```
CREATE TABLE Artist (

artist_id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE Genre (

genre_id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE Album (

album_id INT AUTO_INCREMENT PRIMARY KEY,

title VARCHAR(150) NOT NULL,

artist_id INT NOT NULL,

release_date DATE NOT NULL,

UNIQUE (title, artist_id),

FOREIGN KEY (artist_id) REFERENCES Artist(artist_id)

);

CREATE TABLE Song (

song_id INT AUTO_INCREMENT PRIMARY KEY,

title VARCHAR(100) NOT NULL,

artist_id INT NOT NULL,

album_id INT,

release_date DATE NOT NULL,

UNIQUE (title, artist_id),

FOREIGN KEY (artist_id) REFERENCES Artist(artist_id),

FOREIGN KEY (album_id) REFERENCES Album(album_id)

);
```

```
CREATE TABLE Genre_Song(

genre_id INT NOT NULL,

song_id INT NOT NULL,

FOREIGN KEY (genre_id) REFERENCES Genre(genre_id),

FOREIGN KEY (song_id) REFERENCES Song(song_id)

);

CREATE TABLE User (

user_id INT AUTO_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE Playlist (

playlist_id INT AUTO_INCREMENT PRIMARY KEY,

title VARCHAR(100) NOT NULL,

user_id INT NOT NULL,

created_at DATETIME NOT NULL,

UNIQUE (title, user_id),

FOREIGN KEY (user_id) REFERENCES User(user_id)

);

CREATE TABLE Playlist_Song (

playlist_song_id INT AUTO_INCREMENT PRIMARY KEY,

playlist_id INT NOT NULL,

song_id INT NOT NULL,

FOREIGN KEY (playlist_id) REFERENCES Playlist(playlist_id),

FOREIGN KEY (song_id) REFERENCES Song(song_id)

);
```

```
CREATE TABLE Rating_Song(

rating_id INT AUTO_INCREMENT PRIMARY KEY,

user_id INT NOT NULL,

song_id INT NOT NULL,

rating_value TINYINT NOT NULL CHECK (rating_value BETWEEN 1 AND 5),

rating_date DATE NOT NULL,

UNIQUE (user_id, song_id),

FOREIGN KEY (song_id) REFERENCES Song(song_id)

FOREIGN KEY (user_id) REFERENCES User(user_id)

);

CREATE TABLE Rating_Playlist(

rating_id INT AUTO_INCREMENT PRIMARY KEY,

user_id INT NOT NULL,

song_id INT NOT NULL,

rating_value TINYINT NOT NULL CHECK (rating_value BETWEEN 1 AND 5),

rating_date DATE NOT NULL,

UNIQUE (user_id, playlist_id),

FOREIGN KEY (playlist_id) REFERENCES Playlist(playlist_id)

FOREIGN KEY (user_id) REFERENCES User(user_id)

);

CREATE TABLE Rating_Album(

rating_id INT AUTO_INCREMENT PRIMARY KEY,

user_id INT NOT NULL,

album_id INT NOT NULL,

rating_value TINYINT NOT NULL CHECK (rating_value BETWEEN 1 AND 5),

rating_date DATE NOT NULL,
```

UNIQUE (user_id, album_id),

FOREIGN KEY (album_id) REFERENCES Album(album_id)

FOREIGN KEY (user_id) REFERENCES User(user_id)

);

1. Which 3 genres are most represented in terms of number of songs in that genre? The result must have two columns, named genre and number_of_songs.
    a. SELECT g.name as genre, COUNT(gs.song_id) as number_of_songs
    b. FROM genre_song gs, genre g
    c. WHERE gs.genre_id = g.genre_id
    d. GROUP BY gs.genre_id
    e. ORDER BY number_of_songs DESC
    f. LIMIT 3
2. Find names of artists who have songs that are in albums as well as outside of albums (singles). The result must have one column, named artist_name
    a. SELECT DISTINCT(a.name) as artist_name
    b. FROM Artist a, Song s
    c. WHERE a.artist_id = s.artist_id;
3. What were the top 10 most highly rated albums (highest average user rating) in the period 1990-1999? Break ties using alphabetical order of album names. (Period refers to the rating date, NOT the date of release). The result must have two columns, named album_name and average_user_rating.
    a. SELECT a.title as album_name, avg(r.rating_value) as average_user_rating,

b. FROM Album a, Rating_Album ra,
c. WHERE a.album_id = ra.album_id and ra.rating_date BETWEEN ‘1990-01-01' AND '1999-12-31’,
d. GROUP BY a.album_id,
e. ORDER BY average_user_rating DESC, album_name ASC,
f. LIMIT 10;

