Database Schema:

Create table artist(name varchar(50) **primary key**);

Create table genre(genre_id int auto_increment **primary key,** name varchar(50) not null, unique(name));

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

```
Create table album(album_id int auto_increment,
        title varchar(50) not null,
        artistname varchar(50) not null,
        releasedate date not null,
        foreign key (artistname) references artist(name),
        UNIQUE(title,artistname,releasedate),
        primary key (album_id,title,artistname)
);
Create table song(song_id int auto_increment,
        title varchar(50) not null,
        artistname varchar(50) not null,
        albuminfo int,
        singledate date,
        UNIQUE(title,artistname),
        foreign key (artistname) references artist(name),
        foreign key (albuminfo) references album(album_id),
        primary key (song_id)
);

Create table songgenres(song_id int not null,
        song_genre int not null,
        UNIQUE(song_id, song_genre),
        Foreign key (song_id) references song(song_id),
        Foreign key (song_genre) references genre(genre_id),
        Primary key(song_id)
);


Create table playlists(playlist_id int auto_increment,
        title varchar(50) not null,
        user_id int not null,
        creation_date_time DATETIME not null,
        UNIQUE(title,user_id),
        foreign key (user_id) references users(user_id),
```

```
        Primary key (playlist_id)
);
Create table songplaylists(playlist_id int not null,
        user_id int not null,
        song_id int not null,
        UNIQUE(playlist_id,user_id,song_id),
        foreign key (playlist_id) references playlists(playlist_id),
        foreign key (user_id) references playlists(user_id),
        foreign key (song_id) references song(song_id)
);

Create table albumratings(user_id int,
        album_id int,
        rating int,
    CHECK (rating > 0 AND rating < 6),
        ratedate date,
        UNIQUE(user_id, album_id),
        foreign key (user_id) references users(user_id),
        foreign key (album_id) references album(album_id)
);
Create table songratings(user_id int,
        song_id int,
        rating int,
    CHECK (rating > 0 AND rating < 6),
        ratedate date,
        UNIQUE(user_id, song_id),
        foreign key (user_id) references users(user_id),
        foreign key (song_id) references song(song_id)
);

Create table playlistratings(user_id int,
        playlist_id int,
        rating int,
    CHECK (rating > 0 AND rating < 6),
        ratedate date,
        UNIQUE(user_id, playlist_id),
        foreign key (user_id) references users(user_id),
        foreign key (playlist_id) references playlists(playlist_id)
);

create table songAverageRating (PRIMARY KEY(`song_id`) ) as (
SELECT song.song_id, AVG(songratings.rating) AS 'average_song_rating'
FROM songratings
INNER JOIN song
```

```
ON songratings.song_id = song.song_id
GROUP BY song_id
ORDER BY average_song_rating desc
);
```

Queries(50 points)

1.
```
SELECT genre.name as 'genre', COUNT(songgenres.song_genre) as
'number_of_songs'
FROM songgenres
INNER JOIN genre
ON genre.genre_id = songgenres.song_genre
GROUP BY songgenres.song_genre
ORDER BY number_of_songs  desc
LIMIT 3;
```
2.
```
SELECT song.artistName AS 'artist_name'
FROM song
GROUP BY artist_name;
```

3.
```
 SELECT album.title AS 'album_name', AVG(albumratings.rating) AS
'average_user_rating'
 FROM albumratings
 INNER JOIN album
 ON albumratings.album_id = album.album_id
 WHERE albumratings.ratedate BETWEEN  '1990-01-01' and'1999-12-31'
 GROUP BY album_name
 ORDER BY average_user_rating DESC., album_name ASC
 LIMIT 10;
```

4.
```
select genre.name AS 'genre_name', COUNT(rating) as 'number_of_song_ratings'
from songratings
inner join songgenres
on songratings.song_id = songgenres.song_id
Inner JOIN genre
ON genre.genre_id = songgenres.song_genre
WHERE songratings.ratedate BETWEEN  '1991-01-01' and'1995-12-31'
group by song_genre
ORDER BY number_of_song_ratings desc
LIMIT 3;
```

5.

```
select * from (
select  users.username, playlists.title as 'playlist_title', avg(average_song_rating) as
'average_song_rating'
from songplaylists, songAverageRating, users, playlists
where songplaylists.song_id =songAverageRating.song_id and users.user_id =
songplaylists.user_id and playlists.playlist_id = songplaylists.playlist_id
group by playlist_title
) AS result_table
where result_table.average_song_rating >= 4;
```

6.

```
select users.username, count(rating) as 'number_of_ratings'  from (
select user_id, rating from albumratings
UNION ALL
select user_id, rating from songratings
) as allRatings, users
where users.user_id = allRatings.user_id
group by users.username
order by number_of_ratings desc
limit 5;
```

7.

```
SELECT artistname as 'artist_name', count(title) as 'number_of_songs'
FROM (
select s1.artistname, s1.title
from song as s1, album
where albuminfo is not null and s1.albuminfo = album.album_id and album.releasedate
between '1990-01-01'  and '2010-12-31'
UNION ALL
select s2.artistname, s2.title
from song as s2
where albuminfo is null And singledate between '1990-01-01'  and '2010-12-31'
) as allSongs
group by artist_name
order by number_of_songs desc
limit 10;
```

8.

```
select song.title as 'song_title ', count(songplaylists.playlist_id) as 'number_of_playlists'
from song, songplaylists
where song.song_id = songplaylists.song_id
group by song.title
```

```
order by number_of_playlists desc, song.title ASC
limit 10;
```

9.

```
 select song.title as 'song_title', song.artistname as 'artist_name',
count(songratings.rating) as 'number_of_ratings'
from song, songratings
where song.song_id = songratings.song_id and song.albuminfo is null
group by song.title
order by  number_of_ratings desc
LIMIT 20;
```

10.

```
 select *
from (select s1.artistname as 'artist_title'
        from song as s1, album
        where s1.albuminfo = album.album_id
        UNION ALL
        select s2.artistname as 'artist_title'
        from song as s2
        where s2.albuminfo is null
) as allartists
where allartists.artist_title not in (
        select s1.artistname as 'artist_title'
        from song as s1, album
        where s1.albuminfo = album.album_id and album.releasedate > '1993-01-01'
        UNION ALL
        select s2.artistname as 'artist_title'
        from song as s2
        where s2.albuminfo is null and s2.singledate > '1993-01-01'
)
group by allartists.artist_title;
```