CHAPTER 1

INTRODUCTION

1.INTRODUCTION

1.1 Introduction to DBMS

A database is simply an organized collection of related data, typically stored on disk, and accessible by many concurrent users, it is a logically coherent collection of data with some inherent meaning, representing some aspect of the real world and which is designed, built and populated with data for a specific purpose.

Databases are managed by a Database Management System (DBMS) which is a collection of programs that enables users to create and maintain a database.

Advantages of DBMS:

1. Redundancy is controlled.
2. Unauthorized access is restricted.
3. Providing multiple user interfaces.
4. Enforcing integrity constraints.
5. Providing backup and recovery.

1.2 Introduction to SQL

Structured Query Language (SQL), is a language used to request data from a database which includes database creation, deletion, retrieval of required tables and even manipulation of data held in a relational database management system.

SQL is considered as a Non-Procedural or a High-level language in which the expected result or operation is given without the specific details about how to accomplish the task. So, SQL is a declarative language. Therefore, SQL is designed at a higher conceptual level of operation than procedural languages as procedural languages include only the information about opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file systems, but the lower level logical and physical operations are not specified in SQL.

1.3 Music Library Database:

The primary objective of developing the Music Library Database is to establish a systematic and efficient platform for the comprehensive management of music collections.

1. Systematic Organization: Develop a structured database to systematically organize music collections, allowing users to easily categorize songs based on artist, genre, album, and other relevant criteria.

2. Efficient Retrieval: Enable quick and efficient retrieval of music data, ensuring that users can easily find and access specific songs or albums within the extensive collection.

3. Comprehensive Categorization: Implement a robust categorization system, incorporating metadata for each music file, such as artist names, album titles, and genres, to provide a comprehensive overview of the music library.

4. User-Friendly Interface: Design a user-friendly interface that promotes a seamless and intuitive experience, making it easy for individuals and institutions to interact with and manage their music collections.

5. MySQL Integration: Leverage the capabilities of MySQL to establish a structured and reliable backend system, enhancing the overall accessibility and performance of the Music Library Database.

1.4 Scope and importance of work

Scope of Work:

1. All Kinds of Music: Include lots of different types of music, artists, and albums to make everyone happy.

2. Lots of Info: Write down details about each song, like who made it, when it came out, and what kind of music it is.

3. Make Your Playlists: Help people create, edit, and organize their own playlists of favorite songs.

4. Easy to Use: Design a system that's simple for people to use, so they can easily find and enjoy their music.

5. Search Easily: Make it so people can quickly find exactly the song or album they're looking for.

Importance of Work:

1. Get Organized: Make it easy for people to organize their music, so they don't waste time searching.

2. Enjoy Music More: Create a system that makes listening to music more fun and enjoyable for everyone.

3. Save Time: Help people save time by making it quick and easy to find and manage their music.

4. Pros Use It Too: Even music pros can benefit, making their work more organized and efficient.

5. Always Accessible: Make sure people can always get to their music library easily, whenever they want.

Chapter 2
DESIGN

2.1 Theory of ER Diagram

The Entity–Relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**

An **Entity Relationship Diagram (ERD)** shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data.
An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of database.

ER diagrams are used to sketch out the design of a database.

2.2 Entities

An entity is an 'object' in the real world with an independent existence and an entity type defines a collection (or set) of entities that have the same attributes. Each entity type in the database is described by its name and attributes.

An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.

2.3 Relationships

A relationship among two or more entities represents an association among the entities and whenever an attribute of one entity refers to another entity, there exists a relationship between the two entities.

In a relationship, a foreign key of one table refers to the primary key of the other table and it is represented by a diamond shape in the ER diagram.

2.4 Attributes

An attribute represents some property of interest that further describes an entity and the column header of the table shows the attributes. Each attribute in a table has a certain domain which allows it to accept a certain ‘set of values’ only.

The attribute values of each entity will define its characteristics in the table and is represented by oval in the ER diagram

