TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

***

BÁO CÁO MÔN HỌC CƠ SỞ DỮ LIỆU

ĐỀ TÀI
HỆ THỐNG QUẢN LÝ VÀ PHÁT HÀNH NHẠC SỐ

Nhóm thực hiện:

1. Trần Xuân Bảo - 23020332
2. Hà Xuân Huy - 23020375
3. Phan Hoàng Dũng - 23020346

HÀ NỘI, 12/2024

MỞ ĐẦU

Bài tập lớn môn Cơ sở dữ liệu yêu cầu sinh viên áp dụng các kiến thức đã học để xây dựng một hệ thống quản lý cơ sở dữ liệu hoàn chỉnh, từ việc phân tích yêu cầu, thiết kế mô hình dữ liệu, đến triển khai và kiểm tra hệ thống. Nhóm chúng em đã lựa chọn chủ đề mô phỏng cơ sở dữ liệu của ứng dụng Spotify - một nền tảng phát nhạc trực tuyến phổ biến toàn cầu, để thực hiện bài tập này.

Báo cáo này sẽ trình bày quy trình thiết kế và triển khai cơ sở dữ liệu nhằm đáp ứng các yêu cầu nghiệp vụ của Spotify, bao gồm quản lý người dùng, bài hát, danh sách phát, và khuyến nghị cá nhân hóa. Thông qua dự án, nhóm không chỉ củng cố hiểu biết lý thuyết mà còn tích lũy kinh nghiệm thực tiễn về thiết kế và tối ưu hóa cơ sở dữ liệu trên nền tảng MySQL.

MỤC LỤC

I. Giới thiệu

Trong thời đại công nghệ số hiện nay, các nền tảng phát nhạc trực tuyến đã trở thành một phần quan trọng trong cuộc sống hàng ngày, phần nào thay đổi cách con người tiếp cận âm nhạc và đáp ứng nhu cầu giải trí của hàng trăm triệu người dùng trên toàn cầu. Trong số đó, Spotify nổi bật như một nền tảng được yêu thích và đáng tin cậy bậc nhất, với hơn 500 triệu người dùng hàng tháng. Không chỉ là một ứng dụng phát nhạc, Spotify còn có thể được xem như một hệ thống dữ liệu khổng lồ, có khả năng lưu trữ, quản lý và phân tích thông tin của hàng tỷ bài hát, nghệ sĩ và người dùng khắp thế giới.

Với lượng dữ liệu khổng lồ như vậy và được xử lý mỗi ngày, Spotify thể hiện rõ vai trò của một hệ thống quản lý cơ sở dữ liệu hiệu quả. Nó không chỉ cung cấp các dịch vụ cơ bản như nghe nhạc, tạo các danh sách phát cá nhân mà còn tích hợp các chức năng thông minh như gợi ý bài hát dựa trên sở thích và phân tích xu hướng người dùng. Điều này đòi hỏi một thiết kế cơ sở dữ liệu khoa học, chặt chẽ để đảm bảo khả năng mở rộng và tối ưu hiệu nặng.

Trong bài tập lớn này, nhóm sẽ mô phỏng lại cơ sở dữ liệu cho ứng dụng Spotify, nhằm hiểu rõ hơn về cách tổ chức, thiết kế và khai thác dữ liệu trong các hệ thống lớn. Thông qua việc xây dựng cơ sở dữ liệu này, nhóm không chỉ áp dụng các kiến thức lý thuyết về mô hình hóa và truy vấn SQL mà còn học cách vận dụng chúng vào một bối cảnh thực tiễn.

II. Chi tiết

1. Các thực thể (entities) và quy trình hoạt động (workflows)

a. Các thực thể:

- Hệ thống cơ sở dữ liệu được thiết kế dựa trên các thực thể chính, mỗi thực thể đóng vai trò quan trọng trong việc mô phỏng hoạt động của Spotify:
  - ➢ Người dùng (Users): Đại diện cho người nghe của nền tảng. Bao gồm **user_id, name, email, password, date_joined.**
  - ➢ Bài hát (Songs): Đại diện cho các bài hát. Bao gồm **song_id, title, album_id, duration, genre**.
  - ➢ Nghệ sĩ (Artists): Đại diện cho những người sáng tác bài hát và album. Bao gồm **artist_id, name**.
  - ➢ Bộ sưu tập (Albums): Đại diện cho bộ sưu tập các bài hát. Bao gồm **album_id, album_name.**
  - ➢ Thể loại (Genre): Đại gồm diện cho các thể loại âm nhạc. Bao **genre_id, name**.
  - ➢ Gợi ý (Recommendation): Đại diện cho các gợi ý hệ thống đưa ra. Bao gồm **recommendation_id, user_id, song_id, recommendation_reason, time_stamp**.
  - ➢ Lịch sử nghe (Playback history): Đại diện cho lượt nghe của người dùng. Bao gồm **user_id, song_id, total_time**.
  - ➢ Danh sách phát (Playlists): Đại diện cho danh sách bài hát được người dùng tạo ra. Bao gồm **playlist_id, playlist_name, user_id**.
  - ➢ Gói đăng ký (Subscriptions): Theo dõi các gói đăng ký của người dùng. Bao gồm **subscription_id, user_id, subscription_type, start_date, end_date**.

