

Team Project Final Submission

Team 3: Ali Bahrami, Shalom Bekele, Simranjeet Singh, Robin Godinho, Hemen Gebrekiros

College of Information Studies, University of Maryland, College Park

INST 327 - ESG1

Professor Pkshetry

May 11, 2022

Introduction

Spotify is the music service covered in the database. Our project is focused on creating an algorithm that allows consumers to gain a better experience with an improved algorithm in their Spotify experience. Analytics of music and personalization rankings on their preferred streaming platform. The purpose of our database is to generate a method that will track music to create a better experience for music listeners. The method used in our database will define and deliver what music the users would likely listen to. This method is intended to improve the overall user experience and solve possible gaps in music usability. The components that are included in our music database are monitoring users listening preferences based on the time of day the user may listen to music. Furthermore, present and prospective users' favorite artists and genres will be factors in improving the recommendation system. Our decision to build a database that enhances users' algorithms was driven by the relevance of problem-solving operations and our team's passion for music. The main goal of the database, which may be used in a variety of ways, is to allow users to explore and examine their favorite music selections. Music listeners have become reliant on online streaming platforms for where to get their music. With many companies like Spotify, Apple Music, and Soundcloud becoming more popular, we thought it would be helpful for users to utilize one platform that has all their needs. People are becoming more interactive with their music, and the statistics behind the music they are listening to are becoming more relevant.

Database Description

Our database is targeted at Spotify, and the goal is to improve users' music algorithms providing a better experience for potential users. Our team first approached this project by creating entities and attributes in an excel spreadsheet with assistance from the TA on further database queries that occurred throughout the project. Our choice to build a database that enhances users' algorithms was driven by the relevance of problem-solving operations and our team's interest in music. The main goal of the database, which may be used in a variety of ways, is to allow users to explore and examine their favorite music selections.

We identified primary, composite and foreign keys for our tables and created linking tables that would make it efficient for our lookup tables. Through this process, we were able to create one to many and many to many relationships between tables. This served as a foundation for building our entity-relationship diagram in MySQL.