4. Which were the top 3 most rated genres (this is the number of ratings of songs in genres, not the actual rating scores) in the years 1991-1995? (Years refers to the rating date, NOT the date of release).
   a. SELECT g.name as genre_name, COUNT(rs.song_id) as number_of_songs_ratings
   b. FROM Genre g, Song s, Genre_Song gs, Rating_Song rs
   c. WHERE rs.song_id = s.song_id and s.song_id = gs.song_id and gs.genre_id = g.genre_id and rs.rating_date BETWEEN ‘1990-01-01' AND '1995-12-31’
   d. GROUP BY g.name
   e. LIMIT 3
5. Which users have a playlist that has an average song rating of 4.0 or more? (This is the average of the average song rating for each song in the playlist.) A user may appear multiple times in the result if more than one of their playlists make the cut.
   a. SELECT u.username as username, p.title as playlist_title, AVG(rs.rating_value) as average_song_rating
   b. FROM User u, Playlist p, Playlist_Song ps, Song s, Rating_Song rs
   c. WHERE u.user_id = p.user_id and p.playlist_id = ps.playlist_id and ps.song_id = s.song_id and s.song_id = rs.song_id
   d. GROUP BY p.title
   e. HAVING average_song_rating >= 4

6. Who are the top 5 most engaged users in terms of number of ratings that they have given to songs or albums? (In other words, they have given the most number of ratings to songs or albums combined.)
    a. SELECT u.username as username, COUNT(u.user_id) as number_of_ratings
    b. FROM User u,
        1. (SELECT user_id FROM Rating_Album
        2. UNION
        3. SELECT user_id FROM Rating_Song) as r
    c. WHERE u.user_id = r.user_id
    d. LIMIT 5;

7. Find the top 10 most prolific artists (most number of songs) in the years 1990-2010? Count each song in an album individually.

```
SELECT a.name AS artist_name, COUNT(*) AS number_of_songs
FROM Artist a
JOIN Album al ON a.artist_id = al.artist_id
JOIN Song s ON al.album_id = s.album_id
WHERE al.release_date BETWEEN '1990-01-01' AND '2010-12-31'
GROUP BY a.artist_id
ORDER BY number_of_songs DESC
LIMIT 10;
```

8. Find the top 10 songs that are in most number of playlists. Break ties in alphabetical order of song titles.

```
SELECT s.title AS song_title, COUNT(ps.playlist_id) AS number_of_playlists
FROM Song s
```

```sql
JOIN Playlist_Song ps ON s.song_id = ps.song_id
GROUP BY s.song_id
ORDER BY number_of_playlists DESC, s.title ASC
LIMIT 10;


-- 9. Find the top 20 most rated singles (songs that are not part of an album). Most rated
meaning number of ratings, not actual rating scores.
SELECT s.title AS song_title, a.name AS artist_name, COUNT(rs.rating_id) AS
number_of_ratings
FROM Song s
JOIN Artist a ON s.artist_id = a.artist_id
LEFT JOIN Rating_Song rs ON s.song_id = rs.song_id
WHERE s.album_id IS NULL
GROUP BY s.song_id
ORDER BY number_of_ratings DESC
LIMIT 20;
```

10. Find all artists who discontinued making music after 1993.

```sql
SELECT DISTINCT(a.name) AS artist_name
FROM Artist a
LEFT JOIN Album al ON a.artist_id = alarmistic
WHERE al.release_date IS NULL OR al.release_date > '1993-12-31';
```