Lindsay Prescott
Taylor Mathews
Daniel Atkinson
Tara Leger
Kevin Quebedeaux
David Scheuermann
Greg Johnson

Musicality

Database Design Report

Introduction

With the advent of music streaming services and their subsequent migration to handheld mobile electronic devices, apps like Spotify, Google Play Music, Amazon Music, and Apple Music are literally a swipe away. But, as with most new technology, solving one problem only reveals more problems to be solved. For example, now that music is virtually everywhere, more people are sharing music than ever before. Individuals, from the casual listener to the music aficionado, are making their own playlists geared towards particular activities, social settings, and even friend groups. However, there is no easy way to discover these playlists. Apps like Spotify curate their own playlists and typically feature those over user-created playlists, leaving user-created playlists virtually hidden from discovery. While they do typically allow users to share playlists with their friends (sometimes through other social media networks like Facebook), friends may often have disparate tastes, leaving these users at the mercy of the service's curated playlists. These applications also lack community-oriented features such as the ability to like a playlist or exchange comments about a playlist.

This is where Musicality steps in. Musicality is a social network designed around creating and sharing music playlists. Users will create accounts on this application, add songs to playlists, and share their created playlists on the social network. Other users will in turn be able to discover these playlists, like them if desired, and comment on the playlist posts to share their thoughts with the playlist creator and other users. In this report we outline the design and test implementation of a relational database that would support just such an application.

The rest of the report is comprised of two main sections; database design and database implementation. In describing our database design, we will layout the important elements of the application domain, summarized visually in an

Entity-Relationship Diagram. Then, we list the functional dependencies of our database along with primary and foreign keys. In the second part of the report, we will detail the test implementation of our database with sample data definition statements, insert statements, and all resulting tables, with the sample data in the database. We will also show examples of how the data may be modified via several queries, explaining why some of these queries may be needed in an app like Musicality.

The Application Domain

The application can essentially be split into two different domains: the music side and the social networking side.

On the music side, the application must be able to model the way music is organized in the real world. Music is typically organized as a collection of songs, where songs are assigned to specific artists and often associated with a subcollection of songs, known as albums which are in turn associated with the songs' artist. Songs have a title, a length in time, a genre, and a release year. Artists have a name, and albums have a name and a release date. The application will need to contain a collection of songs and their associated artists and albums in order to allow users to find and select songs they desire on their playlists. Playlists are the heart of our application, and a playlist is essentially just a collection of songs. Playlists, however, will need to be associated with the user who created them in order to facilitate the application's social networking features. Playlists must contain one or more songs. Songs in turn may be single artist authored or collaborative. Songs may appear on multiple albums and they may appear on zero or more playlists.

The heart of any social networking application is social interaction, which means we require users and user interactions. Users would need to be able to enter personal information about themselves, including their names, their date-of-birth, an address, and a short description of themselves as a "bio". The system would create unique identifiers for the user as of now and would also decide their signup-date. For our application, users must be able to "follow" other users, '"post" playlists, "comment" on a post, or "like" a post. These interactions would form the bulk of the social networking aspect of the application.

These basic tables give rise to the need for non-trivial relations between the previously mentioned tables. For example, if users are to follow one another, then we are minimally required to track an ordered pair <user_id1, user_id2> where user1 is understood to follow user2. Likewise, the relationship between users and their posts must also be tracked in a similar way. Three more such non-trivial relations exist between playlists, songs, artists, and albums.

Design Choices

As stated previously, there were essentially two separate portions of the data, the social networking side and the music side. The only connection between the two sides is a post from the social media side must contain a playlist from the song side. To inform the design choices for each side, we looked at some common music apps as well as things involved in most social media applications. For example, when searching for a particular artist on iTunes, it pulls up albums that include the artist as well as songs performed by the artist. These were not always a one to many relationship. An album could contain multiple artists while an artist performs on multiple albums. Also, a song could be performed by multiple artists and an artist can perform multiple songs. This led to many-to-many relationships between Song and Artist as well as Artist and Album. We noticed the same many-to-many relationship between Song and Album as well. We also knew that we would need a many-to-many relationship between a Playlist and Song. As such, we decided to center the music side of the Database around the Song Entity. The many-to-many relationship between Artist and Album can be derived from the other relationships defined in the design.

When looking for the social media aspect of things, we could easily look at something like Facebook, but with our constraint that a post must contain a playlist. There are users of the application, which was the obvious center point for any social media application. The different things a user could do had to be defined. Users can follow other users, create a post, like a post, comment on a post, and make a playlist. In our current design, the table for "Comment" is left as an Entity table for scalability. In the future, Musicality may want to add a relation between User and Comment for "likes" as most social media allows for liking of comments as well.

