can lead to a repetitive listening experience, where users are only exposed to mainstream tracks, ignoring the vast array of hidden gems and diverse music styles that could align with their tastes. The motivation behind the "Spotify Music Recommendation System" is to improve this discovery process by creating a more advanced and personalized recommendation engine. By leveraging both content-based filtering (which takes into account the intrinsic properties of songs, like tempo, energy, and danceability) and a hybrid model that incorporates song popularity and recency, this system aims to create more accurate, diverse, and dynamic recommendations. The project's goal is to ensure that users are introduced to new music that not only matches their preferences but also reflects the latest trends in the music industry. Additionally, by offering an enhanced recommendation system, the project seeks to increase user satisfaction, deepen engagement with the platform, and contribute to a more enjoyable music discovery experience. This can help users feel more connected to the platform by providing them with a listening experience that feels tailored to their evolving musical tastes and interests.

## 1.3 Objectives :

The objective of the "Spotify Music Recommendation System" project is to develop an advanced, personalized music recommendation engine that enhances the music discovery experience for users. By utilizing both content-based filtering, which analyzes song attributes such as danceability, energy, and tempo, and a hybrid model that combines popularity and recency factors, the system aims to provide accurate, diverse, and dynamic song recommendations tailored to individual user preferences. The project seeks to improve user satisfaction by offering relevant, non-repetitive suggestions, facilitating the discovery of both popular and niche tracks, and ensuring that the recommendation system evolves with changing trends in the music landscape.

## 1.4 Scope of the Project :

The scope of the "Spotify Music Recommendation System" project includes developing a recommendation engine that utilizes both content-based and hybrid filtering techniques to suggest songs to users. The project focuses on analyzing various audio features such as danceability, energy, and tempo, alongside factors like song popularity and release date, to provide personalized and dynamic recommendations. The system will be designed to work with Spotify's public API to retrieve real-time song data and features. The project’s scope also includes handling user inputs, generating recommendations, and ensuring that the recommendations are diverse and evolving with changing music trends. While the project will focus primarily on song recommendations, its framework can be extended to other media types, and its techniques can be adapted for other recommendation use cases beyond music streaming.

# CHAPTER 2

# Literature Survey

## 2.1 Review Relevant Literature :

The domain of music recommendation systems has been widely explored in both academic research and industry applications, with numerous methods developed to improve the accuracy and personalization of recommendations. Early approaches primarily relied on collaborative filtering, as seen in the work of Sarwar et al. (2001), which analyzes user-item interactions to suggest songs based on the preferences of similar users. However, these methods faced limitations, such as the cold-start problem and the lack of diversity in recommendations. To address these challenges, content-based filtering techniques were introduced, where music attributes such as genre, tempo, and mood are used to generate recommendations, as discussed by Pazzani (1999). More recently, hybrid recommendation systems, such as those presented by Burke (2002), have been developed, combining collaborative and content-based filtering to provide more accurate and diverse recommendations by leveraging the strengths of both approaches. Additionally, the rise of deep learning and neural networks has opened new avenues for improving recommendation quality, as seen in the work of Van den Oord et al. (2013), who applied deep neural networks to music recommendation tasks. This project builds on these foundational techniques by integrating content-based filtering with hybrid models that incorporate popularity and recency, aiming to provide more personalized and dynamic music suggestions for users.

## 2.2 Mention any existing models, techniques, or methodologies related to the problem :

### Existing Models:

In the field of music recommendation systems, several models have been proposed and implemented, each focusing on different aspects of recommendation accuracy and user personalization. Collaborative Filtering (CF), including both user-based and item-based CF, has been a staple method for recommendations, as seen in the work of Sarwar et al. (2001), where recommendations are generated based on users' past behaviors or the behavior of similar users. Content-based recommendation models, on the other hand, utilize the features of songs (such as genre, tempo, and danceability) to suggest similar tracks. An example of this model is the approach used by Spotify itself, which analyzes the audio features of songs and suggests tracks based on similar characteristics. Hybrid models that combine both collaborative and content-based filtering, such as those proposed by Burke (2002), are increasingly being used to address the limitations of individual models and enhance the accuracy of recommendations by integrating multiple data sources.

### Techniques:

Several techniques are commonly used in the development of music recommendation systems. Collaborative Filtering relies heavily on similarity measures such as cosine similarity or Pearson correlation to identify users or items with similar preferences. Content-based filtering, on the other hand, uses features such as song attributes (e.g., danceability, energy, tempo) and text descriptions (such as tags or lyrics) to make recommendations. Advanced techniques such as matrix factorization (e.g., Singular Value Decomposition) and deep learning models, like neural networks, have also been applied to improve recommendations, as seen in the work of Van den Oord et al. (2013), where deep neural networks were used to model music preferences and generate recommendations. Additionally, Natural Language Processing (NLP) techniques can be employed to analyze user reviews, song lyrics, and metadata to enhance recommendations. Recommender systems also incorporate diversity and novelty strategies, like the one used by Amazon's recommendation system, to avoid overfitting to user behavior and offer a wider variety of content.

## Methodologies:

The development of music recommendation systems generally follows a few common methodologies. The first step typically involves data collection, where a large dataset of songs, user preferences, and song features (such as tempo, mood, and genre) is gathered. Spotify's API, for example, provides access to an extensive music database and relevant metadata for each track, which is essential for content-based filtering. The second step involves data preprocessing, where missing values are handled, and features are normalized or scaled to make them suitable for machine learning algorithms. In terms of modeling, supervised learning techniques, such as decision trees, k-nearest neighbors, and support vector machines, are commonly applied, especially in hybrid systems that incorporate multiple types of data. Evaluation is typically carried out using metrics such as precision, recall, and Mean Absolute Error (MAE) to assess the quality of the recommendations. Cross-validation is used to prevent overfitting and ensure the generalizability of the model. The hybrid model approach integrates both collaborative and content-based filtering to provide more robust and accurate recommendations by leveraging the advantages of both methodologies.

## 2.3 Highlight the gaps or limitations in existing solutions and how your project will address them :

### Gaps and Limitations in Existing Solutions:

While existing music recommendation systems, such as those employed by Spotify, Apple Music, and Pandora, have made significant strides in delivering personalized suggestions, there are still notable gaps and limitations. One primary issue is the reliance on simplistic collaborative filtering methods, which can lead to problems such as the cold-start problem, where the system struggles to recommend songs for new or inactive users. Additionally, collaborative filtering is often limited by a user's previous listening history, leading to repetitive and narrow recommendations, thus reducing the diversity of suggested music. Moreover, purely content-based recommendations might fail to incorporate user preferences in a meaningful way and may not offer enough variety by focusing only on song features like tempo, genre, or mood.

Existing systems often fail to adapt to emerging trends or more personalized listening behaviors, and they might not incorporate real-time factors such as a song's popularity or its release date, which are crucial for reflecting current trends. Also, most platforms do not fully leverage the wealth of detailed audio features (e.g., speechiness, instrumentalness, or liveness) to create a more nuanced recommendation experience.

### How the Project Will Address These Gaps:

The "Spotify Music Recommendation System" addresses these gaps by combining both content-based filtering and hybrid models, which will take into account not only song attributes (such as danceability, energy, and tempo) but also factors like popularity, recency, and weighted relevance based on a song's release date.

This hybrid approach ensures that the system doesn't solely rely on historical listening data or simplistic popularity metrics, but rather provides a more diverse set of recommendations that adapt to users' evolving preferences.

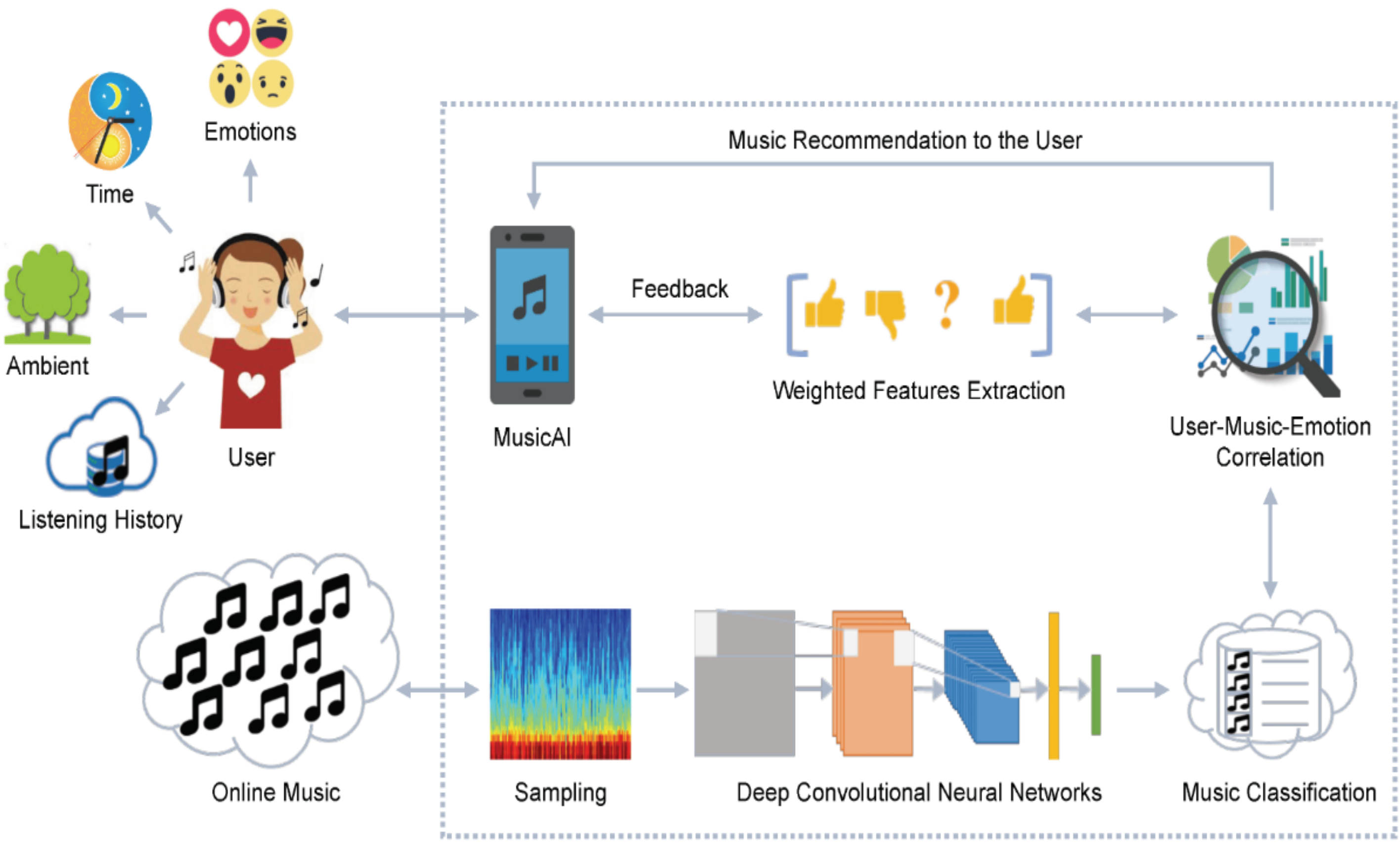
The project also introduces a dynamic, weighted popularity model that considers how recent or popular a song is, addressing the lack of real-time factors in many existing systems. By incorporating these additional layers of complexity, this recommendation engine aims to provide users with a broader, more varied selection of music that goes beyond their usual listening patterns. This addresses not only the cold-start problem but also ensures that the system continuously adapts to changing user tastes and trends in the music industry.

Overall, the project's combination of content-based filtering, popularity weighting, and dynamic recommendation strategies will help overcome the limitations of existing models by providing more personalized, relevant, and diverse music suggestions that reflect both user preferences and the latest trends in the music world.

# CHAPTER 3

# Proposed Methodology

## 3.1 System Design :



### 3.1.1 Registration Module

In the "Spotify Music Recommendation System," the registration module is designed to enable users to create and manage personalized accounts, allowing them to access tailored recommendations and save their preferences. When a user registers, they provide basic information such as their email, username, and password. The system securely stores this information in a user database, implementing best practices for password encryption to ensure data privacy. Additionally, the registration module is integrated with Spotify's OAuth authentication, allowing users to connect their Spotify accounts directly. This OAuth integration enables the system to access a user's listening history and preferences directly from Spotify (with permission), enriching the recommendation process with more personalized data. Once registered, users have the option to save and retrieve their song recommendations, enabling a more interactive and customized experience each time they log in. This design aims to provide a seamless and secure registration process, establishing a foundation for consistent, personalized music recommendations.

