Section 1: Database Schema

```
create table artist (
      id int auto_increment primary key,
      name varchar(100) not null unique,
      in_group bit not null
);

create table album (
      id int auto_increment primary key,
      name varchar(100) not null,
      artist_id int not null,
      foreign key(artist_id) references artist(id),
      release_date date not null,
      unique(name, artist_id)
);

create table song (
      id int auto_increment primary key,
      song_title varchar(100) not null,
      artist_id int not null,
      foreign key(artist_id) references artist(id),
      album_id int,
      foreign key(album_id) references album(id),
      release_date date,
      constraint has_date check (album_id is not null or release_date
      is not null),
      unique(song_title, artist_id)
```

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

create table song_genre (
      id int auto_increment primary key,
      song_id int not null,
      foreign key(song_id) references song(id),
      genre_id int not null,
      foreign key(genre_id) references genre(id)
);

create table user (
      id int auto_increment primary key,
      username varchar(20) unique not null
);

create table playlist(
      id int auto_increment primary key,
      title varchar(100) not null,
      created datetime not null,
      user_id int not null,
      foreign key(user_id) references user(id),
      unique(user_id, title)
);
```

```
create table playlist_song (
	id int auto_increment primary key,
	playlist_id int not null,
	foreign key(playlist_id) references playlist(id),
	song_id int not null,
	foreign key(song_id) references song(id)
);

create table rating(
	id int auto_increment primary key,
	user_id int not null,
	foreign key(user_id ) references user(id),
	created date,
	song_id int,
	foreign key(song_id) references song(id),
	album_id int,
	foreign key(album_id) references album(id),
	playlist_id int,
	foreign key(playlist_id) references playlist(id),
	rating_input int not null check((rating_input > 0) and
	(rating_input< 6)),
	constraint rate_one_item check (
		(((song_id is not null and album_id is null) and playlist_id
	is null)
		or ((album_id is not null and song_id is null) and
	playlist_id is null))
```

```
            or ((playlist_id is not null and song_id is null) and
        album_id is null)
        )
);
```

Section 2: Queries

1. Which 3 genres are most represented in terms of number of songs in that genre?

```
SELECT g.name AS genre, COUNT(sg.song_id) AS number_of_songs

FROM genre g

JOIN song_genre sg ON g.id = sg.genre_id

GROUP BY g.id

ORDER BY COUNT(sg.song_id) DESC

LIMIT 3;
```

2. Find names of artists who have songs that are in albums as well as outside of albums (singles).

```
SELECT DISTINCT a.name AS artist_name

FROM artist a

JOIN song s ON a.id = s.artist_id

GROUP BY a.id

HAVING COUNT(DISTINCT s.album_id IS NOT NULL) > 0 AND
COUNT(DISTINCT s.album_id IS NULL) > 0;
```

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).

```
SELECT al.name AS album_name, AVG(r.rating_input) AS
average_user_rating

FROM album al

JOIN rating r ON al.id = r.album_id

WHERE YEAR(r.created) BETWEEN 1990 AND 1999

GROUP BY al.id
```

```
ORDER BY AVG(r.rating_input) DESC, al.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 the rating date, NOT the date of release).

```
SELECT g.name AS genre_name, COUNT(r.id) AS
number_of_song_ratings

FROM genre g

JOIN song_genre sg ON g.id = sg.genre_id

JOIN song s ON sg.song_id = s.id

JOIN rating r ON s.id = r.song_id

WHERE YEAR(r.created) BETWEEN 1991 AND 1995

GROUP BY g.id

ORDER BY COUNT(r.id) 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.

```
SELECT u.username, p.title AS playlist_title,
AVG(r.rating_input) AS average_song_rating

FROM user u

JOIN playlist p ON u.id = p.user_id

JOIN playlist_song ps ON p.id = ps.playlist_id

JOIN song s ON ps.song_id = s.id

JOIN rating r ON s.id = r.song_id

GROUP BY p.id
```

```
HAVING AVG(r.rating_input) >= 4.0

ORDER BY u.username, p.title;
```

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.)

```
SELECT u.username, COUNT(r.id) AS number_of_ratings

FROM user u

JOIN rating r ON u.id = r.user_id

WHERE r.song_id IS NOT NULL OR r.album_id IS NOT NULL

GROUP BY u.id

ORDER BY COUNT(r.id) 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.

```
SELECT a.name AS artist_name, COUNT(s.id) AS number_of_songs

FROM artist a

JOIN song s ON a.id = s.artist_id

WHERE (s.release_date BETWEEN '1990-01-01' AND '2010-12-31'
OR EXISTS (

        SELECT 1 FROM album al WHERE s.album_id = al.id AND
     al.release_date BETWEEN '1990-01-01' AND '2010-12-31'

))

GROUP BY a.id

ORDER BY COUNT(s.id) DESC

LIMIT 10;
```

8. Find the top 10 songs that are in most number of playlists. Break ties in alphabetical order of song titles.

```
SELECT s.song_title, COUNT(ps.playlist_id) AS
number_of_playlists

FROM song s

JOIN playlist_song ps ON s.id = ps.song_id

GROUP BY s.id

ORDER BY COUNT(ps.playlist_id) DESC, s.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.

```
SELECT s.song_title, a.name AS artist_name, COUNT(r.id) AS
number_of_ratings

FROM song s

JOIN artist a ON s.artist_id = a.id

LEFT JOIN album al ON s.album_id = al.id

JOIN rating r ON s.id = r.song_id

WHERE s.album_id IS NULL

GROUP BY s.id

ORDER BY COUNT(r.id) DESC, s.song_title ASC

LIMIT 20;
```

10. Find all artists who discontinued making music after 1993.

```
SELECT DISTINCT a.name AS artist_name

FROM artist a

WHERE NOT EXISTS (
```

```
        SELECT 1 FROM song s WHERE a.id = s.artist_id AND
      s.release_date > '1993-12-31'

);
```