Logical Design

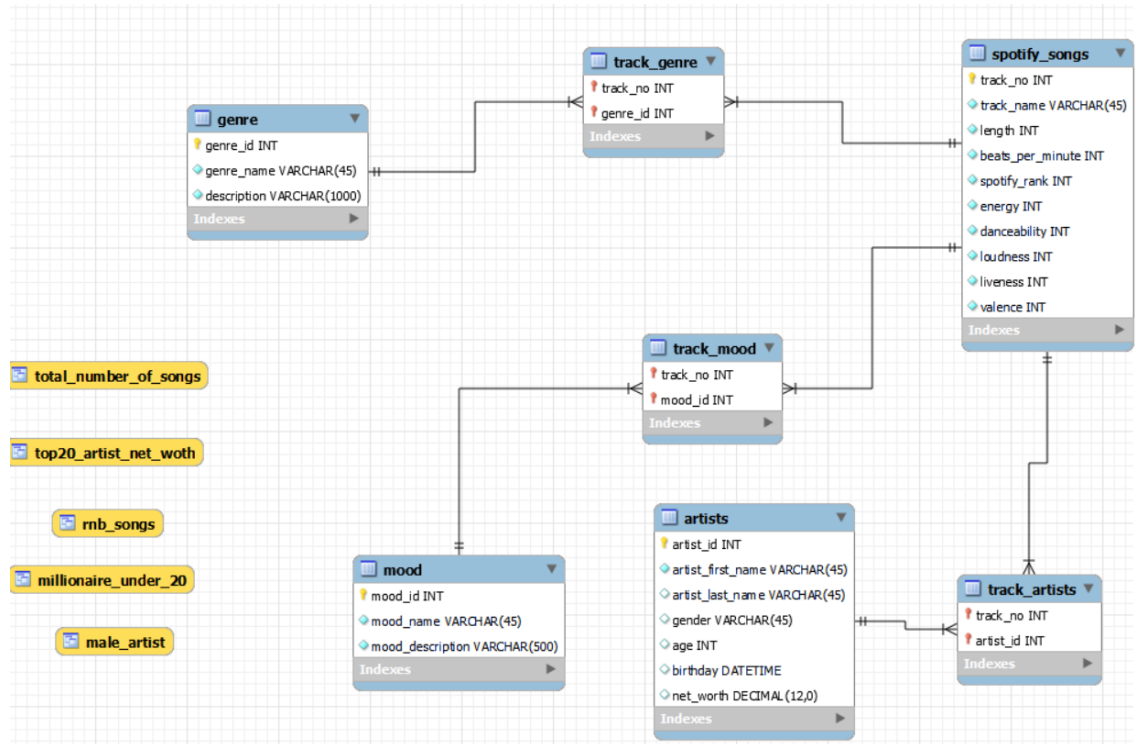


Diagram: Spotify_group_3 Entity Relationship Diagram

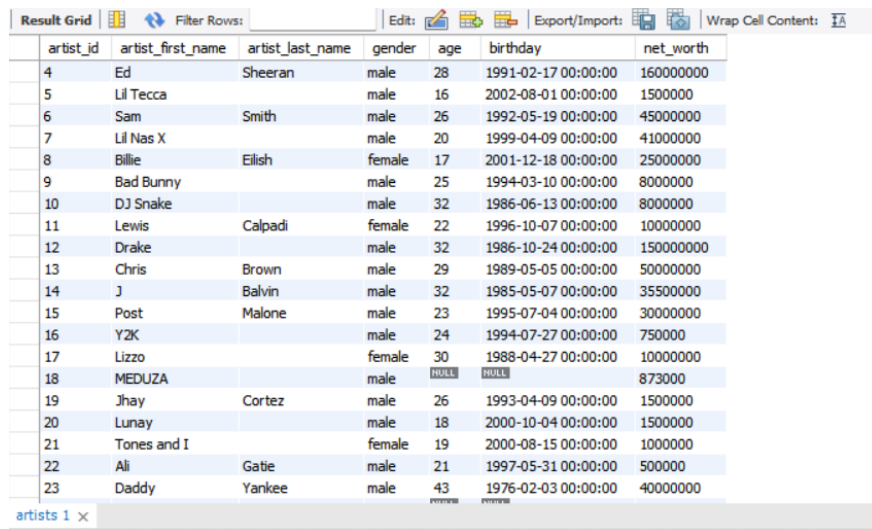
Physical Design

The transition from logical to physical design went smoothly. One major issue we encountered was referencing a table within the entity-relationship diagram itself, which we discovered after we forward engineered. We were able to place the correct columns in the reference tables and forward engineered them to have a database on MySQL. Importing data was not that hard. We discovered that because we built our sample data on Google spreadsheets when we exported it as a CSV, a few values were missing and unorganized, so we exported it into Excel, checked all the data, and then exported it as a CSV file into our database. When we first started importing data, we received the message "0 rows imported." We discovered that it was

due to the order in which we were importing data into our tables after further discussion in our group. We went back to the lecture slides and remembered that we needed to import all of the tables that did not have a foreign key before moving on to our linking tables. After successfully importing all of the data into our tables, we noticed that our stage name column had a few empty values, which we also saw in our Google sheets, but we assumed that MySQL would automatically insert null values. We updated the stage name column by inserting null values where the string is empty.

Sample Data

Most of the data that was inserted to the database can be seen on the Dump file for the Spotify data. Below is the Artists table with over 15 rows of data. The other tables in the database consist of 15 or more rows of data.



artist_id	artist_first_name	artist_last_name	gender	age	birthday	net_worth
4	Ed	Sheeran	male	28	1991-02-17 00:00:00	160000000
5	Lil Tecca		male	16	2002-08-01 00:00:00	1500000
6	Sam	Smith	male	26	1992-05-19 00:00:00	45000000
7	Lil Nas X		male	20	1999-04-09 00:00:00	41000000
8	Billie	Eilish	female	17	2001-12-18 00:00:00	25000000
9	Bad Bunny		male	25	1994-03-10 00:00:00	8000000
10	DJ Snake		male	32	1986-06-13 00:00:00	8000000
11	Lewis	Capaldi	female	22	1996-10-07 00:00:00	10000000
12	Drake		male	32	1986-10-24 00:00:00	150000000
13	Chris	Brown	male	29	1989-05-05 00:00:00	50000000
14	J	Balvin	male	32	1985-05-07 00:00:00	35500000
15	Post	Malone	male	23	1995-07-04 00:00:00	30000000
16	Y2K		female	24	1994-07-27 00:00:00	750000
17	Lizzo		female	30	1988-04-27 00:00:00	10000000
18	MEDUZA		male	NULL	NULL	873000
19	Jhay	Cortez	male	26	1993-04-09 00:00:00	1500000
20	Lunay		male	18	2000-10-04 00:00:00	1500000
21	Tones and I		female	19	2000-08-15 00:00:00	1000000
22	Ali	Gatie	male	21	1997-05-31 00:00:00	500000
23	Daddy	Yankee	male	43	1976-02-03 00:00:00	40000000

Figure: Sample data of the Artist table

Views/Queries

1. Query that shows Artists Under 20 who are Millionaires

```
USE Spotify_Group_3;
CREATE VIEW Millionaire_Under_20 AS
SELECT concat(artist_first_name,' ', artist_last_name) as ArtistsUnder20, net_worth
      FROM artists
      WHERE age<20 AND net_worth> 1000000;
```