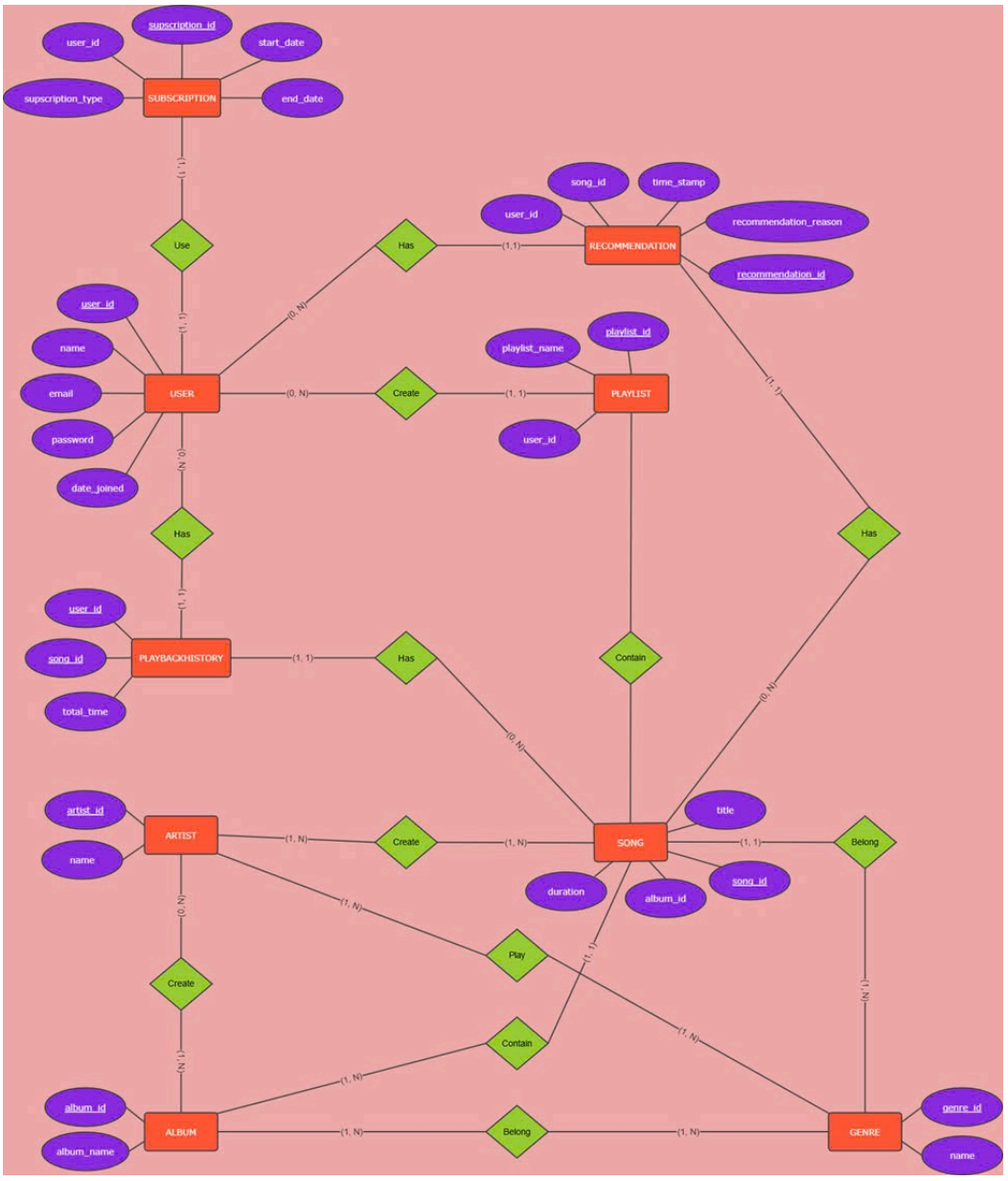
b. Quy trình hoạt động:

- Hệ thống hỗ trợ các quy trình quan trọng, tái hiện lại trải nghiệm người dùng trên nền tảng Spotify:
  - ➢ Người dùng đăng nhập vào hệ thống bằng email và mật khẩu. Sau khi đăng nhập, họ có thể tìm kiếm bài hát, nghệ sĩ, album theo từ khóa. Người dùng cũng có thể tạo và quản lý danh sách phát cá nhân, thêm hoặc xóa bài hát theo ý muốn.
  - ➢ Dựa trên thói quen nghe nhạc, danh sách phát và thể loại yêu thích của người dùng, hệ thống sẽ gợi ý các bài hát, nghệ sĩ, album và thể loại phù hợp, giúp người dùng dễ dàng khám phá các nội dung mới hoặc những nghệ sĩ nổi bật trong dòng nhạc yêu thích.
  - ➢ Người dùng thanh toán để nâng cấp từ tài khoản miễn phí lên premium. Hệ thống tự động quản lý thời hạn gói đăng ký, thông báo người dùng khi gói sắp hết hạn.

➢ Ngoài người dùng, hệ thống cũng hỗ trợ nghệ sĩ đăng nhập bằng email và mật khẩu như người dùng, thực hiện các thao tác như thêm, chỉnh sửa hoặc xóa bài hát, album và thể loại âm nhạc của mình. Nghệ sĩ có thể quản lý nội dung, theo dõi thống kê về lượt nghe đối với các bài hát, album của mình. Điều này giúp nghệ sĩ tương tác trực tiếp với người nghe, đồng thời phát triển sự nghiệp âm nhạc của mình trong môi trường trực tuyến.