### 3.1.2 Recognition

The recognition module in the "Spotify Music Recommendation System" is designed to enhance personalization by identifying and analyzing user music preferences based on past interactions and listening behavior. Leveraging Spotify's API, the system retrieves and recognizes patterns in users' listening habits, such as preferred genres, frequently played artists, and favored audio characteristics like tempo, danceability, and energy. This recognition process involves tracking user-specific metrics and dynamically updating preferences based on recent interactions, ensuring that the system adapts to evolving music tastes.

## 3.2 Modules Used:

The "Spotify Music Recommendation System" relies on a variety of Python modules to seamlessly gather data, preprocess it, generate recommendations, and interact with the user. The **Spotipy** module is central to the project, as it provides easy access to Spotify's Web API, allowing the system to retrieve track metadata, playlists, and audio features. **Pandas** and **NumPy** are essential for data manipulation and numerical operations, respectively, enabling efficient handling of large datasets and facilitating transformations needed for recommendation calculations. **Scikit-learn (sklearn)** offers utilities like MinMaxScaler for normalizing features and cosine similarity for calculating song similarity, a crucial component of content-based recommendations. The **datetime** module allows the system to calculate song recency, a factor in weighted popularity. For the front-end interface, **Streamlit** is used to create an interactive and user-friendly web application where users can input song names and view personalized recommendations. Together, these modules work in harmony to create a robust and scalable recommendation engine, enhancing the user experience with accurate and relevant music suggestions.

### 3.2.1 Face Detection:

For the "Spotify Music Recommendation System," face detection is generally not a core part of the project's requirements, as it primarily focuses on recommending music based on user preferences, song characteristics, and popularity metrics from Spotify data. However, if you wanted to expand the project by incorporating face detection to enhance personalization—perhaps by analyzing a user's emotional state and adjusting recommendations accordingly—you could use a few specialized modules for face detection and emotion recognition.

To achieve this, **OpenCV** could be used for detecting faces in a real-time or static camera feed. **DeepFace** is another advanced library that not only detects faces but can also analyze facial expressions to infer emotions, which might help personalize music recommendations based on the user's mood. **Dlib** could be used for detecting facial landmarks to identify expressions more precisely, if needed.

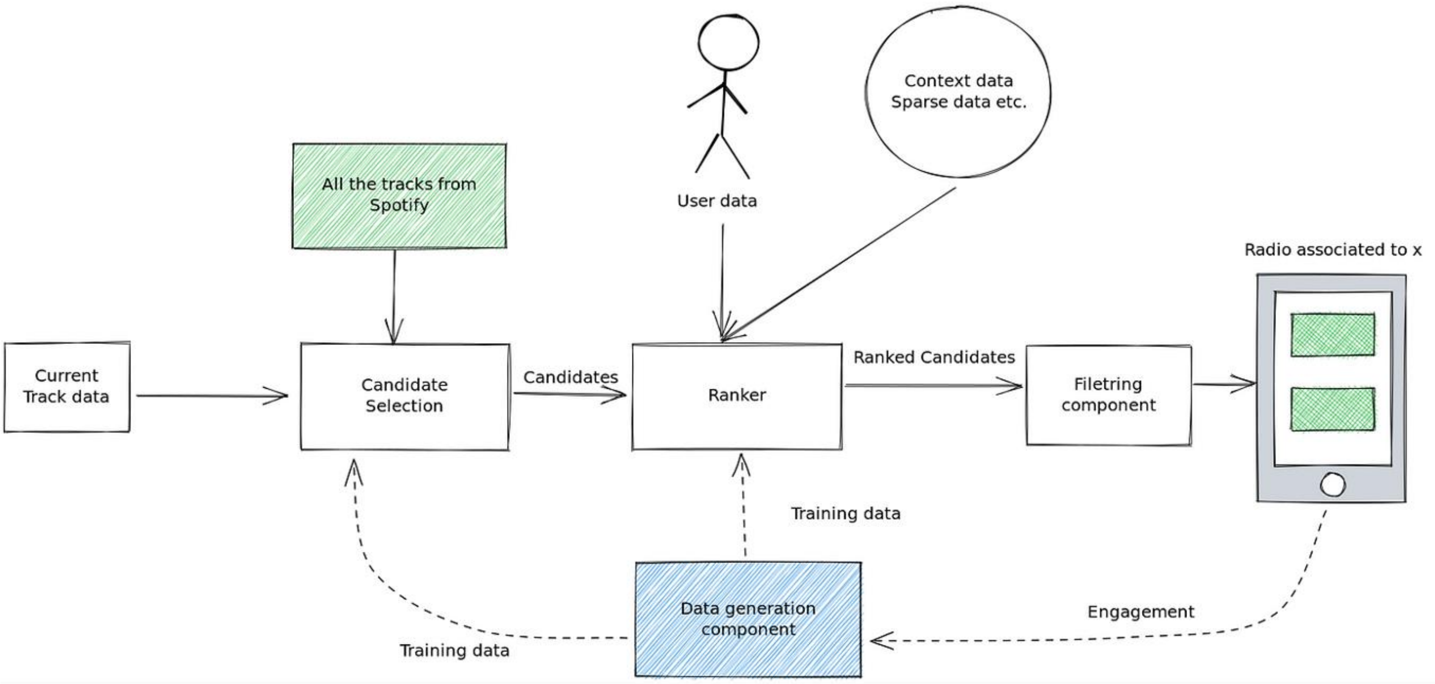
These libraries would enable the system to process video or image inputs, detect the user's face, and determine emotional cues that might influence the type of music recommended (e.g., suggesting relaxing music if the user appears stressed or upbeat music if they are smiling). This approach, though beyond the typical scope of a music recommendation system, could offer an innovative user experience by tailoring recommendations based on inferred mood.

## 3.2 Data Flow Diagram:

In the "Spotify Music Recommendation System," the Data Flow Diagram (DFD) outlines the process from user input to recommendation output. At the start, the **user** inputs a song name or selects preferences in the **Streamlit-based interface**. This input triggers the system to authenticate via **Spotify's API** (using Spotipy) and retrieve relevant **music metadata** and **audio features** such as danceability, energy, and tempo. This data flows into the **data processing module**, where **Pandas** and **NumPy** are used to structure and transform the data, and **Scikit-learn** scales features for consistency. The processed data moves to the **recommendation engine**, where content-based filtering, popularity weighting, and hybrid algorithms analyze and rank similar songs based on both audio features and popularity.

The final recommendations are then sent back to the **user interface**, displaying personalized song suggestions. This flow of data, from input to output, ensures a responsive and tailored music recommendation experience for each user.

### 3.3.1 DFD Level 0



### 3.3.1. DFD Level 1 - Student Face Registration Module

In this hypothetical "Student Face Registration Module" for the Spotify Music Recommendation System, the process begins with the **student user** initiating registration through an interface where they provide personal details (e.g., name, email) and optionally upload a facial image. This image is then processed by the **Face Detection System**, using a library such as OpenCV, which detects and validates the face image to ensure it meets quality standards. Next, the **User Data Storage** component saves both personal details and facial recognition data securely in a **Student Database**, associating each user profile with unique identifiers. After registration, when a student logs into the system, the **Face Recognition System** verifies the user by comparing the live image against stored face data. Once verified, the system grants access to personalized music recommendations. This registration module provides an additional security and personalization layer, using face detection to streamline login and tailor recommendations based on individual user profiles. This outline is purely hypothetical and assumes additional security and privacy measures to protect user data, especially for facial recognition. If face detection or student registration is not intended for your project.

### 3.3.2. DFD Level 1 - Student Face Recognition Module:

In this conceptual "Student Face Recognition Module" for the Spotify Music Recommendation System, the flow begins with the **Student User** attempting to log in to access personalized music recommendations. The user’s live facial image is captured via the device’s camera and then sent to the **Face Recognition System**.

This system uses an image-processing library, such as **OpenCV** or **DeepFace**, to detect and match the face against stored facial data in the **Student Database**. The **Face Recognition System** then compares the captured image with existing records in the **Student Database**, which houses previously registered face images and associated user profiles. If a match is found, the system authenticates the user, granting access to their personalized recommendations on the **Recommendation Engine**. If no match is found, the system either prompts the user to try again or alerts for verification. Upon successful recognition, the system retrieves the student's preferences and song history, tailoring music recommendations accordingly. This module thus adds a layer of personalized access, allowing students to securely log in using face recognition before receiving their customized music suggestions. This DFD Level 1 description is hypothetical and may not be necessary for a music recommendation project. If face recognition isn't part of your intended system design, you can omit this module entirely.

### 3.3.3. DFD Level 1 - Concentration Analysis Module:

In this hypothetical "Concentration Analysis Module" for the Spotify Music Recommendation System, the process begins with the **User** activating the concentration analysis feature while listening to music. The **Concentration Analysis System** utilizes either data from a webcam or sensors (e.g., eye-tracking, heart rate, or EEG devices if available) to monitor signs of the user's concentration levels, like gaze focus, head movements, or physiological responses.

This live data is sent to the **Data Processing Unit**, which uses machine learning models or predefined algorithms to assess concentration indicators. For example, if the user shows high focus, the system might detect signs of concentration and adjust music recommendations to keep them in that state (e.g., suggesting instrumental or ambient music). Conversely, if signs of distraction are detected, the system may suggest more engaging or upbeat music to bring the user back to focus.

The **Concentration Analysis Results** are then sent to the **Recommendation Engine**, which integrates these findings into the recommendation algorithm, adjusting song suggestions based on real-time concentration levels. Finally, the tailored recommendations are displayed back on the **User Interface**, enhancing the user experience by matching music to their current mental state, potentially supporting productivity or relaxation as needed. This concentration analysis module is an optional, advanced feature that could be integrated if your project aims to provide adaptive, mood-based music recommendations. If concentration analysis isn't within the project scope, you may exclude this component

## 3.3 Advantages

The "Spotify Music Recommendation System" offers several advantages by enhancing the user's music discovery experience through personalized recommendations. By leveraging Spotify's vast data resources and audio features, it provides song suggestions that align closely with the user's listening preferences, mood, and trending music. This approach saves users time by reducing the need to search for new music and helps uncover songs and artists they might not otherwise encounter. The system's use of content-based and hybrid recommendation techniques ensures a high degree of personalization while adapting to the user's changing tastes over time. Additionally, the project's use of interactive platforms like Streamlit makes it user-friendly and accessible, enabling seamless interaction for music enthusiasts.

## 3.4 Requirement Specification

The "Spotify Music Recommendation System" requires a range of specifications to ensure its effective functionality and user experience. First, access to Spotify's Web API is necessary for retrieving song metadata, playlists, and audio features, which involves creating a Spotify Developer account and acquiring authentication credentials (client ID and secret). Essential software requirements include Python as the primary programming language, along with specific libraries: **Spotipy** for API interaction, **Pandas** and **NumPy** for data manipulation and analysis, **Scikit-learn** for machine learning utilities (such as scaling and similarity measures), and **Streamlit** to create an interactive web-based interface. The system also requires an internet connection to fetch real-time data from Spotify's API. Additionally, a machine with sufficient processing power and memory is needed to handle the computations and data transformations involved in generating recommendations. These requirements work together to provide a seamless recommendation experience, allowing users to explore and receive suggestions based on various track attributes and personalized filters.

### 3.5.1. Hardware Requirements:

The "Spotify Music Recommendation System" is designed to run on a machine with modest hardware requirements, making it accessible on most standard computing devices. A **computer or laptop** with at least **4GB of RAM** is recommended to efficiently handle data processing and the computations involved in recommendation algorithms, although **8GB of RAM** or higher is ideal for better performance, especially if working with large datasets or multiple simultaneous requests. A **dual-core processor** is sufficient for basic functionality, but a **quad-core processor** or higher is recommended for faster performance during data processing and real-time recommendation generation. A **stable internet connection** is essential to interact with the Spotify API and retrieve data seamlessly. Additionally, **disk storage** of at least **500MB** is required to store temporary files and any local data, though this can vary depending on the scale of data retained locally. This setup ensures that the system runs smoothly, providing users with fast and responsive recommendations.