E-R Diagram

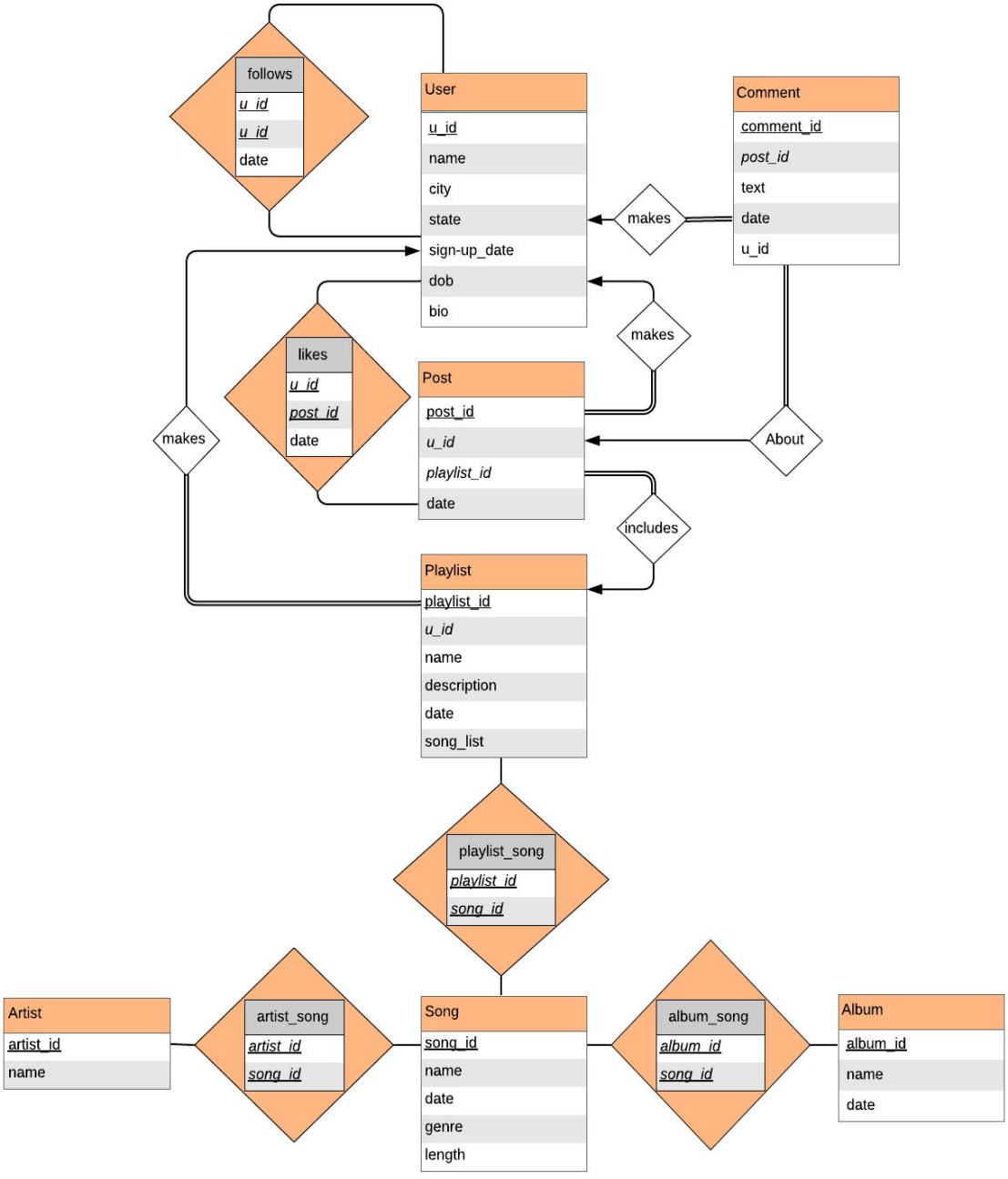


Figure 1. Entity Relationship Diagram

Functional dependencies

The set of non-trivial functional dependencies, F, for each relation are written under each create table statement below. Our database design is in Boyce-Codd normal form (BCNF), as every functional dependency for a given relation is either trivial or derived from a superkey of the relation.

Primary/Foreign Keys

Table	**Primary Key(s)**	**Foreign Key(s)**
User	user_id	NONE
Playlist	playlist_id	user_id
Post	post_id	user_id, playlist_id
Comment	comment_id	post_id
Artist	artist_id	NONE
Song	song_id	NONE
Album	album_id	NONE
Follows	user_id, follows_u_id	user_id, follows_u_id
Likes	user_id, post_id	user_id, post_id
Artist_For	song_id, artist_id	song_id, artist_id
Playlist_Song	playlist_id, song_id	playlist_id, song_id
Album_Song	album_id, song_id	album_id, song_id

Data Definition Statements

User

```
CREATE TABLE user(
    user_id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    city VARCHAR(20) NOT NULL,
    state CHAR(2) NOT NULL,
    signup_date DATE NOT NULL,
    dob DATE NOT NULL,
    bio VARCHAR(250),
    PRIMARY KEY (user_id)
);
```

F = {user_id → (name,city,state,signup_date,dob,bio)}

Playlist

```
CREATE TABLE playlist(
    playlist_id INT NOT NULL AUTO_INCREMENT,
    user_id INT REFERENCES user(user_id),
    name VARCHAR(30) NOT NULL,
    description VARCHAR(250),
CREATE_date DATE,
    PRIMARY KEY (playlist_id)
);
```

F = {playlist_id → (user_id, name, description, creation_date)}

Post

```
CREATE TABLE post(
    post_id INT NOT NULL AUTO_INCREMENT,
    user_id INT REFERENCES user(user_id),
    playlist_id INT REFERENCES playlist(playlist_id),
    date DATE NOT NULL,
    PRIMARY KEY (post_id)
);
```

F = {post_id → (user_id, playlist_id, date)}

Comment

```
CREATE TABLE comment(
    comment_id INT NOT NULL AUTO_INCREMENT,
```

```
    post_id INT REFERENCES post(post_id),
    user_id INT REFERENCES user(user_id),
    text VARCHAR(250) NOT NULL,
    date DATE NOT NULL,
    PRIMARY KEY (comment_id)
  );
```
F = {comment_id → (post_id,user_id,text,date)}

Artist

```
  CREATE TABLE artist(
    artist_id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    PRIMARY KEY (artist_id)
  );
```
F = {artist_id → name}

Song

```
  CREATE TABLE song(
    song_id INT AUTO_INCREMENT NOT NULL,
    title VARCHAR(50) NOT NULL,
    year YEAR NOT NULL,
    genre VARCHAR(20) NOT NULL,
    length TIME NOT NULL,
    PRIMARY KEY (song_id)
  );
```
F = {song_id → (title, year, genre, length)}

Album

```
  CREATE TABLE album(
    album_id INT AUTO_INCREMENT NOT NULL,
    name VARCHAR(60) NOT NULL,
    date DATE NOT NULL,
    PRIMARY KEY (album_id)
  );
```
F = {album_id → (name, date)}

Follows

```
  CREATE TABLE follows(
  user_id INT,
```

```
follows_u_id INT,
date DATE NOT NULL,
FOREIGN KEY (user_id) REFERENCES user(user_id),
FOREIGN KEY (follows_u_id) REFERENCES user(user_id),
PRIMARY KEY (user_id,follows_u_id)
);
```

F = { (user_id, follows_u_id) → date}

Likes

```
CREATE TABLE likes(
    user_id INT,
    post_id INT,
    date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES user(user_id),
    FOREIGN KEY (post_id) REFERENCES post(post_id),
    PRIMARY KEY (user_id, post_id)
);
```

F = { (user_id, post_id) → date}

Artist-for

```
CREATE TABLE artist_for(
    song_id INT,
    artist_id INT,
    FOREIGN KEY (song_id) REFERENCES song(song_id),
    FOREIGN KEY (artist_id) REFERENCES artist(artist_id),
    PRIMARY KEY (song_id, artist_id)
);
```

F does not contain any non-trivial functional dependencies

Playlist_song

```
CREATE TABLE playlist_song(
    playlist_id INT,
    song_id INT,
    FOREIGN KEY (playlist_id) REFERENCES playlist(playlist_id),
    FOREIGN KEY (song_id) REFERENCES song(song_id),
    PRIMARY KEY (playlist_id, song_id)
);
```

F does not contain any non-trivial functional dependencies

Album_song

```
CREATE TABLE album_song(
    album_id INT,
    song_id INT,
    FOREIGN KEY (album_id) REFERENCES album(album_id),
    FOREIGN KEY (song_id) REFERENCES song(song_id),
    PRIMARY KEY (album_id, song_id)
);
```

F does not contain any non-trivial functional dependencies

Sample Insert Statements

User

Insert into user values (NULL, 'Daniel','Baton Rouge', 'Louisiana', STR_TO_DATE('21,5,2013','%d,%m,%Y'), STR_TO_DATE('3,5,1998','%d,%m,%Y'), 'Salutations my fellow music lovers');

Insert into user values (NULL, 'Joe', 'Boston', 'Massachusetts', STR_TO_DATE('16,7,2018','%d,%m,%Y'), STR_TO_DATE('9,9,1968','%d,%m,%Y'), "Music was way much better in my day.");

