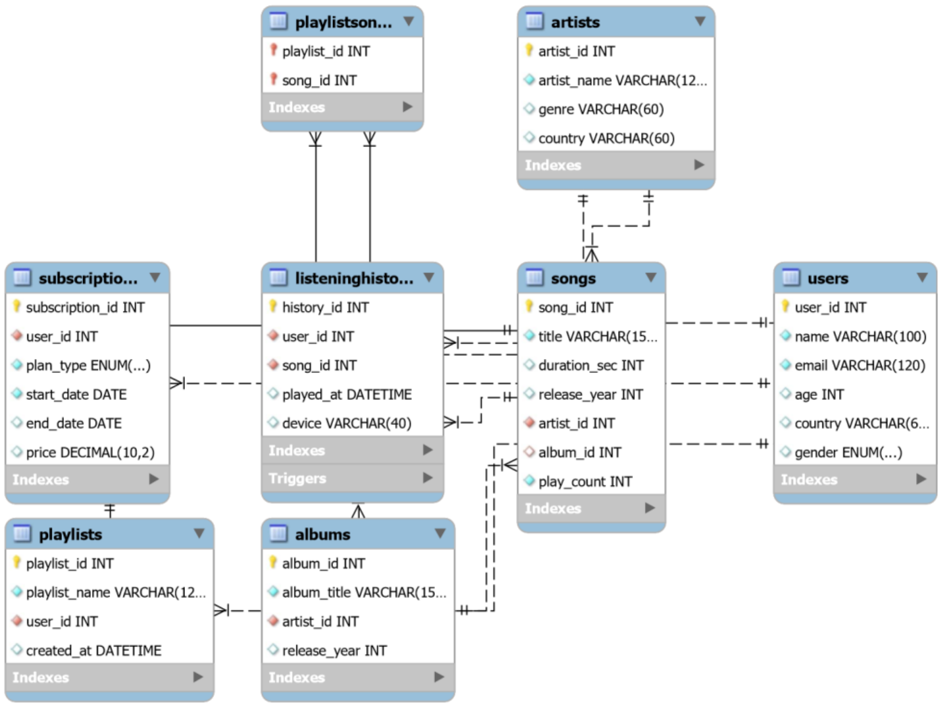
■ Project Report: Music Streaming Analytics Database

1. Introduction

Music streaming platforms generate massive amounts of data daily, such as user activity, listening patterns, subscriptions, and preferences. This project, Music Streaming Analytics Database, is designed in MySQL to simulate a streaming platform and analyze insights such as top songs, average durations, user listening patterns, and subscription details.

The database demonstrates concepts of DDL, DML, constraints, joins, aggregation, sub-queries, triggers, and stored procedures.

2. ER Diagram



3. Database Design

1 Users Table

user_id	name	email	age	country	gender
1	Aarav	aarav@example.com	21	India	Male
2	Maya	maya@example.com	24	USA	Female

3	Liam	liam@example.com	19	Canada	Male
4	Isha	isha@example.com	27	India	Female

2 Artists Table

artist_id	artist_name	genre	country
1	Echo Waves	Pop	USA
2	Raga Roots	Classical	India
3	Neon Drift	EDM	Germany

3 Albums Table

album_id	album_title	artist_id	release_year
1	Blue Horizon	1	2022
2	Morning Ragas	2	2021

4 Songs Table

song_id	title	duration_sec	release_year	artist_id	album_id	play_count
1	Skyline	210	2022	1	1	0
2	Night Drive	180	2022	1	1	0
3	Bhimpalasi	420	2021	2	2	0
4	Trance Gate	230	2023	3	NULL	0

5 Playlists Table

playlist_id	playlist_name	user_id	created_at
1	Chill Mix	1	2025-08-19
2	Focus	2	2025-08-19

6 PlaylistSongs Table

playlist_id	song_id
1	1
1	3
2	2

(Note: The record (2,4) was deleted as per DELETE query.)

7 ListeningHistory Table

history_id	user_id	song_id	played_at	device
1	1	1	2025-08-01 10:00:00	mobile
2	1	3	2025-08-01 21:15:00	desktop
3	2	2	2025-08-02 09:30:00	mobile
4	2	4	2025-08-02 23:55:00	tablet
5	3	1	2025-08-03 19:20:00	mobile
6	4	3	2025-08-04 06:45:00	smart_speaker

8 Subscriptions Table

subscription_id	user_id	plan_type	start_date	end_date	price
1	1	Premium	2025-07-01	2025-08-01	199.00
2	2	Free	2025-07-10	NULL	0.00
3	3	Family	2025-06-15	2025-07-15	299.00

4. Queries and Results

1 Update Query

SQL: UPDATE Users SET country = 'Canada' WHERE email = 'liam@example.com';
■ **Result** → Liam’s country updated from UK → Canada (shown in Users table).

2 Delete Query

SQL: DELETE FROM PlaylistSongs WHERE playlist_id = 2 AND song_id = 4;
■**Result** → Row (2,4) deleted. Remaining records are shown in PlaylistSongs table.

3 LIKE Query

SQL: SELECT song_id, title FROM Songs s JOIN Artists a ON s.artist_id = a.artist_id
WHERE a.genre = 'EDM' AND s.title LIKE '%Drive%';
■**Result** → No EDM song with 'Drive'. Output Table: Empty set.

4 Aggregate Query – Total Plays per Song

SQL: SELECT s.title, COUNT(*) AS play_count FROM ListeningHistory lh JOIN Songs s ON lh.song_id = s.song_id GROUP BY s.title ORDER BY play_count DESC;
■**Result** → Skyline=2, Bhimpalasi=2, Night Drive=1, Trance Gate=1

5 Average Song Duration per Artist

SQL: SELECT a.artist_name, AVG(s.duration_sec) AS avg_duration_sec FROM Songs s JOINArtists a ON s.artist_id = a.artist_id GROUP BY a.artist_name;
■**Result** → Echo Waves=195.0, Raga Roots=420.0, Neon Drift=230.0

6 Sub-Query – Active Listeners

SQL: SELECT u.user_id, u.name, play_ct FROM (SELECT user_id, COUNT(*) AS play_ct FROM ListeningHistory GROUP BY user_id) t JOIN Users u ON u.user_id = t.user_id WHERE t.play_ct > (SELECT AVG(cnt) FROM (SELECT COUNT(*) AS cnt FROM ListeningHistory GROUP BY user_id) x);
■**Result** → Aarav=2, Maya=2

7 Stored Procedure Example

SQL: CALL GetTopSongsByMonth(2025, 8, 3);
■**Result** → Top 3 songs in August 2025 → Skyline=2, Bhimpalasi=2, Night Drive=1

8 Trigger Test

SQL: INSERT INTO ListeningHistory (history_id, user_id, song_id) VALUES (7,1,1);
SELECT song_id, title, play_count FROM Songs WHERE song_id = 1;
■**Result** → Skyline play_count incremented.

5. Conclusion

This project successfully demonstrates the design and implementation of a Music Streaming Analytics Database using MySQL. It showcases:

• Database design with 8 interrelated tables.
• Implementation of DDL, DML, UPDATE, DELETE, ALTER.
• Aggregate queries, sub-queries, joins, LIKE for analysis.
• Advanced features: Stored Procedures & Triggers.

This system can be extended with real-world datasets to analyze user behavior, recommend music, and optimize subscription models.