## Software Requirements:

The "Spotify Music Recommendation System" requires several software components to operate effectively. **Python 3.x** is the primary programming language, as it supports the extensive libraries necessary for data manipulation, machine learning, and web application development. Key libraries include **Spotipy** for accessing the Spotify API, **Pandas** and **NumPy** for data handling and analysis, and **Scikit-learn** for scaling, similarity calculations, and machine learning utilities. Additionally, **Streamlit** is used to create an interactive user interface for a seamless web-based experience, allowing users to enter song details and receive recommendations directly. The system also requires a **Spotify Developer Account** to generate API credentials, enabling access to Spotify's data. To manage package dependencies efficiently, a **package manager like pip** is recommended. For optional development and testing, an **IDE (Integrated Development Environment)** like Visual Studio Code or PyCharm can be used to streamline coding and debugging. With these software requirements, the recommendation system can effectively retrieve, process, and display music suggestions tailored to user preferences.

TechSaksham | SAP | edunet foundation | Microsoft

# CHAPTER 4

# IMPLEMENTATION AND RESULT

| Track Nam | Artists | Album Na | Album ID | Track ID | Popularity | Release Date | Duration ( | Explicit | External U | Danceabi | Energy | Key | Loudness | Mode | Speechine | Acousticn | Instrumer | Liveness | Valence | Tempo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I'm Good ( | David Gue | I'm Good ( | 7M842DN | 4uUG5RX | 94 | 26-08-2022 | 175238 | TRUE | https://op | 0.561 | 0.965 | 7 | -3.673 | 0 | 0.0343 | 0.00383 | 7.07E-06 | 0.371 | 0.304 | 128.04 |
| FE!N (feat | Travis Sco | UTOPIA | 18NOKLk2 | 42Vsgltoc | 93 | 28-07-2023 | 191701 | TRUE | https://op | 0.569 | 0.882 | 3 | -2.777 | 0 | 0.06 | 0.0316 | 0 | 0.142 | 0.201 | 148.038 |
| Boy's a Lia | PinkPanth | Boy's a lia | 6cVfHBcp | 6AQbmUe | 92 | 03-02-2023 | 131013 | TRUE | https://op | 0.696 | 0.809 | 5 | -8.254 | 1 | 0.05 | 0.252 | 0.00013 | 0.248 | 0.857 | 132.962 |
| Quevedo: | Bizarrap, ( | Quevedo: | 4PNqWiJ/ | 2tTmW7R | 92 | 06-07-2022 | 198938 | FALSE | https://op | 0.621 | 0.782 | 2 | -5.548 | 1 | 0.044 | 0.0125 | 0.033 | 0.23 | 0.55 | 128.033 |
| Me Porto | Bad Bunny | Un Veranc | 3RQQmkQ | 6Sq7ltF9C | 91 | 06-05-2022 | 178567 | TRUE | https://op | 0.911 | 0.712 | 1 | -5.105 | 0 | 0.0817 | 0.0901 | 2.68E-05 | 0.0933 | 0.425 | 92.005 |
| Baby Don' | David Gue | Baby Don' | 327tc3Eru | 3BKD1Pw. | 93 | 06-04-2023 | 140018 | FALSE | https://op | 0.602 | 0.91 | 7 | -3.404 | 1 | 0.0308 | 0.00126 | 0.00017 | 0.12 | 0.228 | 127.944 |
| El Mereng | Marshme | El Mereng | 6sU751LC | 51FvjPEGł | 91 | 03-03-2023 | 189357 | FALSE | https://op | 0.775 | 0.677 | 8 | -4.703 | 0 | 0.0442 | 0.0313 | 0.00517 | 0.112 | 0.698 | 124.011 |
| Jimmy Co | Drake, 21 | Honestly, | 3cf4iSSKd | 3F5CgOj3 | 89 | 17-06-2022 | 218365 | TRUE | https://op | 0.529 | 0.673 | 0 | -4.711 | 1 | 0.175 | 0.00031 | 2.41E-06 | 0.093 | 0.366 | 165.921 |
| MONTAGE | S3BZS | MONTAGE | 2HuMAoX | 6njJR3OIp | 89 | 01-05-2023 | 61673 | FALSE | https://op | 0.877 | 0.803 | 9 | -3.195 | 1 | 0.0507 | 0.734 | 0.056 | 0.087 | 0.795 | 130.028 |
| Save Your | The Week | After Hou | 4yP0hdKC | 5QO79kh1 | 89 | 20-03-2020 | 215627 | TRUE | https://op | 0.68 | 0.826 | 0 | -5.487 | 1 | 0.0309 | 0.0212 | 1.24E-05 | 0.543 | 0.644 | 118.051 |
| BABY HELL | Rauw Alej | BABY HELL | 5KDgQ8sk | 2SOvWt6i | 89 | 23-06-2023 | 222462 | TRUE | https://op | 0.773 | 0.892 | 1 | -3.732 | 0 | 0.0483 | 0.166 | 3.10E-06 | 0.429 | 0.842 | 130.007 |
| METAMOR | INTERWOF | METAMOR | 3apQZbgV | 2ksyzVfUO | 89 | 25-11-2021 | 142839 | TRUE | https://op | 0.593 | 0.641 | 7 | -12.727 | 0 | 0.0992 | 0.426 | 0.901 | 0.122 | 0.147 | 175.014 |
| TitÃ- Me P | Bad Bunny | Un Veranc | 3RQQmkQ | 1IHWI5La | 89 | 06-05-2022 | 243717 | TRUE | https://op | 0.65 | 0.715 | 5 | -5.198 | 0 | 0.253 | 0.0993 | 0.00029 | 0.126 | 0.187 | 106.672 |
| Rich Flex | Drake, 21 | Her Loss | 5MS3MvV | 1bDbXMy | 88 | 04-11-2022 | 239360 | TRUE | https://op | 0.561 | 0.52 | 11 | -9.342 | 0 | 0.244 | 0.0503 | 1.86E-06 | 0.355 | 0.424 | 153.15 |
| Escapism. | RAYE, 070 | My 21st C | 3U8n8LzB | 5mHdCZt\ | 88 | 03-02-2023 | 272373 | TRUE | https://op | 0.538 | 0.742 | 2 | -5.355 | 1 | 0.114 | 0.138 | 4.67E-05 | 0.0934 | 0.25 | 96.107 |
| Murder In | Kordhell | Murder In | 68GI09qA | 6qyS9qBy | 88 | 21-01-2022 | 145000 | TRUE | https://op | 0.712 | 0.972 | 10 | -0.514 | 1 | 0.112 | 0.00547 | 7.06E-05 | 0.128 | 0.568 | 119.966 |
| Area Code | Kaliii | Area Code | 6uk3hBYb | 7sliFe6W | 87 | 17-03-2023 | 139326 | TRUE | https://op | 0.823 | 0.388 | 1 | -10.867 | 1 | 0.491 | 0.0187 | 0 | 0.0876 | 0.507 | 154.569 |
| Doja | Central Ce | Doja | 6oECjagks | 3LtpKP5a | 87 | 21-07-2022 | 97393 | TRUE | https://op | 0.911 | 0.573 | 6 | -7.43 | 1 | 0.288 | 0.38 | 0 | 0.403 | 0.972 | 140.04 |
| SICKO MO | Travis Sco | ASTROWO | 41GuZcan | 2xLMifQCj | 87 | 03-08-2018 | 312820 | TRUE | https://op | 0.834 | 0.73 | 8 | -3.714 | 1 | 0.222 | 0.00513 | 0 | 0.124 | 0.446 | 155.008 |
| Lay Low | TiÃ«sto | Lay Low | 0EYKSXXT | 0zKbDrEX | 86 | 06-01-2023 | 153443 | FALSE | https://op | 0.534 | 0.855 | 1 | -4.923 | 0 | 0.183 | 0.0607 | 0.00026 | 0.346 | 0.42 | 122.06 |
| Just Wanr | Lil Uzi Ver | Just Wanr | 2FD6g8bX | 4FyesJzVp | 86 | 17-10-2022 | 123891 | TRUE | https://op | 0.486 | 0.545 | 11 | -7.924 | 1 | 0.0336 | 0.0652 | 0.00474 | 0.0642 | 0.0385 | 150.187 |
| Gato de N | Ã'engo Flc | Gato de N | 2GS2h80[ | 54ELExv5 | 86 | 22-12-2022 | 227013 | TRUE | https://op | 0.892 | 0.662 | 8 | -3.894 | 1 | 0.162 | 0.169 | 1.24E-06 | 0.363 | 0.607 | 93.976 |
| Give It To | Matt Sass | Give It To | 1jbRY71kc | 5ZduaRci | 86 | 22-10-2021 | 102861 | FALSE | https://op | 0.874 | 0.869 | 1 | -5.996 | 0 | 0.0315 | 0.00116 | 0.00238 | 0.164 | 0.726 | 126.027 |
| Push Up - | Creeds | Push Up (I | 3v5BP6gP | 3AjSfp5FD | 86 | 31-03-2023 | 139300 | FALSE | https://op | 0.767 | 0.83 | 7 | -8.78 | 1 | 0.206 | 0.209 | 0.836 | 0.0582 | 0.187 | 75.023 |
| Players | Coi Leray | Players | 4cAAsw7r | 6UN73IYd | 86 | 30-11-2022 | 139560 | TRUE | https://op | 0.954 | 0.516 | 6 | -5.817 | 1 | 0.16 | 0.03 | 7.54E-06 | 0.0504 | 0.624 | 105.001 |
