Homework Assignment 4

1. Database Schema

```
CREATE DATABASE IF NOT EXISTS `music`;

CREATE TABLE artists
(
    id BIGINT AUTO_INCREMENT COMMENT 'Artist id' PRIMARY KEY,
    artist_name VARCHAR(64) DEFAULT '' NOT NULL COMMENT 'Artist name',
    CONSTRAINT artists_name_uindex UNIQUE (artist_name)
) COMMENT 'artists';

CREATE TABLE albums
(
    id BIGINT AUTO_INCREMENT COMMENT 'Album id' PRIMARY KEY,
    artist_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Artist id',
    release_date DATE NOT NULL COMMENT 'Release date',
    album_name VARCHAR(255) DEFAULT '' NOT NULL COMMENT 'Album name',
    CONSTRAINT albums_album_name_artist_id_uindex UNIQUE (album_name, artist_id),
    CONSTRAINT albums_artists_id_fk FOREIGN KEY (artist_id) REFERENCES artists (id)
) COMMENT 'albums';

CREATE TABLE genres
(
```

```
    id INT AUTO_INCREMENT COMMENT 'Genre id' PRIMARY KEY,
    genre_name VARCHAR(64) DEFAULT '' NOT NULL COMMENT 'Genres name'
) COMMENT 'genres';

CREATE TABLE songs
(
    id BIGINT AUTO_INCREMENT COMMENT 'Song id' PRIMARY KEY,
    title VARCHAR(255) DEFAULT '' NOT NULL COMMENT 'Song title',
    artist_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Artist id',
    album_id BIGINT NULL COMMENT 'Album id',
    release_date DATE NOT NULL COMMENT 'Release date',
    CONSTRAINT songs_title_artist_id_uindex UNIQUE (title, artist_id),
    CONSTRAINT songs_albums_id_fk FOREIGN KEY (album_id) REFERENCES albums (id),
    CONSTRAINT songs_artists_id_fk FOREIGN KEY (artist_id) REFERENCES artists (id)
) COMMENT 'songs';

CREATE TABLE song_genres
(
    id BIGINT AUTO_INCREMENT COMMENT 'Song genre id' PRIMARY KEY,
    song_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Song id',
    genre_id INT DEFAULT 0 NOT NULL COMMENT 'Genre id',
    CONSTRAINT song_genres_song_id_genre_id_uindex UNIQUE (song_id, genre_id),
    CONSTRAINT song_genres_genres_id_fk FOREIGN KEY (genre_id) REFERENCES genres (id),
    CONSTRAINT song_genres_songs_id_fk FOREIGN KEY (song_id)
```

```
REFERENCES songs (id)
) COMMENT 'song genres';

CREATE TABLE users
(
    id BIGINT AUTO_INCREMENT COMMENT 'User id' PRIMARY KEY,
    username VARCHAR(255) DEFAULT '' NOT NULL COMMENT 'Username',
    CONSTRAINT users_username_uindex UNIQUE (username)
) COMMENT 'users';

CREATE TABLE playlists
(
    id BIGINT AUTO_INCREMENT COMMENT 'Playlist id' PRIMARY KEY,
    user_id BIGINT DEFAULT 0 NOT NULL COMMENT 'User id',
    title VARCHAR(255) DEFAULT '' NOT NULL COMMENT 'Title',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL COMMENT 'Created date',
    CONSTRAINT playlists_user_id_title_uindex UNIQUE (user_id, title),
    CONSTRAINT playlists_users_id_fk FOREIGN KEY (user_id) REFERENCES users (id)
) COMMENT 'playlists';

CREATE TABLE playlist_songs
(
    id BIGINT AUTO_INCREMENT COMMENT 'Playlist song id' PRIMARY KEY,
    playlist_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Playlist id',
    song_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Song id',
    CONSTRAINT playlist_songs_playlist_id_song_id_uindex UNIQUE
```

```
(playlist_id, song_id),
    CONSTRAINT playlist_songs_playlists_id_fk FOREIGN KEY (playlist_id) REFERENCES playlists (id),
    CONSTRAINT playlist_songs_songs_id_fk FOREIGN KEY (song_id) REFERENCES songs (id)
) COMMENT 'playlist songs';

CREATE TABLE ratings
(
    id BIGINT AUTO_INCREMENT COMMENT 'Rating id' PRIMARY KEY,
    user_id BIGINT DEFAULT 0 NOT NULL COMMENT 'User id',
    rating_type ENUM ('album', 'song', 'playlist') NOT NULL COMMENT 'Rating type',
    target_id BIGINT DEFAULT 0 NOT NULL COMMENT 'Rating target id',
    score TINYINT UNSIGNED DEFAULT '5' NOT NULL COMMENT 'Score: 1,2,3,4, or 5',
    created_date DATE NOT NULL COMMENT 'Created date',
    CONSTRAINT ratings_users_id_fk FOREIGN KEY (user_id) REFERENCES users (id)
) COMMENT 'ratings';
```

