DATA MANAGEMENT

FOR DATASCIENCE

01:198:210:03

HOMEWORK-4

Prof & T.A.: -

Sesh Venugopal

Fatima Jahara

Team Details: -

Raj Shah (ras637)

Mili Patel (mp2173)

Grader: -

Harshankar

Database Schema (60 pts)

```
/* ----- ARTISTS TABLE ----- */
CREATE TABLE artist (
    artist_id INT UNSIGNED AUTO_INCREMENT,
    name      VARCHAR(255) NOT NULL,
    PRIMARY KEY (artist_id),
    UNIQUE KEY (name)
);

/* ----- ALBUMS TABLE ----- */
CREATE TABLE album (
    album_id     INT UNSIGNED AUTO_INCREMENT,
    artist_id    INT UNSIGNED NOT NULL,
    title        VARCHAR(255) NOT NULL,
    release_date DATE NOT NULL,
    PRIMARY KEY (album_id),
    UNIQUE KEY uniq_album_per_artist (artist_id, title),
    FOREIGN KEY (artist_id) REFERENCES artist(artist_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

/* ----- SONGS TABLE ----- */
CREATE TABLE song (
    song_id      INT UNSIGNED AUTO_INCREMENT,
    artist_id    INT UNSIGNED NOT NULL,
    album_id     INT UNSIGNED DEFAULT NULL,   /* NULL → single */
```

```
    title        VARCHAR(255) NOT NULL,
    release_date DATE NOT NULL,
    PRIMARY KEY (song_id),
    UNIQUE KEY uniq_song_per_artist (artist_id, title),
    FOREIGN KEY (artist_id) REFERENCES artist(artist_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (album_id)  REFERENCES album(album_id)
        ON DELETE SET NULL ON UPDATE CASCADE
);

/* ----- GENRES TABLE ----- */
CREATE TABLE genre (
    genre_id INT UNSIGNED AUTO_INCREMENT,
    name     VARCHAR(100) NOT NULL,
    PRIMARY KEY (genre_id),
    UNIQUE KEY (name)
);

/* ----- SONG↔GENRE LINK TABLE ----- */
CREATE TABLE song_genre (
    song_id  INT UNSIGNED NOT NULL,
    genre_id INT UNSIGNED NOT NULL,
    PRIMARY KEY (song_id, genre_id),
    FOREIGN KEY (song_id)  REFERENCES song(song_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (genre_id) REFERENCES genre(genre_id)
        ON DELETE CASCADE ON UPDATE CASCADE
```

```
);

/* ----- USERS TABLE ----- */
CREATE TABLE user_account (
    user_id  INT UNSIGNED AUTO_INCREMENT,
    username VARCHAR(100) NOT NULL,
    PRIMARY KEY (user_id),
    UNIQUE KEY (username)
);

/* ----- PLAYLISTS TABLE ----- */
CREATE TABLE playlist (
    playlist_id INT UNSIGNED AUTO_INCREMENT,
    user_id     INT UNSIGNED NOT NULL,
    title       VARCHAR(255) NOT NULL,
    created_at  DATETIME NOT NULL,
    PRIMARY KEY (playlist_id),
    UNIQUE KEY uniq_title_per_user (user_id, title),
    FOREIGN KEY (user_id) REFERENCES user_account(user_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

/* ----- PLAYLIST↔SONG LINK TABLE ----- */
CREATE TABLE playlist_song (
    playlist_id INT UNSIGNED NOT NULL,
    song_id     INT UNSIGNED NOT NULL,
    PRIMARY KEY (playlist_id, song_id),
```

```
    FOREIGN KEY (playlist_id) REFERENCES playlist(playlist_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (song_id)     REFERENCES song(song_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

/* ----- SONG RATINGS TABLE ----- */
CREATE TABLE song_rating (
    rating_id   INT UNSIGNED AUTO_INCREMENT,
    user_id     INT UNSIGNED NOT NULL,
    song_id     INT UNSIGNED NOT NULL,
    rating      TINYINT UNSIGNED NOT NULL CHECK (rating BETWEEN 1 AND
5),
    rating_date DATE NOT NULL,
    PRIMARY KEY (rating_id),
    FOREIGN KEY (user_id) REFERENCES user_account(user_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (song_id)  REFERENCES song(song_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

/* ----- ALBUM RATINGS TABLE ----- */
CREATE TABLE album_rating (
    rating_id   INT UNSIGNED AUTO_INCREMENT,
    user_id     INT UNSIGNED NOT NULL,
    album_id    INT UNSIGNED NOT NULL,
```

```
    rating      TINYINT UNSIGNED NOT NULL CHECK (rating BETWEEN 1 AND
5),
    rating_date DATE NOT NULL,
    PRIMARY KEY (rating_id),
    FOREIGN KEY (user_id)  REFERENCES user_account(user_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (album_id) REFERENCES album(album_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

/* ----- PLAYLIST RATINGS TABLE ----- */
CREATE TABLE playlist_rating (
    rating_id   INT UNSIGNED AUTO_INCREMENT,
    user_id     INT UNSIGNED NOT NULL,
    playlist_id INT UNSIGNED NOT NULL,
    rating      TINYINT UNSIGNED NOT NULL CHECK (rating BETWEEN 1 AND
5),
    rating_date DATE NOT NULL,
    PRIMARY KEY (rating_id),
    FOREIGN KEY (user_id)    REFERENCES user_account(user_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (playlist_id) REFERENCES playlist(playlist_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Queries (60 points)