| Bad Mem | MEDUZA, . | Bad Mem | 44aG7QL\ | 3rb0tMq4 | 85 | 22-07-2022 | 148629 | FALSE | https://op | 0.607 | 0.767 | 5 | -6.069 | 0 | 0.0474 | 0.118 | 0 | 0.122 | 0.662 | 123.998 |
| Princess [ | Ice Spice, | Princess [ | 2Q7WBQ7 | 0ZxhtATQ | 85 | 14-04-2023 | 172125 | TRUE | https://op | 0.898 | 0.676 | 9 | -5.196 | 1 | 0.187 | 0.14 | 0 | 0.101 | 0.742 | 147.991 |
| Ferrari | James Hy | Ferrari | 6moZ4sN | 4zN21mb | 85 | 01-04-2022 | 186662 | FALSE | https://op | 0.847 | 0.69 | 1 | -7.877 | 0 | 0.0493 | 0.0127 | 6.00E-05 | 0.0526 | 0.692 | 125.004 |
| Pepas | Farruko | Pepas | 2A5ksnhz | 5fwSHITE\ | 85 | 24-06-2021 | 287120 | TRUE | https://op | 0.762 | 0.766 | 7 | -3.955 | 1 | 0.0343 | 0.00776 | 6.98E-05 | 0.128 | 0.442 | 130.001 |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HIGHEST II | Travis Sco | HIGHEST II | 2uDTi1PIp | 3eekarcy7 | 85 | 04-10-2019 | 175721 | TRUE | https://op | 0.598 | 0.427 | 7 | -8.764 | 0 | 0.0317 | 0.0546 | 5.83E-06 | 0.21 | 0.0605 | 76.469 |
| Runnin | 21 Savage | SAVAGE M | 6wTyGUW | 5SWnsxjh | 85 | 02-10-2020 | 195906 | TRUE | https://op | 0.819 | 0.626 | 10 | -4.574 | 0 | 0.202 | 0.00748 | 0.101 | 0.167 | 0.415 | 143.01 |
| LET GO | Central Ce | LET GO | 1QYPAEk2 | 3zkyus0nj | 85 | 15-12-2022 | 175890 | TRUE | https://op | 0.735 | 0.449 | 2 | -9.933 | 0 | 0.383 | 0.859 | 0 | 0.213 | 0.514 | 146.016 |
| Godzilla (f | Eminem, J | Music To E | 4otkd9As | 7FIWs0pq | 84 | 17-01-2020 | 210800 | TRUE | https://op | 0.808 | 0.745 | 10 | -5.26 | 0 | 0.342 | 0.145 | 0 | 0.292 | 0.829 | 165.995 |
| Freestyle | Lil Baby | Too Hard | 75OAPP0e | 5BbdKBZC | 84 | 01-12-2017 | 162053 | TRUE | https://op | 0.877 | 0.517 | 2 | -5.426 | 0 | 0.0706 | 0.217 | 0 | 0.143 | 0.255 | 119.996 |
| Belly Danc | Imanbek, | Belly Danc | 2npvQTpy | 7fZBQnc0 | 84 | 18-02-2022 | 151475 | FALSE | https://op | 0.845 | 0.797 | 1 | -4.984 | 1 | 0.139 | 0.0582 | 5.57E-06 | 0.167 | 0.422 | 121.985 |
| 10:35 | TiÃ«sto, T | 10:35 | 77wWx9s | 6BePGk3e | 84 | 03-11-2022 | 172253 | FALSE | https://op | 0.696 | 0.793 | 8 | -5.733 | 1 | 0.097 | 0.0683 | 3.78E-06 | 0.18 | 0.698 | 120.003 |
| The Box | Roddy Ric | Please Exc | 52u4anZb | 0nbXyq5T | 84 | 06-12-2019 | 196653 | TRUE | https://op | 0.896 | 0.586 | 10 | -6.687 | 0 | 0.0559 | 0.104 | 0 | 0.79 | 0.642 | 116.971 |
| Tarot | Bad Bunny | Un Veranc | 3RQQmkQ | 41oY4WC | 83 | 06-05-2022 | 237895 | TRUE | https://op | 0.795 | 0.684 | 11 | -3.971 | 0 | 0.0419 | 0.0225 | 0 | 0.658 | 0.419 | 114.011 |
| Knife Talk | Drake, 21 | Certified L | 3SpBlxme | 2BcMwX1 | 83 | 03-09-2021 | 242966 | TRUE | https://op | 0.849 | 0.424 | 5 | -9.579 | 0 | 0.324 | 0.0635 | 0 | 0.0834 | 0.153 | 145.887 |
| Dior | Pop Smok | Meet The | 6d1vGZsr | 79s5XnCN | 83 | 26-07-2019 | 216387 | TRUE | https://op | 0.548 | 0.805 | 7 | -5.732 | 1 | 0.351 | 0.212 | 0.00039 | 0.408 | 0.648 | 142.094 |
| Mood (fea | 24kGoldn, | El Dorado | 270o30h7 | 4jPy3lORL | 83 | 26-03-2021 | 140533 | TRUE | https://op | 0.701 | 0.716 | 7 | -3.671 | 0 | 0.0361 | 0.174 | 0 | 0.324 | 0.732 | 91.007 |
| Call It Lov | Felix Jaeh | Call It Lov | 5c3YGhnf | 5YdnOm5 | 83 | 16-09-2022 | 154561 | FALSE | https://op | 0.616 | 0.841 | 5 | -4.779 | 0 | 0.076 | 0.0559 | 0.00217 | 0.417 | 0.714 | 110.029 |
| Moth To A | Swedish H | Paradise A | 2Dbe9L75 | 0VO8gYVD | 83 | 15-04-2022 | 234000 | FALSE | https://op | 0.553 | 0.659 | 8 | -7.295 | 1 | 0.0391 | 0.0027 | 0 | 0.105 | 0.105 | 120.146 |
| Deep Dow | Alok, Ella I | Deep Dow | 3KpxpdyS | 7MlhUdNJ | 83 | 17-06-2022 | 165753 | FALSE | https://op | 0.687 | 0.818 | 0 | -4.221 | 1 | 0.0778 | 0.0112 | 0 | 0.248 | 0.886 | 125.952 |
| La Jumpa | ArcÃ¡ngel, | La Jumpa | 6LOhj1aK | 5MxFWjui | 83 | 30-11-2022 | 255693 | TRUE | https://op | 0.713 | 0.703 | 8 | -5.769 | 1 | 0.194 | 0.298 | 0 | 0.321 | 0.576 | 123.06 |
| Words (fe | Alesso, Za | Words (fe | 66W7mt0 | 1bgKMxPC | 83 | 22-04-2022 | 142677 | FALSE | https://op | 0.739 | 0.586 | 10 | -5.079 | 0 | 0.0472 | 0.0245 | 0.00025 | 0.308 | 0.444 | 124.026 |
| PUFFIN ON | Future | I NEVER LII | 6tE9Dnp2 | 1qMMYpV | 82 | 29-04-2022 | 172933 | TRUE | https://op | 0.883 | 0.657 | 8 | -5.748 | 1 | 0.305 | 0.0603 | 0 | 0.128 | 0.284 | 124.992 |
| DÃKITI | Bad Bunny | DÃKITI | 43dl8hP5 | 47EiUVwL | 83 | 30-10-2020 | 205090 | TRUE | https://op | 0.731 | 0.573 | 4 | -10.059 | 0 | 0.0544 | 0.401 | 5.22E-05 | 0.113 | 0.145 | 109.928 |
| INDUSTRY | Lil Nas X, J | INDUSTRY | 622NFw5Y | 27NovPIU | 82 | 23-07-2021 | 212000 | TRUE | https://op | 0.736 | 0.704 | 3 | -7.409 | 0 | 0.0615 | 0.0203 | 0 | 0.0501 | 0.894 | 149.995 |
| REACT | Switch Dis | REACT | 3opvHAj8 | 1UPHCP5Y | 82 | 13-01-2023 | 201146 | FALSE | https://op | 0.737 | 0.899 | 6 | -3.189 | 0 | 0.0556 | 0.0329 | 0 | 0.186 | 0.358 | 125.992 |
| The Motto | TiÃ«sto, A | The Motto | 278z9UXJ | 18asYwW | 82 | 04-11-2021 | 164819 | FALSE | https://op | 0.754 | 0.763 | 7 | -4.627 | 0 | 0.0435 | 0.0301 | 2.23E-05 | 0.0901 | 0.464 | 117.953 |
| MONEY | LISA | LALISA | 66OYt73n | 7hU3IHwj | 82 | 10-09-2021 | 168228 | TRUE | https://op | 0.831 | 0.554 | 1 | -9.998 | 0 | 0.218 | 0.161 | 6.12E-05 | 0.152 | 0.396 | 140.026 |
| N95 | Kendrick L | Mr. Moral | 79ONNoS | 0fX4oNGB | 82 | 13-05-2022 | 195950 | TRUE | https://op | 0.79 | 0.67 | 1 | -5.527 | 1 | 0.105 | 0.377 | 2.32E-06 | 0.119 | 0.408 | 139.956 |
| Need to K | Doja Cat | Planet He | 1nAQbHe | 3Vi5XqYrr | 82 | 25-06-2021 | 210560 | TRUE | https://op | 0.664 | 0.609 | 1 | -6.509 | 1 | 0.0707 | 0.304 | 0 | 0.0926 | 0.194 | 130.041 |
| Private La | Don Tolive | Love Sick | 26z5Ilzd1 | 52NGJPcL | 81 | 24-02-2023 | 238253 | TRUE | https://op | 0.843 | 0.669 | 1 | -4.105 | 0 | 0.0639 | 0.0846 | 1.28E-05 | 0.105 | 0.435 | 136.979 |
| MONTERO | Lil Nas X | MONTERO | 6pOiDiuD | 1SC5rEoYI | 82 | 17-09-2021 | 137704 | TRUE | https://op | 0.593 | 0.503 | 8 | -6.725 | 0 | 0.22 | 0.293 | 0 | 0.405 | 0.71 | 178.781 |
| Esta Vida | Marshme | Esta Vida | 5C82uAeA | 5OSGdSxt | 81 | 13-04-2023 | 209136 | FALSE | https://op | 0.755 | 0.676 | 5 | -7.349 | 1 | 0.0377 | 0.0418 | 0.00176 | 0.204 | 0.461 | 123.98 |
| Thunder | Gabry Por | Thunder | 35Q09Y0t | 2USlegnFJ | 81 | 07-05-2021 | 160000 | FALSE | https://op | 0.67 | 0.896 | 1 | -4.673 | 1 | 0.058 | 0.0342 | 3.21E-05 | 0.344 | 0.403 | 101.216 |
| Vegas (Fro | Doja Cat | Vegas (Fro | 2Q5DPv9 | 0hquQWY | 82 | 06-05-2022 | 182907 | TRUE | https://op | 0.801 | 0.601 | 8 | -7.574 | 0 | 0.255 | 0.0777 | 3.23E-05 | 0.145 | 0.74 | 159.969 |