Playlist

Insert into playlist values (NULL, 1, 'My playlist', "This is a playlist of my favorite songs, hope you enjoy!", STR_TO_DATE('1,7,2019','%d,%m,%Y'));

Insert into playlist values (NULL, 2, 'Playlist for the ages', "Only listen to this playlist if you're prepared to have your socks blown off", STR_TO_DATE('2,7,2019','%d,%m,%Y'));

Post

Insert into post values (NULL,1,1, STR_TO_DATE('1,8,2019','%d,%m,%Y'));

Insert into post values (NULL,1,2, STR_TO_DATE('5,8,2019','%d,%m,%Y'));

Comment

Insert into comment values (NULL, 1, 1, "Thanks for listening!", STR_TO_DATE('7,11,2019','%d,%m,%Y'));

Insert into comment values(NULL, 4, 6, "I listen to this playlist at the gym.", STR_TO_DATE('10,11,2019','%d,%m,%Y'));

Artist

Insert into artist values (NULL, 'Drake');

Insert into artist values (NULL, 'The Beatles');

Song

Insert into song values (NULL,'Hotline Bling', '2016', 'Hip-Hop/Rap', '0:4:27');

Insert into song values (NULL,'Hey Jude', '1968', 'Rock', '0:3:58');

Album

Insert into album values (NULL, 'Views', STR_TO_DATE('29,4,2016','%d,%m,%Y'));

Insert into album values (NULL, 'Hey Jude', STR_TO_DATE('11,5,1975','%d,%m,%Y'));

Follows

Insert into follows values (1, 1, STR_TO_DATE('2,4,2019','%d,%m,%Y');

Insert into follows values (1, 2, STR_TO_DATE('3,4,2019','%d,%m,%Y');

Likes

Insert into likes values (10, 7, STR_TO_DATE('24, 8, 2019','%d,%m,%Y'));

Insert into likes values (4, 12, STR_TO_DATE('7,9,2019','%d,%m,%Y'));

Artist_For

Insert into artist_for values (7, 6);

Insert into artist_for values (8, 7);

Playlist_Song

Insert into playlist_song values (5, 5);

Insert into playlist_song values (5, 10);

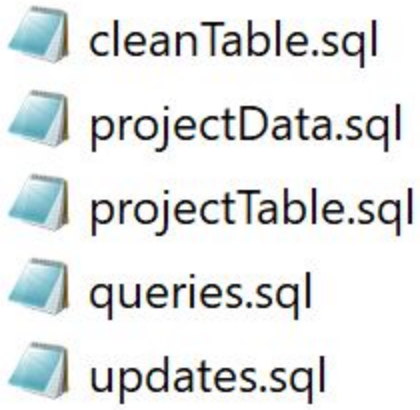
Album_Song

Insert into album_song values (11, 13);

Insert into album_song values (12, 6);

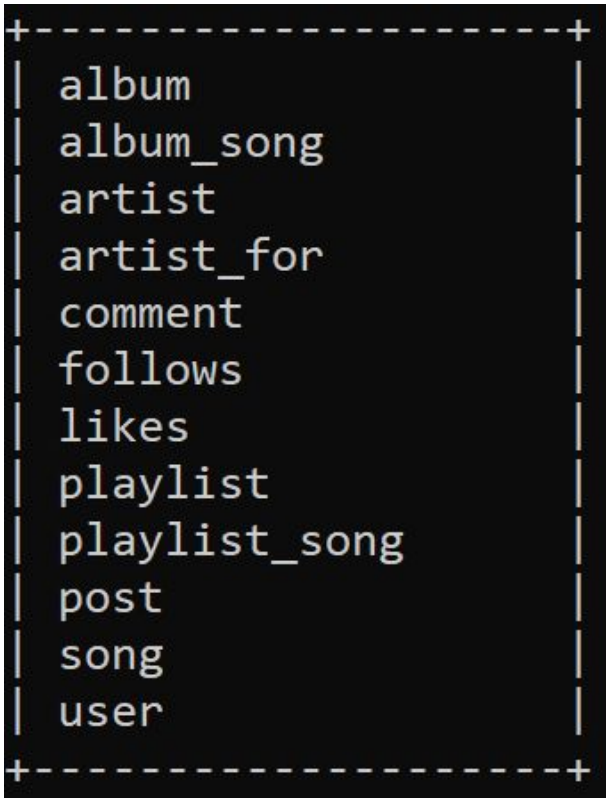
Implementation

To implement and test our design, we used the LSU classes server with MariaDB using some sample data we created. We created five .sql documents for ease of testing.



We first would upload the projectTable.sql file, which contained all of our insert table statements, followed by the projectData.sql file, which contained insert statements for all of our initial sample data. We required the use of the cleanTable.sql document to easily remove all the tables from the database and restart the process when we adjusted and added new data or refined a requirement on the creation of a table. For example, when we discovered the TIME construct in MariaDB for song length we changed our insert statement in the song table from an int type to the TIME construct. We also had to restructure our insert statements to be in the correct format. Having the cleanTable.sql file made testing the changes easy. Once we were certain our sample data was sufficient, we were ready to upload and test the sample update statements in the updates.sql file, followed by the sample queries in queries.sql. Below are the screenshots of our initial sample data, followed by the updates and the results. Finally, we have our sample queries, the results, and why we might need the query in Musicality.

Tables



User:

user_id	name	city	state	signup_date	dob	bio
1	Daniel	Baton Rouge	LA	2013-05-21	1998-05-03	Salutations my fellow music lovers
2	Josh	Encino	CA	2015-07-16	1995-02-01	Drake, where's The Doors?
3	Joe	Boston	MA	2018-07-16	1968-09-09	Music was way much better in my day.
4	Carl	Sacramento	CA	2017-10-17	1999-11-08	Jeffery Epstein didn't kill himself
5	James	Phoenix	AZ	2013-10-13	2000-05-04	Like my playlists please
6	Spencer	Baton Rouge	LA	2017-10-17	1995-04-07	You better call 3872323 right now
7	Jeb	Jackson	MS	2019-09-12	1950-01-03	My name is Jeb but you can call me Jebra
8	Ricky	New York	NY	2017-07-07	1980-02-01	I'm not a pessimist. I'm an optometrist.
9	Ed	Peach Creek	ME	2017-04-08	1995-06-04	Work that body work that body don't you go hurt nobody.
10	Frank	Philadelphia	PA	2017-04-11	1953-04-15	Can I offer you some Gregg Allman in this trying time?
11	Emma	Denver	CO	2017-04-08	1995-06-04	S[he] Be[lie]ve[d]
12	John Fortnite Kennedy	Brookline	MA	2019-01-08	1942-05-06	The ignorance of one voter in a playlist sharing service impairs the security of all

