Author: Payal Gami : psg60
Author: Kev Sharma : kks107

Schema

Create the Schema - DDL

```
DROP DATABASE IF EXISTS Music;
CREATE DATABASE IF NOT EXISTS Music;
USE Music;


create table Artist (
        artist_name varchar(150) PRIMARY KEY
);




create table Album (
        album_id int unsigned AUTO_INCREMENT PRIMARY KEY,
        album_name varchar(250) NOT NULL,
        artist_name varchar(150) NOT NULL,
        album_release_date date NOT NULL,
        UNIQUE (album_name, artist_name),
        FOREIGN KEY (artist_name) REFERENCES Artist (artist_name) on delete cascade on update
        cascade
);
```

```
create table Song (
        song_id int unsigned AUTO_INCREMENT PRIMARY KEY,
        title varchar(100) NOT NULL,
        artist_name varchar(150) NOT NULL,
        song_release_date date,
        album_id int unsigned,

        UNIQUE (title, artist_name),

        FOREIGN KEY (artist_name) REFERENCES Artist (artist_name) on delete cascade on update
        cascade,
        FOREIGN KEY (album_id) REFERENCES Album (album_id) on delete cascade,

        # Make sure that if the song isn't in an album, it has song_release_date attribute populated
        CONSTRAINT check_release_date check((album_id is not null and song_release_date is null) or
        (album_id is null and song_release_date is not null))
);



create table Genre (
        gname varchar(100) PRIMARY KEY
);



# Song ==<in>-- Genre

create table SongInGenre(
        gname varchar(100),
        song_id int unsigned,
        PRIMARY KEY (gname, song_id),
        FOREIGN KEY (song_id) REFERENCES Song (song_id) on delete cascade,
        FOREIGN KEY (gname) REFERENCES Genre (gname) on delete cascade on update cascade
);
```

User of music db information (user, playlists, ratings)

```
create table User_m (
        username varchar(50) PRIMARY KEY
);
```

Playlist has User_m as identifying owner

```
create table Playlist (
        playlist_id int unsigned AUTO_INCREMENT PRIMARY KEY,
        playlist_title varchar(250) NOT NULL,
        created_dt datetime NOT NULL,
        username varchar(50) NOT NULL,
        UNIQUE (playlist_title, username),
        FOREIGN KEY (username) REFERENCES User_m (username) on delete cascade on update
        cascade
);
```

Playlist ==<>-- Song

```
create table has (
        playlist_id int unsigned,
        song_id int unsigned,
        primary key (playlist_id, song_id),
        foreign key (playlist_id) references Playlist (playlist_id) on delete cascade,
        foreign key (song_id) references Song (song_id) on delete cascade
);
```

ratings tables below
User --<>-- song

```
create table r_song(
        username varchar(50),
        song_id int unsigned,
        rating smallint NOT NULL check(rating >= 1 and rating <= 5) ,
        rated_on date NOT NULL,
        primary key (username, song_id),
        foreign key (username) references User_m (username) on delete cascade on update cascade,
        foreign key (song_id) references Song (song_id) on delete cascade
);
```

User ---<>--- Album

```
create table r_album(
        username varchar(50),
        album_id int unsigned,
        rating smallint NOT NULL check(rating >= 1 and rating <= 5) ,
        rated_on date NOT NULL,
        primary key (username, album_id),
        foreign key (username) references User_m (username) on delete cascade on update cascade,
        foreign key (album_id) references Album (album_id) on delete cascade
);
```

User ---<>--- Playlist

```
create table r_playlist(
        username varchar(50),
        playlist_id int unsigned,
        rating smallint NOT NULL check(rating >= 1 and rating <= 5),
        rated_on date NOT NULL,
        primary key (username, playlist_id),
        foreign key (username) references User_m (username) on delete cascade on update cascade,
        foreign key (playlist_id) references Playlist (playlist_id) on delete cascade
);
```

Queries

1.

```
SELECT SG.gname as genre, count(*) as number_of_songs
FROM SongInGenre SG
GROUP BY SG.gname
ORDER BY number_of_songs DESC
LIMIT 3;
```

2.

```
SELECT DISTINCT artist_name
FROM Song
WHERE song_release_date IS NOT NULL # confirmed as Single
        AND artist_name IN (
                # artist_name of Songs that are in Album
                SELECT S.artist_name
                FROM Song S, Album A
                WHERE S.album_id is not null AND S.album_id = A.album_id
);
```

3.

```
SELECT A.album_name, AVG(R.rating) as average_user_rating
FROM Album A JOIN r_album R on A.album_id = R.album_id
WHERE YEAR(R.rated_on) between 1990 AND 1999
GROUP BY A.album_name, A.artist_name # same album by two artists should not be paired
ORDER BY
        average_user_rating DESC,
        A.album_name ASC
LIMIT 10;
```

4.

```
SELECT SG.gname as genre_name, count(*) as number_of_song_ratings
FROM r_song R, SongInGenre SG
WHERE R.song_id = SG.song_id AND YEAR(R.rated_on) between 1991 AND 1995
GROUP BY SG.gname
LIMIT 3;
```

5

```
SELECT P.username, P.playlist_title, avg(T1.avg_rating_of_song) as average_song_rating
FROM Playlist P, has H, (
                    SELECT song_id, avg(rating) as avg_rating_of_song
                    FROM r_song
                    GROUP BY song_id
          ) as T1
WHERE P.playlist_id = H.playlist_id AND H.song_id = T1.song_id
GROUP BY H.playlist_id
HAVING avg(T1.avg_rating_of_song) >= 4.0;
```

6.

```
SELECT username, count(*) as number_of_ratings
FROM r_album JOIN r_song USING (username)
GROUP BY username
ORDER BY number_of_ratings DESC
LIMIT 5;
```

7.

```
SELECT T.artist_name, count(*) as number_of_songs
FROM (
                SELECT S.artist_name
                FROM Song S LEFT JOIN Album A ON S.album_id = A.album_id
                WHERE (YEAR(S.song_release_date) between 1990 AND 2010) or
                (YEAR(A.album_release_date) between 1990 AND 2010)
        ) as T
GROUP BY T.artist_name
ORDER BY number_of_songs DESC
LIMIT 10;
```

#8.

```
SELECT S.title as song_title, count(*) as number_of_playlists
FROM has H JOIN Song S ON H.song_id = S.song_id
GROUP BY H.song_id
ORDER BY
        number_of_playlists DESC,
        song_title ASC
LIMIT 10;
```

9

```
SELECT S.title as song_title, S.artist_name, count(*) as number_of_ratings
FROM r_song R, Song S
WHERE R.song_id = S.song_id AND S.album_id is null
GROUP BY S.song_id
ORDER BY number_of_ratings DESC
LIMIT 20;
```

10.

```
SELECT artist_name as artist_title
FROM Artist
WHERE artist_name NOT IN (
        # Exists if produced a song/album after 1993
        SELECT DISTINCT S.artist_name
        FROM Song S left join Album A on S.album_id = A.album_id
        WHERE (YEAR(S.song_release_date) > 1993) or (YEAR(A.album_release_date) > 1993)
);
```