2. Mô hình thực thể - quan hệ (ER Model)

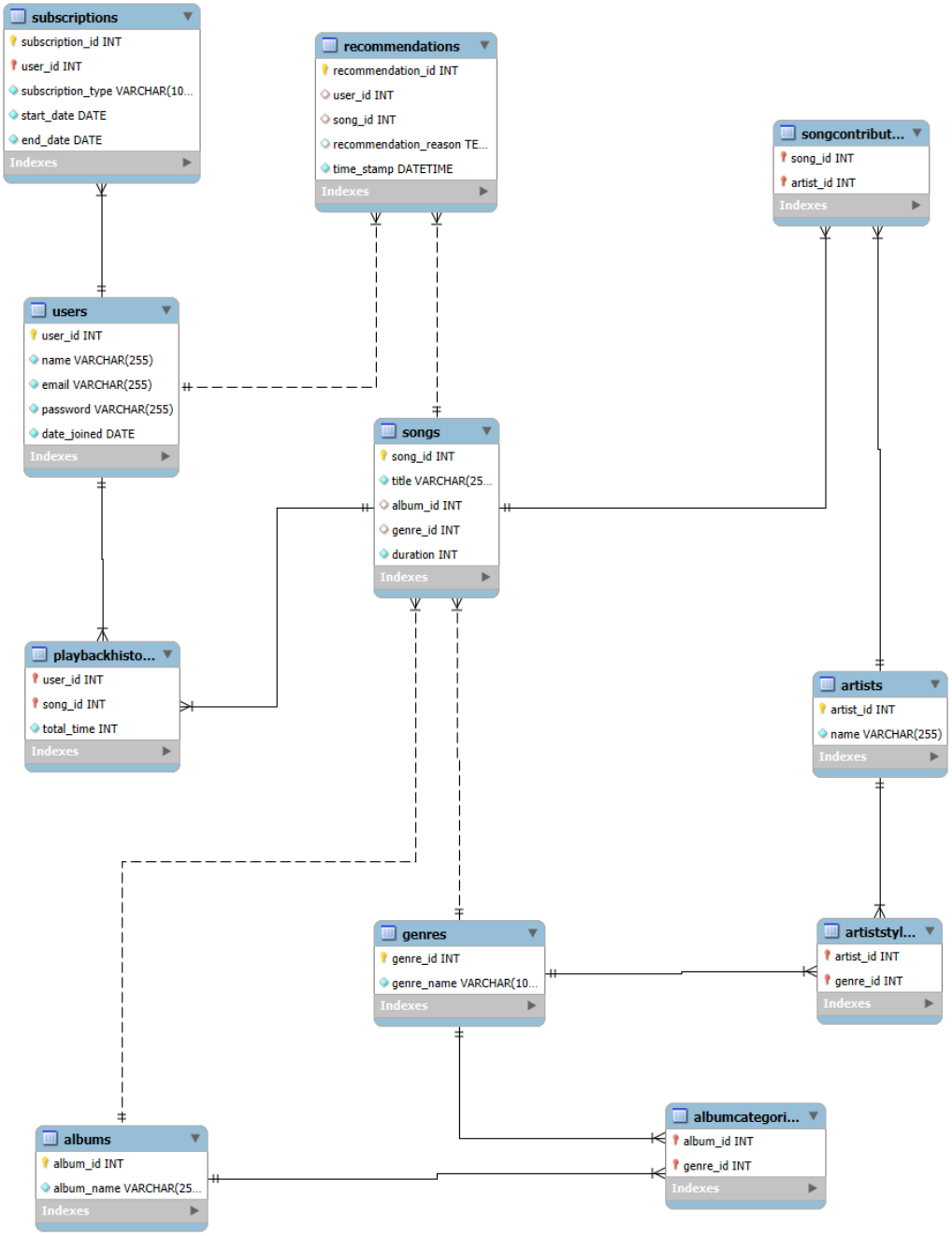
- Để hiểu rõ cách thức hoạt động của hệ thống Spotify, nhóm đã xây dựng một mô hình thực thể - quan hệ nhằm mô tả các thực thể chính và các mối quan hệ giữa chúng trong hệ thống. Mô hình này giúp chúng ta hình dung rõ ràng cách thức tổ chức dữ liệu và luồng thông tin giữa các thực thể, từ đó hỗ trợ triển khai cơ sở dữ liệu một cách khoa học và hiệu quả.



- Mô hình ER được đề xuất không chỉ mang lại cái nhìn trực quan về cấu trúc dữ liệu mà còn định hướng rõ ràng cho việc chuyển đổi sang mô hình quan hệ và triển khai cơ sở dữ liệu. Bằng cách xác định một cách chi tiết các thực thể và mối quan hệ trong hệ thống, mô hình này đảm bảo tính linh hoạt và khả năng mở rộng trong tương lai, đáp ứng những yêu cầu phức tạp của một hệ thống quản lý nhạc số hiện đại như Spotify.