Playlist:

playlist_id	user_id	name	description	creation_date
1	1	My playlist	This is a playlist of my favorite songs, hope you enjoy!	2019-07-01
2	2	Playlist for the ages	Only listen to this playlist if you are prepared to have your socks blown off	2019-07-02
3	3	Best of rock	A playlist for true rock & roll lovers	2019-07-03
4	4	Worst of Becky	This is a playlist of all the songs that annoy my ex-girlfriend Becky. Screw you Becky.	2019-07-04
5	5	Inspiration	A playlist of songs that inspire you to work.	2019-07-05
6	6	Hell	That playlist your hear at the gates of hell.	2019-07-06
7	7	Best of pop	A playlist that contains only the best songs that pop has to offer.	2019-07-07
8	8	Gold	Songs I like to play for my pet goldfish.	2019-07-08
9	8	Golden memories	Songs I like to play in remembrance of my pet goldfish	2019-07-13
10	9	Golden	Dedicated to my professor	2019-07-14
11	10	How do you cre	How do you create a playlist	2019-07-15
12	11	My favorite songs	How do you create a playlist	2019-07-16

Post:

post_id	user_id	playlist_id	date
1	1	1	2019-08-01
2	1	2	2019-08-05
3	2	3	2019-08-11
4	3	4	2019-09-11
5	4	5	2019-09-08
6	4	6	2019-09-19
7	4	7	2019-09-30
8	5	8	2019-06-30
9	6	9	2019-06-19
10	7	10	2019-05-20
11	8	11	2019-04-20
12	9	12	2019-04-17
13	10	13	2019-04-14
14	11	14	2019-04-04
15	12	15	2019-04-01

Comment:

comment_id	post_id	user_id	text	date
1	1	2	Anyone else listening to this from Australia?	2019-12-05
2	1	1	Thanks for listening!	2019-11-07
3	1	2	I didn’t say it was good, idiot.	2019-11-08
4	2	4	Nice playlist!	2019-11-06
5	2	5	I like the songs in this playlist.	2019-11-07
6	3	5	This playlist SUCKS!	2019-11-09
7	4	6	I listen to this playlist at the gym.	2019-11-10
8	4	1	This playlist brought back a previous heart condition I had.	2019-11-15
9	5	3	My child loves this playlist!	2019-09-01
10	5	4	Yeah, because only children can find enjoyment out of these songs.	2019-09-02
11	6	3	Yeah it’s okay, I guess.	2019-09-03
12	6	10	This playlist is the definition of adequate.	2019-09-04
13	7	12	First!	2019-09-05
14	7	2	Did you really think commenting that was a good use of your time?	2019-09-06
15	8	3	Very good playlist, nice job!	2019-09-07
16	8	4	A good playlist.	2019-09-08
17	9	1	This is such a fun playist!	2019-09-09
18	9	2	I think you meant playlist.	2019-09-10
19	9	1	Yeah, no kidding.	2019-09-11
20	9	8	Be sensitive, I`m grieving.	2019-09-11
21	10	8	Listening to this on my ride to work!	2019-09-12
22	10	5	Hope you typed that out at a red light then.	2019-09-13
23	10	8	Nope!	2019-09-14
24	10	6	r/madlads	2019-09-15
25	10	5	r/ihavereddit	2019-09-16
26	11	9	My husbands love listening to this!	2019-09-17
27	12	10	Good job on this!	2019-09-18
28	12	11	Nice.	2019-09-19
29	13	5	Sounds like what was played at my dad's funeral	2019-09-20
30	14	6	Anyone else never want to listen to this playlist again?	2019-09-21
31	9	11	Oh no! Not your wonderful goldfish!	2019-09-12
32	9	8	Yeah... Goldy Goldenfin was special. I miss him.	2019-09-12
33	9	11	I am soooooooo sorry for your loss.	2019-09-13
34	15	3	I didn`t like it.	2019-09-22

Artist:

artist_id	name
1	Drake
2	Taylor Swift
3	Rihanna
4	Richard Wagner
5	A Great Big World
6	Aqua
7	Rick Astley
8	Patrick Wilson
9	Three Days Grace
10	Aerosmith
11	The Beatles
12	Def Leppard
13	Bruno Mars
14	Justin Timberlake
15	Chris Stapleton
16	Emmy Rossum
17	Pink Floyd
18	Radiohead
19	Bowling For Soup
20	Pinkfong
21	Queen
22	Andrew Lloyd Webber

Song:

song_id	title	year	genre	length
1	Hotline Bling	2016	Hip-Hop/Rap	00:04:27
2	Only Girl (In the World)	2010	Pop	00:03:55
3	Ride of the Valkyries	1979	soundtrack	00:05:31
4	Say Something	2018	Pop	00:04:38
5	Barbie Girl	1997	Pop	00:03:17
6	Never Gonna Give You Up	1987	Pop	00:03:32
7	Think of Me	2004	soundtrack	00:03:39
8	Last to Know	2009	Alt. Rock	00:03:27
9	Walk This Way	1975	Rock	00:03:40
10	Hey Jude	1968	Rock	00:03:58
11	Pour Some Sugar on Me	1987	Classic Rock	00:04:27
12	Shake It Off	2014	Pop	00:03:39
13	Bad Blood	2014	Pop	00:03:31
14	Comfortably Numb	1979	Progressive Rock	00:06:23
15	Paranoid Android	1997	Alt. Rock	00:06:27
16	1985	2004	Alt. Rock	00:03:13
17	Baby Shark	2016	Childrens	00:01:46
18	Girl All the Bad Guys Want	2002	Alt. Rock	00:03:33
19	High School Never Ends	2006	Alt. Rock	00:03:29
20	We Will Rock You	1977	Classic Rock	00:02:02
21	Bohemian Rhapsody	1975	Classic Rock	00:05:55
22	Bicycle Race	1978	Classic Rock	00:03:01
23	Fat Bottom Girls	1978	Classic Rock	00:04:16
24	What`s My Name	2010	Pop	00:04:24

Album:

album_id	name	date
1	Views	2016-04-29
2	1989	2014-10-27
3	Loud	2010-11-12
4	The Ride of the Valkyries	1988-11-23
5	Aquarium	1997-03-26
6	Whenever You Need Somebody	1987-11-16
7	The Phantom of the Opera	2004-11-23
8	Life Starts Now	2009-09-22
9	Toys in the Attic	1975-04-08
10	Hey Jude	1975-05-11
11	Hysteria	1987-09-08
12	The Wall	1979-11-30
13	OK Computer	1997-05-21
14	Comfortably Numb	1980-06-23
15	A Hangover You Don`t Deserve	2004-09-14
16	The Great Burrito Extortion Case	2006-11-07
17	Drunk Enough to Dance	2002-08-20
18	Pinkfong Animal Songs	2017-07-27
19	Jazz	1978-11-10
20	A Night at the Opera	1975-11-21
21	News of the World	1977-10-28
22	The Platinum Collection (Greatest Hits I, II, & III)	2000-11-13

