```sql
DROP DATABASE IF EXISTS greenlinerecords;
CREATE DATABASE greenlinerecords;

USE greenlinerecords;

-- project --
DROP TABLE IF EXISTS project;
CREATE TABLE project (
   project_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   project_name VARCHAR(50) NOT NULL,
   completed TINYINT(1) NOT NULL,
   PRIMARY KEY (project_id)
);


-- genre --
DROP TABLE IF EXISTS genre;
CREATE TABLE genre (
   genre_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   genre_name VARCHAR(50) UNIQUE NOT NULL,
   PRIMARY KEY (genre_id)
);


-- genre_of_project --
DROP TABLE IF EXISTS genre_of_project;
CREATE TABLE genre_of_project (
   project_id INT NULL UNIQUE,
   genre_id INT NOT NULL UNIQUE,
   PRIMARY KEY (project_id , genre_id),
   INDEX genre_of_project_genre_idx (genre_id ASC),
   INDEX genre_of_project_project_idx (project_id ASC),
   FOREIGN KEY (project_id)
       REFERENCES project (project_id),
   FOREIGN KEY (genre_id)
       REFERENCES genre (genre_id)
);


-- member --
DROP TABLE IF EXISTS club_member;
CREATE TABLE club_member (
   member_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   email VARCHAR(50) NOT NULL UNIQUE,
   lastname VARCHAR(50) NOT NULL UNIQUE,
   firstname VARCHAR(50) NOT NULL UNIQUE,
   general_meetings_attended INT NOT NULL DEFAULT 0,
   PRIMARY KEY (member_id)
```

```
);


-- ar_rep --
DROP TABLE IF EXISTS ar_rep;
CREATE TABLE ar_rep (
   rep_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   member_id INT NOT NULL UNIQUE,
   PRIMARY KEY (rep_id),
   INDEX ar_rep_club_member (member_id ASC),
   FOREIGN KEY (member_id)
       REFERENCES club_member (member_id)
);


-- artist --
DROP TABLE IF EXISTS artist;
CREATE TABLE artist (
   artist_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   rep_id INT NOT NULL,
   artist_name VARCHAR(80) NOT NULL,
   PRIMARY KEY (artist_id),
   INDEX artist (rep_id ASC),
   FOREIGN KEY (rep_id)
       REFERENCES ar_rep (rep_id)
);


-- artist_writes_project --
DROP TABLE IF EXISTS artist_writes_project;
CREATE TABLE artist_writes_project (
   project_id INT NULL,
   artist_id INT NOT NULL,
   PRIMARY KEY (project_id , artist_id),
   INDEX artist_writes_project_artist_idx (artist_id ASC),
   INDEX artist_writes_project_project_idx (project_id ASC),
   FOREIGN KEY (project_id)
       REFERENCES project (project_id),
   FOREIGN KEY (artist_id)
       REFERENCES artist (artist_id)
);


-- song --
DROP TABLE IF EXISTS song;
CREATE TABLE song (
   song_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   song_name VARCHAR(50) NOT NULL,
```

```
   PRIMARY KEY (song_id)
);


-- artist_writes_song --
DROP TABLE IF EXISTS artist_writes_song;
CREATE TABLE artist_writes_song (
   song_id INT NULL UNIQUE,
   artist_id INT NOT NULL UNIQUE,
   PRIMARY KEY (song_id , artist_id),
   INDEX artist_writes_song_artist_idx (artist_id ASC),
   INDEX artist_writes_song_song_idx (song_id ASC),
   FOREIGN KEY (song_id)
       REFERENCES song (song_id),
   FOREIGN KEY (artist_id)
       REFERENCES artist (artist_id)
);


-- track --
DROP TABLE IF EXISTS track;
CREATE TABLE track (
   song_id INT NOT NULL,
   project_id INT NOT NULL,
   PRIMARY KEY (song_id , project_id),
   INDEX track_project_idx (project_id ASC),
   INDEX track_song_idx (song_id ASC),
   FOREIGN KEY (song_id)
       REFERENCES song (song_id),
   FOREIGN KEY (project_id)
       REFERENCES project (project_id)
);


-- engineer --
DROP TABLE IF EXISTS engineer;
CREATE TABLE engineer (
   engineer_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   member_id INT UNIQUE NOT NULL,
   level ENUM('Assistant', 'Lead', 'EIT') NOT NULL,
   PRIMARY KEY (engineer_id),
   INDEX engineer_club_member_idx (member_id ASC),
   FOREIGN KEY (member_id)
       REFERENCES club_member (member_id)
);


-- project_assignment --
```

```
DROP TABLE IF EXISTS project_assignment;
CREATE TABLE project_assignment (
   project_id INT NULL,
   engineer_id INT NOT NULL,
   PRIMARY KEY (project_id , engineer_id),
   INDEX project_assignment_engineer_idx (engineer_id ASC),
   INDEX project_assignment_project_idx (project_id ASC),
   FOREIGN KEY (project_id)
       REFERENCES project (project_id),
   FOREIGN KEY (engineer_id)
       REFERENCES engineer (engineer_id)
);


-- recording_session --
DROP TABLE IF EXISTS recording_session;