TechSaksham | SAP | edunet foundation | Microsoft

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 Reasor | Nathan D: | 21 Reasor | 118PKNjh | 1RF02Cf8 | 81 | 29-04-2022 | 155253 | FALSE | https://op | 0.621 | 0.785 | 6 | -4.499 | 1 | 0.11 | 0.0165 | 5.92E-06 | 0.0512 | 0.779 | 123.96 |
| Goosebur | Travis Sco | Goosebur | 3SdFuYwy | 5uEYRdElI | 81 | 15-01-2021 | 162803 | TRUE | https://op | 0.841 | 0.593 | 1 | -7.846 | 1 | 0.0379 | 0.418 | 0 | 0.124 | 0.808 | 124.917 |
| WORTH N | TWISTED, | WORTH N | 3iqdAceIK | 65tX8MBo | 81 | 26-01-2023 | 164629 | TRUE | https://op | 0.604 | 0.601 | 2 | -7.887 | 0 | 0.0628 | 0.15 | 0.00058 | 0.168 | 0.156 | 139.88 |
| Nonstop | Drake | Scorpion | 1ATL5GLy | 0TlLq3IA8 | 81 | 29-06-2018 | 238614 | TRUE | https://op | 0.912 | 0.412 | 7 | -8.074 | 1 | 0.123 | 0.0165 | 0.0126 | 0.104 | 0.423 | 154.983 |
| Move You | Ã–wnboss | Move You | 4l9wMVL4 | 6GomT97 | 81 | 29-10-2021 | 157445 | FALSE | https://op | 0.848 | 0.821 | 2 | -5.408 | 0 | 0.0527 | 0.0169 | 0.0004 | 0.0962 | 0.249 | 125.051 |
| LOKERA | Rauw Alej | LOKERA | 4vJcqwIGI | 79HZAZNr | 80 | 25-07-2022 | 195294 | TRUE | https://op | 0.834 | 0.828 | 11 | -2.657 | 0 | 0.0452 | 0.21 | 5.91E-06 | 0.103 | 0.58 | 102.019 |
| In Ha Moo | Ice Spice | In Ha Moo | 0CQzO0dL | 0yUaLqhs | 80 | 06-01-2023 | 129362 | TRUE | https://op | 0.768 | 0.74 | 0 | -6.595 | 1 | 0.336 | 0.696 | 6.81E-06 | 0.23 | 0.532 | 141.059 |
| Where Di | Jax Jones, | Where Di | 5vSLX6JIja | 3sa06xVN | 80 | 04-02-2022 | 177689 | FALSE | https://op | 0.763 | 0.782 | 7 | -4.541 | 0 | 0.0346 | 0.182 | 7.08E-06 | 0.293 | 0.502 | 127.034 |
| La Llevo A | Chris Jedi, | La Llevo A | 0WEtvIRZI | 6DoL1yYlv | 80 | 20-05-2022 | 254920 | TRUE | https://op | 0.795 | 0.845 | 9 | -3.809 | 0 | 0.135 | 0.105 | 1.14E-05 | 0.172 | 0.769 | 170.023 |
| OUT OUT ( | Joel Corry | OUT OUT ( | 5wJb3DBS | 6Dy1jexK | 80 | 13-08-2021 | 162604 | FALSE | https://op | 0.787 | 0.833 | 8 | -4.403 | 1 | 0.0478 | 0.018 | 0.00747 | 0.0374 | 0.796 | 123.97 |
| Element | Pop Smok | Meet The | 4MZnolldo | 57BGVV6v | 80 | 07-02-2020 | 135747 | TRUE | https://op | 0.772 | 0.878 | 2 | -4.22 | 1 | 0.324 | 0.0301 | 2.18E-06 | 0.251 | 0.305 | 61.311 |
| Heaven Ta | Swedish H | Paradise / | 2Dbe9L75 | 3nEHrvNN | 80 | 15-04-2022 | 214071 | FALSE | https://op | 0.665 | 0.738 | 0 | -5.931 | 0 | 0.0379 | 0.0124 | 0.00598 | 0.0923 | 0.0389 | 125.037 |
| Come & G | Juice WRL | Legends N | 6n9DKpO: | 2Y0wPrPC | 79 | 10-07-2020 | 205485 | TRUE | https://op | 0.625 | 0.814 | 0 | -5.181 | 1 | 0.0657 | 0.0172 | 0 | 0.158 | 0.535 | 144.991 |
| Red Ruby | Nicki Mina | Red Ruby | 0zCHOD0z | 4ZYAU4A2 | 79 | 03-03-2023 | 214445 | TRUE | https://op | 0.696 | 0.733 | 1 | -6.181 | 1 | 0.256 | 0.115 | 0 | 0.111 | 0.292 | 98.355 |
| Little Girl | CHINCHILL | Little Girl | 1vTHLsE6I | 5OLcsBO2 | 79 | 21-04-2023 | 188596 | TRUE | https://op | 0.706 | 0.675 | 1 | -6.351 | 0 | 0.237 | 0.177 | 0 | 0.0852 | 0.516 | 159.961 |
| Where Yo | John Sumi | Where Yo | 4bIEy1wD | 4qDpLaFG | 79 | 03-03-2023 | 236000 | FALSE | https://op | 0.56 | 0.832 | 9 | -6.432 | 0 | 0.0363 | 0.00953 | 0.00541 | 0.546 | 0.0818 | 126 |
| ULTRA SOL | PolimÃ¡ W | ULTRA SOL | 7Jsxzl8oZ | 6wtZPYBI | 79 | 16-06-2022 | 322347 | FALSE | https://op | 0.909 | 0.824 | 1 | -4.627 | 1 | 0.0797 | 0.0777 | 0.00387 | 0.0638 | 0.585 | 109.97 |
| Tell Me W | Supermoc | Tell Me W | 6CTjQWx5 | 7jrMFjEq0 | 79 | 02-09-2022 | 171429 | FALSE | https://op | 0.561 | 0.928 | 6 | -5.812 | 0 | 0.0371 | 0.087 | 0.894 | 0.332 | 0.182 | 125.988 |
| Remembe | Becky Hill, | Remembe | 6DHfD3rZ | 4IaAKIq9Z | 79 | 18-06-2021 | 160766 | FALSE | https://op | 0.612 | 0.862 | 8 | -2.903 | 1 | 0.037 | 0.041 | 0 | 0.0907 | 0.354 | 123.849 |
| Way 2 Sex | Drake, Fut | Certified L | 3SpBlxme | 0k1WUmI | 79 | 03-09-2021 | 257605 | TRUE | https://op | 0.803 | 0.597 | 11 | -6.035 | 0 | 0.141 | 0.00062 | 4.50E-06 | 0.323 | 0.331 | 136.008 |
| All By Mys | Alok, Sigal | All By Mys | 3IAmnwQ | 5Hp4xFihc | 78 | 07-10-2022 | 171778 | FALSE | https://op | 0.662 | 0.848 | 0 | -4.338 | 0 | 0.0346 | 0.0932 | 7.62E-06 | 0.241 | 0.773 | 123.041 |
| Go Hard | Lil Baby | Go Hard | 0lvoLgzBq | 1lzJxbARc | 78 | 03-05-2023 | 217610 | TRUE | https://op | 0.849 | 0.581 | 6 | -5.464 | 0 | 0.261 | 0.0178 | 0 | 0.092 | 0.274 | 142.015 |
| Drugs Fro | Mau P | Drugs Fro | 060SvgMz | 0w7JPIp7 | 78 | 16-09-2022 | 235786 | FALSE | https://op | 0.683 | 0.932 | 8 | -10.402 | 1 | 0.0513 | 0.0389 | 0.605 | 0.0615 | 0.553 | 125.002 |
| SHAKE SUI | DaBaby | CALL DA FI | 3oDobVN | 3FhZPYvM | 78 | 05-05-2023 | 124806 | TRUE | https://op | 0.783 | 0.768 | 11 | -8.471 | 0 | 0.157 | 0.00198 | 0 | 0.171 | 0.267 | 143.071 |
| Astronaut | Masked W | Astronaut | 7vus4Q8r | 3Ofmpyhv | 78 | 06-01-2021 | 132780 | TRUE | https://op | 0.778 | 0.695 | 4 | -6.865 | 0 | 0.0913 | 0.175 | 0 | 0.15 | 0.472 | 149.996 |
| When Iâ€ | Alesso, Ka | When Iâ€ | 5itVTi6rl3 | 5902W4u | 78 | 29-12-2021 | 161267 | FALSE | https://op | 0.685 | 0.886 | 0 | -4.179 | 1 | 0.034 | 0.028 | 0 | 0.481 | 0.615 | 125.034 |
| Lie | NF | Perceptio | 1KOmHyN | 07FkzikE6 | 78 | 06-10-2017 | 209213 | FALSE | https://op | 0.513 | 0.661 | 5 | -6.419 | 1 | 0.24 | 0.161 | 0 | 0.246 | 0.179 | 94.791 |
| One in a N | Bebe Rext | One in a N | 65L5VcKG | 3YfGTvsTA | 77 | 04-08-2023 | 160530 | FALSE | https://op | 0.454 | 0.931 | 11 | -4.083 | 1 | 0.0321 | 0.238 | 0.00062 | 0.289 | 0.462 | 137.945 |
| Rainfall (P | Tom Santa | Rainfall (P | 4VanY5i4I | 1M8t1j3K | 77 | 18-02-2022 | 166570 | FALSE | https://op | 0.767 | 0.862 | 5 | -5.464 | 0 | 0.0606 | 0.14 | 0.0092 | 0.252 | 0.509 | 128.039 |
| Another L | Tom Odell | Another L | 1QJtVGIN | 4Et6tUEO | 77 | 13-05-2022 | 185366 | FALSE | https://op | 0.79 | 0.764 | 4 | -4.685 | 0 | 0.0737 | 0.0836 | 0.0871 | 0.132 | 0.473 | 123.046 |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Got It On I | Pop Smok | Shoot For | 7e7t0MCr | 25zlnOAzt | 77 | 03-07-2020 | 164580 | TRUE | https://op | 0.688 | 0.647 | 2 | -7.258 | 1 | 0.19 | 0.00815 | 1.65E-05 | 0.095 | 0.195 | 88.834 |
| Kernkraft | Topic, A7S | Kernkraft | 2NIChqkij | 3kcKIOkQ | 77 | 17-06-2022 | 165800 | FALSE | https://op | 0.623 | 0.727 | 11 | -5.57 | 0 | 0.0562 | 0.184 | 2.02E-05 | 0.309 | 0.4 | 125.975 |
| pushin P ( | Gunna, Fu | DS4EVER | 02uWB8K | 3XOalgus | 77 | 13-01-2022 | 136267 | TRUE | https://op | 0.773 | 0.422 | 1 | -4.572 | 0 | 0.187 | 0.00783 | 0.00693 | 0.129 | 0.488 | 77.502 |
| MONEY OI | Elley Duh | MONEY OI | 1nlaLDMF | 1p0jBDjx | 77 | 20-01-2023 | 145667 | FALSE | https://op | 0.84 | 0.667 | 8 | -7.126 | 1 | 0.0843 | 0.0439 | 1.34E-05 | 0.179 | 0.292 | 120.044 |
| Cynical | twocolors | Cynical | 2FSVbytdz | 2grS8y9w | 77 | 30-06-2023 | 191365 | FALSE | https://op | 0.69 | 0.84 | 11 | -6.947 | 1 | 0.0573 | 0.0268 | 3.06E-05 | 0.363 | 0.493 | 129.935 |
| Motley Cr | Post Malo | Motley Cr | 4tokbQaF | 40uMln2z | 77 | 09-07-2021 | 184213 | TRUE | https://op | 0.797 | 0.631 | 3 | -3.818 | 0 | 0.0786 | 0.0904 | 3.71E-06 | 0.0998 | 0.288 | 129.915 |
| Marianela | HUGEL, M | Marianela | 5As1VmPI | 5bZjb7xKc | 76 | 25-11-2022 | 145766 | FALSE | https://op | 0.828 | 0.893 | 1 | -3.344 | 1 | 0.0496 | 0.033 | 0.00535 | 0.0811 | 0.602 | 124.043 |
| PLAYA DEL | Quevedo, | PLAYA DEL | 1MgW79L | 2t6lxTASa | 76 | 15-12-2022 | 237525 | FALSE | https://op | 0.793 | 0.736 | 7 | -3.254 | 0 | 0.0469 | 0.0822 | 0 | 0.109 | 0.656 | 112.993 |
| Levitating | Dua Lipa, | Levitating | 04m06Kh | 463CkQjx | 76 | 01-10-2020 | 203064 | FALSE | https://op | 0.702 | 0.825 | 6 | -3.787 | 0 | 0.0601 | 0.00883 | 0 | 0.0674 | 0.915 | 102.977 |
| family tie | Baby Keer | family tie | 3HqmX8h | 7Bpx2vsW | 76 | 27-08-2021 | 252070 | TRUE | https://op | 0.711 | 0.611 | 1 | -5.453 | 1 | 0.329 | 0.00575 | 0 | 0.231 | 0.144 | 134.14 |
| Lionheart | Joel Corry | Lionheart | 68U7cani | 5vlzH0ps6 | 76 | 21-10-2022 | 186689 | FALSE | https://op | 0.628 | 0.967 | 8 | -2.43 | 1 | 0.0538 | 0.0217 | 0.00157 | 0.336 | 0.349 | 125.982 |

```python
import requests
import base64

CLIENT_ID = '3718b3ee547542b48915cd97527ae072'
CLIENT_SECRET = '873b12eaa6934c8a97a10e9a3e59853f'

client_credentials = f"{CLIENT_ID}:{CLIENT_SECRET}"
client_credentials_base64 = base64.b64encode(client_credentials.encode())

token_url = 'https://accounts.spotify.com/api/token'
headers = {
    'Authorization': f'Basic {client_credentials_base64.decode()}'
}
data = {
    'grant_type': 'client_credentials'
}
response = requests.post(token_url, data=data, headers=headers)

if response.status_code == 200:
    access_token = response.json()['access_token']
    print("Access token obtained successfully.")
    print(f"Access Token: {access_token}")
else:
    print("Error obtaining access token.")
    print(f"Response Code: {response.status_code}")
    print(f"Response Content: {response.text}")
    exit()
```

```
Access token obtained successfully.
Access Token: BQAJZ7gNVAkS2yyAaDj725Ld59z2wYYwik5bYMPmm_oGzLaPo88wJZveMbipPSjZfTnVUYuSeosgXE96UTbuRWbKC7laL6_Dp90WFp6f5DmkOYY5W7s
```

TechSaksham SAP edunet foundation Microsoft

[2]
```
%pip install spotipy
%pip install pandas
%pip install numpy
%pip install streamlit
```

```
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Collecting streamlit
  Downloading streamlit-1.40.0-py2.py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (from streamlit) (1.4)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (24.1)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (10.4.0)
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.20.3)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (17.0.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (13.9.3)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.12.2)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.1.43)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-packages (from streamlit) (6.3.3)
Collecting watchdog<6,>=2.1.5 (from streamlit)
  Downloading watchdog-5.0.3-py3-none-manylinux2014_x86_64.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.9/41.9 kB 1.6 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (3.1.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (4.23.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (0.12.1)
```