Follows:

user_id	follows_u_id	date
1	1	2019-04-02
1	2	2019-04-03
1	3	2019-04-04
2	1	2019-04-05
3	3	2019-04-06
3	4	2019-04-07
4	3	2019-04-08
5	4	2019-04-09
5	6	2019-04-10
5	7	2019-04-11
5	8	2019-04-12
6	8	2019-04-13
7	9	2019-04-14
8	10	2019-04-15
8	11	2019-04-16
9	12	2019-05-19
10	1	2019-05-21
10	2	2019-05-22
10	3	2019-05-13
10	12	2019-05-20
11	12	2019-05-29

Likes:

user_id	post_id	date
1	3	2019-04-12
1	13	2019-07-15
2	4	2019-05-12
2	12	2019-02-28
3	6	2019-05-27
3	7	2019-11-09
4	12	2019-09-07
4	13	2019-03-16
5	10	2019-05-29
5	15	2019-10-09
7	5	2019-06-30
8	8	2019-08-01
10	3	2019-06-02
10	7	2019-08-24
12	10	2019-10-20

Artist_For:

song_id	artist_id
1	1
2	3
3	4
4	5
5	6
6	7
7	8
7	16
7	22
8	9
9	10
10	11
11	12
12	2
13	2
14	17
15	18
16	19
17	20
18	19
19	19
20	21
21	21
22	21
23	21
24	1
24	3

Playlist_Song:

playlist_id	song_id
1	1
1	2
1	3
2	1
2	2
2	4
3	3
3	4
3	5
4	6
4	7
4	8
5	5
5	7
5	10
6	1
6	3
6	17
7	2
7	5
7	8
8	5
8	9
8	10
9	5
9	10
9	11
9	20
9	21
9	22
9	23
10	1
10	4
10	7

Album_Song:

album_id	song_id
1	1
2	12
2	13
3	2
3	24
4	3
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	15
13	15
15	16
16	19
17	18
18	17
19	22
19	23
20	21
21	20
22	20
22	21
22	22
22	23

Updates

Change all songs with genre "Alt. Rock" to "Alternative Rock".

```
UPDATE song
SET genre = "Alternative Rock"
WHERE song.genre = "Alt. Rock";
```

```
Query OK, 5 rows affected (0.001 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

song_id	title	year	genre	length
1	Hotline Bling	2016	Hip-Hop/Rap	00:04:27
2	Only Girl (In the World)	2010	Pop	00:03:55
3	Ride of the Valkyries	1979	soundtrack	00:05:31
4	Say Something	2018	Pop	00:04:38
5	Barbie Girl	1997	Pop	00:03:17
6	Never Gonna Give You Up	1987	Pop	00:03:32
7	Think of Me	2004	soundtrack	00:03:39
8	Last to Know	2009	Alternative Rock	00:03:27
9	Walk This Way	1975	Rock	00:03:40
10	Hey Jude	1968	Rock	00:03:58
11	Pour Some Sugar on Me	1987	Classic Rock	00:04:27
12	Shake It Off	2014	Pop	00:03:39
13	Bad Blood	2014	Pop	00:03:31
14	Comfortably Numb	1979	Progressive Rock	00:06:23
15	Paranoid Android	1997	Alternative Rock	00:06:27
16	1985	2004	Alternative Rock	00:03:13
17	Baby Shark	2016	Childrens	00:01:46
18	Girl All the Bad Guys Want	2002	Alternative Rock	00:03:33
19	High School Never Ends	2006	Alternative Rock	00:03:29
20	We Will Rock You	1977	Classic Rock	00:02:02
21	Bohemian Rhapsody	1975	Classic Rock	00:05:55
22	Bicycle Race	1978	Classic Rock	00:03:01
23	Fat Bottom Girls	1978	Classic Rock	00:04:16
24	What`s My Name	2010	Pop	00:04:24

Update user 3's city and state to New Orleans, LA.

```
UPDATE user
SET city = "New Orleans", state = "LA"
WHERE user.user_id = 3;
```

```
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

user_id	name	city	state	signup_date	dob
1	Daniel	Baton Rouge	LA	2013-05-21	1998-05-03
2	Josh	Encino	CA	2015-07-16	1995-02-01
3	Joe	New Orleans	LA	2018-07-16	1968-09-09
4	Carl	Sacramento	CA	2017-10-17	1999-11-08
5	James	Phoenix	AZ	2013-10-13	2000-05-04
6	Spencer	Baton Rouge	LA	2017-10-17	1995-04-07
7	Jeb	Jackson	MS	2019-09-12	1950-01-03
8	Ricky	New York	NY	2017-07-07	1980-02-01
9	Ed	Peach Creek	ME	2017-04-08	1995-06-04
10	Frank	Philadelphia	PA	2017-04-11	1953-04-15
11	Emma	Denver	CO	2017-04-08	1995-06-04
12	John Fortnite Kennedy	Brookline	MA	2019-01-08	1942-05-06

Update playlist description, name for playlist 11.

UPDATE playlist
SET description = "I figured out how to create a new playlist!", name = "My new playlist"
WHERE playlist.playlist_id = 11;

```
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

playlist_id	user_id	name	description	creation_date
1	1	My playlist	This is a playlist of my favorite songs, hope you enjoy!	2019-07-01
2	2	Playlist for the ages	Only listen to this playlist if you are prepared to have your socks blown off	2019-07-02
3	3	Best of rock	A playlist for true rock & roll lovers	2019-07-03
4	4	Worst of Becky	This is a playlist of all the songs that annoy my ex-girlfriend Becky. Screw you Becky.	2019-07-04
5	5	Inspiration	A playlist of songs that inspire you to work.	2019-07-05
6	6	Hell	That playlist your hear at the gates of hell.	2019-07-06
7	7	Best of pop	A playlist that contains only the best songs that pop has to offer.	2019-07-07
8	8	Gold	Songs I like to play for my pet goldfish.	2019-07-08
9	8	Golden memories	Songs I like to play in remembrance of my pet goldfish	2019-07-13
10	9	Golden	Dedicated to my professor	2019-07-14
11	10	My new playlist	I figured out how to create a new playlist!	2019-07-15
12	11	My favorite songs	How do you create a playlist	2019-07-16

Update comment text for comment_id number 5 to "EDIT: Thanks for the gold, kind stranger!"

UPDATE comment
SET text = CONCAT(text, " EDIT: Thanks for the gold, kind stranger!")
WHERE comment.comment_id = 5;

