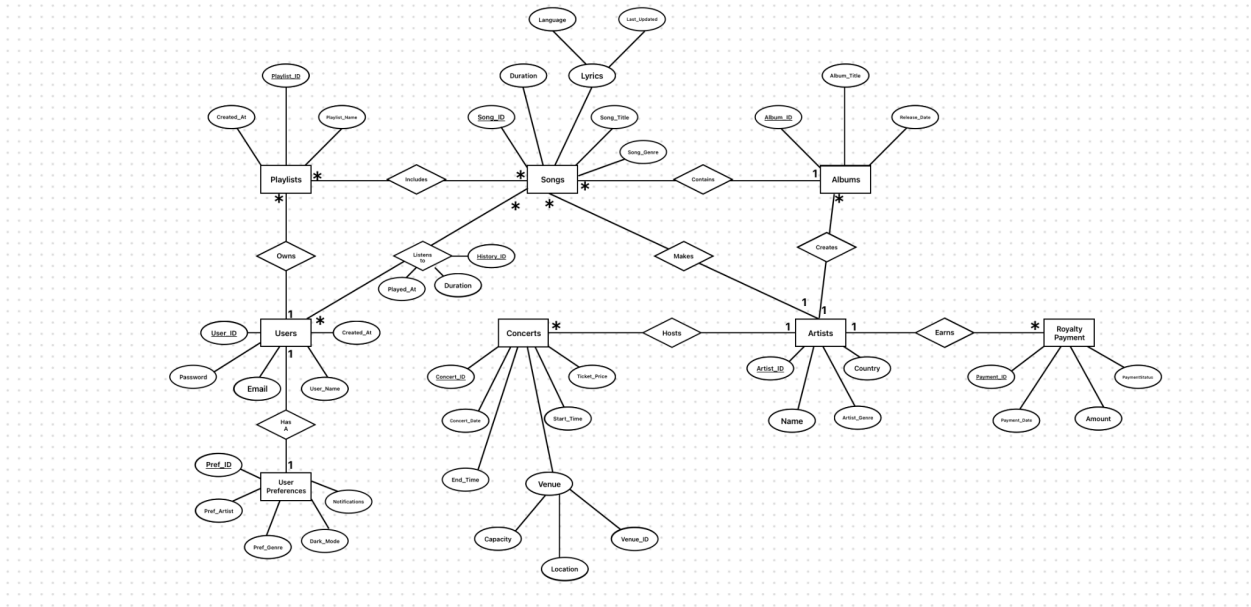

Table Conversion:

1. 1st Normal Form : If a table has no repeated groups, it is in 1NF.
   All tables are already in 1NF.
2. 2nd Normal Form : If a table is in 1NF and every non-key attribute is fully dependent on the primary key, then it is in 2NF.
   All tables are already in 2NF.
3. 3rd Normal Form : If the table is in 2NF and has no transitive dependencies, then it is in 3NF.
   Here, we separate the Concert and Venue tables to remove transitive dependencies. This gives us Concert(Concert_ID, Date, Start_Time, End_Time, Venue_ID) and Venue(Venue_ID, Capacity, Location). We separate the Songs and Lyrics tables getting : Songs(Song_ID, Duration, Song_Title, Song_Genre, Album_ID, Artist_ID, Lyric_ID) and Lyrics(Lyric_ID, Lyrics, Language, Last_Updated).

   Tables:
   CREATE TABLE Artist(
   Artist_ID INT PRIMARY KEY,
   Name VARCHAR(20) NOT NULL,
   Artist_Genre VARCHAR(20),
   Country VARCHAR(20));

   CREATE TABLE Albums(
   Album_ID INT PRIMARY KEY,
   Title varchar(20) not null,
   Release_Date date,
   Artist_ID int,
   foreign key (Artist_ID) references Artist(Artist_ID));

```
CREATE TABLE Songs (
            song_id INT PRIMARY KEY,
            title VARCHAR(100) NOT NULL,
            duration TIME,
            album_id INT,
            artist_id INT,
            FOREIGN KEY (album_id) REFERENCES Albums(album_id),
            FOREIGN KEY (artist_id) REFERENCES Artists(artist_id));

CREATE TABLE Users (
             user_id INT PRIMARY KEY,
             username VARCHAR(50) NOT NULL,
             email VARCHAR(100) UNIQUE NOT NULL,
             created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

CREATE TABLE Playlists (
   playlist_id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id INT,
   FOREIGN KEY (user_id) REFERENCES Users(user_id));

CREATE TABLE Playlist_Songs (
   playlist_id INT,
   song_id INT,
   added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (playlist_id, song_id),
   FOREIGN KEY (playlist_id) REFERENCES Playlists(playlist_id),
   FOREIGN KEY (song_id) REFERENCES Songs(song_id));

CREATE TABLE User_Preferences (
   preference_id INT PRIMARY KEY,
   user_id INT,
   preferred_genre VARCHAR(50),
   preferred_artists VARCHAR(255),
   notifications_enabled BOOLEAN DEFAULT TRUE,
   dark_mode BOOLEAN DEFAULT FALSE,
   FOREIGN KEY (user_id) REFERENCES Users(user_id));

CREATE TABLE Genres (
   genre_id INT PRIMARY KEY,
   name VARCHAR(50) NOT NULL);
```

```
CREATE TABLE Song_Genres (
    song_id INT,
    genre_id INT,
    PRIMARY KEY (song_id, genre_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id),
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id));

CREATE TABLE Song_Lyrics (
    song_id INT PRIMARY KEY,
    lyrics TEXT NOT NULL,
    language VARCHAR(50),
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (song_id) REFERENCES Songs(song_id));

CREATE TABLE Venues (
    venue_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL,
    capacity INT,
    contact_info VARCHAR(100));

CREATE TABLE Concerts (
    concert_id INT PRIMARY KEY,
    artist_id INT,
    venue_id INT,
    concert_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME,
    ticket_price DECIMAL(10, 2),
    FOREIGN KEY (artist_id) REFERENCES Artists(artist_id),
    FOREIGN KEY (venue_id) REFERENCES Venues(venue_id));

CREATE TABLE User_Song_History (
    history_id INT PRIMARY KEY,
    user_id INT,
    song_id INT,
    played_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    duration_played INT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id));

CREATE TABLE royalty_payments (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    song_id INT,
```

```
artist_id INT,
album_id INT,
payment_date DATE,
amount DECIMAL(10, 2),
status VARCHAR(20) -- e.g., 'Paid', 'Pending', 'Failed');
```