✓ 0s completed at 5:16PM

```
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.11)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27->streamlit) (2024.8.30)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit) (2.18.0)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair<6,>=4.0->streamlit) (3.0.2)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.20.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.16.0)
Downloading streamlit-1.40.0-py2.py3-none-any.whl (8.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 55.0 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 83.5 MB/s eta 0:00:00
Downloading watchdog-5.0.3-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.3/79.3 kB 7.9 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.40.0 watchdog-5.0.3
```

```python
import pandas as pd
import spotipy
from spotipy.oauth2 import SpotifyOAuth

def get_trending_playlist_data(playlist_id, access_token):
    sp = spotipy.Spotify(auth=access_token)

    playlist_tracks = sp.playlist_tracks(playlist_id, fields='items(track(id,name,artists,album(id,name)))')

    music_data = []
    for track_info in playlist_tracks.get('items', []):
        track = track_info.get('track')
        if not track:  # Check if track info is available
            continue

        track_name = track.get('name')
        artists = ', '.join([artist['name'] for artist in track.get('artists', [])])
        album_name = track.get('album', {}).get('name')
        album_id = track.get('album', {}).get('id')
        track_id = track.get('id')

        audio_features = sp.audio_features(track_id)[0] if track_id else None

        try:
            album_info = sp.album(album_id) if album_id else None
            release_date = album_info.get('release_date') if album_info else None
        except Exception as e:
            print(f"Error fetching album info: {e}")
            release_date = None
```

```
        try:
            track_info = sp.track(track_id) if track_id else None
            popularity = track_info.get('popularity') if track_info else None
        except Exception as e:
            print(f"Error fetching track info: {e}")
            popularity = None

        track_data = {
            'Track Name': track_name,
            'Artists': artists,
            'Album Name': album_name,
            'Album ID': album_id,
            'Track ID': track_id,
            'Popularity': popularity,
            'Release Date': release_date,
            'Duration (ms)': audio_features.get('duration_ms') if audio_features else None,
            'Explicit': track_info.get('explicit', None) if track_info else None,
            'External URLs': track_info.get('external_urls', {}).get('spotify', None) if track_info else None,
            'Danceability': audio_features.get('danceability') if audio_features else None,
            'Energy': audio_features.get('energy') if audio_features else None,
            'Key': audio_features.get('key') if audio_features else None,
            'Loudness': audio_features.get('loudness') if audio_features else None,
            'Mode': audio_features.get('mode') if audio_features else None,
            'Speechiness': audio_features.get('speechiness') if audio_features else None,
            'Acousticness': audio_features.get('acousticness') if audio_features else None,
            'Instrumentalness': audio_features.get('instrumentalness') if audio_features else None,
            'Liveness': audio_features.get('liveness') if audio_features else None,
            'Valence': audio_features.get('valence') if audio_features else None,
            'Tempo': audio_features.get('tempo') if audio_features else None,
        }

        music_data.append(track_data)

    df = pd.DataFrame(music_data)

    return df
```

```python
playlist_id = '37i9dQZF1DX76Wlfdnj7AP'

music_df = get_trending_playlist_data(playlist_id, access_token)

print(music_df)
```

```
96       Astronaut In The Ocean  7vus4Q8r5DS2D11JClxEsA
97  Another Love (Tiësto Remix)  1QltVGINGbK94CKgUsY2Ga
98                   Tarantella  0nxNoLQWFwsRqTfLh7R0sq
99            MONEY ON THE DASH  1nlaLDMPSzXL8G5LPBDhwd

                  Track ID  Popularity Release Date  Duration (ms)  Explicit  \
0   5vNRhkKd0yEAg8suGBpjeY          97   2024-10-18         169917     False
1   2PnlsTsOTLE5jnBnNe2K0A          90   2024-09-05         198428      True
2   6AI3ezQ4o3HUoP6Dhudph3          88   2024-05-04         274192      True
3   5RePVWy39tLpHH0wwXgBsK          86   2024-10-28         165413      True
4   0QA00aPt3BV10qoMIs3meW          86   2024-07-09         198667      True
..                     ...         ...          ...            ...       ...
95  0w7JPlp7eEQI2EKW3ayXrv          69   2022-09-16         235786     False
96  3Ofmpyhv5UAQ70mENzB277          69   2021-01-06         132780      True
97  4Et6tUEO7biKx2EfJXpNj1          69   2022-05-13         185366     False
98  4pBuhoBxgwBpkkOhfT6p6N          69   2024-03-29         145554     False
99  1p0jBDjxORjYNJyAphBRpE          68   2023-01-20         145667     False

                                        External URLs  ...  Energy  Key  \
0   https://open.spotify.com/track/5vNRhkKd0yEAg8s...  ...   0.783    0
1   https://open.spotify.com/track/2PnlsTsOTLE5jnB...  ...   0.872    7
2   https://open.spotify.com/track/6AI3ezQ4o3HUoP6...  ...   0.472    1
3   https://open.spotify.com/track/5RePVWy39tLpHH0...  ...   0.878    1
4   https://open.spotify.com/track/0QA00aPt3BV10qo...  ...   0.745    4
..                                                ...  ...     ...  ...
95  https://open.spotify.com/track/0w7JPlp7eEQI2EK...  ...   0.928    8
96  https://open.spotify.com/track/3Ofmpyhv5UAQ70m...  ...   0.695    4
97  https://open.spotify.com/track/4Et6tUEO7biKx2E...  ...   0.764    4
98  https://open.spotify.com/track/4pBuhoBxgwBpkkO...  ...   0.861    9
99  https://open.spotify.com/track/1p0jBDjxORjYNJy...  ...   0.667    8

    Loudness  Mode  Speechiness  Acousticness  Instrumentalness  Liveness  \
0     -4.477     0       0.2600       0.02830          0.000000    0.3550
1     -3.344     1       0.0336       0.01560          0.000000    0.1210
2     -7.001     1       0.0776       0.01070          0.000000    0.1410
3     -4.099     1       0.3100       0.00289          0.000000    0.3810
4     -3.202     0       0.1610       0.02350          0.000000    0.3630
..       ...   ...          ...           ...               ...       ...
95   -10.380     1       0.0527       0.03160          0.727000    0.0576
96    -6.865     0       0.0913       0.17500          0.000000    0.1500
97    -4.685     0       0.0737       0.08360          0.087100    0.1320
98    -5.509     1       0.0781       0.13900          0.000026    0.3320
99    -7.126     1       0.0843       0.04390          0.000013    0.1790

    Valence    Tempo
0     0.939  149.027
1     0.806  184.115
2     0.214  101.061
3     0.514   82.557
4     0.262  180.098
..      ...      ...
95    0.631  124.986
96    0.472  149.996
97    0.473  123.846
98    0.426  140.072
99    0.292  120.044

[100 rows x 21 columns]
```

```
music_df
```

| | Track Name | Artists | Album Name | Album ID | Track ID | Popularity | Release Date | Duration (ms) | Explicit |
|---|---|---|---|---|---|---|---|---|---|
| 0 | APT. | ROSÉ, Bruno Mars | APT. | 2IYQwwgxgOIn7t3iF6ufFD | 5vNRhkKd0yEAg8suGBpjeY | 97 | 2024-10-18 | 169917 | False |
| 1 | The Emptiness Machine | Linkin Park | The Emptiness Machine | 6W0Gabv5f3ugnckc6YgfJQ | 2PnlsTsOTLE5jnBnNe2K0A | 90 | 2024-09-05 | 190428 | True |
| 2 | Not Like Us | Kendrick Lamar | Not Like Us | 5JjnoGJyOxfSZUZtk2rRwZ | 6AI3ezQ4o3HUoP6Dhudph3 | 88 | 2024-05-04 | 274192 | True |
| 3 | Rah Tah Tah | Tyler, The Creator | CHROMAKOPIA | 0U28P0QVB1QRxpqp5IHOlH | 5RePVWy39tLpHH0WwXgBsK | 86 | 2024-10-28 | 165413 | True |
| 4 | Big Dawgs | Hanumankind, Kalmi | Big Dawgs | 6Yw4204wbgmpsGTzjXBhYD | 0OA00aPt3BV10qeMIs3meW | 86 | 2024-07-09 | 190667 | True |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | Drugs From Amsterdam | Mau P | Drugs From Amsterdam | 060SvgMzLKrNzpvVLK5gSo | 0w7JPlp7eEQI2EKW3ayXrv | 69 | 2022-09-16 | 235786 | False |
| 96 | Astronaut In The Ocean | Masked Wolf | Astronaut In The Ocean | 7vus4Q8r5DS2Dl1JClxEsA | 3Ofmpyhv5UAQ70mENzB277 | 69 | 2021-01-06 | 132780 | True |
| 97 | Another Love - Tiësto Remix | Tom Odell, Tiësto | Another Love (Tiësto Remix) | 1QltVGlNGbK94CKgUsY2Ga | 4Et6tUEO7biKxZEfJXpNj1 | 69 | 2022-05-13 | 185366 | False |
| 98 | Tarantella | Gabry Ponte, KEL | Tarantella | 0nxNoLQWFwsRqTfLh7R0sq | 4pBuhoBxgwBpkkOhfT6p6N | 69 | 2024-03-29 | 145554 | False |
| 99 | MONEY ON THE DASH | Elley Duhé, Whethan | MONEY ON THE DASH | 1nlaLDMPSzXL8G5LPBDhwd | 1p0jBDjxORjYNJyAphBRpE | 68 | 2023-01-20 | 145667 | False |

| | External URLs | ... | Energy | Key | Loudness | Mode | Speechiness | Acousticness | Instrumentalness | Liveness | Valence | Tempo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | https://open.spotify.com/track/5vNRhkKd0yEAg8s... | ... | 0.783 | 0 | -4.477 | 0 | 0.2600 | 0.02830 | 0.000000 | 0.3550 | 0.939 | 149.027 |
| 1 | https://open.spotify.com/track/2PnlsTsOTLE5jnB... | ... | 0.872 | 7 | -3.344 | 1 | 0.0336 | 0.01560 | 0.000000 | 0.1210 | 0.806 | 184.115 |
| 2 | https://open.spotify.com/track/6AI3ezQ4o3HUoP6... | ... | 0.472 | 1 | -7.001 | 1 | 0.0776 | 0.01070 | 0.000000 | 0.1410 | 0.214 | 101.061 |
| 3 | https://open.spotify.com/track/5RePVWy39tLpHH0... | ... | 0.878 | 1 | -4.099 | 1 | 0.3100 | 0.00289 | 0.000000 | 0.3810 | 0.514 | 82.557 |
| 4 | https://open.spotify.com/track/0OA00aPt3BV10qe... | ... | 0.745 | 4 | -3.202 | 0 | 0.1610 | 0.02350 | 0.000000 | 0.3630 | 0.262 | 180.098 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | https://open.spotify.com/track/0w7JPlp7eEQI2EK... | ... | 0.928 | 8 | -10.380 | 1 | 0.0527 | 0.03160 | 0.727000 | 0.0576 | 0.631 | 124.986 |
| 96 | https://open.spotify.com/track/3Ofmpyhv5UAQ70m... | ... | 0.695 | 4 | -6.865 | 0 | 0.0913 | 0.17500 | 0.000000 | 0.1500 | 0.472 | 149.996 |
| 97 | https://open.spotify.com/track/4Et6tUEO7biKxZE... | ... | 0.764 | 4 | -4.685 | 0 | 0.0737 | 0.08360 | 0.087100 | 0.1320 | 0.473 | 123.046 |
| 98 | https://open.spotify.com/track/4pBuhoBxgwBpkkO... | ... | 0.861 | 9 | -5.509 | 1 | 0.0781 | 0.13900 | 0.000026 | 0.3320 | 0.426 | 140.072 |
| 99 | https://open.spotify.com/track/1p0jBDjxORjYNJy... | ... | 0.667 | 8 | -7.126 | 1 | 0.0843 | 0.04390 | 0.000013 | 0.1790 | 0.292 | 120.044 |

100 rows × 21 columns

```
print(music_df)
```

```
                   Track Name                Artists  \
0                        APT.       ROSÉ, Bruno Mars
1       The Emptiness Machine            Linkin Park
2                 Not Like Us         Kendrick Lamar
3                 Rah Tah Tah     Tyler, The Creator
4                   Big Dawgs     Hanumankind, Kalmi
..                        ...                    ...
95       Drugs From Amsterdam                  Mau P
96     Astronaut In The Ocean            Masked Wolf
97  Another Love - Tiësto Remix      Tom Odell, Tiësto
98                 Tarantella       Gabry Ponte, KEL
99          MONEY ON THE DASH   Elley Duhé, Whethan

                   Album Name                Album ID  \
0                        APT.  2IYQwwgxgOIn7t3iF6ufFD
1       The Emptiness Machine  6W0Gabv5f3ugnckc6YgfJQ
2                 Not Like Us  5JjnoGJyOxfS2U2tk2rRw2
3                 CHROMAKOPIA  0U28P0QVB1QRxpqp5IHOlH
4                   Big Dawgs  6Yw4204wbgmpsGTzjXBhYD
..                        ...                     ...
95       Drugs From Amsterdam  060SvgMzLKrNzpvVLK5gSo
96     Astronaut In The Ocean  7vus4Q8r5DS2Dl1JClxEsA
97  Another Love (Tiësto Remix)  1QltVGlNGbK94CKgUsY2Ga
98                 Tarantella  0nxNoLQWFwsRqTfLh7R0sq
99          MONEY ON THE DASH  1nlaLDMPS2XL8G5LPBDhwd

                  Track ID  Popularity Release Date  Duration (ms)  Explicit  \
0   5vNRhkKd0yEAg8suGBpjeY          97   2024-10-18         169917     False
1   2PnlsTsOTLE5jnBnNe2K0A          90   2024-09-05         190428      True
2   6AI3ezQ4o3HUoP6Dhudph3          88   2024-05-04         274192      True
3   5RePVWy39tLpHH0WwXgBsK          86   2024-10-28         165413      True
4   0OA00aPt3BV10qeMIs3meW          86   2024-07-09         190667      True
..                     ...         ...          ...            ...       ...
95  0w7JPlp7eEQI2EKW3ayXrv          69   2022-09-16         235786     False
96  3OFmpyhv5UAQ70mENzB277          69   2021-01-06         132780      True
97  4Et6tUEO7biKxZEfJXpNj1          69   2022-05-13         185366     False
98  4pBuhoBxgwBpkkOhfT6p6N          69   2024-03-29         145554     False
99  1p0jBDjxORjYNJyAphBRpE          68   2023-01-20         145667     False

                                        External URLs  ...  Energy  Key  \
0   https://open.spotify.com/track/5vNRhkKd0yEAg8s...  ...   0.783    0
1   https://open.spotify.com/track/2PnlsTsOTLE5jnB...  ...   0.872    7
2   https://open.spotify.com/track/6AI3ezQ4o3HUoP6...  ...   0.472    1
3   https://open.spotify.com/track/5RePVWy39tLpHH0...  ...   0.878    1
4   https://open.spotify.com/track/0OA00aPt3BV10qe...  ...   0.745    4
..                                                ...  ...     ...  ...
95  https://open.spotify.com/track/0w7JPlp7eEQI2EK...  ...   0.928    8
96  https://open.spotify.com/track/3OFmpyhv5UAQ70m...  ...   0.695    4
97  https://open.spotify.com/track/4Et6tUEO7biKx2E...  ...   0.764    4
98  https://open.spotify.com/track/4pBuhoBxgwBpkkO...  ...   0.861    9
99  https://open.spotify.com/track/1p0jBDjxORjYNJy...  ...   0.667    8

    Loudness  Mode  Speechiness  Acousticness  Instrumentalness  Liveness  \
0     -4.477     0       0.2600       0.02830          0.000000    0.3550
1     -3.344     1       0.0336       0.01560          0.000000    0.1210
2     -7.001     1       0.0776       0.01070          0.000000    0.1410
3     -4.099     1       0.3100       0.00289          0.000000    0.3810
4     -3.202     0       0.1610       0.02350          0.000000    0.3630
```

```
music_df.isnull().sum()
```

