music_db

December 11, 2021

[ ]:
```
## SCHEMA ##

<pre>
create table artists (
    id int auto_increment primary key,
    artist_name varchar(50) not null,
    unique (artist_name)
);

create table albums (
    id int auto_increment primary key,
    album_name varchar(50) not null,
    release_date date not null,
    artist_id int primary key,
    foreign key (artist_id) references artists(id)
);

create table genres (
    id int auto_increment primary key,
    genre varchar(20) not null
);

alter table songs (
    id int auto_increment primary key,
    song_title varchar(20) not null,
    type varchar(2) not null COMMENT'1--albums 2--singles',
    release_date date not null,
    title_id int not null,
    covers_id int not null,
    album_id int not null,
    foreign key (title_id) references artists(id),
    foreign key (covers_id) references artists(id)
);


create table song_genre (
    song_id int not null,
    genre_id int not null,
    primary key (song_id, genre_id),
    foreign key (song_id) references songs(id),
    foreign key (genre_id) references genres(id)
);

create table users (
    id int auto_increment primary key,
    username varchar(50) not null,
    unique (username)
);

create table playlists (
    id int auto_increment primary key,
    playlist_title varchar(50) not null,
    created_date date not null,
    user_id int not null,
    foreign key (user_id) references users(id),
    unique (playlist_title, user_id)
);

create table playlist_song (
    playlist_id int not null,
    song_id int not null,
    primary key (playlist_id, song_id),
    foreign key (playlist_id) references playlists(id),
    foreign key (song_id) references songs(id)
);

create table ratings (
    id int auto_increment primary key,
    type varchar(2) not null COMMENT '1--songs 2--albums 3--playlists',
    song_album_playlist_id int not null,
    rating tinyint not null,
    created_date date not null,
    user_id int not null,
    foreign key (user_id) references users(id)
);
</pre>
```

[ ]:

```
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.

<pre>
select genre, count(*) as "number_of_songs"
from genres, song_genre
```

```
where genre_id = genres.id
group by genre
order by number_of_songs desc limit 3;
</pre>
```

[ ]:
```
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

<pre>
select artist_name
from artists
where id in (
        select albums.artist_id
        from albums, songs
        where songs.artist_id = albums.artist_id
        );
</pre>
```

[ ]:
```
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.

<pre>
select album_name, round(avg(rating),1) as average_user_rating
from ratings, albums
where year(release_date) BETWEEN '1990' and '1999'
    and type = '1'
    and song_album_playlist_id = albums.id
group by album_name
order by album_name desc limit 10;
</pre>
```

[ ]:
```
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 rating date, NOT date of release)
The result must have two columns, named genre_name and number_of_song_ratings.

<pre>
select genres.genre as genre_name, count(rating) as number_of_song_ratings
from ratings, song_genre, genres
where type = '2'
    and year(created_date) BETWEEN '1991' and '1995'
    and ratings.song_album_playlist_id = song_genre.song_id
    and song_genre.genre_id = genres.id
group by genre
```

```
order by number_of_song_ratings desc limit 3;
</pre>
```

[ ]:
```
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.
The result must 3 columns named username, playlist_title, average_song_rating

<pre>
select username, playlist_title, average_song_rating
from (select round(avg(rating),1) as average_song_rating,␣
 ↪song_album_playlist_id from ratings where type = '2'
group by song_album_playlist_id having average_song_rating >= 4.0) t1,
    playlist_song t2,
    playlists t3,
    users t4
where t2.song_id = t1.song_album_playlist_id
    and t2.playlist_id = t3.id
    and t3.user_id = t4.id;
</pre>
```

[ ]:
```
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.)
The result must have 2 columns, named username and number_of_ratings.

<pre>
select username, number_of_ratings
from (select count(*)number_of_ratings, user_id
      from ratings where type = '1' or type = '2'
      group by user_id
      order by number_of_ratings desc limit 5) t1,
      users t2
where t1.user_id = t2.id;
</pre>
```

[ ]:
```
7) Find the top 10 most prolific artists (most number of songs) in the years␣
 ↪1990-2010? Count each song in an album individually.
The result must have 2 columns, named artist_name and number_of_songs.

<pre>
select artist_name, count(*) as "number_of_songs"
from artists, songs
where artist_id = artists.id and year(release_date) between 1990 and 2010
group by artist_name
order by number_of_songs desc limit 10;
```

```
</pre>
```

```
[ ]: 8) Find the top 10 songs that are in most number of playlists. Break ties in␣
      ↪alphabetical order of song titles.
     The result must have a 2 columns, named song_title and number_of_playlists.
     <pre>
     select song_title, number_of_playlists
     from (select count(*) number_of_playlists, song_id
         from playlist_song
         group by song_id
         order by number_of_playlists desc limit 10) t1 ,
         songs t2
     where t1.song_id = t2.id
     order by song_title;
     </pre>
```

```
[ ]: 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.

     <pre>
     select t2.song_title, t3.artist_name, number_of_ratings
     from (
         select count(*) number_of_ratings, song_album_playlist_id from ratings
         where song_album_playlist_id in(
             select id from songs where type = '2')
         group by song_album_playlist_id order by number_of_ratings desc limit 20)␣
      ↪t1 ,
             songs t2,
             artists t3
     where t1.song_album_playlist_id = t2.id
     and t2.title_id = t3.id;
     </pre>
```

```
[ ]: 10) Find all artists who discontinued making music after 1993.
     The result should be a single column named artist_title

     <pre>
     select artist_name as artist_title
     from artists
     where id not in (
         select distinct title_id
         from songs
         where year(release_date) > 1993);
     </pre>
```