TABLES:

```
create table Artist (
          name varchar(50) not null,
          primary key(name)
);

create table Album (
         album_name not null,
         release_date not null,
         artist_name not null,
         primary key (album_name, artist_name),
         foreign key (artist_name) references Artist
);

create table Song (
          title varchar(50) not null,
          release_date date null,
          artist_name varchar(50) not null,
          album_name varchar(50) null,
          primary key(title, artist_name),
          foreign key (artist_name) references Artist(name),
          foreign key(album_name) references Album(album_name)
);

create table Genre(
         title varchar(50) not null,
         primary key(title)

);

create table Song_Genre(
         song_title varchar(50) not null,
         song_artist varchar(50) not null,
         genre_title varchar(50) not null,
         primary key(song_title, song_artist, genre_title),
         foreign key(song_title) references Song(title),
         foreign key (song_artist) references Song(artist_name),
         foreign key(genre_title) references Genre(title)
);

create table User(
         username varchar(50) not null,
         primary key(username)
);

create table Playlist(
	title varchar(50) not null,
	date_time datetime(yyyy-mm-dd) not null,
	user varchar(50) not null,
	primary key(title, user),
	foreign key(user) references User(username)
);

create table Playlist_Song(
	playlist_title varchar(50) not null,
	playlist_user varchar(50) not null,
	song_title varchar(50) not null,
	song_artist varchar(50) not null,
	primary key(playlist_title, playlist_user, song_title, song_artist),
	foreign key(playlist_title) references Playlist(title),
	foreign key(playlist_user) references Playlist(user),
	foreign key(song_title) references Song(title),
	foreign key(song_artist) references Song(artist_name)
);

create table Rating(
	value int not null,
	date date not null,
	user varchar(50) not null,
	song_title varchar(50) not null,
	song_artist varchar(50) not null,
	foreign key(user) references User(username),
	foreign key(song_title) references Song(title),
	foreign key(song_artist) references Song(artist_name)
);
```

QUERIES:

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.

```
SELECT
	genre_Title AS genre,
	count(*) AS number_of_songs
FROM
	Song_Genre
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). The result must have one column, named artist_name

```
SELECT
            DISTINCT(artist_name) AS artist_name
FROM
            Song
WHERE
            album_name IS NOT NULL AND
            artist_name IN (SELECT artist_name FROM Song WHERE album_name IS NULL);
```

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.

```
SELECT
            DISTINCT(Song.album_name) AS album_name,
FROM
            Song
            (SELECT
                    r.Song_title,
                    r.Artist_name,
                    CONCAT(r.Song_title, '-', r.artist_name) AS SONG_KEY
                    AVG(val) AS avg_rating
             FROM
                    'Rating' r
             WHERE
                    Rating.Date_time BETWEEN CAST('1990-01-01' AS DATE) AND CAST('1999-
12-31' AS DATE)
             GROUP BY
                    SONG_KEY
              ORDER BY
                    avg_rating DESC
             ) AS avg_Rating
WHERE
           avg_Rating.Song_title = Song.Title AND
           avg_Rating.Song_artist = Song.Artist_name
ORDER BY
           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? (Years refers to rating date, NOT date of release) The result must have two columns, named genre_name and number_of_song_ratings.

```
SELECT
	Song_Genre.genre_title AS genre_name,
	COUNT(*) AS number_of_song_ratings
FROM
	Song_Genre, Rating
WHERE
	Song_Genre.song_title = Rating.song_title AND
	Song_Genre.song_artist = Rating.song_artist
	Rating.date_time BETWEEN CAST('1991-01-01' AS DATE) AND CAST('1995-12-31' AS
DATE)
GROUP BY
	Song_Genre.genre_title
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.
The result must 3 columns named **username**, **playlist_title**, **average_song_rating**

```
SELECT
	value, round(avg(value),1) as average_song_rating
 FROM
	Rating
WHERE
	Rating.date_time BETWEEN CAST('2010-01-01' AS DATE) AND CAST('2015-12-31' AS
DATE)
GROUP BY
	user
HAVING
	average_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.) The result must have 2 columns, named username and number_of_ratings.

```
SELECT
	user AS username,
            Count(*) AS number_of_ratings
FROM
            Rating
GROUP BY
            user
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. The result must have 2 columns, named artist_name and number_of_songs.

```
SELECT
           artist_Name AS artist_name,
           Count(*) AS number_of_songs
FROM
        Song
WHERE
         release_date BETWEEN CAST('1990-01-01' AS DATE) AND CAST('2010-12-31' AS
DATE)
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. The result must have a 2 columns, named song_title and number_of_playlists.

```
SELECT
         song_title AS song_title,
         CONCAT(Song.song_title, '-', Song.artist_name) AS SONG_KEY,
         Count(*) AS number_of_playlists
FROM
         Song, Playlist_Song
WHERE
          Playlist_Song.song_title = Song.title AND
          Playlist_Song.song_artist = Song.artist_name
GROUP BY
          SONG_KEY
ORDER BY
         number_of_playlists DESC
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. The result must have 3 columns, named song_title, artist_name, number_of_ratings.

```
SELECT
        Song.song_title,
         Song.artist_name,
       CONCAT(Song.song_title, '-', Song.artist_name) AS SONG_KEY,
        COUNT(*) AS number_of_ratings
FROM
         Song, Rating
WHERE
          Song.album_name IS NULL
GROUP BY
          SONG_KEY
ORDER BY
           number_of_ratings DESC
LIMIT 20;
```

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

```
SELECT
          DISTINCT(artist_name)
FROM
         Song
WHERE
          NOT IN (SELECT artist_name FROM Song WHERE release_date > CAST('1993-12-31'
AS DATE));
```