ER DIAGRAM

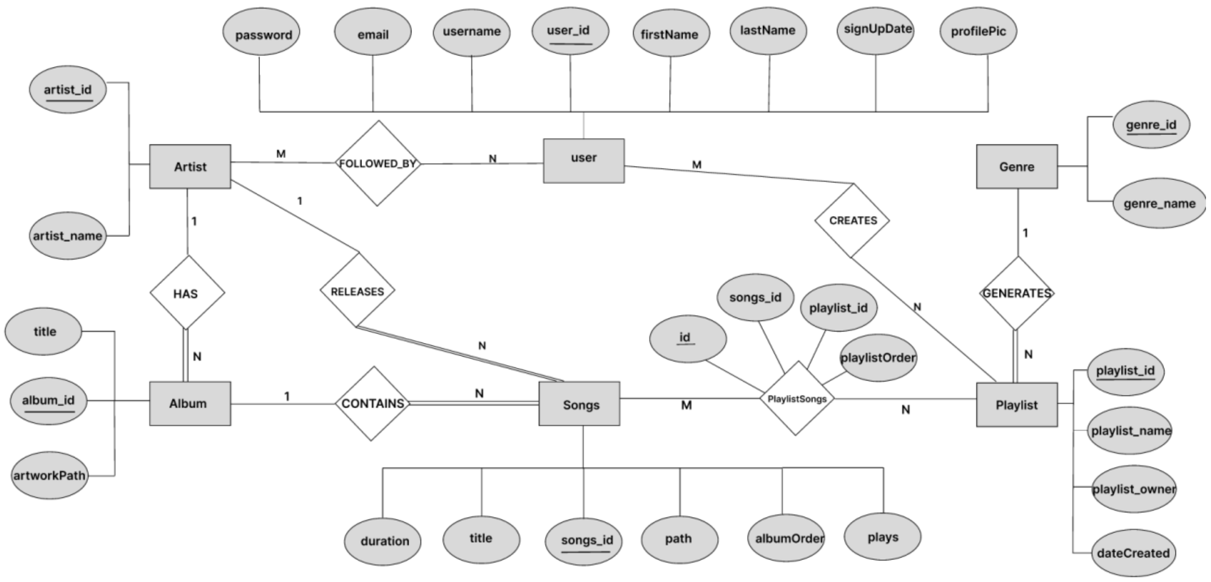


Figure :ER diagram for the database

SCHEMA DIAGRAM

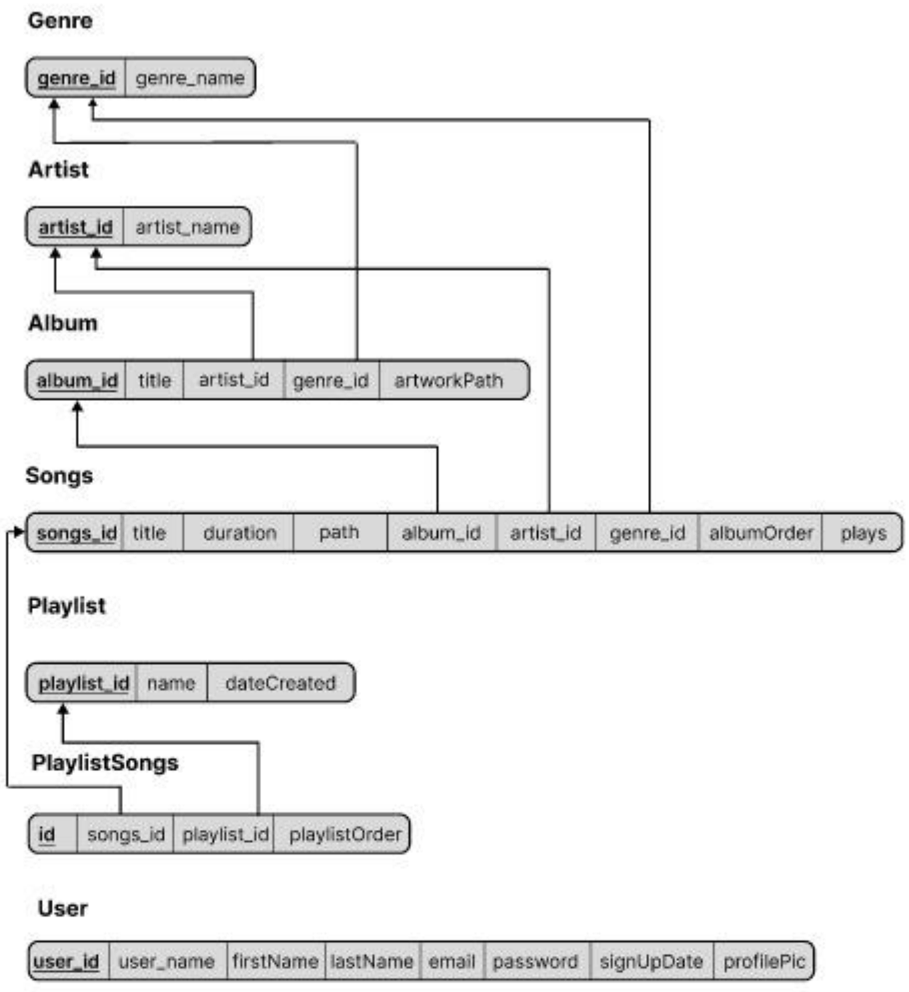


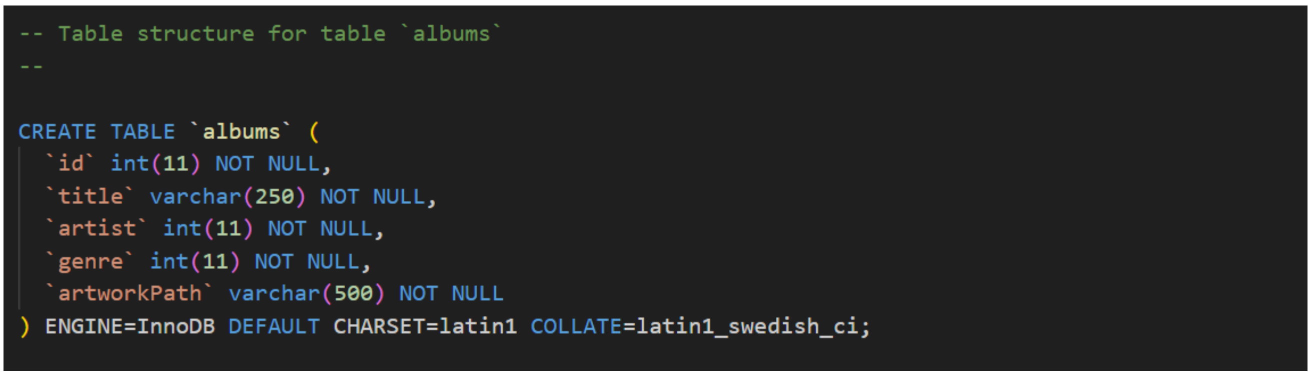
Figure: Schema diagram

LIST OF TABLES

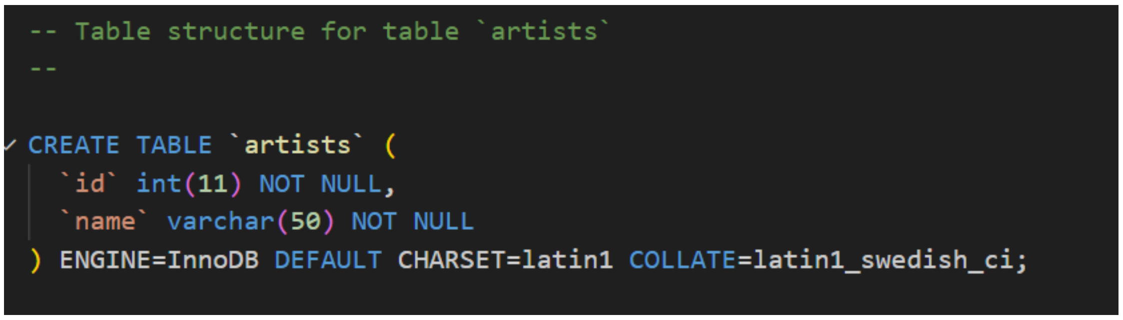
CHAPTER 3

IMPLEMENTATION

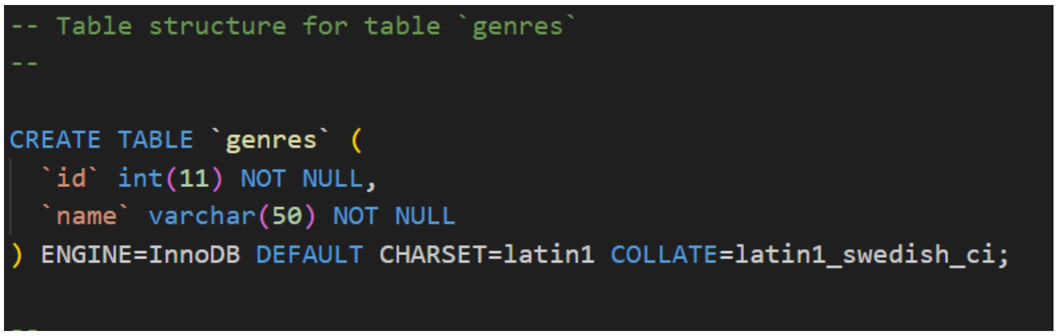
Create table command & Insertion tables values:



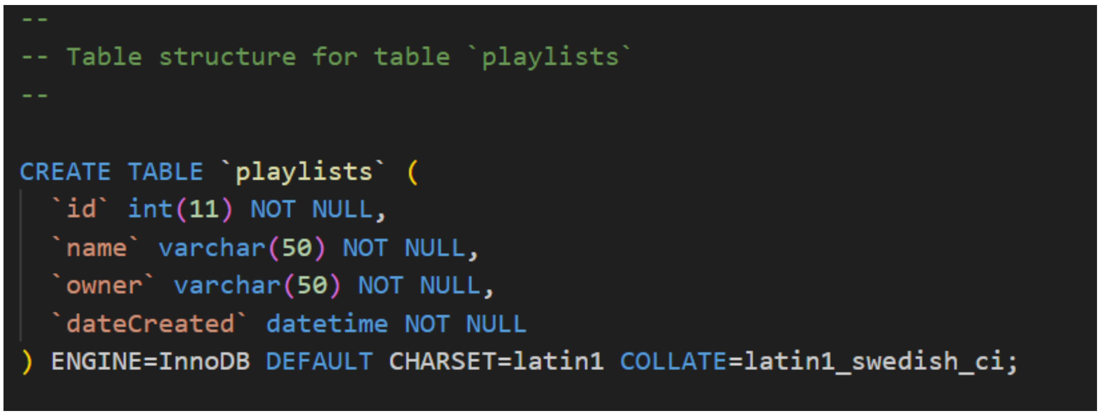
```sql
-- Table structure for table `albums`
--

CREATE TABLE `albums` (
  `id` int(11) NOT NULL,
  `title` varchar(250) NOT NULL,
  `artist` int(11) NOT NULL,
  `genre` int(11) NOT NULL,
  `artworkPath` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



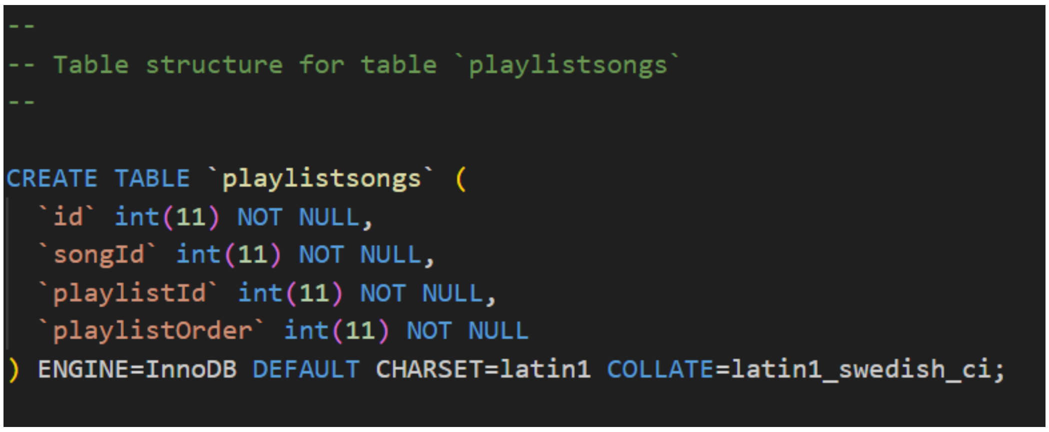
```sql
-- Table structure for table `artists`
--

CREATE TABLE `artists` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



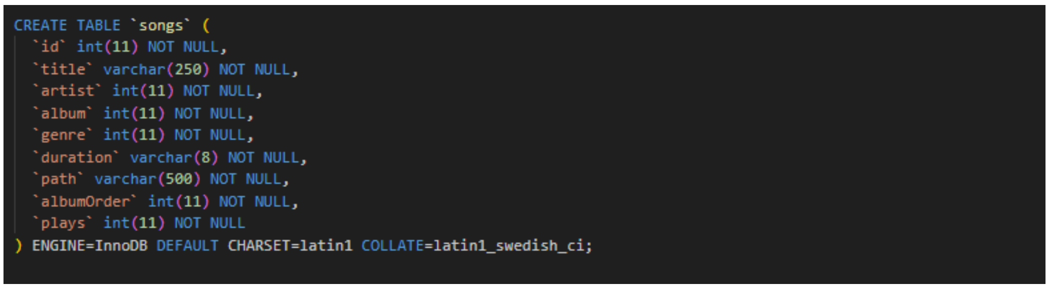
```sql
-- Table structure for table `genres`
--

CREATE TABLE `genres` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



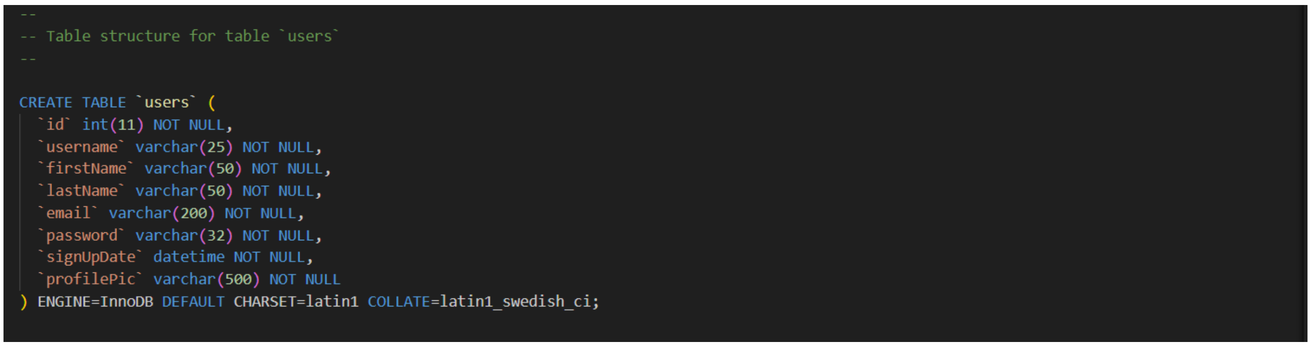
```sql
--
-- Table structure for table `playlists`
--

CREATE TABLE `playlists` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `owner` varchar(50) NOT NULL,
  `dateCreated` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