```
================================================================================
-- 1. Top 3 genres by number of songs
================================================================================

SELECT g.name AS genre,
       COUNT(*) AS number_of_songs
FROM   genre g
JOIN   song_genre sg ON g.genre_id = sg.genre_id
GROUP  BY g.genre_id
ORDER  BY number_of_songs DESC
LIMIT  3;


================================================================================
-- 2. Artists with songs both in albums and as singles
================================================================================

SELECT DISTINCT a.name AS artist_name
FROM   artist a
JOIN   song s1 ON s1.artist_id = a.artist_id
WHERE  s1.album_id IS NOT NULL
  AND  EXISTS (SELECT 1
               FROM   song s2
               WHERE  s2.artist_id = a.artist_id
                 AND  s2.album_id IS NULL);

================================================================================
-- 3. Top-10 albums (avg rating) rated between 1990-1999
================================================================================
```

```
SELECT al.title AS album_name,
       AVG(ar.rating) AS average_user_rating
FROM   album al
JOIN   album_rating ar ON ar.album_id = al.album_id
WHERE  ar.rating_date BETWEEN '1990-01-01' AND '1999-12-31'
GROUP  BY al.album_id
ORDER  BY average_user_rating DESC,
          al.title ASC
LIMIT  10;

=========================================================================
-- 4. Top-3 most-rated genres (ratings on songs) in 1991-1995
=========================================================================
SELECT g.name AS genre_name,
       COUNT(*) AS number_of_song_ratings
FROM   song_rating sr
JOIN   song_genre sg ON sg.song_id = sr.song_id
JOIN   genre g ON g.genre_id = sg.genre_id
WHERE  sr.rating_date BETWEEN '1991-01-01' AND '1995-12-31'
GROUP  BY g.genre_id
ORDER  BY number_of_song_ratings DESC
LIMIT  3;

=========================================================================
-- 5. Playlists whose songs average ≥ 4.0
=========================================================================
SELECT u.username,
       p.title AS playlist_title,
```

```
       ROUND(AVG(avg_s.avg_song_rating),2) AS average_song_rating
FROM   playlist p
JOIN   user_account u ON u.user_id = p.user_id
JOIN   playlist_song ps ON ps.playlist_id = p.playlist_id
JOIN   ( SELECT song_id, AVG(rating) AS avg_song_rating
         FROM   song_rating
         GROUP  BY song_id ) AS avg_s ON avg_s.song_id = ps.song_id
GROUP  BY p.playlist_id
HAVING average_song_rating >= 4.0;

=============================================================================
-- 6. Top-5 users by total # song+album ratings
=============================================================================
SELECT u.username,
       (IFNULL(sr.cnt,0) + IFNULL(ar.cnt,0)) AS number_of_ratings
FROM   user_account u
LEFT   JOIN (SELECT user_id, COUNT(*) AS cnt
             FROM   song_rating
             GROUP  BY user_id) sr ON sr.user_id = u.user_id
LEFT   JOIN (SELECT user_id, COUNT(*) AS cnt
             FROM   album_rating
             GROUP  BY user_id) ar ON ar.user_id = u.user_id
ORDER  BY number_of_ratings DESC
LIMIT  5;
```

```
================================================================
-- 7. Top-10 prolific artists (songs 1990-2010)
================================================================
SELECT a.name AS artist_name,
       COUNT(*) AS number_of_songs
FROM   song s
JOIN   artist a ON a.artist_id = s.artist_id
WHERE  s.release_date BETWEEN '1990-01-01' AND '2010-12-31'
GROUP  BY a.artist_id
ORDER  BY number_of_songs DESC
LIMIT  10;

================================================================
-- 8. Top-10 songs appearing in most playlists
================================================================
SELECT s.title AS song_title,
       COUNT(*) AS number_of_playlists
FROM   playlist_song ps
JOIN   song s ON s.song_id = ps.song_id
GROUP  BY ps.song_id
ORDER  BY number_of_playlists DESC,
         s.title ASC
LIMIT  10;

================================================================
-- 9. Top-20 most-rated singles (songs not in albums)
================================================================
SELECT s.title AS song_title,
       a.name  AS artist_name,
```

```
       COUNT(sr.rating_id) AS number_of_ratings
FROM   song s
JOIN   artist a ON a.artist_id = s.artist_id
JOIN   song_rating sr ON sr.song_id = s.song_id
WHERE  s.album_id IS NULL
GROUP  BY s.song_id
ORDER  BY number_of_ratings DESC
LIMIT  20;


=============================================================================
-- 10. Artists who stopped releasing songs after 1993
=============================================================================
SELECT a.name AS artist_name
FROM   artist a
JOIN   song s ON s.artist_id = a.artist_id
GROUP  BY a.artist_id
HAVING MAX(s.release_date) <= '1993-12-31';
```