```
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

comment_id	post_id	user_id	text	date
1	1	2	Anyone else listening to this from Australia?	2019-12-05
2	1	1	Thanks for listening!	2019-11-07
3	1	2	I didn’t say it was good, idiot.	2019-11-08
4	2	4	Nice playlist!	2019-11-06
5	2	5	I like the songs in this playlist. EDIT: Thanks for the gold, kind stranger!	2019-11-07
6	3	5	This playlist SUCKS!	2019-11-09
7	4	6	I listen to this playlist at the gym.	2019-11-10
8	4	1	This playlist brought back a previous heart condition I had.	2019-11-15
9	5	3	My child loves this playlist!	2019-09-01
10	5	4	Yeah, because only children can find enjoyment out of these songs.	2019-09-02
11	6	3	Yeah it’s okay, I guess.	2019-09-03
12	6	10	This playlist is the definition of adequate.	2019-09-04
13	7	12	First!	2019-09-05
14	7	2	Did you really think commenting that was a good use of your time?	2019-09-06
15	8	3	Very good playlist, nice job!	2019-09-07
16	8	4	A good playlist.	2019-09-08
17	9	1	This is such a fun playist!	2019-09-09
18	9	2	I think you meant playlist.	2019-09-10
19	9	1	Yeah, no kidding.	2019-09-11
20	9	8	Be sensitive, I`m grieving.	2019-09-11
21	10	8	Listening to this on my ride to work!	2019-09-12
22	10	5	Hope you typed that out at a red light then.	2019-09-13
23	10	8	Nope!	2019-09-14
24	10	6	r/madlads	2019-09-15
25	10	5	r/ihavereddit	2019-09-16
26	11	9	My husbands love listening to this!	2019-09-17
27	12	10	Good job on this!	2019-09-18
28	12	11	Nice.	2019-09-19
29	13	5	Sounds like what was played at my dad's funeral	2019-09-20
30	14	6	Anyone else never want to listen to this playlist again?	2019-09-21
31	9	11	Oh no! Not your wonderful goldfish!	2019-09-12
32	9	8	Yeah... Goldy Goldenfin was special. I miss him.	2019-09-12
33	9	11	I am sooooooooo sorry for your loss.	2019-09-13
34	15	3	I didn`t like it.	2019-09-22

Update user bio for user 1 to "Check out my playlists…or else!"

UPDATE user
SET bio = "Check out my playlists...or else!"
WHERE user.user_id = 1;

```
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

user_id	name	city	state	signup_date	dob	bio
1	Daniel	Baton Rouge	LA	2013-05-21	1998-05-03	Check out my playlists...or else!
2	Josh	Encino	CA	2015-07-16	1995-02-01	Drake, where’s The Doors?
3	Joe	New Orleans	LA	2018-07-16	1968-09-09	Music was way much better in my day.
4	Carl	Sacramento	CA	2017-10-17	1999-11-08	Jeffery Epstein didn’t kill himself
5	James	Phoenix	AZ	2013-10-13	2000-05-04	Like my playlists please
6	Spencer	Baton Rouge	LA	2017-10-17	1995-04-07	You better call 3872323 right now
7	Jeb	Jackson	MS	2019-09-12	1950-01-03	My name is Jeb but you can call me Jebra
8	Ricky	New York	NY	2017-07-07	1980-02-01	I’m not a pessimist. I’m an optometrist.
9	Ed	Peach Creek	ME	2017-04-08	1995-06-04	Work that body work that body don’t you go hurt nobody.
10	Frank	Philadelphia	PA	2017-04-11	1953-04-15	Can I offer you some Gregg Allman in this trying time?
11	Emma	Denver	CO	2017-04-08	1995-06-04	S[he] Be[lie]ve[d]
12	John Fortnite Kennedy	Brookline	MA	2019-01-08	1942-05-06	The ignorance of one voter in a playlist sharing service impairs the security of all

Update the song "Comfortably Numb" to be in album "The Wall"

```
UPDATE album_song
SET song_id = (SELECT song_id FROM song WHERE title = "Comfortably Numb")
WHERE album_id = (SELECT album_id FROM album WHERE name = "The Wall");
```

```
Query OK, 1 row affected (0.000 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

album_id	song_id
1	1
2	12
2	13
3	2
3	24
4	3
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	14
13	15
15	16
16	19
17	18
18	17
19	22
19	23
20	21
21	20
22	20
22	21
22	22
22	23

Queries

Get all songs in playlist 1, including album name and artist name.

This query would be used to show a user the playlist contents.

```
SELECT song.title, artist.name AS artist_name, album.name AS album_name, song.length
FROM song
NATURAL JOIN artist_for
INNER JOIN artist ON artist_for.artist_id = artist.artist_id
NATURAL JOIN album_song
INNER JOIN album ON album_song.album_id = album.album_id
WHERE song.song_id IN (SELECT playlist_song.song_id FROM playlist_song WHERE
playlist_song.playlist_id = 1);
```

title	artist_name	album_name	length
Hotline Bling	Drake	Views	00:04:27
Only Girl (In the World)	Rihanna	Loud	00:03:55
Ride of the Valkyries	Richard Wagner	The Ride of the Valkyries	00:05:31

3 rows in set (0.065 sec)

Get the name and length of all playlists, and the name of the user who owns each playlist.

This query would be used to list playlist in order of time it might take to listen to in its entirety. A user may want this as they could be looking for a good playlist for a specific timed event, like their commute to work or a workout.

```
SELECT playlist.name, user.name, (SELECT
SEC_TO_TIME(SUM(TIME_TO_SEC(song.length)))) AS playlist_length
FROM playlist_song
INNER JOIN playlist
ON playlist_song.playlist_id = playlist.playlist_id
INNER JOIN song
ON playlist_song.song_id = song.song_id
INNER JOIN user
ON playlist.user_id = user.user_id
GROUP BY playlist.playlist_id
ORDER BY playlist_length DESC;
```

```
+-----------------------+---------+-----------------+
| name                  | name    | playlist_length |
+-----------------------+---------+-----------------+
| Golden memories       | Ricky   | 00:26:56        |
| My playlist           | Daniel  | 00:13:53        |
| Best of rock          | Joe     | 00:13:26        |
| Playlist for the ages | Josh    | 00:13:00        |
| Golden                | Ed      | 00:12:44        |
| Hell                  | Spencer | 00:11:44        |
| Gold                  | Ricky   | 00:10:55        |
| Inspiration           | James   | 00:10:54        |
| Best of pop           | Jeb     | 00:10:39        |
| Worst of Becky        | Carl    | 00:10:38        |
+-----------------------+---------+-----------------+
10 rows in set (0.000 sec)
```

Get playlist_id, playlist name, and the name of the user who created playlist that contains a song from Drake.

This query can be adapted to any artist. A user may want to search for particular playlist that contain their favorite artists. Giving the playlist name and creator can help the user make a decision on which playlist they would want to observe.