3. Chuyển đổi thành mô hình quan hệ và chuẩn hóa lên 3NF

- Dựa trên mô hình ER đã xác định, nhóm sẽ chuyển đổi các thực thể và mối quan hệ thành các bảng trong mô hình quan hệ và xác định các phụ thuộc hàm của từng bảng:



- Các bảng thực thể:
    - Users (user_id (PK), name, email, password, date_joined)
        - Phụ thuộc hàm: user_id → name, email, password, date_joined
    - Songs (song_id (PK), title, album_id (FK), duration, genre_id)
        - Phụ thuộc hàm: song_id → title, album_id, duration
    - Artists (artist_id (PK), name, country)
        - Phụ thuộc hàm: artist_id → name, country
    - Albums (album_id (PK), album_name, release_date)
        - Phụ thuộc hàm: album_id → album_name, release_date
    - Genres (genre_id (PK), genre_name)
        - Phụ thuộc hàm: genre_id → genre_name
    - Recommendations (recommendation_id (PK), user_id (FK), song_id (FK), recommendation_reason, time_stamp)
        - Phụ thuộc hàm: recommendation_id → user_id, song_id, recommendation_reason, time_stamp
    - PlaybackHistory (user_id (PK, FK), song (PK, FK), total_time)
        - Phụ thuộc hàm: user_id, song_id → total_time
    - Subscriptions (subscription_id (PK), user_id (PK), subscription_type, start_date, end_date.)
        - Phụ thuộc hàm: subscription_id → user_id, subscription_type, start_date, end_date
- Các bảng trung gian:
    - SongContributor *(Many-to-Many giữa Song và Artist)*: SongContributor(song_id (FK), artist_id (FK))
    - SongCategories *(Many-to-Many giữa Song và Genre)*: SongCategories(song_id (FK), genre_id (FK))
    - AlbumCategories *(Many-to-Many giữa Album và Genre)*: AlbumCategories(album_id (FK), genre_id (FK))
    - ArtistStyles *(Many-to-Many giữa Artist và Genre)*: ArtistStyles(artist_id (FK), genre_id (FK))
- Chuẩn hóa lên 3NF: Có thể thấy toàn bộ các bảng trong cơ sở dữ liệu đã được chuẩn hóa lên dạng chuẩn 3NF, đảm bảo rằng mỗi bảng đều tuân thủ các nguyên tắc cơ bản của thiết kế cơ sở dữ liệu. Cụ thể, tất cả các phụ thuộc hàm đều được kiểm tra và xác nhận rằng không có phụ thuộc bắc cầu, đồng thời mọi thuộc tính không khóa chỉ phụ thuộc trực tiếp vào khóa chính của bảng. Ngoài ra, các bảng trung gian trong hệ thống được thiết kế để duy trì mối quan hệ nhiều-nhiều mà không tạo ra sự dư thừa dữ liệu. Việc chuẩn hóa này giúp cơ sở dữ liệu đạt tính toàn vẹn dữ liệu cao, giảm thiểu rủi ro bất thường trong thao tác cập nhật, đồng thời cải thiện hiệu quả lưu trữ và truy vấn.

4. **Tạo cơ sở dữ liệu và các bảng**

- Việc thiết kế cơ sở dữ liệu đóng vai trò quan trọng trong việc tổ chức và lưu trữ thông tin liên quan đến người dùng, bài hát, thể loại, khuyến nghị và các loại đăng ký, từ đó tạo nền tảng vững chắc cho các chức năng của hệ thống. Trong phần này, nhóm sẽ sử dụng hệ quản trị cơ sở dữ liệu là MySQL Workbench và dùng các câu lệnh SQL tương thích để tạo các bảng trong cơ sở dữ liệu:

```
CREATE DATABASE csdl;

USE csdl;

CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    date_joined DATE NOT NULL
);


CREATE TABLE Albums (
    album_id INT AUTO_INCREMENT PRIMARY KEY,
    album_name VARCHAR(255) NOT NULL
);

CREATE TABLE Genres (
    genre_id INT AUTO_INCREMENT PRIMARY KEY,
    genre_name VARCHAR(100) NOT NULL
);

CREATE TABLE Songs (
    song_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    album_id INT,
    genre_id INT,
    duration INT NOT NULL,
    FOREIGN KEY (album_id) REFERENCES Albums(album_id),
     FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
);
```

```sql
CREATE TABLE Artists (
    artist_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE SongContributor (
    song_id INT,
    artist_id INT,
    PRIMARY KEY (song_id, artist_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id),
    FOREIGN KEY (artist_id) REFERENCES Artists(artist_id)
);

CREATE TABLE AlbumCategories (
    album_id INT,
    genre_id INT,
    PRIMARY KEY (album_id, genre_id),
    FOREIGN KEY (album_id) REFERENCES Albums(album_id),
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
);

CREATE TABLE ArtistStyles (
    artist_id INT,
    genre_id INT,
    PRIMARY KEY (artist_id, genre_id),
    FOREIGN KEY (artist_id) REFERENCES Artists(artist_id),
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
);

CREATE TABLE Recommendations (
    recommendation_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    song_id INT,
    recommendation_reason TEXT,
    time_stamp DATETIME NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);
```

```sql
CREATE TABLE PlaybackHistory (
    user_id INT,
    song_id INT,
    total_time INT NOT NULL,
    PRIMARY KEY (user_id, song_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (song_id) REFERENCES Songs(song_id)
);

CREATE TABLE Subscriptions (
    subscription_id INT AUTO_INCREMENT,
    user_id INT,
    subscription_type VARCHAR(100) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    PRIMARY KEY (subscription_id, user_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

- Cấu trúc này tạo ra một cơ sở dữ liệu linh hoạt, dễ mở rộng và phù hợp để triển khai các tính năng nâng cao như gợi ý bài hát, theo dõi lịch sử phát nhạc, và quản lý gói đăng ký. Với thiết kế này, nhóm sẽ tiếp tục phát triển các tính năng ứng dụng dựa trên cơ sở dữ liệu mạnh mẽ và hiệu quả.

5. Tạo ràng buộc

- Ràng buộc (constraints) là một phần quan trọng trong việc thiết kế và quản lý cơ sở dữ liệu, nhằm đảm bảo tính toàn vẹn và chất lượng của dữ liệu. Trong phần này, nhóm sẽ sử dụng các lệnh `ALTER TABLE` để bổ sung các ràng buộc trên các bảng đã tạo trước đó, giúp cải thiện khả năng kiểm soát dữ liệu và đảm bảo các quy tắc nghiệp vụ được thực thi.

