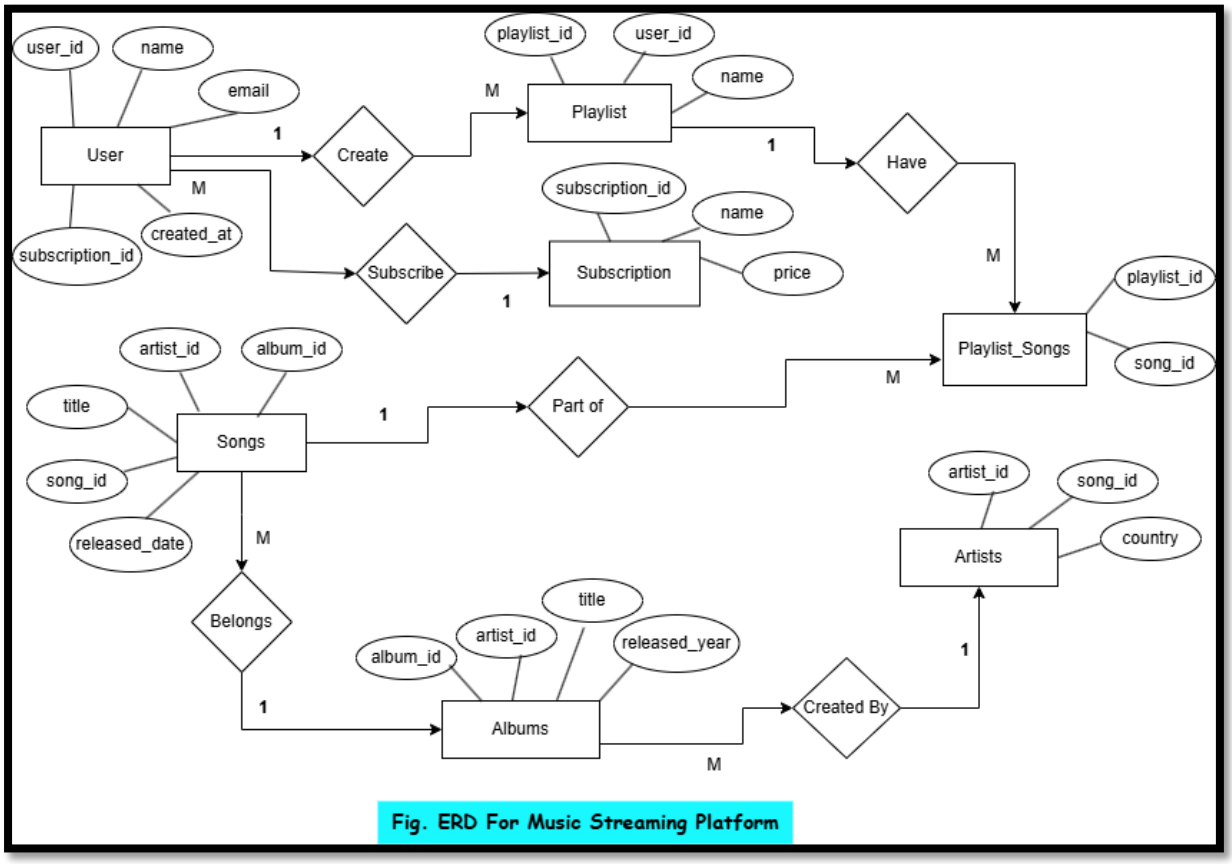
Database Assignment on Music Streaming Platform

❖ **Step 1: Design an ERD for the assigned topic.**
Entities: Users, Songs, Playlists, Subscriptions, Artists, Albums



❖ **Step 2: Identify Entities, Attributes, Relationships & Cardinality**

Entities, Attributes:

- **Users** (user_id, name, email, password, subscription_id, created_at)
- **Songs** (song_id, title, artist_id, album_id, genre, duration, release_date)
- **Artists** (artist_id, name, country, debut_year)
- **Albums** (album_id, title, artist_id, release_year)
- **Playlists** (playlist_id, user_id, name, created_at)
- **Subscriptions** (subscription_id, name, price, duration)

Relationships:

1. A user can have multiple playlists.
2. A playlist can have multiple songs.
3. A song belongs to an album and is created by an artist.
4. Users can subscribe to a plan.
5. Users can like songs and playlists.

❖ **Step 3: Normalize the Database (1NF, 2NF, 3NF)**

Initial Table Structure

Before normalization, we might have a table like this:

user_id	user_name	email	playlist_id	playlist_name	song_id	song_title	artist_name	album_name	Subscription
1	Alice	alice@email.com	101	Chill Vibes	201	Blank Space	Taylor Swift	1989	Premium
1	Alice	alice@email.com	101	Chill Vibes	202	Shape of You	Ed Sheeran	Divide	Premium
2	Bob	bob@email.com	102	Workout Mix	203	Perfect	Ed Sheeran	Divide	Free

Problems in Present Table:

- **Repeating groups**: Multiple songs under the same playlist for a user.
- **Multivalued attributes**: A playlist contains multiple songs, violating atomicity.

