CS210 - Homework 4
Kruti Shah & Vaishnavi Manthena

Database Schema

create table artist (id smallint auto_increment primary key, artistName varchar(30) not null unique);

create table album (id smallint auto_increment primary key, albumName varchar(30) not null, artistId smallint not null, releaseDate date not null, Foreign key (artistId) references artist(id), unique (albumName, artistId));

create table song (id smallint auto_increment primary key, songName varchar(30) not null, artistId smallint not null, Foreign key (artistId) references artist(id), unique (songName, artistId));

create table single_song (id smallint not null unique, releaseDate date not null, Foreign key (id) references song(id));

create table album_song (id smallint not null unique, albumId smallint not null, Foreign key (id) references song(id), Foreign key (albumId) references album(id));

create table genre (id smallint auto_increment primary key, genreName varchar(20) not null unique);

create table song_to_genre (songId smallint not null, genreId smallint not null, Foreign key (songId) references song(id), Foreign key (genreId) references genre(id), unique (songId, genreId));

create table user (id smallint auto_increment primary key, userName varchar(30) not null unique);

create table playlist (id smallint auto_increment primary key, playlistName varchar(30) not null, userId smallint not null, date date not null, time time not null, Foreign key (userId) references user(id), unique (playlistName, userId));

create table playlist_to_songs (playlistId smallint not null, songId smallint not null, Foreign key (playlistId) references playlist(id), Foreign key (songId) references song(id), unique (playlistId, songId));

```
create table album_rating(userId smallint not null, albumId smallint not null, rating tinyint not
null, date date not null, Foreign key(userId) references user(id), Foreign key (albumId)
references album(id), unique(userId, albumId), check(rating >= 1 and rating <= 5) );
```

```
create table song_rating(userId smallint not null, songId smallint not null, rating tinyint not null,
date date not null, Foreign key (userId) references user(id), Foreign key (songId) references
song(id), unique(userId, songId), check(rating >= 1 and rating <= 5));
```

```
create table playlist_rating(userId smallint not null, playlistId smallint not null, rating tinyint not
null, date date not null, Foreign key (userId) references user(id), Foreign key (playlistId)
references playlist(id), unique(userId, playlistId), check(rating >= 1 and rating <= 5) );
```

Queries

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

```
SELECT genreName as genre, count(*) as number_of_songs
FROM genre JOIN song_to_genre
ON genre.id = song_to_genre.genreId
GROUP BY genre
ORDER BY number_of_songs DESC
LIMIT 3;
```

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

```
SELECT artistName as artist_name
FROM artist
WHERE id in (SELECT distinct artistId
                    FROM single_song JOIN song
                    ON song.id = single_song.id)
          and id in(SELECT distinct artistId
                         FROM album JOIN album_song
                         ON album.id = album_song.albumId);
```

3. What were the top 10 most highly rated albums (highest average user rating) in the period 1990-1999?

```
SELECT albumName as album_name, average_user_rating
FROM album JOIN (SELECT albumId, AVG(rating) as average_user_rating
                FROM ( SELECT * from album_rating where year(date) >= 1990 AND
                 year(date) <= 1999 ) as A GROUP BY albumId) as album_avg
        ON album.id = album_avg.albumId
        ORDER BY average_user_rating DESC, album_name
        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?

```
SELECT genre.genreName as genre_name, count(*) as number_of_song_ratings
FROM genre, song_to_genre, song_rating
WHERE genre.id = song_to_genre.genreId AND song_to_genre.songID = song_rating.songID
AND year(song_rating.date) <= 1995 AND year(song_rating.date) >= 1991
GROUP BY genre_name
ORDER BY number_of_song_ratings DESC
LIMIT 3;
```

5. Which users have a playlist that has an average song rating of 4.0 or more?

```
SELECT userId as username, playlistName as playlist_title, average_playlist_rating as
average_song_rating
FROM
playlist JOIN
( SELECT playlistId, AVG(average_song) as average_playlist_rating FROM playlist_to_songs
JOIN (SELECT songId, AVG(rating) as average_song FROM song_rating GROUP BY songId)
as A
ON playlist_to_songs.songId = A.songId GROUP BY playlistId HAVING
average_playlist_rating >= 4.0) as B
ON playlist.id = B.playlistId
ORDER BY average_song_rating DESC;
```

6. Who are the top 5 most engaged users in terms of the number of ratings that they have given to songs or albums?

```
SELECT userName as username, number_of_ratings
FROM user JOIN
(SELECT userId, sum(count) as number_of_ratings
FROM
((SELECT userId, count(*) as count from song_rating GROUP BY userId)
UNION ALL
(SELECT userId, count(*) as count from album_rating GROUP BY userId)) A
GROUP BY userId)  B
ON user.id = B.userId
ORDER BY number_of_ratings DESC
LIMIT 5;
```

7. Find the top 10 most prolific artists (most number of songs) in the years 1990-2010?

```
SELECT artistName as artist_name, number_of_songs
FROM artist
JOIN
( SELECT artistId, SUM(count) as number_of_songs  FROM ( (SELECT artistId, count(*) as
count FROM single_song JOIN song  ON song.id = single_song.id  WHERE year(releaseDate)
>= 1990 and year(releaseDate) <= 2010 GROUP BY artistId)  UNION ALL
 (SELECT artistId, COUNT(*) as count  FROM album_song JOIN  album ON
album_song.albumId = album.id  WHERE year(releaseDate) >= 1990 and year(releaseDate) <=
2010 GROUP BY artistId) ) A GROUP BY artistId )B
ON artist.id = B.artistId
ORDER BY number_of_songs DESC
LIMIT 10;
```

8. Find the top 10 songs that are in the most number of playlists.

```
SELECT song.songName as song_title, count(*) as number_of_playlists
FROM song JOIN playlist_to_songs ON song.id = playlist_to_songs.songID
GROUP BY song_title
ORDER BY number_of_playlists DESC, song_title ASC
LIMIT 10;
```

9. Find the top 20 most rated singles (songs that are not part of an album).

```
SELECT song_title, artistName as artist_name, number_of_ratings
FROM artist JOIN (
SELECT songName as song_title, artistId, number_of_ratings
FROM song JOIN (SELECT songId, Count(*) as number_of_ratings FROM song_rating
                              WHERE songId in (SELECT DISTINCT id from single_song)
                              GROUP BY songId ) as songCount
                    ON song.id = songCount.songId) A
ON artist.id = A.artistId
ORDER BY number_of_ratings DESC
LIMIT 20;
```

10. Find all artists who discontinued making music after 1993

```
SELECT artistName as artist_title
FROM artist
WHERE Id not in (SELECT artistId from album where year(releaseDate) >= 1993)  AND Id not
in (SELECT DISTINCT artistId FROM song join single_song ON song.id = single_song.id
where year(releaseDate) >= 1993);
```