```sql
USE csdl;

-- Ensures each album name is unique
ALTER TABLE Albums
ADD CONSTRAINT unique_album_name UNIQUE (album_name);
```

```
-- Validates that the song's duration is greater than zero
ALTER TABLE Songs
ADD CONSTRAINT check_song_duration CHECK (duration > 0);

-- Ensures the subscription_type column contains valid values
like 'Free', 'Premium', etc
ALTER TABLE Subscriptions
ADD CONSTRAINT check_subscription_type CHECK
(subscription_type IN ('Free', 'Standard', 'Premium'));
```

- Việc thêm các ràng buộc như `UNIQUE`, `CHECK` không chỉ đảm bảo rằng dữ liệu trong cơ sở dữ liệu luôn hợp lệ mà còn hỗ trợ phát hiện và ngăn chặn các lỗi dữ liệu ngay từ giai đoạn nhập liệu. Đây là bước quan trọng trong việc xây dựng một cơ sở dữ liệu đáng tin cậy, tạo tiền đề vững chắc cho các hoạt động xử lý và phân tích dữ liệu sau này.

6. Chèn dữ liệu vào các bảng

- Việc chèn dữ liệu là một bước thiết yếu để kiểm tra và vận hành cơ sở dữ liệu sau khi thiết kế các bảng và ràng buộc. Trong phần này, nhóm sẽ thực hiện việc thêm dữ liệu mẫu vào từng bảng trong cơ sở dữ liệu. Các dữ liệu này sẽ được sử dụng để kiểm tra tính nhất quán của hệ thống, khả năng truy vấn, và mối quan hệ giữa các bảng.

```
USE csdl;

INSERT INTO Users (user_id, name, email, password,
date_joined) VALUES
(1, 'John Doe', 'john.doe@example.com', 'pass123',
'2024-01-01'),
(2, 'Jane Smith', 'jane.smith@example.com', 'secure456',
'2024-01-15'),
(3, 'Alice Johnson', 'alice.j@example.com', 'qwerty789',
'2024-02-10'),
(4, 'Bob Williams', 'bob.w@example.com', 'abc123',
'2024-02-20'),
(5, 'Charlie Brown', 'charlie.b@example.com', 'zxcvb456',
'2024-03-05'),
....
```

```
INSERT INTO Albums (album_id, album_name) VALUES
(1, 'Part & Parcel (Recorded Delivery)'),
(2, 'Bailamos Otra Vez'),
(3, 'Sol (From "Ponniyin Selvan Part-1") [Original Motion
Picture Soundtrack]'),
(4, 'A Thought Evoked'),
(5, 'Isles'),
....

INSERT INTO Genres (genre_id, genre_name) VALUES
(1, 'acoustic'),
(2, 'afrobeat'),
(3, 'ambient'),
(4, 'blues'),
(5, 'brazil'),
....

INSERT INTO Songs (song_id, title, album_id, genre_id,
duration) VALUES
(1, 'Di Papakawalan', 9, 1, 244),
(2, 'Laro Laro Laro', 9, 1, 174),
(3, 'Vem', 15, 2, 73),
(4, 'Feitiço', 15, 2, 262),
(5, 'Lambe-Lambe', 15, 2, 205),
....

INSERT INTO Artists (artist_id, name) VALUES
(1, 'Maris Racal'),
(2, 'Samuca e a Selva'),
(3, 'Illy'),
(4, 'Onã'),
(5, 'Bicep'),
....

INSERT INTO SongContributor (song_id, artist_id) VALUES
(1, 1),
(2, 1),
(3, 2),
```

```
(4, 2),
(5, 2),
....

INSERT INTO AlbumCategories (album_id, genre_id) VALUES
(9, 1),
(15, 2),
(5, 3),
(21, 3),
(11, 4),
(20, 4),
....

INSERT INTO ArtistStyles (artist_id, genre_id) VALUES
(1, 1),
(2, 2),
(3, 2),
(4, 2),
(5, 3),
....

INSERT INTO PlaybackHistory (user_id, song_id, total_time)
VALUES
(1, 1, 120), (1, 2, 150), (1, 3, 180), (1, 4, 160), (1, 5,
200),
(2, 6, 180), (2, 7, 120), (2, 8, 140), (2, 9, 220), (2, 10,
180),
(3, 11, 250), (3, 12, 200), (3, 13, 210), (3, 14, 230), (3,
15, 260),
(4, 16, 140), (4, 17, 160), (4, 18, 170), (4, 19, 180), (4,
20, 220),
(5, 21, 250), (5, 22, 240), (5, 23, 230), (5, 24, 200), (5,
25, 170),
....

INSERT INTO Recommendations (user_id, song_id,
recommendation_reason, time_stamp) VALUES
(1, 1, 'Great acoustic vibe for a calm day', '2024-12-11
10:15:00'),
```

```
(1, 2, 'Catchy melody, perfect for a morning drive',
'2024-12-11 10:20:00'),
(2, 3, 'Great for chilling with friends', '2024-12-11
11:00:00'),
(2, 4, 'Feels like a perfect summer song', '2024-12-11
11:10:00'),
(3, 5, 'Nice to listen after a long day', '2024-12-11
12:00:00'),
....