2. Query that shows Male artist in order of net worth

```
USE Spotify_Group_3;
CREATE VIEW Male_artist AS
SELECT concat(artist_first_name, " ",artist_last_name) as Male_Artist, gender, net_worth
      FROM artists
      WHERE gender ="male"
      order by net_worth;
```

3. Query that lists all of the R&B Songs available on the database

```
USE Spotify_Group_3;
CREATE VIEW RnB_songs AS
SELECT track_name, genre_name
      FROM spotify_songs
      INNER JOIN genre
      ON genre_id = genre.genre_id
      WHERE genre_name = 'R&B';
```

4. Query that retrieves Artist with more than 25 songs

```
USE Spotify_Group_3;
CREATE VIEW Total_number_of_songs AS
SELECT
      artists.artist_last_name,
      COUNT(spotify_songs.track_no) AS NumberOfOrders
      FROM
      spotify_songs
      INNER JOIN
      artists ON artist_id = artists.artist_id
      WHERE
      artist_last_name = 'eilish'
      OR artist_last_name = 'billie'
```

```
GROUP BY artist_last_name
HAVING COUNT(spotify_songs.track_no) > 25;
```

5. Query that retrieves top 20 Artists Net worth from highest to lowest

```
USE Spotify_Group_3;
CREATE VIEW Top20_artist_net_worth AS
SELECT
    CONCAT(artist_first_name, ' ', artist_last_name) AS Artist,
    gender,
    age,
    CAST(birthday AS DATE) AS Artist_birthday,
    net_worth
FROM
    artists
ORDER BY net_worth DESC
LIMIT 20;
```

View Name	Req. A	Req. B	Req. C	Req. D	Req. E
Query 1	X	X			
Query 2	X	X			
Query 3	X	X		X	
Query 4	X	X	X	X	
Query 5	X		X		

Changes from Original Design

Our database was constructed similarly to the initial plan for our project, focused on one major music platform which was Spotify. Our database focuses on allowing users to arrange data about an album, artist, or track based on factors such as platform streams, track name, artist name, genre, and finally chart and playlist rating. We solely concentrated on highly significant components of music in our databases, such as where the musicians were from, a genre with a description, and the time the user listens to a specific genre. This emphasis on more relevant details allows our database to be much more useful and concise as our main goal with our project is to allow users to gain a more specific algorithm that caters to their music needs. Music has become very well integrated into our society and is continuously evolving. Right now, there are no simple programs that compare the precise statistics generated by different music platforms, which we believe will be extremely beneficial to our generation and in inspiring users to be adventurous and interact with genres and artists of the music that they may not know. This algorithm can help users gain a broader music library while providing exposure to independent artists. Linking tables were one significant component that we overlooked in our first design. We needed to search up tables, but we couldn't figure out how connecting tables would help our database. We realized that those connecting tables would have a many-to-many relationship with our look-up tables after talking with the instructor. When we first started querying the database, we discovered that we could link tables together using those tables.

Lessons Learned

We encountered issues when building our streaming database. To fix challenges, we enlisted the assistance of our instructor and TA. As previously indicated, one challenge we encountered was determining the relevance of our connecting tables and how to access them through our lookup tables. Another difficulty we encountered was during the data import procedure. We discovered that part of the data for the Artists table would not import while we were creating our entity-relationship diagram. After further analysis with a review of previously provided coursework, our team was able to pinpoint the issue, which was our data type. We utilized spaces for our long table names, and when we investigated the problem, we discovered that it was because of the spaces we used and that we would have to use an underscore to access data from the database. We assumed we would have to start again with our entity-relationship diagram, but happily, the lecturer explained that we could leverage MySQL functionality by clicking on the wrench symbol next to the table name. We were able to adjust the datatype length and a few column names thanks to this discovery. We might also examine our foreign keys to have a better idea of when our queries failed. Backing up the database was also fascinating to learn, and it came in helpful every time we wanted to make a change so that other team members could benefit from the new database. Overall, we learned a lot while putting together our database. With assistance from the TA, we were able to learn a lot in just a few weeks. There were times when we thought developing the database would be too difficult, but in the end, we learned many new skills in the process.

Potential Future Work

If our team spent more time researching the music algorithm database, we could add additional features to make the algorithm better. We would be able to obtain a deeper knowledge of the music algorithm and consumer demands if we had additional research. This would change our platform's capabilities by providing more accurate data. Our team would concentrate on learning about consumers' music preferences to develop a specific algorithm. Within several months, our team would most likely evaluate user patterns, extending and reaching new audiences and areas. More tables containing additional information about the learned data would be added to our database as future updates and additions. Our team recognizes that there is always a potential for growth and development.