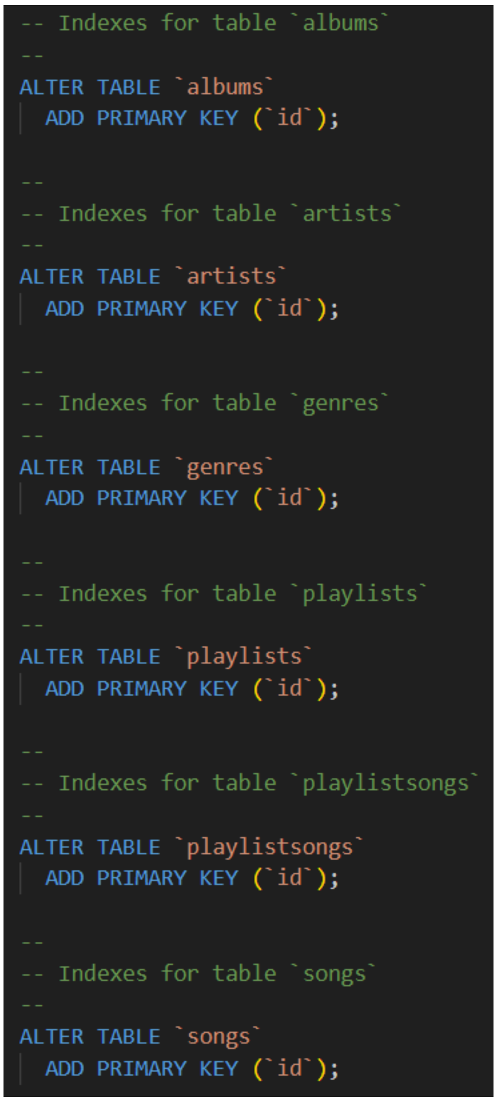
```sql
--
-- Table structure for table `playlistsongs`
--

CREATE TABLE `playlistsongs` (
  `id` int(11) NOT NULL,
  `songId` int(11) NOT NULL,
  `playlistId` int(11) NOT NULL,
  `playlistOrder` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



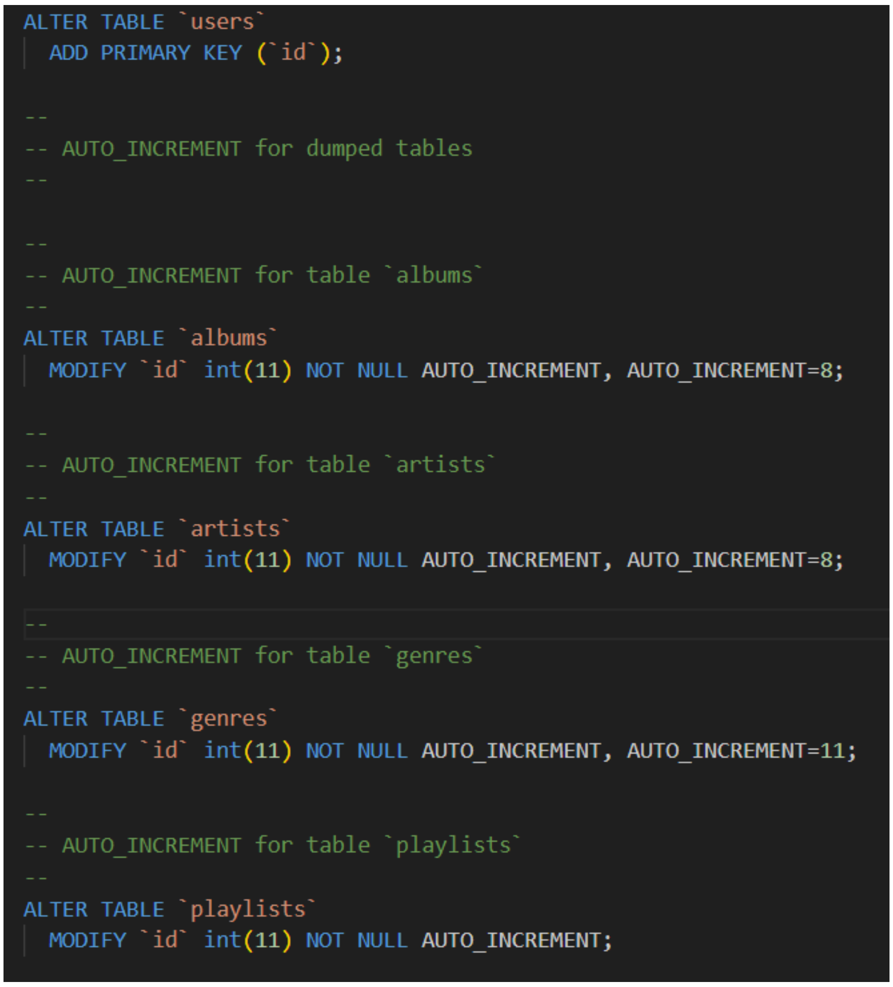
```sql
CREATE TABLE `songs` (
  `id` int(11) NOT NULL,
  `title` varchar(250) NOT NULL,
  `artist` int(11) NOT NULL,
  `album` int(11) NOT NULL,
  `genre` int(11) NOT NULL,
  `duration` varchar(8) NOT NULL,
  `path` varchar(500) NOT NULL,
  `albumOrder` int(11) NOT NULL,
  `plays` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



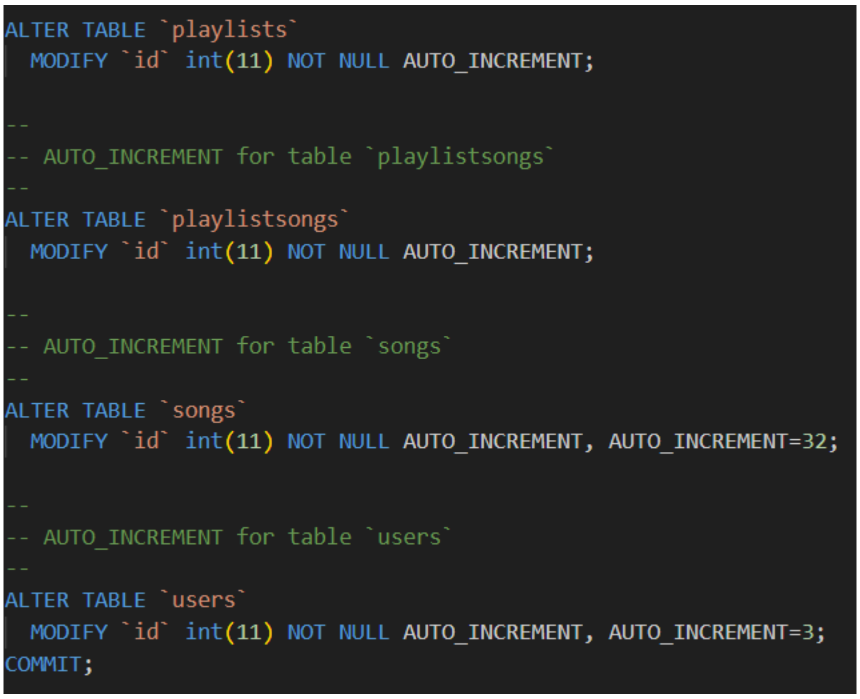
```sql
--
-- Table structure for table `users`
--

CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `username` varchar(25) NOT NULL,
  `firstName` varchar(50) NOT NULL,
  `lastName` varchar(50) NOT NULL,
  `email` varchar(200) NOT NULL,
  `password` varchar(32) NOT NULL,
  `signUpDate` datetime NOT NULL,
  `profilePic` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```



```sql
-- Indexes for table `albums`
--
ALTER TABLE `albums`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `artists`
--
ALTER TABLE `artists`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `genres`
--
ALTER TABLE `genres`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `playlists`
--
ALTER TABLE `playlists`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `playlistsongs`
--
ALTER TABLE `playlistsongs`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `songs`
--
ALTER TABLE `songs`
  ADD PRIMARY KEY (`id`);
```



```
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `albums`
--
ALTER TABLE `albums`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

--
-- AUTO_INCREMENT for table `artists`
--
ALTER TABLE `artists`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

--
-- AUTO_INCREMENT for table `genres`
--
ALTER TABLE `genres`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `playlists`
--
ALTER TABLE `playlists`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```



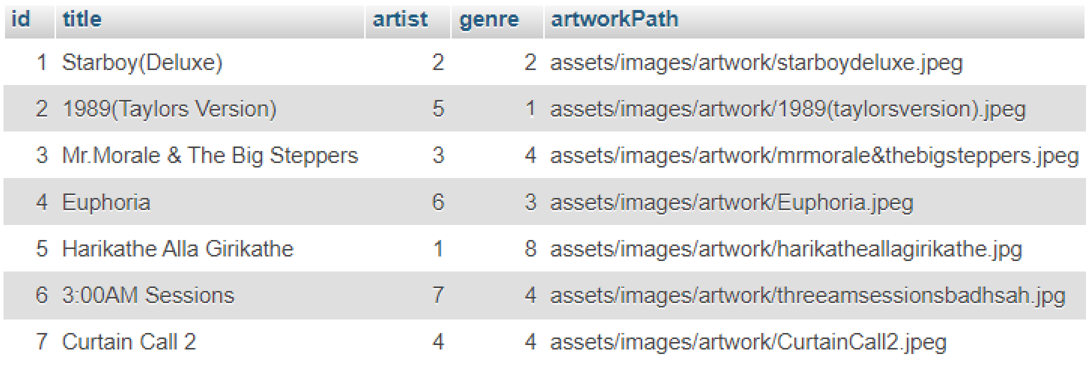
```
ALTER TABLE `playlists`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `playlistsongs`
--
ALTER TABLE `playlistsongs`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `songs`
--
ALTER TABLE `songs`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=32;

--
-- AUTO_INCREMENT for table `users`
--
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
COMMIT;
```