INSERT INTO Subscriptions (user_id, subscription_type,
start_date, end_date) VALUES
(1, 'Premium', '2024-01-01', '2025-01-01'),
(2, 'Standard', '2024-02-01', '2025-02-01'),
(3, 'Free', '2024-03-01', '2024-06-01'),
(4, 'Premium', '2024-04-01', '2025-04-01'),
(5, 'Standard', '2024-05-01', '2025-05-01'),
....
```

- Dữ liệu mẫu này đóng vai trò quan trọng trong quá trình phát triển và thử nghiệm các tính năng ứng dụng, đảm bảo rằng hệ thống cơ sở dữ liệu hoạt động ổn định và đáp ứng các yêu cầu nghiệp vụ.

7. Thực hiện các truy vấn

- Nhóm sẽ trình bày các truy vấn SQL với nhiều yêu cầu khác nhau, bao gồm sử dụng `INNER JOIN`, `OUTER JOIN`, subquery trong `WHERE` và `FROM`, cũng như `GROUP BY` với các hàm tổng hợp. Những truy vấn này không chỉ minh họa khả năng linh hoạt của SQL mà còn cho thấy cách tận dụng các công cụ mạnh mẽ để làm việc với dữ liệu phức tạp.

a. Truy vấn với `INNER JOIN`:

```
-- INNER JOIN: retrieves the names of users and the songs
they have played
SELECT Users.name AS user_name, Songs.title AS song_title
FROM Users
INNER JOIN PlaybackHistory ON Users.user_id =
```

```
PlaybackHistory.user_id
INNER JOIN Songs ON PlaybackHistory.song_id = Songs.song_id;
```

user_name	song_title
Alice Johnson	Vermelhos de Ciclos
Alice Johnson	Sincronia
Alice Johnson	Apricots
Alice Johnson	Lido
Alice Johnson	Sentient
Bob Williams	Atlas
Bob Williams	Sleigh Ride
Bob Williams	Rudolph The Red-Nosed Reindeer
Bob Williams	Hola les lolos
Bob Williams	O Fracasso Não É o Fim
Charlie Brown	I Know You (feat. Bastille) - Vigiland Remix
Charlie Brown	End of The Day
Charlie Brown	Pearl

- Câu truy vấn SQL trên sử dụng `INNER JOIN` để lấy tên người dùng và các bài hát mà họ đã nghe, với điều kiện rằng cả người dùng và bài hát phải có bản ghi liên kết trong các bảng `Users`, `PlaybackHistory` và `Songs`. Câu truy vấn sẽ chỉ trả về các kết quả có sự khớp giữa các bảng này, đảm bảo rằng mỗi người dùng chỉ được liệt kê nếu họ đã phát ít nhất một bài hát, và mỗi bài hát chỉ được liệt kê nếu nó đã được phát bởi người dùng nào đó.

b. Truy vấn với `OUTER JOIN`:

```
-- OUTER JOIN: fetch all songs and their associated
recommendations, even if some songs do not have any
recommendations
SELECT Songs.title, Users.name AS user_name,
Recommendations.recommendation_reason,
Recommendations.time_stamp
FROM Songs
LEFT OUTER JOIN Recommendations ON Songs.song_id =
Recommendations.song_id
LEFT OUTER JOIN Users ON Recommendations.user_id =
Users.user_id;
```

title	user_name	recommendation_reason	time_stamp
Lido	Emma Taylor	Perfect background music for work	2024-12-11 16:05:00
Sentient	Fiona White	For fans of upbeat pop tracks	2024-12-11 16:30:00
Sentient	Fiona White	For fans of upbeat pop tracks	2024-12-11 16:30:00
Atlas	Fiona White	The perfect track for a workout	2024-12-11 17:00:00
Atlas	Fiona White	The perfect track for a workout	2024-12-11 17:00:00
Sleigh Ride	George Black	A remix you don't want to miss	2024-12-11 17:15:00
Sleigh Ride	George Black	A remix you don't want to miss	2024-12-11 17:15:00
Rudolph The R...	George Black	It's Christmas time! Perfect for the season	2024-12-11 18:00:00
Rudolph The R...	George Black	It's Christmas time! Perfect for the season	2024-12-11 18:00:00
Hola les lolos	Hannah Scott	For fans of alternative rock	2024-12-11 18:10:00
Hola les lolos	Hannah Scott	For fans of alternative rock	2024-12-11 18:10:00
O Fracasso Nã...	Hannah Scott	An iconic song from a legendary album	2024-12-11 19:00:00
O Fracasso Nã...	Hannah Scott	An iconic song from a legendary album	2024-12-11 19:00:00
I Know You (fe...	Ian Clark	Best dancehall beat of the year	2024-12-11 19:30:00
I Know You (fe...	Ian Clark	Best dancehall beat of the year	2024-12-11 19:30:00
End of The Day	Ian Clark	Funky vibes for your party	2024-12-11 20:00:00

- Câu truy vấn trên sử dụng `LEFT OUTER JOIN` để truy xuất tất cả các bài hát và các gợi ý cho người dùng liên quan, ngay cả khi một số bài hát không có gợi ý. Cụ thể, nó kết hợp ba bảng: `Songs`, `Recommendations`, và `Users`. Kết quả trả về sẽ bao gồm tên bài hát, tên người dùng được gợi ý, lý do gợi ý, và dấu thời gian. Nếu bài hát không có gợi ý, các trường thông tin về người dùng và gợi ý sẽ là `NULL`, đảm bảo rằng tất cả bài hát đều được liệt kê.

c. Dùng truy vấn con trong `WHERE`:

```
-- SUBQUERY IN WHERE: retrieves the users who have played
songs from a specific genre (e.g., 'Pop')
SELECT Users.name AS user_name
FROM Users
WHERE Users.user_id IN (
    SELECT DISTINCT PlaybackHistory.user_id
    FROM PlaybackHistory
    JOIN Songs ON PlaybackHistory.song_id = Songs.song_id
    WHERE Songs.genre_id = (SELECT genre_id FROM Genres WHERE
genre_name = 'Pop')
);
```

user_name
Laura Walker

- Câu truy vấn sử dụng subquery trong mệnh đề `WHERE` để truy xuất những người dùng đã nghe các bài hát thuộc thể loại cụ thể, ví dụ như 'Pop'. Subquery đầu tiên tìm ra các `user_id` của người dùng đã chơi bài hát có thể loại 'Pop', bằng cách kết hợp bảng `PlaybackHistory` và `Songs`, và sử dụng `genre_id` lấy từ bảng `Genres`. Sau đó, truy vấn chính sử dụng mệnh đề `IN` để lọc ra những người dùng có `user_id` khớp với kết quả từ subquery. Kết quả trả về là tên của những người dùng đã nghe bài hát thuộc thể loại 'Pop'.

d. Dùng truy vấn con trong `FROM`:

```
-- SUBQUERY IN FROM: calculates the total number of songs
played by each user
SELECT UserSongs.user_name, COUNT(UserSongs.song_id) AS
total_songs_played
FROM (
    SELECT Users.name AS user_name, PlaybackHistory.song_id
    FROM Users
    JOIN PlaybackHistory ON Users.user_id =
PlaybackHistory.user_id
) AS UserSongs
GROUP BY UserSongs.user_name;
```

user_name	total_songs_played
John Doe	5
Jane Smith	5
Alice Johnson	5
Bob Williams	5
Charlie Brown	5
Daniel Green	5
Emma Taylor	5
Fiona White	5
George Black	5
Hannah Scott	5
Ian Clark	5
Julia Adams	5
Kevin Moore	5
Laura Walker	5
Michael Hall	5
Natalie King	5

- Câu truy vấn sử dụng truy vấn con trong mệnh đề `FROM` để tính tổng số bài hát mà mỗi người dùng đã nghe. Truy vấn con đầu tiên kết hợp bảng `Users` và `PlaybackHistory` để lấy tên người dùng (`user_name`) và các `song_id` mà họ đã chơi. Kết quả từ subquery này được đặt tên là `UserSongs`. Sau đó, truy vấn chính sử dụng `COUNT` để đếm số lượng bài hát mà mỗi người dùng đã chơi

và nhóm kết quả theo `user_name`. Kết quả trả về là tên người dùng và tổng số bài hát mà họ đã chơi.

e. Truy vấn sử dụng `GROUP BY` và hàm tổng hợp (aggregate functions):

```
-- GROUP BY and Aggregate Functions: retrieves the total number of songs per genre and the average song duration in each genre
SELECT Genres.genre_name, COUNT(Songs.song_id) AS total_songs, AVG(Songs.duration) AS average_duration
FROM Songs
JOIN Genres ON Songs.genre_id = Genres.genre_id
GROUP BY Genres.genre_name;
```

genre_name	total_songs	average_duration
acoustic	2	209.0000
afrobeat	10	216.2000
ambient	4	249.7500
blues	3	201.0000
brazil	1	289.0000
british	1	183.0000
chill	2	174.5000
country	1	122.0000
dancehall	1	184.0000
death-metal	2	246.0000
detroit-techno	2	393.5000
drum-and-b...	1	192.0000
dub	1	245.0000
electronic	2	213.0000
forro	1	200.0000
german	2	217.5000

- Câu truy vấn sử dụng `GROUP BY` và các hàm tổng hợp để lấy tổng số bài hát và thời gian trung bình của bài hát theo thể loại. Truy vấn kết hợp bảng `Songs` và `Genres` thông qua cột `genre_id`. Sau đó, sử dụng hàm `COUNT` để đếm tổng số bài hát (`total_songs`) và hàm `AVG` để tính thời gian trung bình (`average_duration`) của bài hát trong mỗi thể loại. Kết quả trả về là tên thể loại và tổng số bài hát cũng như thời gian trung bình của các bài hát trong mỗi thể loại.

8. Giao dịch sử dụng `ROLLBACK`

- Đoạn mã SQL dưới đây bắt đầu một giao dịch sử dụng `START TRANSACTION`, sau đó thực hiện việc chèn một người dùng mới vào bảng

`Users` với các thông tin như tên, email, mật khẩu và ngày gia nhập. Mặc dù người dùng được thêm vào bảng, nhưng thay vì xác nhận giao dịch, nhóm sử dụng lệnh `ROLLBACK` để hủy bỏ toàn bộ các thay đổi đã thực hiện trong giao dịch này.

```
USE csdl;

START TRANSACTION;

INSERT INTO Users (name, email, password, date_joined)
VALUES ('Test User', 'test.user@example.com',
'testpassword123', '2024-12-11');

ROLLBACK;
```

- Sau khi thực hiện lệnh `ROLLBACK`, tất cả các thay đổi trong giao dịch, bao gồm việc chèn người dùng mới vào bảng `Users`, sẽ bị hủy bỏ. Điều này có nghĩa là không có bất kỳ thay đổi nào được ghi nhận vào cơ sở dữ liệu, và trạng thái ban đầu của cơ sở dữ liệu được phục hồi.

9. Tạo một Trigger

- Đoạn mã SQL dưới đây tạo một trigger có tên `set_date_joined` để tự động thiết lập giá trị `date_joined` cho người dùng mới khi họ được thêm vào bảng `Users`. Trigger này sẽ kiểm tra nếu giá trị `date_joined` không được chỉ định (NULL), nó sẽ tự động gán giá trị hiện tại của ngày tháng (`CURDATE()`) cho trường `date_joined`.