1. First Normal Form (1NF):

Rules of 1NF

1. Each column must contain atomic (indivisible) values.
2. Each row must have a unique identifier (Primary Key).
3. No repeating groups or arrays.

To achieve 1NF, we separate playlists and songs into different rows instead of grouping them together.

user_id	user_name	email	playlist_id	playlist_name	song_id	song_title	artist_name	album_name	subscription
1	Alice	alice@email.com	101	Chill Vibes	201	Blank Space	Taylor Swift	1989	Premium
1	Alice	alice@email.com	101	Chill Vibes	202	Shape of You	Ed Sheeran	Divide	Premium
2	Bob	bob@email.com	102	Workout Mix	203	Perfect	Ed Sheeran	Divide	Free

After This :

- Each field contains only atomic values.
- Each row represents a single piece of information (no grouping of songs).

Issues still present:

- **Partial dependencies:** Playlist details depend only on playlist_id, not on user_id.
- **Transitive dependencies:** subscription depends on user_id, which is not directly related to playlists. Hence we try to for the improvement using Second Normal Form.

2. Second Normal Form (2NF)

Rules of 2NF

1. The table must be in **1NF**.
2. **Remove Partial Dependencies** (A column should depend on the whole primary key, not just a part of it).
3. Create separate tables for independent entities.

Here, Partial Dependencies are exist because of:

- subscription only depends on **user_id**, not on **playlist_id** or **song_id.**
- **artist_name** and **album_name** depend on **song_id**, not on **playlist_id.**

Users Table

user_id	user_name	email	subscription
1	Alice	alice@email.com	Premium
2	Bob	bob@email.com	Free

Playlists Table

playlist_id	user_id	playlist_name
101	1	Chill Vibes
102	2	Workout Mix

Artists Table

artist_id	artist_name
1	Taylor Swift
2	Ed Sheeran

Albums Table

album_id	album_name
1	1989
2	Divide

Songs Table

song_id	song_title	artist_id	album_id
201	Blank Space	1	1
202	Shape of You	2	2
203	Perfect	2	2

Playlist-Songs (Bridge Table)

playlist_id	song_id
101	201
101	202
102	203

After this know all tables are in 2NF;

- Removed partial dependencies: Playlist and user details are in separate tables.
- Each table focuses on a single entity with relevant attributes.

Still we noticed that **Transitive dependencies** is present in subscription should have its own table since it depends on user_id indirectly.

3. Third Normal Form (3NF)

Rules of 3NF

1. The table must be in 2NF.
2. Remove transitive dependencies (A non-key column should not depend on another non-key column).

Identifying Transitive Dependencies

- subscription depends on **user_id**, but **user_id** should only reference a subscription ID.
- Move subscription details into a separate table.

Users Table

user_id	user_name	email	subscription_id
1	Alice	alice@email.com	1
2	Bob	bob@email.com	2

Subscriptions Table

subscription_id	subscription_name	price
1	Premium	9.99
2	Free	0.00

Songs Table

song_id	song_title	artist_id	album_id
201	Blank Space	1	1
202	Shape of You	2	2
203	Perfect	2	2

Playlists Table

playlist_id	user_id	playlist_name
101	1	Chill Vibes
102	2	Workout Mix

Artists Table

artist_id	artist_name
1	Taylor Swift
2	Ed Sheeran

Albums Table

album_id	album_name
1	1989
2	Divide

Playlist-Songs (Bridge Table)

playlist_id	song_id
101	201
101	202
102	203

Now, in above tables no Transitive Dependencies are Present;

- Better database efficiency and maintainability.
- Each table focuses only on its specific role.

❖ Step 4: Implement SQL Database Schema

Sample Create Query:

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),  email VARCHAR(100) UNIQUE,
    password    VARCHAR(255),    subscription_id INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (subscription_id) REFERENCES Subscriptions(subscription_id));
```

❖ **Step 5: Insert Sample Data**