Snap Shots

ALBUM

album table has a attributes **id, title ,artist , genre , artworkPath** with ‘id’ as the primary key as used shown in table

STRUCTURE OF ALBUM

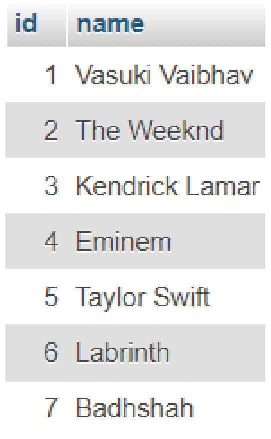


id	title	artist	genre	artworkPath
1	Starboy(Deluxe)	2	2	assets/images/artwork/starboydeluxe.jpeg
2	1989(Taylors Version)	5	1	assets/images/artwork/1989(taylorsversion).jpeg
3	Mr.Morale & The Big Steppers	3	4	assets/images/artwork/mrmorale&thebigsteppers.jpeg
4	Euphoria	6	3	assets/images/artwork/Euphoria.jpeg
5	Harikathe Alla Girikathe	1	8	assets/images/artwork/harikatheallagirikathe.jpg
6	3:00AM Sessions	7	4	assets/images/artwork/threeamsessionsbadhsah.jpg
7	Curtain Call 2	4	4	assets/images/artwork/CurtainCall2.jpeg

ARTISTS

artists table has a attributes **id , name** with ‘**id**’ as the primary key as used shown in the table

STRUCTURE OF ARTISTS

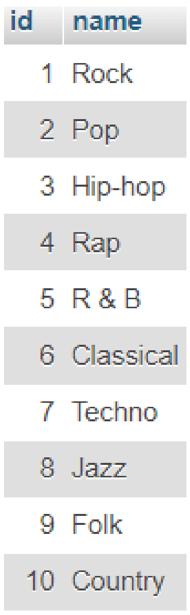


id	name
1	Vasuki Vaibhav
2	The Weeknd
3	Kendrick Lamar
4	Eminem
5	Taylor Swift
6	Labrinth
7	Badhshah

GENRES

GENRES table has the attribute **id , name** with ‘**id**’ as the primary key as used shown in the table

STRUCTURE OF GENRES

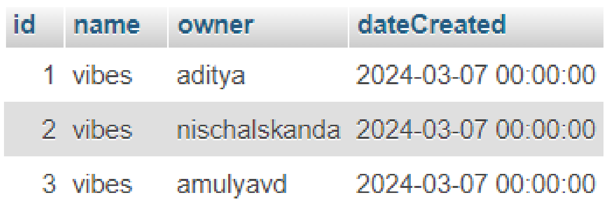


id	name
1	Rock
2	Pop
3	Hip-hop
4	Rap
5	R & B
6	Classical
7	Techno
8	Jazz
9	Folk
10	Country

PLAYLIST

PLAYLIST table has the attribute **id, name, owner , dateCreated**

STRUCTURE OF PAYMENT

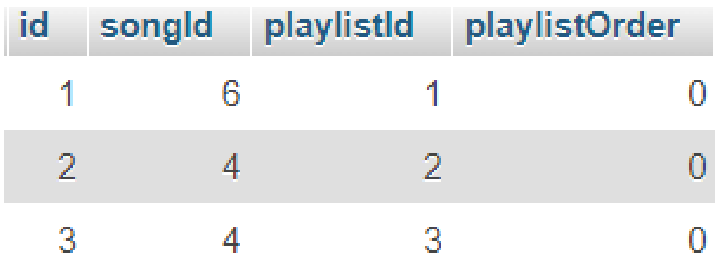


id	name	owner	dateCreated
1	vibes	aditya	2024-03-07 00:00:00
2	vibes	nischalskanda	2024-03-07 00:00:00
3	vibes	amulyavd	2024-03-07 00:00:00

PLAYLISTSONGS

playlistsongs table has the attribute **id , songid , playlistid , playlistOrder**

STRUCTURE OF STOCKS

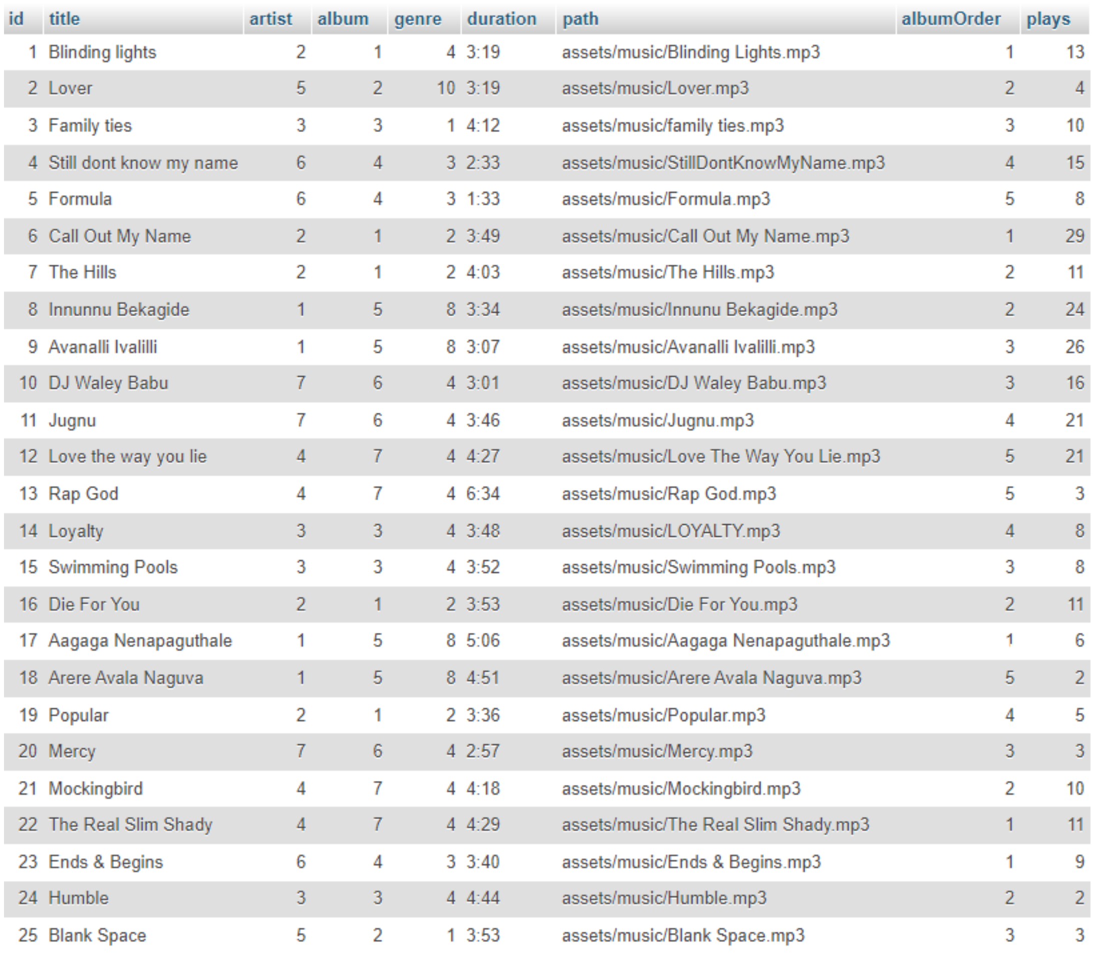


id	songId	playlistId	playlistOrder
1	6	1	0
2	4	2	0
3	4	3	0

SONGS

songs table has the **attribute id, title , artist , genre , duration , path , albumOrder** plays as shown in table