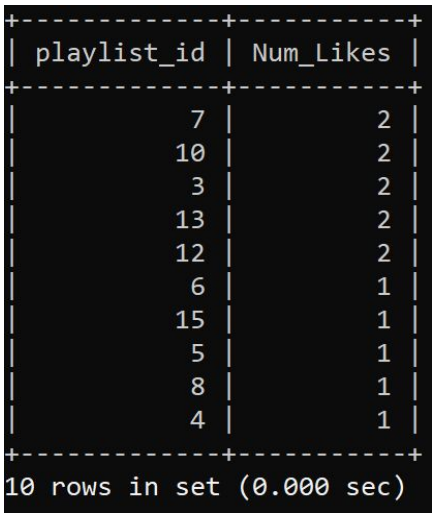
```
SELECT playlist.playlist_id, playlist.name, user.name
FROM playlist_song
INNER JOIN playlist
ON playlist_song.playlist_id = playlist.playlist_id
INNER JOIN artist_for
ON playlist_song.song_id = artist_for.song_id
INNER JOIN artist
ON artist_for.artist_id = artist.artist_id
INNER JOIN user
ON playlist.user_id = user.user_id
WHERE artist.name = "Drake";
```

```
+-------------+-----------------------+---------+
| playlist_id | name                  | name    |
+-------------+-----------------------+---------+
|           1 | My playlist           | Daniel  |
|           2 | Playlist for the ages | Josh    |
|           6 | Hell                  | Spencer |
|          10 | Golden                | Ed      |
+-------------+-----------------------+---------+
4 rows in set (0.000 sec)
```

Get a list of playlist_id sorted by total number of likes, in descending order.

This query would be useful to sort the most liked playlist on the platform. We could easily create a Top 10 Playlists page for users to see what is most popular.

```
SELECT post.playlist_id, COUNT(likes.post_id) AS
Num_Likes
FROM likes
INNER JOIN post
ON likes.post_id = post.post_id
GROUP BY post.playlist_id
ORDER BY Num_Likes DESC;
```



playlist_id	Num_Likes
7	2
10	2
3	2
13	2
12	2
6	1
15	1
5	1
8	1
4	1

10 rows in set (0.000 sec)

Get songs that are in the most playlists, including artist and album name

This is a metric that would show which song is the most popular addition to any playlist. This could help us create a Top 10 Songs page for users to see which songs are popular at a given moment.

```
SELECT song.song_id, song.title, artist.name AS Artist_Name, album.name AS
Album_Name, COUNT(DISTINCT playlist_id) AS Num_Playlists
FROM playlist_song
NATURAL JOIN song
NATURAL JOIN artist_for
INNER JOIN artist
ON artist_for.artist_id = artist.artist_id
NATURAL JOIN album_song
INNER JOIN album
ON album_song.album_id = album.album_id
GROUP BY (song.song_id)
ORDER BY Num_Playlists DESC;
```

song_id	title	Artist_Name	Album_Name	Num_Playlists
5	Barbie Girl	Aqua	Aquarium	5
1	Hotline Bling	Drake	Views	4
10	Hey Jude	The Beatles	Hey Jude	3
7	Think of Me	Patrick Wilson	The Phantom of the Opera	3
3	Ride of the Valkyries	Richard Wagner	The Ride of the Valkyries	3
2	Only Girl (In the World)	Rihanna	Loud	3
8	Last to Know	Three Days Grace	Life Starts Now	2
22	Bicycle Race	Queen	Jazz	1
17	Baby Shark	Pinkfong	Pinkfong Animal Songs	1
9	Walk This Way	Aerosmith	Toys in the Attic	1
6	Never Gonna Give You Up	Rick Astley	Whenever You Need Somebody	1
21	Bohemian Rhapsody	Queen	The Platinum Collection (Greatest Hits I, II, & III)	1
11	Pour Some Sugar on Me	Def Leppard	Hysteria	1
23	Fat Bottom Girls	Queen	Jazz	1
20	We Will Rock You	Queen	The Platinum Collection (Greatest Hits I, II, & III)	1

15 rows in set (0.042 sec)

Get the artists that are in the most playlists.

This query would get a metric for the most popular artist in a playlist. It could be used to create a Top 10 Artists page for users to discover the popular artists of any given moment.

SELECT artist.artist_id, artist.name AS Artist_Name, COUNT(DISTINCT playlist_id) as Num_Playlists
FROM playlist_song
NATURAL JOIN artist_for
INNER JOIN artist
ON artist_for.artist_id = artist.artist_id
GROUP BY (artist.artist_id)
ORDER BY Num_Playlists DESC;

artist_id	Artist_Name	Num_Playlists
6	Aqua	5
1	Drake	4
4	Richard Wagner	3
5	A Great Big World	3
22	Andrew Lloyd Webber	3
11	The Beatles	3
16	Emmy Rossum	3
8	Patrick Wilson	3
3	Rihanna	3
9	Three Days Grace	2
20	Pinkfong	1
21	Queen	1
10	Aerosmith	1
12	Def Leppard	1
7	Rick Astley	1

15 rows in set (0.000 sec)

Get the albums in the most playlists

This would get the albums that are the most popular, and show up the most often in current playlists. It could be used as a Top 10 Albums metric.

```
SELECT album.album_id, album.name AS Album_Name, artist.name AS Artist_Name,
COUNT(DISTINCT playlist_id) AS Num_Playlists
FROM playlist_song
NATURAL JOIN artist_for
INNER JOIN artist
ON artist_for.artist_id = artist.artist_id
NATURAL JOIN album_song
INNER JOIN album
ON album_song.album_id = album.album_id
GROUP BY (album.album_id)
ORDER BY Num_Playlists DESC;
```

album_id	Album_Name	Artist_Name	Num_Playlists
5	Aquarium	Aqua	5
1	Views	Drake	4
3	Loud	Rihanna	3
10	Hey Jude	The Beatles	3
4	The Ride of the Valkyries	Richard Wagner	3
7	The Phantom of the Opera	Andrew Lloyd Webber	3
8	Life Starts Now	Three Days Grace	2
20	A Night at the Opera	Queen	1
18	Pinkfong Animal Songs	Pinkfong	1
21	News of the World	Queen	1
6	Whenever You Need Somebody	Rick Astley	1
19	Jazz	Queen	1
11	Hysteria	Def Leppard	1
9	Toys in the Attic	Aerosmith	1
22	The Platinum Collection (Greatest Hits I, II, & III)	Queen	1

15 rows in set (0.000 sec)

Get all song titles, artist name, album name, and genre from songs that exceed 2 minutes.