2. Queries

2.1. Which 3 genres are most represented in terms of number of songs in that genre?

```
SELECT g.genre_name AS genre, COUNT(song_id) AS number_of_songs
FROM song_genres sg
   LEFT JOIN genres g ON g.id = sg.genre_id
GROUP BY sg.genre_id
```

```
ORDER BY number_of_songs DESC
LIMIT 3;
```

2.2. Find names of artists who have songs that are in albums as well as outside of albums (singles).

```
SELECT DISTINCT a.artist_name
FROM songs s1
   LEFT JOIN artists a ON s1.artist_id = a.id
WHERE album_id IS NULL
   AND artist_id IN (SELECT DISTINCT artist_id FROM songs s2 WHERE s2.album_id IS NOT NULL);
```

2.3. 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).

```
SELECT a.album_name, AVG(r.score) AS average_user_rating
FROM ratings r
   LEFT JOIN albums a ON a.id = r.target_id
WHERE r.rating_type = 'album'
   AND r.created_date BETWEEN '1990-01-01' AND '1999-12-31'
GROUP BY r.target_id, a.album_name
ORDER BY average_user_rating DESC, album_name DESC
LIMIT 10;
```

2.4. 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).

```
SELECT g.genre_name, COUNT(*) AS number_of_song_ratings
FROM ratings r
   LEFT JOIN song_genres sg ON r.target_id = sg.song_id
   LEFT JOIN genres g ON g.id = sg.genre_id
WHERE r.rating_type = 'song'
   AND r.created_date BETWEEN '1991-01-01' AND '1995-12-31'
GROUP BY sg.genre_id
ORDER BY number_of_song_ratings DESC
LIMIT 3;
```

2.5. 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.

```
SELECT u.username, p.title AS playlist_title, AVG(r.score) AS average_song_rating
FROM playlist_songs ps
   LEFT JOIN ratings r ON r.target_id = ps.song_id AND r.rating_type = 'song'
   LEFT JOIN playlists p ON ps.playlist_id = p.id
   LEFT JOIN users u ON u.id = p.user_id
GROUP BY ps.playlist_id
HAVING average_song_rating >= 4.0;
```

2.6. 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.)

```
SELECT u.username, COUNT(*) AS number_of_ratings
FROM ratings r
   LEFT JOIN users u ON u.id = r.user_id
```

```
WHERE r.rating_type IN ('song', 'album')
GROUP BY r.user_id
ORDER BY number_of_ratings DESC
LIMIT 5;
```

2.7. 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.artist_name, COUNT(s.id) AS number_of_songs
FROM songs s
   LEFT JOIN artists a ON s.artist_id = a.id
WHERE release_date BETWEEN '1990-01-01' AND '2010-12-31'
GROUP BY s.artist_id
ORDER BY number_of_songs DESC
LIMIT 10;
```

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

```
SELECT s.title, COUNT(ps.playlist_id) AS number_of_playlists
FROM playlist_songs ps
   LEFT JOIN songs s ON ps.song_id = s.id
GROUP BY ps.song_id, s.title
ORDER BY number_of_playlists DESC, s.title ASC
LIMIT 10;
```

2.9. 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. The result must have 3 columns, named song_title, artist_name, number_of_ratings.

```
SELECT s.title AS song_title, a.artist_name, COUNT(r.id) AS number_of_ratings
FROM ratings r
   LEFT JOIN songs s ON r.target_id = s.id
   LEFT JOIN artists a ON s.artist_id = a.id
WHERE r.rating_type = 'song'
   AND s.album_id IS NULL
GROUP BY r.target_id
LIMIT 20;
```

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

```
SELECT s.artist_id
FROM songs s
GROUP BY s.artist_id
HAVING MAX(s.release_date) < '1994-01-01';
```