STRUCTURE OF SONGS

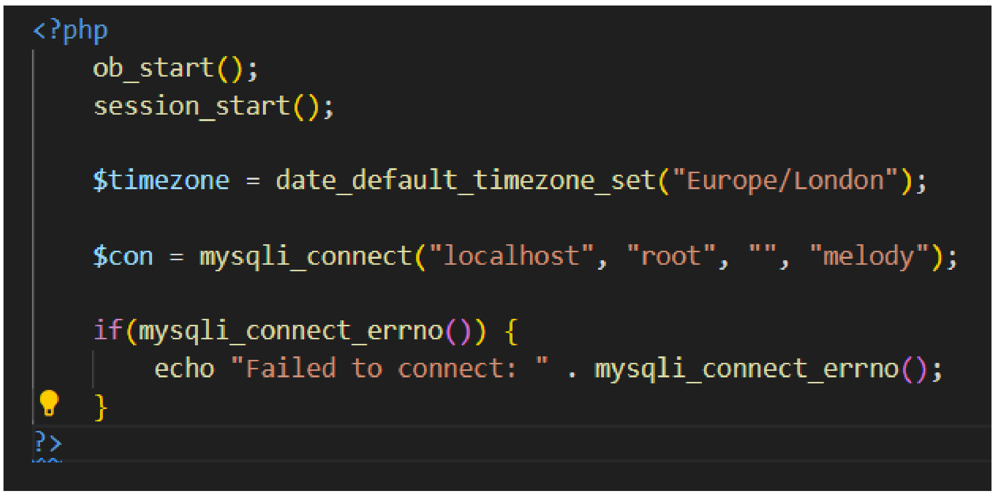


id	title	artist	album	genre	duration	path	albumOrder	plays
1	Blinding lights	2	1	4	3:19	assets/music/Blinding Lights.mp3	1	13
2	Lover	5	2	10	3:19	assets/music/Lover.mp3	2	4
3	Family ties	3	3	1	4:12	assets/music/family ties.mp3	3	10
4	Still dont know my name	6	4	3	2:33	assets/music/StillDontKnowMyName.mp3	4	15
5	Formula	6	4	3	1:33	assets/music/Formula.mp3	5	8
6	Call Out My Name	2	1	2	3:49	assets/music/Call Out My Name.mp3	1	29
7	The Hills	2	1	2	4:03	assets/music/The Hills.mp3	2	11
8	Innunnu Bekagide	1	5	8	3:34	assets/music/Innunu Bekagide.mp3	2	24
9	Avanalli Ivalilli	1	5	8	3:07	assets/music/Avanalli Ivalilli.mp3	3	26
10	DJ Waley Babu	7	6	4	3:01	assets/music/DJ Waley Babu.mp3	3	16
11	Jugnu	7	6	4	3:46	assets/music/Jugnu.mp3	4	21
12	Love the way you lie	4	7	4	4:27	assets/music/Love The Way You Lie.mp3	5	21
13	Rap God	4	7	4	6:34	assets/music/Rap God.mp3	5	3
14	Loyalty	3	3	4	3:48	assets/music/LOYALTY.mp3	4	8
15	Swimming Pools	3	3	4	3:52	assets/music/Swimming Pools.mp3	3	8
16	Die For You	2	1	2	3:53	assets/music/Die For You.mp3	2	11
17	Aagaga Nenapaguthale	1	5	8	5:06	assets/music/Aagaga Nenapaguthale.mp3	1	6
18	Arere Avala Naguva	1	5	8	4:51	assets/music/Arere Avala Naguva.mp3	5	2
19	Popular	2	1	2	3:36	assets/music/Popular.mp3	4	5
20	Mercy	7	6	4	2:57	assets/music/Mercy.mp3	3	3
21	Mockingbird	4	7	4	4:18	assets/music/Mockingbird.mp3	2	10
22	The Real Slim Shady	4	7	4	4:29	assets/music/The Real Slim Shady.mp3	1	11
23	Ends & Begins	6	4	3	3:40	assets/music/Ends & Begins.mp3	1	9
24	Humble	3	3	4	4:44	assets/music/Humble.mp3	2	2
25	Blank Space	5	2	1	3:53	assets/music/Blank Space.mp3	3	3

CODE SNIPPETS

1. DATABASE CONNECTION

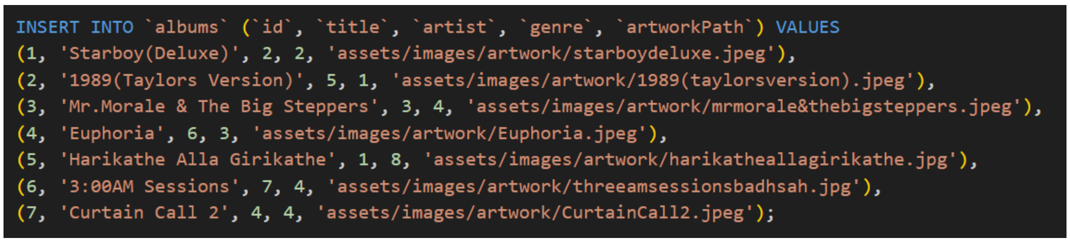
The connect() / mysqli_connect() function opens a new connection to the MySQL server with the following syntax : mysqli_connect(host, username, password, dbname, port, socket).



```php
<?php
    ob_start();
    session_start();

    $timezone = date_default_timezone_set("Europe/London");

    $con = mysqli_connect("localhost", "root", "", "melody");

    if(mysqli_connect_errno()) {
        echo "Failed to connect: " . mysqli_connect_errno();
    }
?>
```

2. INSERT QUERY

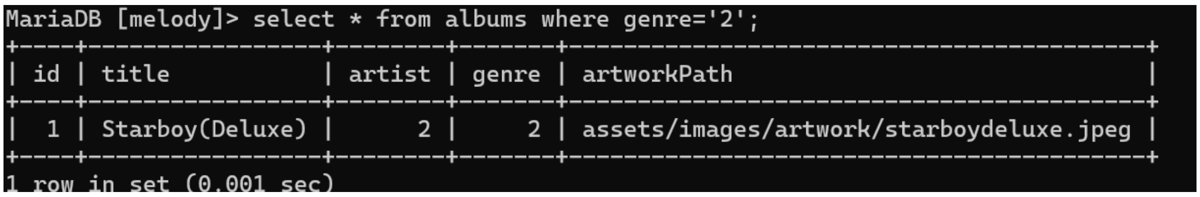
This query is used to insert the values for albums.



```sql
INSERT INTO `albums` (`id`, `title`, `artist`, `genre`, `artworkPath`) VALUES
(1, 'Starboy(Deluxe)', 2, 2, 'assets/images/artwork/starboydeluxe.jpeg'),
(2, '1989(Taylors Version)', 5, 1, 'assets/images/artwork/1989(taylorsversion).jpeg'),
(3, 'Mr.Morale & The Big Steppers', 3, 4, 'assets/images/artwork/mrmorale&thebigsteppers.jpeg'),
(4, 'Euphoria', 6, 3, 'assets/images/artwork/Euphoria.jpeg'),
(5, 'Harikathe Alla Girikathe', 1, 8, 'assets/images/artwork/harikatheallagirikathe.jpg'),
(6, '3:00AM Sessions', 7, 4, 'assets/images/artwork/threeamsessionsbadhsah.jpg'),
(7, 'Curtain Call 2', 4, 4, 'assets/images/artwork/CurtainCall2.jpeg');
```

