CS210

Assignment4

GeonJae Baek

Schema

```
create table artist
(
id int auto_increment primary key,
Name VARCHAR(100) not null unique
);

create table album
(
id int auto_increment primary key,
title VARCHAR(100) not null,
artist_id int not null,
Foreign key (artist_id) references artist(id),
unique(title, artist_id),
releasedate date not null
);

create table song
(
id int auto_increment primary key,
title VARCHAR(100) not null,
```

```
artist_id int not null,
Foreign key (artist_id) references artist(id),
unique(title, artist_id),
album_id int not null,
foreign key (album_id) references album(id),
Genre enum('hip-hop', 'jazz', 'k-pop', 'pop', 'rock', 'classic', 'top-hit', 'R&B') default 'k-pop',
releasedate date,
constraint releasedate1 check(album_id is null and releasedate is not null or album_id is not null and releasedate is null)
);

create table user
(
id int auto_increment primary key,
name VARCHAR(100) not null unique
);

create table playlist
(
id int auto_increment primary key,
title VARCHAR(100) not null,
createtime time not null,
createdate date not null,
user_id int,
Foreign key (user_id) references user(id),
```

```
unique(title, user_id)
);

create table playlistsong
(
PL_id int not null,
Foreign key(PL_id) references playlist(id),
song_id int not null,
Foreign key (song_id) references song(id)
);

create table rating_song
(
user_id int not null,
Foreign key (user_id) references user(id),
song_id int not null,
Foreign key (song_id) references song(id),
unique(user_id, song_id),
rating int not null check(rating >= 1 and rating <= 5),
ratingdate date not null
);

create table rating_album
(
user_id int not null,
```

```
Foreign key (user_id) references user(id),
album_id int not null,
Foreign key (album_id) references album(id),
unique(user_id, album_id),
rating int not null check(rating >= 1 and rating <= 5),
ratingdate date not null
);

create table rating_PL
(
user_id int not null,
Foreign key (user_id) references user(id),
PL_id int not null,
Foreign key (PL_id) references playlist(id),
unique(user_id, PL_id),
rating int not null check(rating >= 1 and rating <= 5),
ratingdate date not null
);
```

Query

1. **Select genre, count(*) as number_of_songs from song group by genre order by number_of_songs desc limit 3;**
2. **select artist.name as artist_name from artist, song where artist.id = song.artist_id and song.album_id is not null and artist.id in(select artist.id from artist, song where artist.id = song.artist_id and song.album_id is null);**
3. **select a.title as album_name, avg(ar.rating) as average_user_rating from album a, rating_album ar where a.id = ar.album_id and year(ar.ratingdate)>=1990 and year(ar.ratingdate) <= 1999 group by album_name order by average_user_rating desc , album_name asc limit 10;**
4. **select s.genre as genre_name, count(*) as number_of_song_ratings from song s, rating_song r where s.id = r.song_id and year(r.ratingdate) >= 1991 and year(r.ratingdate) <= 1995 group by s.genre order by number_of_song_ratings desc limit 3;**
5. **select u.name as username, p.title as playlist_title, avg(sr.rating) as average_song_rating from user u, playlist p, song s, playlistsong sp, rating_song sr where u.id = p.user_id and p.id = sp.PL_id and s.id = sp.song_id and s.id = sr.song_id group by u.name, p.title having avg(sr.rating) > 4;**
6. **select user.name as username, sum(num) as number_of_ratings from(select sr.user_id, count(*) as num from rating_song sr group by sr.user_id union select ar.user_id, count(*) as num from rating_album ar group by ar.user_id) as p, user where user.id = p.user_id group by username order by number_of_ratings desc limit 5;**
7. **select artist.name as artist_name, count(*) as number_of_songs from artist, (select title, releasedate, artist_id from song where album_id is null union select distinct song.title, album.releasedate, album.artist_id from song, album where song.album_id = album.id) as p where p.artist_id = artist.id and year(p.releasedate) >= 1990 and year(p.releasedate) <= 2010 group by artist_name order by number_of_songs desc limit 10;**

8. **select song.title as song_title , count(*) as number_of_plays from song, playlistsong sp where song.id = sp.song_id group by song_title order by number_of_plays desc, song_title asc limit 10;**
9. **select song.title as song_title, artist.name as artist_name, count(*) as number_of_ratings from song, artist, rating_song where song.artist_id = artist.id and song.id = rating_song.song_id and song.album_id is null group by song_title, artist_name limit 20;**
10. **select artist.name as artist_title from artist where artist.name not in (select distinct artist.name as artist_title from artist, (select title, releasedate, artist_id from song where album_id is null union select distinct song.title, album.releasedate, album.artist_id from song, album where song.album_id = album.id) as p where artist.id = p.artist_id and year(p.releasedate) > 1993);**