CREATE TABLE recording_session (
   recording_session_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   project_id INT NOT NULL,
   date DATETIME NOT NULL,
   PRIMARY KEY (recording_session_id , project_id),
   INDEX recording_session_project_idx (project_id ASC),
   FOREIGN KEY (project_id)
       REFERENCES project (project_id)
);


-- assigned_recording_session --
DROP TABLE IF EXISTS assigned_recording_session;
CREATE TABLE assigned_recording_session (
   engineer_id INT NOT NULL,
   recording_session_id INT NULL,
   PRIMARY KEY (engineer_id , recording_session_id),
   INDEX assigned_recording_session_recording_session_idx (recording_session_id
ASC),
   INDEX assigned_recording_session_engineer_idx (engineer_id ASC),
   FOREIGN KEY (engineer_id)
       REFERENCES engineer (engineer_id),
   FOREIGN KEY (recording_session_id)
       REFERENCES recording_session (recording_session_id)
);


-- live_session --
DROP TABLE IF EXISTS live_session;
CREATE TABLE live_session (
   live_session_id INT NOT NULL UNIQUE AUTO_INCREMENT,
   date DATETIME NOT NULL,
```

```
    PRIMARY KEY (live_session_id)
);


-- event --
DROP TABLE IF EXISTS `event`;
CREATE TABLE `event` (
    event_id INT NOT NULL UNIQUE AUTO_INCREMENT,
    date DATETIME NOT NULL,
    description VARCHAR(700) NULL,
    turnout ENUM('Low', 'Medium', 'High') NULL,
    PRIMARY KEY (event_id)
);


-- booking --
DROP TABLE IF EXISTS booking;
CREATE TABLE booking (
    event_id INT NULL,
    artist_id INT NOT NULL,
    PRIMARY KEY (event_id , artist_id),
    INDEX booking_artist_idx (artist_id ASC),
    INDEX booking_event_idx (event_id ASC),
    FOREIGN KEY (event_id)
        REFERENCES `event` (event_id),
    FOREIGN KEY (artist_id)
        REFERENCES artist (artist_id)
);


-- assigned_live_session --
DROP TABLE IF EXISTS assigned_live_session;
CREATE TABLE assigned_live_session (
    live_session_id INT NULL,
    engineer_id INT NULL,
    PRIMARY KEY (live_session_id , engineer_id),
    INDEX assigned_live_session_engineer_idx (engineer_id ASC),
    INDEX assigned_live_session_live_session_idx (live_session_id ASC),
    FOREIGN KEY (live_session_id)
        REFERENCES live_session (live_session_id),
    FOREIGN KEY (engineer_id)
        REFERENCES engineer (engineer_id)
);


-- release --
DROP TABLE IF EXISTS `release`;
CREATE TABLE `release` (
```

```
    release_id INT NOT NULL AUTO_INCREMENT,
    project_id INT NOT NULL,
    plays INT NOT NULL DEFAULT 0,
    release_date DATE NOT NULL,
    PRIMARY KEY (release_id),
    INDEX release_project_idx (project_id ASC),
    FOREIGN KEY (project_id)
        REFERENCES project (project_id)
);


-- department --
DROP TABLE IF EXISTS department;
CREATE TABLE department (
    department_id INT NOT NULL,
    dept_head_id INT NOT NULL,
    title VARCHAR(30) NULL,
    PRIMARY KEY (department_id , dept_head_id),
    INDEX department_club_member_idx (dept_head_id ASC),
    FOREIGN KEY (dept_head_id)
        REFERENCES club_member (member_id)
);


-- department_membership --
DROP TABLE IF EXISTS department_membership;
CREATE TABLE department_membership (
    member_id INT NOT NULL,
    department_id INT NOT NULL,
    dept_meetings_attended INT NOT NULL DEFAULT 0,
    PRIMARY KEY (member_id , department_id),
    INDEX department_membership_department_idx (department_id ASC),
    INDEX department_membership_club_member_idx (member_id ASC),
    FOREIGN KEY (member_id)
        REFERENCES club_member (member_id),
    FOREIGN KEY (department_id)
        REFERENCES department (department_id)
);


-- eboard_member --
DROP TABLE IF EXISTS eboard_member;
CREATE TABLE eboard_member (
    title VARCHAR(30) NOT NULL,
    member_id INT NOT NULL,
    eboard_id INT NOT NULL UNIQUE AUTO_INCREMENT,
    INDEX eboard_member_club_member_idx (member_id ASC),
    PRIMARY KEY (eboard_id),
```

```
    FOREIGN KEY (member_id)
        REFERENCES club_member (member_id)
);


-- link --
DROP TABLE IF EXISTS link;
CREATE TABLE link (
    type ENUM('Bandcamp', 'Soundcloud', 'Spotify', 'Apple Music') NOT NULL,
    url VARCHAR(300) NOT NULL,
    link_id INT NOT NULL AUTO_INCREMENT,
    release_id INT NOT NULL,
    PRIMARY KEY (link_id),
    INDEX fk_Link_Release1_idx (release_id ASC),
    FOREIGN KEY (release_id)
        REFERENCES `release` (release_id)
);
```