3. SELECT QUERY

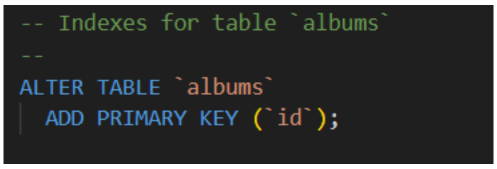
In this query,all the details are fetched using SELECT* command.



```
MariaDB [melody]> select * from albums where genre='2';
+----+-----------------+--------+-------+-----------------------------------------+
| id | title           | artist | genre | artworkPath                             |
+----+-----------------+--------+-------+-----------------------------------------+
|  1 | Starboy(Deluxe) |      2 |     2 | assets/images/artwork/starboydeluxe.jpeg |
+----+-----------------+--------+-------+-----------------------------------------+
1 row in set (0.001 sec)
```

4. ALTER QUERY

Here the alter query is called to update the patient of an already existing based on its name and and patient id respectively.



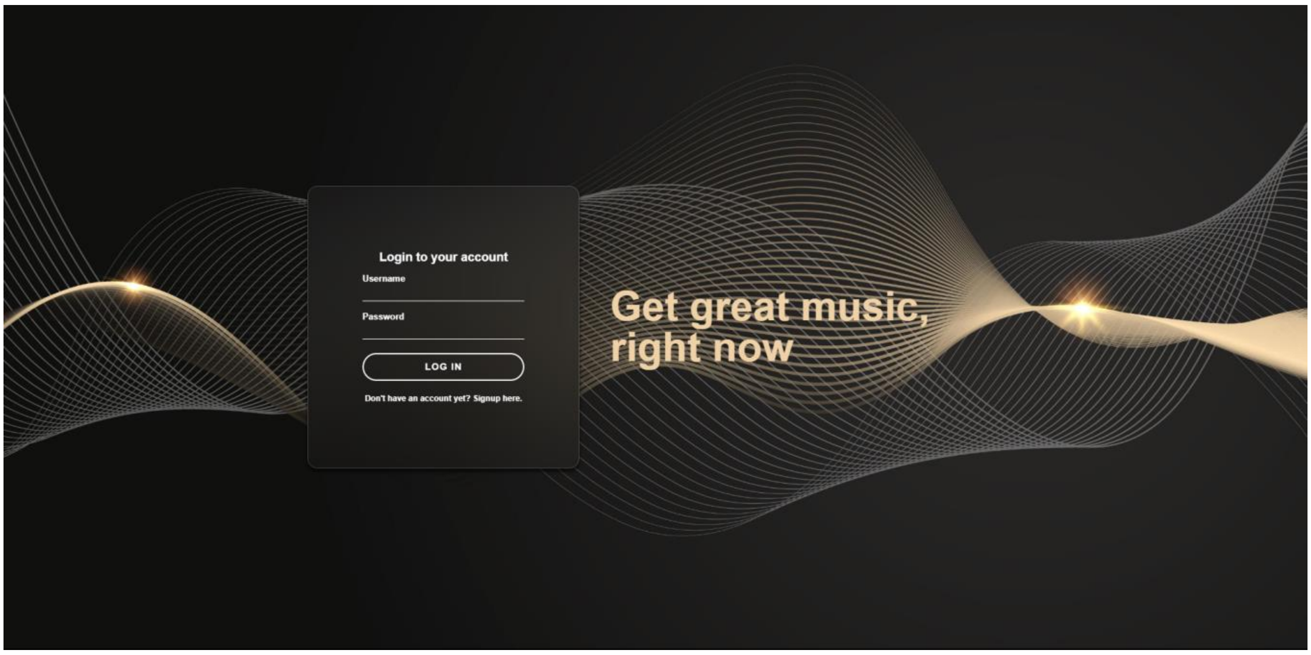
```sql
-- Indexes for table `albums`
--
ALTER TABLE `albums`
  ADD PRIMARY KEY (`id`);
```

CHAPTER 4

RESULTS AND SNAPSHOTS

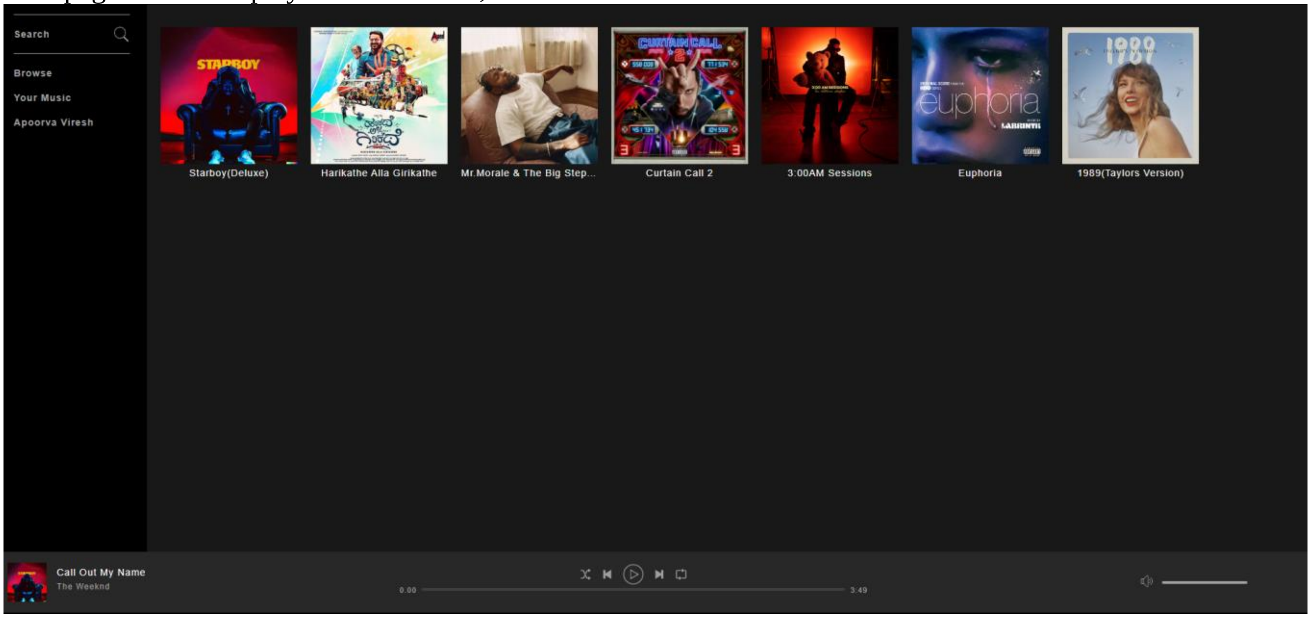
HOME PAGE

This is the first window when the application is executed.