| | 0 |
|---|---|
| Track Name | 0 |
| Artists | 0 |
| Album Name | 0 |
| Album ID | 0 |
| Track ID | 0 |
| Popularity | 0 |
| Release Date | 0 |
| Duration (ms) | 0 |
| Explicit | 0 |
| External URLs | 0 |
| Danceability | 0 |
| Energy | 0 |
| Key | 0 |
| Loudness | 0 |
| Mode | 0 |
| Speechiness | 0 |
| Acousticness | 0 |
| Instrumentalness | 0 |
| Liveness | 0 |
| Valence | 0 |
| Tempo | 0 |

dtype: int64

```
[30] import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
     from datetime import datetime
     from sklearn.metrics.pairwise import cosine_similarity
```

```
[31] data = music_df
```

```
[32] def calculate_weighted_popularity(release_date):
         release_date = datetime.strptime(release_date, '%Y-%m-%d')

         time_span = datetime.now() - release_date

         weight = 1 / (time_span.days + 1)
         return weight
```

```
[33] def calculate_weighted_popularity(release_date):
         if release_date is None:
             return 0

         try:
             release_date = datetime.strptime(release_date, '%Y-%m-%d')
         except ValueError:
             return 0

         time_span = datetime.now() - release_date

         weight = 1 / (time_span.days + 1)
         return weight
```

```
[34] data['Weighted Popularity'] = data['Release Date'].apply(calculate_weighted_popularity)
```

```
scaler = MinMaxScaler()

music_features = music_df[['Danceability', 'Energy', 'Key', 'Loudness',
                           'Mode', 'Speechiness', 'Acousticness',
                           'Instrumentalness', 'Liveness', 'Valence',
                           'Tempo']].fillna(0).values

music_features_scaled = scaler.fit_transform(music_features)

music_features_scaled_df = pd.DataFrame(music_features_scaled,
                                        columns=['Danceability', 'Energy', 'Key',
                                                 'Loudness', 'Mode', 'Speechiness',
                                                 'Acousticness', 'Instrumentalness',
                                                 'Liveness', 'Valence', 'Tempo'])
```

```
[36] music_features_scaled_df
```

```
music_features_scaled_df
```

1 to 25 of 100 entries Filter

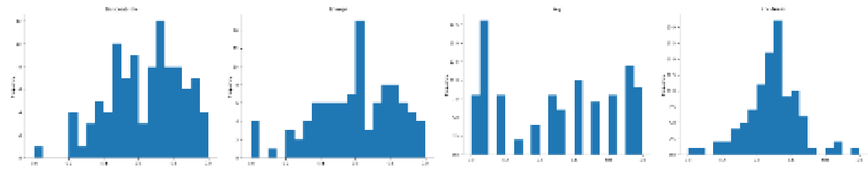
| index | Danceability | Energy | Key | Loudness | Mode | Speechiness | Acousticness | Instrumentalness | Liveness | Valence | Tempo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.7287581699346407 | 0.6407599309153714 | 0.0 | 0.5488402482848742 | 0.0 | 0.5060658578856153 | 0.03302256831187706 | 0.0 | 0.4278574247007759 | 0.9646491697911086 | 0.7142764079345949 |
| 1 | 0.22058823529411764 | 0.7944732297063904 | 0.6363636363636364 | 0.641375367526952 | 1.0 | 0.01559792027729636 | 0.01804072936782538 | 0.0 | 0.12008417729843482 | 0.8221746116764864 | 1.0 |
| 2 | 0.9264705882352943 | 0.10362694300518138 | 0.09090909090909091 | 0.34269846455406716 | 1.0 | 0.11091854419410746 | 0.01226033481460859 | 0.0 | 0.14638958305931865 | 0.1880021424745581 | 0.3236865248688968 |
| 3 | 0.53921568627451 | 0.8048359240069086 | 0.09090909090909091 | 0.5797125122508984 | 1.0 | 0.6143847487001733 | 0.003047093700195708 | 0.0 | 0.46205445218992497 | 0.5093733261917515 | 0.1730073938959643 |
| 4 | 0.2173202614379085 | 0.5751295336787566 | 0.36363636363636365 | 0.6529728846782097 | 0.0 | 0.2915944540727903 | 0.027360140994440205 | 0.0 | 0.43837958700512947 | 0.2394215318693091 | 0.9672893391094752 |
| 5 | 0.7271241830065361 | 0.44041450777202085 | 0.6363636363636364 | 0.3736524011760862 | 1.0 | 0.15576256499133448 | 0.016881057010026034 | 0.3239740820734341 | 0.06102854136525055 | 0.6207820032137118 | 0.5594931761180417 |
| 6 | 0.5718954248366014 | 0.5215889464594129 | 0.09090909090909091 | 0.5374060764456059 | 0.0 | 0.009748700173310233 | 0.04387555400363104 | 0.0 | 0.2660791792713402 | 0.552222817354044 | 0.5591430246571774 |
| 7 | 0.37745098039215697 | 0.8721934369602765 | 1.0 | 0.6884188173799412 | 0.0 | 0.19627383015597924 | 0.06015503254126199 | 0.0 | 0.6027883730106536 | 0.4954472415640065 | 0.3230839386339207 |
| 8 | 0.23692810457516333 | 0.7202072538860104 | 0.9090909090909092 | 0.6036426004573667 | 0.0 | 0.08535528596187175 | 0.01992820514030433 | 0.009892008639308855 | 0.1805866105484677 | 0.4997321906802357 | 0.6342382984267614 |
| 9 | 0.5016339869281047 | 0.765112262521589 | 0.6363636363636364 | 0.5175596210388761 | 1.0 | 0.0 | 0.008567960334696643 | 0.0 | 0.2831776930159147 | 0.7418318157471879 | 0.542938340770659 |
| 10 | 0.5130718954248367 | 0.3834196891191711 | 0.9090909090909092 | 0.6066644887291734 | 0.0 | 0.016681109185441932 | 0.1860260731184521 | 4.08207343412527e-06 | 0.08036301459950018 | 0.33583288698446706 | 0.8037686068857692 |
| 11 | 0.3758169934640524 | 0.9550949913644214 | 0.6363636363636364 | 0.6145050637046716 | 0.0 | 0.017114384748700175 | 0.004155985716527092 | 7.63498920086393e-06 | 0.448901749309483 | 0.2844134975897161 | 0.5433780658610468 |
| 12 | 0.5049019607843138 | 0.24179620034542326 | 0.18181818181818182 | 0.45066971577915704 | 1.0 | 0.22227036395147312 | 0.02500079627884151 | 0.0 | 0.11745363672234643 | 0.07873594001071238 | 0.6810038760952413 |
| 13 | 0.7173202614379087 | 0.7720207253886012 | 0.9090909090909092 | 0.6382718065991506 | 0.0 | 0.5385615251299828 | 0.42667923410951847 | 0.0 | 0.8461133762988293 | 0.6443492233529727 | 0.5574655548679197 |
| 14 | 0.7728758169934643 | 0.9412780656303973 | 0.0 | 0.4692094086899705 | 1.0 | 0.016464471403812832 | 0.1470968853110737 | 0.018790496760259175 | 0.11087728528212548 | 0.9892878272094269 | 0.5270512361160873 |
| 15 | 0.5686274509803922 | 0.7132987910189984 | 0.0 | 0.46635086573015344 | 1.0 | 0.009748700173310233 | 0.02464689457150171 | 1.3066954643628508e-05 | 0.6751282388530843 | 0.648634172469202 | 0.46202078108204947 |
| 16 | 0.7549019607843138 | 0.6338514680483595 | 0.8181818181818182 | 0.5380594576935641 | 0.0 | 0.06867417677642981 | 0.32876642841217285 | 0.00033585313174946 | 0.373931342890964 | 0.6946973754686664 | 0.6570714308980163 |
| 17 | 0.5016339869281047 | 0.526770293609672 | 0.7272727272727273 | 0.44046063377981043 | 1.0 | 0.024046793760831887 | 0.0008175129439549462 | 1.93304535637149e-06 | 0.10956201499408127 | 0.4108194965184789 | 0.6562734112895345 |
| 18 | 0.843137254901961 | 0.3626943005181348 | 0.09090909090909091 | 0.36344331917673955 | 0.0 | 0.13843154246100522 | 0.14001885116427762 | 0.0 | 0.3357885045376824 | 0.194429566148902 | 0.5593466010879125 |
| 19 | 0.7892156862745098 | 0.4559585492227981 | 1.0 | 0.533077425677883 | 0.0 | 0.44324090121317156 | 0.008001717602952957 | 1.3822894168466521e-05 | 0.11745363672234643 | 0.29298339582217464 | 0.8200140060584344 |
| 20 | 0.897058823529412 | 0.02763385146804842 | 0.8181818181818182 | 0.23823913753675263 | 1.0 | 0.3890814558058925 | 0.17540902189825797 | 5.4535637149028076e-05 | 0.11087728528212548 | 0.3154793786823782 | 0.31510374254910267 |
| 21 | 0.9885620915032683 | 0.8203799654576859 | 0.8181818181818182 | 0.6890721986278994 | 0.0 | 0.09077123050259966 | 0.03408427343389647 | 2.0302375809935205e-06 | 0.0374852032092595 | 0.9110873058382432 | 0.5349337155141527 |
| 22 | 0.4803921568627453 | 0.5544041450777203 | 0.18181818181818182 | 0.19928128062724593 | 1.0 | 0.048526863084922024 | 0.006692281285795684 | 0.0010615550755939524 | 0.4541628304616598 | 0.2201392608462775 | 0.9020797368163904 |
| 23 | 0.44281045751633985 | 0.8601036269430055 | 0.6363636363636364 | 0.6384750081672655 | 1.0 | 0.009532062391681119 | 0.001124227756982776 | 0.00018790496760259177 | 0.11876890701039061 | 0.20299946438136052 | 0.542598332367024 |
| 24 | 0.4950980392156864 | 0.6027633851468049 | 0.4545454545454546 | 0.5263802678863116 | 1.0 | 0.045060658578856154 | 0.005972681147538083 | 0.0 | 0.6830198605813494 | 0.7996786288162828 | 0.5590371649131949 |

Show 25 per page

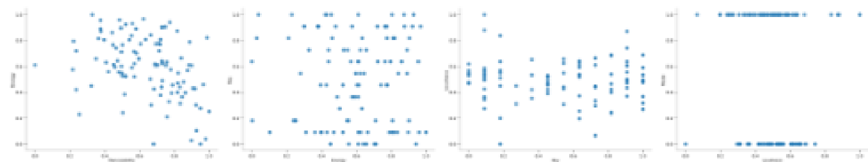
1 2 3 4

Like what you see? Visit the data table notebook to learn more about interactive tables.