This query shows how the relations all work together to show complete song information that is also filterable by different things. In this example, we want songs longer than 2 minutes, but this query could be adjusted to sort by or filter by any of these attributes.

```
SELECT song.song_id, song.title, artist.name AS Artist_Name, album.name AS
Album_Name, song.genre, song.length
FROM song
NATURAL JOIN artist_for
INNER JOIN artist
ON artist_for.artist_id = artist.artist_id
NATURAL JOIN album_song
INNER JOIN album
ON album_song.album_id = album.album_id
WHERE song.length > '0:2:00';
```

song_id	title	Artist_Name	Album_Name	genre	length
1	Hotline Bling	Drake	Views	Hip-Hop/Rap	00:04:27
24	What`s My Name	Drake	Loud	Pop	00:04:24
12	Shake It Off	Taylor Swift	1989	Pop	00:03:39
13	Bad Blood	Taylor Swift	1989	Pop	00:03:31
2	Only Girl (In the World)	Rihanna	Loud	Pop	00:03:55
24	What`s My Name	Rihanna	Loud	Pop	00:04:24
3	Ride of the Valkyries	Richard Wagner	The Ride of the Valkyries	soundtrack	00:05:31
5	Barbie Girl	Aqua	Aquarium	Pop	00:03:17
6	Never Gonna Give You Up	Rick Astley	Whenever You Need Somebody	Pop	00:03:32
7	Think of Me	Patrick Wilson	The Phantom of the Opera	soundtrack	00:03:39
8	Last to Know	Three Days Grace	Life Starts Now	Alternative Rock	00:03:27
9	Walk This Way	Aerosmith	Toys in the Attic	Rock	00:03:40
10	Hey Jude	The Beatles	Hey Jude	Rock	00:03:58
11	Pour Some Sugar on Me	Def Leppard	Hysteria	Classic Rock	00:04:27
7	Think of Me	Emmy Rossum	The Phantom of the Opera	soundtrack	00:03:39
14	Comfortably Numb	Pink Floyd	The Wall	Progressive Rock	00:06:23
15	Paranoid Android	Radiohead	OK Computer	Alternative Rock	00:06:27
16	1985	Bowling For Soup	A Hangover You Don`t Deserve	Alternative Rock	00:03:13
18	Girl All the Bad Guys Want	Bowling For Soup	Drunk Enough to Dance	Alternative Rock	00:03:33
19	High School Never Ends	Bowling For Soup	The Great Burrito Extortion Case	Alternative Rock	00:03:29
20	We Will Rock You	Queen	News of the World	Classic Rock	00:02:02
20	We Will Rock You	Queen	The Platinum Collection (Greatest Hits I, II, & III)	Classic Rock	00:02:02
21	Bohemian Rhapsody	Queen	A Night at the Opera	Classic Rock	00:05:55
21	Bohemian Rhapsody	Queen	The Platinum Collection (Greatest Hits I, II, & III)	Classic Rock	00:05:55
22	Bicycle Race	Queen	Jazz	Classic Rock	00:03:01
22	Bicycle Race	Queen	The Platinum Collection (Greatest Hits I, II, & III)	Classic Rock	00:03:01
23	Fat Bottom Girls	Queen	Jazz	Classic Rock	00:04:16
23	Fat Bottom Girls	Queen	The Platinum Collection (Greatest Hits I, II, & III)	Classic Rock	00:04:16
7	Think of Me	Andrew Lloyd Webber	The Phantom of the Opera	soundtrack	00:03:39

29 rows in set (0.000 sec)

Get all songs that have multiple artists.

This query is solely to show the need for the song to artist relation table. In our sample data alone, there are two songs with multiple artists attached to them.

```
SELECT song.song_id, song.title, COUNT(DISTINCT artist_id) AS Num_Artists
FROM artist_for
NATURAL JOIN song
GROUP BY (song_id)
HAVING Num_Artists > 1;
```

song_id	title	Num_Artists
7	Think of Me	3
24	What`s My Name	2

2 rows in set (0.074 sec)

Get a list of artists sorted by the number of albums they have.

This can easily show the total number of albums an artist has contributed to. A user may want to see the most prolific artist from which to create a playlist.

```
SELECT artist_id, name, COUNT(DISTINCT album_id) AS Num_Albums
FROM artist_for
NATURAL JOIN artist
NATURAL JOIN album_song
GROUP BY (artist_id)
ORDER BY Num_Albums DESC;
```

artist_id	name	Num_Albums
21	Queen	4
19	Bowling For Soup	3
1	Drake	2
17	Pink Floyd	1
9	Three Days Grace	1
3	Rihanna	1
18	Radiohead	1
10	Aerosmith	1
4	Richard Wagner	1
11	The Beatles	1
6	Aqua	1
20	Pinkfong	1
12	Def Leppard	1
7	Rick Astley	1
16	Emmy Rossum	1
8	Patrick Wilson	1
2	Taylor Swift	1
22	Andrew Lloyd Webber	1

18 rows in set (0.000 sec)

Get the 5 genres with the most songs, displaying the genre and number of songs in that genre.

This could be used to choose which genres to display first to the users. The genre with more songs has more variety to create playlists within.

```
SELECT genre, COUNT(song_id) AS Num_Songs
FROM song
GROUP BY genre
ORDER BY Num_Songs DESC LIMIT 5;
```

```
+-------------------+-----------+
| genre             | Num_Songs |
+-------------------+-----------+
| Pop               |         7 |
| Alternative Rock  |         5 |
| Classic Rock      |         5 |
| soundtrack        |         2 |
| Rock              |         2 |
+-------------------+-----------+
5 rows in set (0.000 sec)
```

Get all comments for a post_id 9.

This query would be used to display all the comments for the post under the post itself.

```
SELECT name, comment.*
FROM comment
NATURAL JOIN user
WHERE post_id = 9;
```

```
+--------+------------+---------+---------+-----------------------------------------------+------------+
| name   | comment_id | post_id | user_id | text                                          | date       |
+--------+------------+---------+---------+-----------------------------------------------+------------+
| Daniel |         17 |       9 |       1 | This is such a fun playist!                   | 2019-09-09 |
| Josh   |         18 |       9 |       2 | I think you meant playlist.                   | 2019-09-10 |
| Daniel |         19 |       9 |       1 | Yeah, no kidding.                             | 2019-09-11 |
| Ricky  |         20 |       9 |       8 | Be sensitive, I`m grieving.                   | 2019-09-11 |
| Emma   |         31 |       9 |      11 | Oh no! Not your wonderful goldfish!           | 2019-09-12 |
| Ricky  |         32 |       9 |       8 | Yeah... Goldy Goldenfin was special.  I miss him. | 2019-09-12 |
| Emma   |         33 |       9 |      11 | I am sooooooooo sorry for your loss.          | 2019-09-13 |
+--------+------------+---------+---------+-----------------------------------------------+------------+
7 rows in set (0.000 sec)
```

Get all comments from user 1, Daniel.

This query would be useful for a User to see their own contributions to the platform, as well as an administrator reviewing a report on the user in question.

```
SELECT name, comment.*
FROM comment
NATURAL JOIN user
WHERE user_id = 1;
```

```
+--------+------------+---------+---------+-----------------------------------------------------------+------------+
| name   | comment_id | post_id | user_id | text                                                      | date       |
+--------+------------+---------+---------+-----------------------------------------------------------+------------+
| Daniel |          2 |       1 |       1 | Thanks for listening!                                     | 2019-11-07 |
| Daniel |          8 |       4 |       1 | This playlist brought back a previous heart condition I had. | 2019-11-15 |
| Daniel |         17 |       9 |       1 | This is such a fun playist!                               | 2019-09-09 |
| Daniel |         19 |       9 |       1 | Yeah, no kidding.                                         | 2019-09-11 |
+--------+------------+---------+---------+-----------------------------------------------------------+------------+
4 rows in set (0.000 sec)
```