```
-- TRIGGER: automatically set date_joined to the current date
when a new user is added
USE csdl;

DELIMITER $$

CREATE TRIGGER set_date_joined
BEFORE INSERT ON Users
FOR EACH ROW
BEGIN
```

```
    IF NEW.date_joined IS NULL THEN
        SET NEW.date_joined = CURDATE();
    END IF;
END$$

DELIMITER ;
```

- Với trigger này, nhóm đảm bảo rằng mỗi khi một người dùng mới được thêm vào bảng `Users`, trường `date_joined` sẽ được tự động điền ngày hiện tại, giúp duy trì tính nhất quán và chính xác trong cơ sở dữ liệu mà không cần can thiệp thủ công.

10. Tạo một thủ tục (Procedure)

- Đoạn mã SQL dưới đây tạo một thủ tục (procedure) có tên `GetUsersBySubscription` để truy vấn thông tin người dùng theo loại đăng ký. Thủ tục này nhận tham số đầu vào `subscription_type`, sau đó sử dụng câu lệnh `SELECT` để lấy thông tin tất cả người dùng từ bảng `Users`, kết hợp với bảng `Subscriptions` dựa trên `user_id`, và lọc kết quả theo loại đăng ký được chỉ định.

```
-- PROCEDURE: get users by subscription type
USE csdl;

DELIMITER $$

CREATE PROCEDURE GetUsersBySubscription(IN subscription_type
VARCHAR(50))
BEGIN
    SELECT u.*
    FROM Users u
    JOIN Subscriptions s ON s.user_id = u.user_id
    WHERE s.subscription_type = subscription_type;
END$$

DELIMITER ;
```

```
CALL GetUsersBySubscription("Free");
```

user_id	name	email	password	date_joined
3	Alice Johnson	alice.j@example.com	qwerty789	2024-02-10
7	Emma Taylor	emma.t@example.com	welcome1	2024-03-12
9	George Black	george.b@example.com	dragon77	2024-03-18
13	Kevin Moore	kevin.m@example.com	football9	2024-03-30
15	Michael Hall	michael.h@example.com	passw0rd	2024-04-03
17	Oliver Evans	oliver.e@example.com	letmein22	2024-04-07
20	Rachel Foster	rachel.f@example.com	unicorn5	2024-04-15
4	Bob Williams	bob.w@example.com	abc123	2024-02-20
6	Daniel Green	daniel.g@example.com	hello123	2024-03-10
8	Fiona White	fiona.w@example.com	testpass	2024-03-15

- Khi gọi thủ tục với tham số "Free", câu truy vấn sẽ trả về tất cả người dùng có loại đăng ký là "Free". Thủ tục này giúp dễ dàng truy xuất danh sách người dùng theo các loại đăng ký khác nhau mà không cần viết lại câu truy vấn mỗi lần.

III. Kết luận

Hệ thống quản lý và phát hành nhạc số Spotify đã cung cấp một nền tảng hiệu quả để quản lý dữ liệu người dùng, bài hát, khuyến nghị và các loại đăng ký. Với các tính năng như thêm người dùng, quản lý lịch sử phát nhạc, và phân loại bài hát theo thể loại, hệ thống giúp tối ưu hóa trải nghiệm người dùng và hỗ trợ phát triển các tính năng như khuyến nghị nhạc cá nhân hóa. Việc sử dụng các công cụ như trigger, procedure, và các truy vấn SQL phức tạp giúp tự động hóa các tác vụ và đảm bảo tính nhất quán của dữ liệu. Tổng thể, hệ thống này không chỉ cải thiện khả năng quản lý mà còn nâng cao sự tiện lợi trong việc cung cấp và phát hành nhạc số, đáp ứng nhu cầu ngày càng cao của người dùng trong thời đại công nghệ hóa, hiện đại hóa.

Trong tương lai, hệ thống có thể được nâng cấp và mở rộng với nhiều cải tiến nhằm đáp ứng nhu cầu ngày càng cao của người dùng và các yêu cầu kỹ thuật trong thời đại số. Trước hết, việc tích hợp các công nghệ phân tích dữ liệu tiên tiến như học máy và khai thác dữ liệu sẽ giúp hệ thống dự đoán hành vi người dùng chính xác hơn, từ đó đưa ra các gợi ý bài hát và nghệ sĩ phù hợp hơn với sở thích cá nhân. Đồng thời, tăng cường bảo mật và quyền riêng tư cũng là một yếu tố không thể thiếu. Việc áp dụng các cơ chế mã hóa dữ liệu, quản lý truy cập và giám sát an ninh sẽ giúp bảo vệ tối đa thông tin người dùng. Ngoài ra, việc bổ sung các tính năng mới như tích hợp podcast, quản lý sự kiện âm nhạc, hoặc xây dựng cộng đồng người dùng cũng là một hướng đi tiềm năng để nâng cao trải nghiệm và sự gắn kết của người dùng trên nền tảng. Những định hướng này không chỉ giúp hệ thống đáp ứng tốt hơn nhu cầu hiện tại mà còn tạo tiền đề vững chắc cho sự phát triển lâu dài và bền vững trong tương lai.