Eric Yau: ey164
Nana-Acheampong Afriyie: na700

Schema:

```
CREATE TABLE Artist (
   Artist_ID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Genre (
   Genre_ID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE User (
   User_ID INT AUTO_INCREMENT PRIMARY KEY,
   Username VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Album (
   Album_ID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
   Artist_ID INT NOT NULL,
   Release_Date DATE NOT NULL,
   UNIQUE (Name, Artist_ID),
   FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);

CREATE TABLE Song (
   Song_ID INT AUTO_INCREMENT PRIMARY KEY,
   Title VARCHAR(255) NOT NULL,
   Artist_ID INT NOT NULL,
   Album_ID INT,
   Release_Date DATE,
   UNIQUE (Title, Artist_ID),
   FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID),
   FOREIGN KEY (Album_ID) REFERENCES Album(Album_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 Song(Song_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);

CREATE TABLE Playlist (
    Playlist_ID INT AUTO_INCREMENT PRIMARY KEY,
    User_ID INT NOT NULL,
    Title VARCHAR(255) NOT NULL,
    Creation_Date DATETIME NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(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 Playlist(Playlist_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);

CREATE TABLE Rating (
    Rating_ID INT AUTO_INCREMENT PRIMARY KEY,
    User_ID INT NOT NULL,
    Song_ID INT,
    Album_ID INT,
    Playlist_ID INT,
    Rating TINYINT NOT NULL CHECK (Rating BETWEEN 1 AND 5),
    Rating_Date DATE NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID),
    FOREIGN KEY (Album_ID) REFERENCES Album(Album_ID),
    FOREIGN KEY (Playlist_ID) REFERENCES Playlist(Playlist_ID)
);
```

Queries:

1.

```
SELECT Genre.Name AS genre, COUNT(*) AS number_of_songs
FROM Song_Genre
JOIN Genre ON Song_Genre.Genre_ID = Genre.Genre_ID
```

```
GROUP BY Genre.Name
ORDER BY number_of_songs DESC
LIMIT 3;

2.
SELECT DISTINCT Artist.Name AS artist_name
FROM Artist
WHERE Artist_ID IN (SELECT Artist_ID FROM Song WHERE Album_ID IS NULL)
  AND Artist_ID IN (SELECT Artist_ID FROM Song WHERE Album_ID IS NOT
NULL);

3.
SELECT Album.Name AS album_name, AVG(Rating.Rating) AS
average_user_rating
FROM Rating
JOIN Album ON Rating.Album_ID = Album.Album_ID
WHERE YEAR(Rating_Date) BETWEEN 1990 AND 1999
GROUP BY Album.Name
ORDER BY average_user_rating DESC, Album.Name
LIMIT 10;

4.
SELECT Genre.Name AS genre_name, COUNT(Rating.Rating_ID) AS
number_of_song_ratings
FROM Rating
JOIN Song ON Rating.Song_ID = Song.Song_ID
JOIN Song_Genre ON Song.Song_ID = Song_Genre.Song_ID
JOIN Genre ON Song_Genre.Genre_ID = Genre.Genre_ID
WHERE YEAR(Rating_Date) BETWEEN 1991 AND 1995
GROUP BY Genre.Name
ORDER BY number_of_song_ratings DESC
LIMIT 3;

5.
SELECT User.Username, Playlist.Title AS playlist_title, AVG(Rating.Rating) AS
average_song_rating
FROM Playlist
JOIN Playlist_Song ON Playlist.Playlist_ID = Playlist_Song.Playlist_ID
JOIN Song ON Playlist_Song.Song_ID = Song.Song_ID
JOIN Rating ON Song.Song_ID = Rating.Song_ID
JOIN User ON Playlist.User_ID = User.User_ID
GROUP BY Playlist.Playlist_ID
HAVING average_song_rating >= 4.0;
```

6.
```
SELECT User.Username, COUNT(*) AS number_of_ratings
FROM Rating
JOIN User ON Rating.User_ID = User.User_ID
WHERE Song_ID IS NOT NULL OR Album_ID IS NOT NULL
GROUP BY User.Username
ORDER BY number_of_ratings DESC
LIMIT 5;
```

7.
```
SELECT Artist.Name AS artist_name, COUNT(Song.Song_ID) AS
number_of_songs
FROM Song
JOIN Artist ON Song.Artist_ID = Artist.Artist_ID
WHERE YEAR(Release_Date) BETWEEN 1990 AND 2010
GROUP BY Artist.Name
ORDER BY number_of_songs DESC
LIMIT 10;
```

8.
```
SELECT Song.Title AS song_title, COUNT(Playlist_Song.Playlist_ID) AS
number_of_playlists
FROM Playlist_Song
JOIN Song ON Playlist_Song.Song_ID = Song.Song_ID
GROUP BY Song.Title
ORDER BY number_of_playlists DESC, Song.Title
LIMIT 10;
```

9.
```
SELECT Song.Title AS song_title, Artist.Name AS artist_name,
COUNT(Rating.Rating_ID) AS number_of_ratings
FROM Rating
JOIN Song ON Rating.Song_ID = Song.Song_ID
JOIN Artist ON Song.Artist_ID = Artist.Artist_ID
WHERE Song.Album_ID IS NULL
GROUP BY Song.Title, Artist.Name
ORDER BY number_of_ratings DESC
LIMIT 20;
```

10.
```
SELECT Artist.Name AS artist_name
FROM Artist
WHERE Artist_ID NOT IN (SELECT Artist_ID FROM Song WHERE
YEAR(Release_Date) > 1993);
```