Login to your account
Username
Password
LOG IN
Don't have an account yet? Signup here.
Get great music, right now

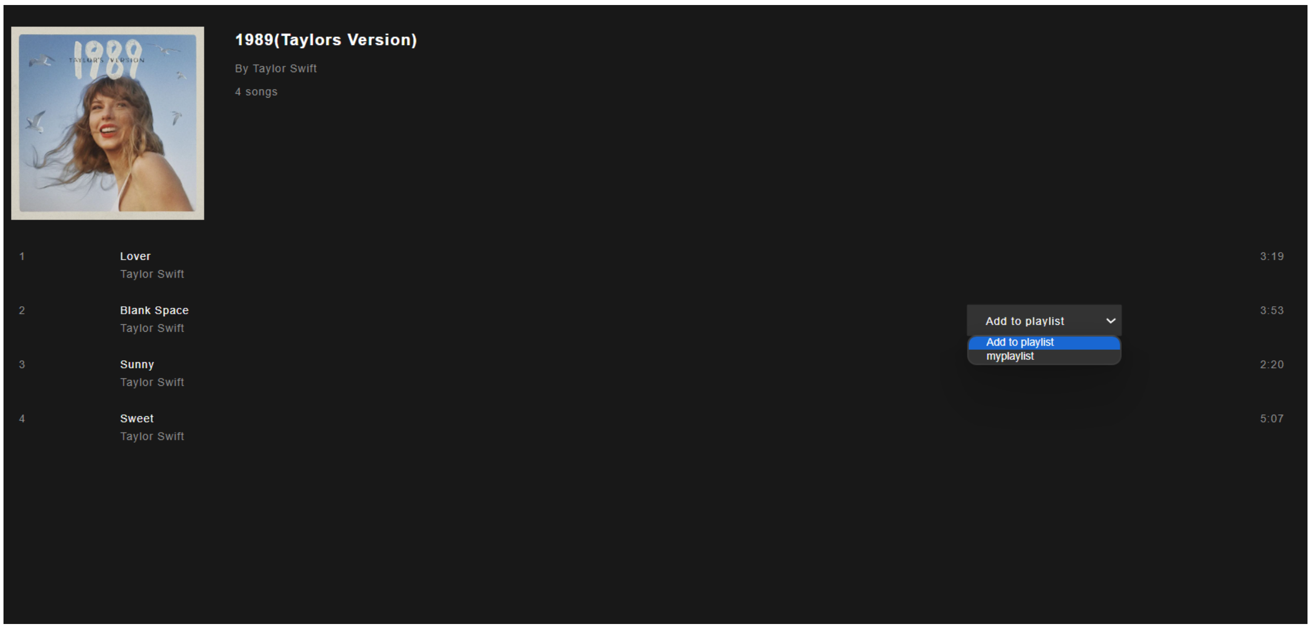
DASHBOARD

This page shows the playlists of the user, albums and user details



ALBUM 1

This page shows how a user can add a song to his playlist, in this example the user is adding the song ‘Blank Space’ to his playlist ‘myplaylist’

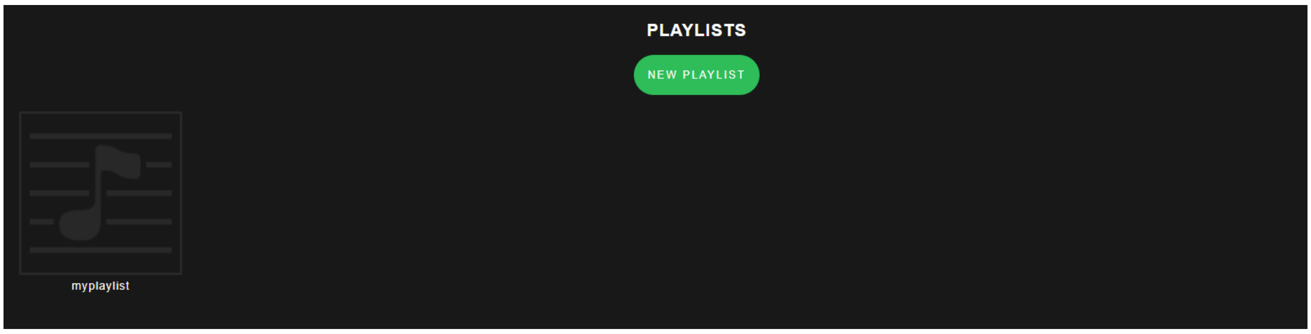


1989(Taylors Version)

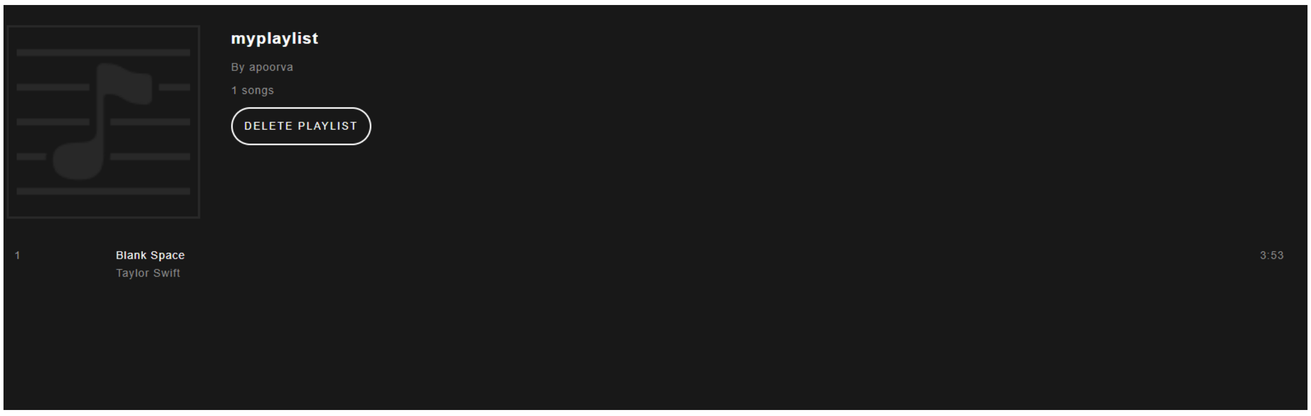
By Taylor Swift

4 songs

1	Lover Taylor Swift	3:19
2	Blank Space Taylor Swift	3:53
3	Sunny Taylor Swift	2:20
4	Sweet Taylor Swift	5:07



PLAYLISTS



myplaylist

By apoorva

1 songs

1	Blank Space Taylor Swift	3:53

CHAPTER 5

CONCLUSION & REFERENCES

CONCLUSION

In conclusion, the implementation of a music library database is instrumental in revolutionizing the way we organize, access, and appreciate musical content. This technology not only streamlines the management of vast musical collections but also enhances the overall user experience. By providing efficient search and categorization tools, users can effortlessly navigate through an extensive array of songs, albums, and artists. The database fosters a sense of connectivity, allowing music enthusiasts to explore diverse genres and discover new favorites.

Furthermore, the integration of metadata and advanced tagging systems ensures accuracy and precision in cataloging, facilitating personalized playlists and recommendations. As technology continues to evolve, the music library database stands as a testament to the symbiotic relationship between innovation and art, underscoring the significance of organized and accessible repositories in the digital age. In essence, the music library database harmonizes the intricacies of music curation, ultimately enriching our auditory journey and reinforcing the timeless power of music in our lives.

REFERENCES

1) MySQL Database

https://www.mysql.com/downloads/

2) PHP

http://php.net/

3) Design of registeration form

https://youtu.be/2HVKizgcfjo?si=7S

baE7UVFW1agKk5