CS210 - Assignment 4 - Someshwar Ramesh Babu & Shantanu Jain

Database Schema (50 pts)

- **Artists Table**
  - create table artists (id int auto_increment primary key, name varchar(50) not null unique);
- **Songs Table**
  - create table songs(id int auto_increment primary key, artist_id int not null, foreign key (artist_id) references artists(id), title varchar(50) not null, release_date date, album_id int, foreign key(album_id) references albums(album_id), unique(artist_id, title), constraint chk_date check ((release_date is null and album_id is not null) or (release_date is not null and album_id is null)));
- **Albums Table**
  - create table albums(album_id int auto_increment primary key, title varchar(50) not null, artist_id int not null unique, foreign key (artist_id) references artists(id), release_date date, unique(title, artist_id));
- **Playlists Table**
  - create table playlists (id int auto_increment primary key, user_id int not null, title varchar(50) not null, time datetime, foreign key (user_id) references users(id), unique(user_id, title));
- **Ratings Table**
  - create table ratings(user_id int not null, rating int not null, album_id int, playlist_id int, song_id int, date date, foreign key (user_id) references users(id), foreign key (album_id) references albums(album_id), foreign key (playlist_id) references playlists(id), foreign key (song_id) references songs(id), unique(user_id, album_id), unique(user_id, playlist_id), unique(user_id, song_id), constraint chk_rating check (rating >= 1 and rating <= 5));
- **Genres Table**
  - create table genres(song_id int not null, genre varchar(30) not null, foreign key(song_id) references songs(id), unique(song_id, genre));

- **Users Table**
  - `create table users (id int auto_increment primary key, username varchar(50) not null unique);`
- **Songs in Playlist Table**
  - `create table playlistsongs(playlist_id int not null, song_id int not null, foreign key (playlist_id) references playlists(id), foreign key (song_id) references songs(id), unique(playlist_id, song_id));`

Queries (50 pts)

1. Which 3 genres are most represented in terms of number of songs in that genre?
   a. `select genre, count(genre) as 'number_of_songs' from genres 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).
   a. `select artists.name as 'artist_name' from songs inner join artists on songs.artist_id=artists.id inner join albums on artists.id=albums.artist_id where (songs.release_date is not null and albums.artist_id=artists.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.
   a. `select distinct albums.title as 'album_name', avg(ratings.rating) as 'average_user_rating' from albums inner join ratings on albums.album_id=ratings.album_id where (year(ratings.date) >= 1990 and year(ratings.date) <= 1999) group by album_name order by average_user_rating desc, album_name asc 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?
   a. `select genres.genre as 'genre_name', count(ratings.song_id) as 'number_of_song_ratings' from genres inner join ratings on genres.song_id=ratings.song_id where (year(ratings.date) >= 1991 and year(ratings.date) <= 1995) 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? (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 users.username as 'username', playlists.title as 'playlist_title', avg(ratings.rating) as 'average_song_rating' from ratings inner join playlistsongs on ratings.song_id=playlistsongs.song_id inner join playlists on playlistsongs.playlist_id=playlists.id inner join users on playlists.user_id=users.id group by username, playlist_title having average_song_rating >= 4.0;
```
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 users.username, count(ratings.rating) as 'number_of_ratings' from ratings inner join users on ratings.user_id=users.id where (ratings.playlist_id is null) group by users.username 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? Count each song in an album individually.
   a.
```
select artists.name as 'artist_name', count(songs.id) as 'number_of_songs' from songs inner join artists on songs.artist_id=artists.id inner join albums on songs.album_id=albums.album_id where ((songs.album_id is not null and year(albums.release_date) >= 1990 and year(albums.release_date) <= 2010) or (songs.album_id is null and year(songs.release_date) >= 1990 and year(songs.release_date) <= 2010)) group by artist_name 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.
   a.
```
select songs.title as 'song_title', count(playlistsongs.song_id) as 'number_of_playlists' from playlistsongs inner join songs on playlistsongs.song_id=songs.id 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). Most rated meaning number of ratings, not actual rating scores.

a.
```
select songs.title as 'song_title', artists.name as 'artist_name', count(ratings.song_id) as 'number_of_ratings' from ratings inner join songs on ratings.song_id=songs.id inner join artists on songs.artist_id=artists.id where (songs.album_id is null) group by song_title, artist_name order by number_of_ratings desc limit 20;
```

10. Find all artists who discontinued making music after 1993.
    a.
```
select distinct artists.name as 'artist_title' from songs inner join artists on songs.artist_id=artists.id inner join albums on artists.id=albums.artist_id where (year(songs.release_date) <= 1993 and year(albums.release_date) <= 1993);
```