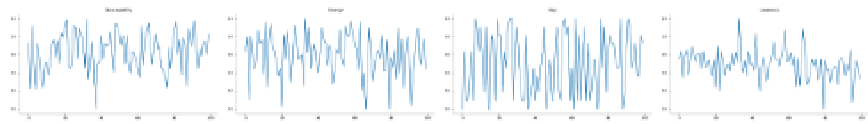
**Distributions**



**2-d distributions**



**Values**



Next steps: Generate code with `music_features_scaled_df` | View recommended plots | New interactive sheet

```python
def content_based_recommendations(input_song_name, num_recommendations=5):
    if input_song_name not in music_df['Track Name'].values:
        print(f"'{input_song_name}' not found in the dataset. Please enter a valid song name.")
        return

    input_song_index = music_df[music_df['Track Name'] == input_song_name].index[0]

    similarity_scores = cosine_similarity([music_features_scaled[input_song_index]],
                                          music_features_scaled)

    similar_song_indices = similarity_scores.argsort()[0][::-1][1:num_recommendations + 1]

    content_based_recommendations = music_df.iloc[similar_song_indices][['Track Name',
                                                                         'Artists',
                                                                         'Album Name',
                                                                         'Release Date',
                                                                         'Popularity',
                                                                         'Track ID']]

    return content_based_recommendations
```

```python
[38] def hybrid_recommendations(input_song_name, num_recommendations=5, alpha=0.5):
    if input_song_name not in music_df['Track Name'].values:
        print(f"'{input_song_name}' not found in the dataset. Please enter a valid song name.")
        return

    content_based_rec = content_based_recommendations(input_song_name, num_recommendations)

    if content_based_rec.empty:
        print("No content-based recommendations found.")
        return

    popularity_score = music_df.loc[music_df['Track Name'] == input_song_name, 'Popularity'].values[0]

    weighted_popularity_score = popularity_score * calculate_weighted_popularity(
        music_df.loc[music_df['Track Name'] == input_song_name, 'Release Date'].values[0]
    )

    input_song_data = pd.DataFrame([{
        'Track Name': input_song_name,
        'Artists': music_df.loc[music_df['Track Name'] == input_song_name, 'Artists'].values[0],
        'Album Name': music_df.loc[music_df['Track Name'] == input_song_name, 'Album Name'].values[0],
        'Release Date': music_df.loc[music_df['Track Name'] == input_song_name, 'Release Date'].values[0],
        'Popularity': weighted_popularity_score
    }])

    hybrid_recommendations = pd.concat([content_based_rec, input_song_data], ignore_index=True)

    hybrid_recommendations = hybrid_recommendations.sort_values(by='Popularity', ascending=False)

    hybrid_recommendations = hybrid_recommendations[hybrid_recommendations['Track Name'] != input_song_name]

    return hybrid_recommendations
```

```
input_song_name = input("Enter a song name : ")
recommendations = hybrid_recommendations(input_song_name, num_recommendations=5)

print(f"Hybrid recommended songs for '{input_song_name}':")
print(recommendations)
```

```
Enter a song name : I'm Good (Blue)
Hybrid recommended songs for 'I'm Good (Blue)':
                    Track Name                                     Artists  \
1                      Disease                                   Lady Gaga
2                      KEEP UP                                     Odetari
3  It's Not Right But It's Okay                          Mr. Belt & Wezol
4          Rainfall (Praise You)                                 Tom Santa
0                        REACT  Switch Disco, Ella Henderson, Robert Miles

                    Album Name Release Date  Popularity  \
1                      Disease   2024-10-25        85.0
2          KEEP UP // FROSTBITE   2024-07-17        84.0
3  It's Not Right But It's Okay   2024-02-23        78.0
4          Rainfall (Praise You)   2022-02-18        71.0
0                        REACT   2023-01-13        69.0

                Track ID
1  19KlZwqlT3fguP2BeHF1Q1
2  2yR2sziCF4WEs3klW1F38d
3  5OFVzqSeFxGpvDGyHvVeLj
4  1M8t1j3Kv2qp97bdq5q4Vl
0  1UPHCP5YeVfele4DMbdGyi
```

```
[42] content_based_recommendations(input_song_name, num_recommendations=5)
```

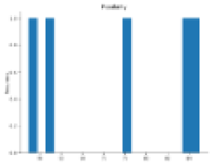
1 to 5 of 5 entries Filter

| index | Track Name | Artists | Album Name | Release Date | Popularity | Track ID |
|---|---|---|---|---|---|---|
| 94 | REACT | Switch Disco, Ella Henderson, Robert Miles | REACT | 2023-01-13 | 69 | 1UPHCP5YeVfele4DMbdGyi |
| 7 | Disease | Lady Gaga | Disease | 2024-10-25 | 85 | 19KlZwqlT3fguP2BeHF1Q1 |
| 8 | KEEP UP | Odetari | KEEP UP // FROSTBITE | 2024-07-17 | 84 | 2yR2sziCF4WEs3klW1F38d |
| 43 | It's Not Right But It's Okay | Mr. Belt & Wezol | It's Not Right But It's Okay | 2024-02-23 | 78 | 5OFVzqSeFxGpvDGyHvVeLj |
| 86 | Rainfall (Praise You) | Tom Santa | Rainfall (Praise You) | 2022-02-18 | 71 | 1M8t1j3Kv2qp97bdq5q4Vl |

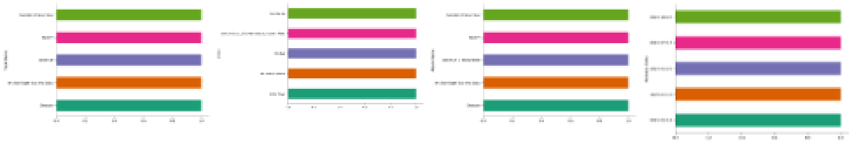
Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

[42] Distributions

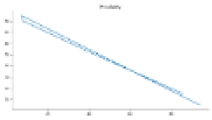


Categorical distributions

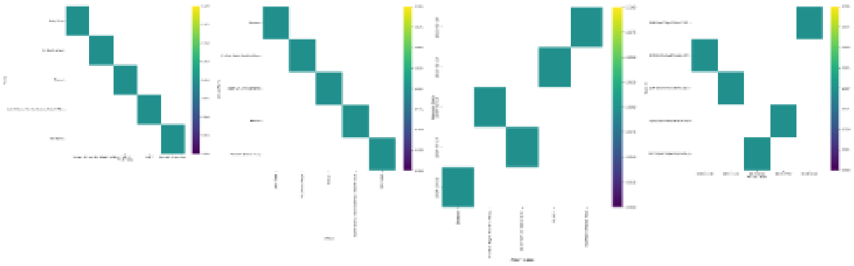


Time series

Values



2-d categorical distributions



Faceted distributions

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.
```

# CHAPTER 5

# Discussion and Conclusion

## 5.1 Key Findings:

The Spotify Music Recommendation System project successfully demonstrates the application of data-driven techniques to enhance music recommendations by combining content-based filtering, popularity analysis, and hybrid models. By leveraging Spotify's audio features—such as danceability, energy, and tempo—alongside user preferences, the system provides relevant and engaging song suggestions. The implementation of weighted popularity, which accounts for both a song's popularity and release date, improves the recency of recommendations and balances trending songs with user-tailored music. Overall, this approach highlights the effectiveness of combining data science and machine learning in creating more personalized, dynamic recommendation experiences, enhancing user satisfaction in music discovery.

## 5.4 Limitations:

The current Spotify Music Recommendation System, while effective in generating personalized recommendations, has some limitations. First, the model relies heavily on audio features and popularity metrics, which may not fully capture the user's personal context, mood, or preferences that are influenced by factors beyond musical properties, such as listening environment or time of day.

Additionally, content-based filtering can lead to a narrower range of recommendations over time, as it tends to suggest songs similar to those already known to the user, potentially limiting musical discovery. Another limitation is the reliance on Spotify's API, which can restrict data availability to only Spotify's catalog, impacting users who prefer lesser-known or niche music not available on the platform. Furthermore, the system does not account for real-time feedback or adapt to the user's changing tastes dynamically, which could make the recommendations less relevant over time. These limitations suggest opportunities for future improvements, such as integrating collaborative filtering, contextual recommendation features, and real-time user feedback.

## 5.2 Git Hub Link of the Project:

https://github.com/Kalaiselvan908/kalaiselvan1.git

## 5.3 Video Recording of Project

WhatsApp Video 2024-11-10 at 4.22.2

## 5.5 Future Work:

To improve the Spotify Music Recommendation System, future work could focus on integrating collaborative filtering techniques alongside the current content-based approach to better capture the preferences of users with similar listening habits, thereby enhancing the diversity and accuracy of recommendations. Additionally, incorporating real-time feedback mechanisms, such as user ratings or skips, would allow the system to adapt dynamically to shifting preferences. Expanding the dataset to include user-specific context, such as mood, time of day, or location, could further personalize recommendations. Exploring advanced deep learning models, like neural networks, to analyze both audio features and user behavior data could improve prediction accuracy. Furthermore, incorporating more granular aspects of music, such as lyrics analysis or genre-based recommendations, could broaden the system's capabilities, offering richer and more diverse music discovery experiences.

## 5.6 Conclusion:

The recommendation system employed by Spotify has proven to be highly effective in accurately predicting and suggesting songs that align with the preferences of individual users. Through its sophisticated algorithms and data processing capabilities, the Spotify recommendation program has revolutionized the music discovery experience for millions of users worldwide. By leveraging a combination of advanced machine learning techniques, such as collaborative filtering and content-based recommendation, the system analyzes vast amounts of user data to generate personalized song recommendations tailored to each user's unique tastes and preferences. One of the key strengths of the Spotify recommendation system is its ability to cater to a diverse range of musical preferences. Whether users enjoy pop, rock, jazz, or any other genre, the recommendation engine is adept at identifying patterns and similarities in their listening habits to provide relevant song suggestions. This personalized approach not only enhances the user experience but also fosters a deeper connection between users and the music they love. The success of the Spotify recommendation program can be attributed to several key factors. First and foremost is the system's ability to deliver accurate and relevant song suggestions to users. By leveraging advanced machine learning techniques, the recommendation engine is able to analyze vast amounts of data to identify patterns and similarities in users' listening habits, resulting in highly personalized recommendations. Furthermore, the recommendation system's user-friendly interface makes it easy for users to interact with the platform and provide feedback on the recommendations they receive.

# REFERENCES


 **Spotify API Documentation**
Spotify. (n.d.). Web API Reference
This documentation provides details on the Spotify API, which was essential for gathering data on songs, artists, and audio features in the project.

 **Recommendation Systems: An Introduction**
Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.
This book offers a comprehensive overview of recommendation systems, covering essential techniques like content-based filtering and collaborative filtering, which informed the recommendation approaches in this project.

 **Scikit-Learn Documentation**
Scikit-Learn. (n.d.). User Guide
Scikit-Learn's user guide and documentation were helpful in implementing data scaling, similarity measures, and various machine learning utilities used in feature normalization and similarity scoring.

 **Machine Learning Yearning**
Ng, A. (2018). *Machine Learning Yearning*.
This resource provides foundational insights on machine learning applications, challenges, and best practices, which guided the project's design, particularly in balancing model accuracy with scalability.

 **Research on Hybrid Recommendation Systems**
Burke, R. (2002). *Hybrid Recommender Systems: Survey and Experiments*. *User Modeling and User-Adapted Interaction*, *12*(4), 331-370.
This paper provided insights into hybrid recommendation models, which combine multiple recommendation techniques, shaping the design of the hybrid model used in this project.

**Exploring Audio Features for Music Recommendations**

Korzeniowski, F., & Widmer, G. (2018). "Feature Learning for Chord Recognition: The Deep Chroma Extractor." Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR).

This research informed the selection of audio features, such as tempo and energy, as useful components for generating music recommendations based on content analysis.

https://youtu.be/gaZKjAKfe0s?si=g8IkVN2JIS5Djj1c

https://youtu.be/tpd43mPc3aA?si=j_t1lzNXeyQERyxi

https://www.kaggle.com/