Umar Khattak (uk50)
Matthew Lieberman (mrl196)
Data Management for Data Science
CS210 Assignment 4

Database Schema (50 pts)

```
CREATE TABLE IF NOT EXISTS Artist(
        Id MEDIUMINT NOT NULL AUTO_INCREMENT,
        Name VARCHAR(50) NOT NULL
        PRIMARY KEY (Id)
)

CREATE TABLE IF NOT EXISTS Song(
        Id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (Id),
        Title VARCHAR(50) NOT NULL,
        ReleaseDate DATE NOT NULL,
        ArtistName VARCHAR(50) NOT NULL,
        FOREIGN KEY (ArtistName) REFERENCES Artist.Name,
        AlbumName VARCHAR(50) NULL,
        FOREIGN KEY (Title) REFERENCES Album.Title
)

CREATE TABLE IF NOT EXISTS Album(
        Id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (Id),
        Title VARCHAR(50) NOT NULL,
        ReleaseDate DATE NOT NULL,
        FOREIGN KEY (Title) REFERENCES Artist.Name
)

CREATE TABLE IF NOT EXISTS User(
        Id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (Id),
        UserName VARCHAR(50) NOT NULL
        UserRating TINYINT NOT NULL,
        FOREIGN KEY (UserRating) REFERENCES Rating.Rating
        SongRating INT NULL,
        FOREIGN KEY (SongRating) REFERENCES Song.Id
        AlbumRating INT NULL,
        FOREIGN KEY (AlbumRating) REFERENCES Album.Id
        PlaylistRating BIGINT NULL,
        FOREIGN KEY (PlaylistRating) REFERENCES Playlist.Id
)
```

```
CREATE TABLE IF NOT EXISTS Playlist(
        Id BIGINT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (Id),
        Name VARCHAR(50) NOT NULL,
        SongName INT NOT NULL,
        FOREIGN KEY (SongName) REFERENCES Song.Id
        UserName INT NOT NULL,
        FOREIGN KEY (UserName) REFERENCES User.Id
        DateAndTime DATETIME NOT NULL
)

CREATE TABLE IF NOT EXISTS Rating(
        Id INT NOT NULL ATUO_INCREMENT,
        PRIMARY KEY (Id)
        Rating TINYINT NOT NULL,
        CHECK(Id >=1 AND Id <=5),
        Date DATE NOT NULL
)

CREATE TABLE IF NOT EXISTS Genre(
        Name SMALLINT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (Name),
        SongName VARCHAR (50) NOT NULL,
        FOREIGN KEY (SongName) REFERNECES Song.Title
)
```

<u>Queries (50 points)</u>

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
        COUNT(Name) AS number_of_songs
FROM Song, Genre
WHERE Genre.SongName == Song.Title
GROUP BY Genre
ORDER BY COUNT(Name) 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**

```
FROM Artist, Song
```

```
SELECT Artist.Name AS artist_name
FROM Song, Artist
WHERE Song.AlbumName IS NOT NULL
GROUP BY Artist.Name;
```

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**.

```
FROM Album, Rating
SELECT Album.Title AS album_name, AVG(Rating.Rating) AS
average_user_rating
FROM Album, User, Rating
WHERE Album.Id == User.AlbumRating
AND Rating.Date BETWEEN ‘1990-01-01’ AND ‘1999-12-31’
GROUP BY Album.Title
ORDER BY AVG(Rating.Rating) DESC
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 Genre AS genre_name,
COUNT(Rating) AS number_of_song_ratings
FROM User, Genre, Rating
WHERE Genre.SongName == User.SongRating
AND Rating.Date BETWEEN ‘1991-01-01’ AND ‘1995-12-31’
GROUP BY Genre
ORDER BY COUNT(Rating.Rating) 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**

```
FROM User, Playlist
SELECT UserName AS username, Playlist.Name AS playlist_title
FROM Playlist, User, Rating
WHERE User.UserRating == Rating.Id
AND User.Playlist.Rating == Playlist.Id AND AVG(Rating.Rating) >= 4
```

```
GROUP BY Playlist.Name;
```

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**.

```
FROM User, Rating
SELECT User.UserName AS username, Rating.Id AS number_of_ratings
FROM User, Rating
WHERE User.UserRating == Rating.Id
GROUP BY username
ORDER BY COUNT(Rating.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. The result must have 2 columns, named **artist_name** and **number_of_songs**.

```
FROM Artist, Song
SELECT Artist.Name AS artist_name
COUNT(Song.Id) AS number_of_songs
FROM Artist, Song
WHERE Artists.Name = Song.ArtistName AND Song.ReleaseDate
BETWEEN ‘1990-01-01’ AND ‘2010-12-31’
GROUP BY Artist.Name
ORDER BY COUNT(Song.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. The result must have a 2 columns, named **song_title** and **number_of_playlists**.

```
FROM Song, Playlist
SELECT Song.Title AS song_title,
COUNT(Playlist.SongName) AS number_of_playlists
FROM Song, Playlist
WHERE Playlist.SongName == Song.Id
GROUP BY Song.Title
ORDER BY COUNT(Playlist.SongName) 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**.

```
FROM Song, Artist, Rating
SELECT Song.Title AS song_title, Artist.Name AS artist_name,
COUNT(Rating.Id) AS number_of_ratings
WHERE Song.AlbumName IS NULL
GROUP BY Song.Title
ORDER BY COUNT(Rating.Id) DESC
LIMIT 20;
```

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

```
FROM Artist, Song
SELECT Artist.Name AS artist_title
WHERE Song.ReleaseDate <= ‘1993-01-01’;
```