```
INSERT INTO Albums (album_id, title)
VALUES (1, 'Aashiqui 2'), (2, 'Ae Dil Hai Mushkil'), (3, 'Tamasha'),
(4, 'Kal Ho Naa Ho'), (5, 'Kabir Singh');
```

Albums Table (Sample Table) **// For All Tables Refer Folder "Tables"**

album_id	title	artist_id	release_year
1	Aashiqui 2	1	2013
2	Ae Dil Hai	2	2016
3	Tamasha	3	2015
4	Kal Ho Naa	4	2003
5	Kabir Sing	5	2019
6	Shershaah	NULL	NULL
7	Brahmastr	NULL	NULL
8	Animal	NULL	NULL
9	Half Girlfr	NULL	NULL
10	Raees	NULL	NULL

❖ **Step 6: Write Basic Queries**

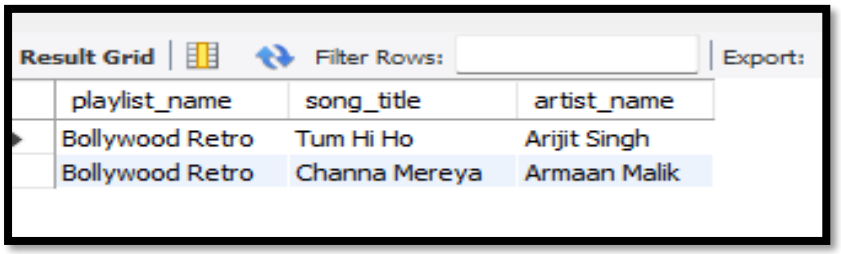
- **Get all users and their subscriptions:**

```
SELECT u.name, u.email, s.name AS subscription
FROM Users u
JOIN Subscriptions s ON u.subscription_id = s.subscription_id;
```

name	email	subscription
Rahul Sharma	rahul.sharma@email.com	Premium
Amit Singh	amit.singh@email.com	Premium
Ravi Kumar	ravi.kumar@email.com	Premium
Ananya Pandey	ananya.pandey@email.com	Premium
Kiran Joshi	kiran.joshi@email.com	Premium
Meera Kapoor	meera.kapoor@email.com	Premium
Deepak Choudhary	deepak.choudhary@email.com	Premium
Rohan Joshi	rohan.joshi@email.com	Premium
Vikas Gupta	vikas.gupta@email.com	Premium
Harshita Reddy	harshita.reddy@email.com	Premium
Rohini Iyer	rohini.iyer@email.com	Premium
Ankita Patil	ankita.patil@email.com	Premium
Pooja Hegde	pooja.hegde@email.com	Premium
Vikrant Massey	vikrant.massey@email.com	Premium
Anjali Chauhan	anjali.chauhan@email.com	Premium
Priya Verma	priya.verma@email.com	Free
Neha Mehta	neha.mehta@email.com	Free
Vikram Batra	vikram.batra@email.com	Free

- **Get all songs in a playlist**

```
SELECT  p.name AS playlist_name,
s.title AS song_title,  a.name AS artist_name    FROM Playlists p
JOIN Playlist_Songs ps ON p.playlist_id = ps.playlist_id
JOIN Songs s ON ps.song_id = s.song_id
JOIN Artists a ON s.artist_id = a.artist_id  WHERE p.playlist_id = 101;
```

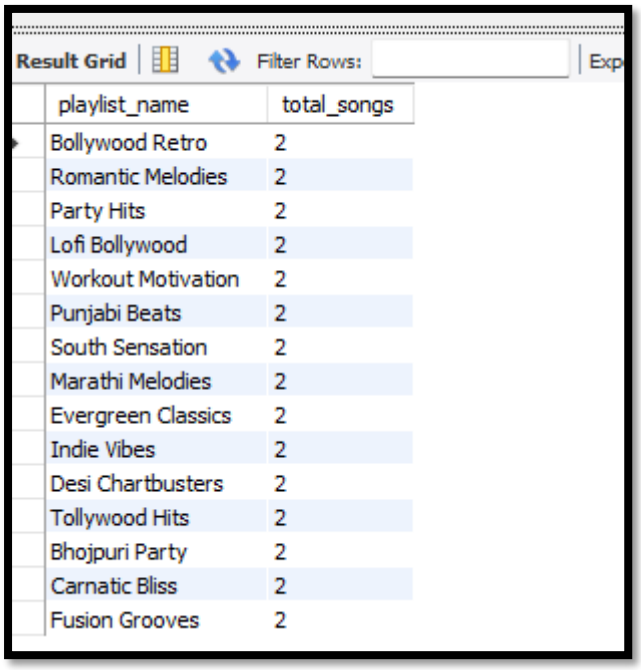


Result Grid | Filter Rows: | Export:

playlist_name	song_title	artist_name
Bollywood Retro	Tum Hi Ho	Arijit Singh
Bollywood Retro	Channa Mereya	Armaan Malik

- **Get total number of songs per playlist**

```
SELECT p.name AS playlist_name, COUNT(ps.song_id) AS total_songs
FROM Playlists p  JOIN Playlist_Songs ps ON p.playlist_id = ps.playlist_id
GROUP BY p.name;
```



Result Grid | Filter Rows: | Exp

playlist_name	total_songs
Bollywood Retro	2
Romantic Melodies	2
Party Hits	2
Lofi Bollywood	2
Workout Motivation	2
Punjabi Beats	2
South Sensation	2
Marathi Melodies	2
Evergreen Classics	2
Indie Vibes	2
Desi Chartbusters	2
Tollywood Hits	2
Bhojpuri Party	2
Carnatic Bliss	2
Fusion Grooves	2