AT Music
Database Schema

Admin.sql:

```
CREATE TABLE IF NOT EXISTS admin (
    admin_id BIGSERIAL PRIMARY KEY NOT NULL,
    admin_name VARCHAR(50) NOT NULL UNIQUE,
    admin_password VARCHAR(50) NOT NULL
);
```

Album.sql

```
CREATE TABLE IF NOT EXISTS album (
    album_id SERIAL PRIMARY KEY NOT NULL,
    album_name VARCHAR(255) NOT NULL,
    artist_id INTEGER NOT NULL,
    album_year INTEGER,
    album_artwork VARCHAR(255),
    FOREIGN KEY (artist_id) REFERENCES artist (artist_id),
    CONSTRAINT unique_album_name_year UNIQUE (album_name, album_year)
);
```

Artist.sql

```
CREATE TABLE artist (
    artist_id BIGSERIAL NOT NULL PRIMARY KEY,
    artist_name VARCHAR(100),
    alias VARCHAR(100),
    artist_intro_video VARCHAR(255),
    artist_image VARCHAR(255),
    small_biography TEXT
);
```

Awards.sql

```
CREATE TABLE IF NOT EXISTS awards_list (
    award_id BIGSERIAL PRIMARY KEY NOT NULL,
    award_name VARCHAR(50) NOT NULL,
    award_category VARCHAR(50) NOT NULL,
    category_description VARCHAR(255) NOT NULL
);
```

Chat.sql

```
CREATE TABLE IF NOT EXISTS chat (
   chat_id SERIAL PRIMARY KEY,
   sender_id INTEGER NOT NULL,
   receiver_id INTEGER NOT NULL,
   message bytea NOT NULL,
   message_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
   FOREIGN KEY (sender_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
   FOREIGN KEY (receiver_id) REFERENCES user_db(user_id) ON DELETE CASCADE
);
```

Friends.sql

```
CREATE TABLE IF NOT EXISTS friends (
   friendship_id SERIAL PRIMARY KEY,
   user1 INTEGER NOT NULL,
   user2 INTEGER NOT NULL,
   date_connected TIMESTAMP NOT NULL,
   CONSTRAINT fk_user1 FOREIGN KEY (user1) REFERENCES user_db(user_id) ON
DELETE CASCADE,
   CONSTRAINT fk_user2 FOREIGN KEY (user2) REFERENCES user_db(user_id) ON
DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS friend_request (
   request_id SERIAL PRIMARY KEY,
   sender INTEGER NOT NULL,
   recipient INTEGER NOT NULL,
   request_sent TIMESTAMP NOT NULL,
   CONSTRAINT fk_sender FOREIGN KEY (sender) REFERENCES user_db(user_id) ON
DELETE CASCADE,
   CONSTRAINT fk_recipient FOREIGN KEY (recipient) REFERENCES user_db(user_id) ON
DELETE CASCADE
);
```

Genre.sql

```
CREATE TABLE IF NOT EXISTS genre (
   genre_id BIGSERIAL PRIMARY KEY NOT NULL,
   genre_name VARCHAR(50) NOT NULL,
   genre_image_url VARCHAR(255)
);
```

Like_table.sql

```sql
CREATE TABLE IF NOT EXISTS liked_song (
   user_id INTEGER NOT NULL,
   song_id INTEGER NOT NULL,
   PRIMARY KEY (user_id, song_id), -- Define composite primary key
   FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
   FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS liked_album (
   user_id INTEGER NOT NULL,
   album_id INTEGER NOT NULL,
   PRIMARY KEY (user_id, album_id), -- Define composite primary key
   FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
   FOREIGN KEY (album_id) REFERENCES album(album_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS liked_artist (
   user_id INTEGER NOT NULL,
   artist_id INTEGER NOT NULL,
   PRIMARY KEY (user_id, artist_id), -- Define composite primary key
   FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
   FOREIGN KEY (artist_id) REFERENCES artist(artist_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS liked_genre (
   user_id INTEGER NOT NULL,
   genre_id INTEGER NOT NULL,
   PRIMARY KEY (user_id, genre_id), -- Define composite primary key
   FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
   FOREIGN KEY (genre_id) REFERENCES genre(genre_id) ON DELETE CASCADE
);
```

News.sql

```sql
CREATE TABLE IF NOT EXISTS artist_news (
   id SERIAL PRIMARY KEY,
   title VARCHAR(255) NOT NULL,
   artist_id INTEGER NOT NULL,
   content TEXT NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (artist_id) REFERENCES artist(artist_id) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS song_news (
    id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    song_id INTEGER NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE
);
```

People.sql

```
CREATE TABLE IF NOT EXISTS people(
    person_id BIGSERIAL PRIMARY KEY NOT NULL,
    name VARCHAR(50) NOT NULL,
    nationality VARCHAR(50) NOT NULL,
    gender VARCHAR(10),
    biography TEXT
);

CREATE TABLE IF NOT EXISTS composer (
    composer_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    FOREIGN KEY (composer_id) REFERENCES people(person_id),
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE,
    PRIMARY KEY (composer_id, song_id)
);

CREATE TABLE IF NOT EXISTS producer (
    producer_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    FOREIGN KEY (producer_id) REFERENCES people(person_id),
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE,
    PRIMARY KEY (producer_id, song_id)
);

CREATE TABLE IF NOT EXISTS lyricist (
    lyricist_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    FOREIGN KEY (lyricist_id) REFERENCES people(person_id),
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE,
    PRIMARY KEY (lyricist_id, song_id)
);
```

Platform_rec_table.sql

```
CREATE TABLE platform (
   platform_id BIGSERIAL PRIMARY KEY NOT NULL,
   platform_name VARCHAR(50) NOT NULL,
   total_visit INTEGER
);

CREATE TABLE rec_type (
   rectype_id BIGSERIAL PRIMARY KEY NOT NULL,
   rectype_name VARCHAR(30) NOT NULL,
   total_visit INTEGER
);

CREATE TABLE IF NOT EXISTS platform_song (
   platform_id INTEGER REFERENCES platform(platform_id) ON DELETE CASCADE NOT
NULL,
   song_id INTEGER NOT NULL
);

CREATE TABLE IF NOT EXISTS recording_song (
   rectype_id INTEGER REFERENCES rec_type(rectype_id) ON DELETE CASCADE NOT
NULL,
   song_id INTEGER NOT NULL
);
```

Playlist.sql

```
CREATE TABLE IF NOT EXISTS user_playlist (
   playlist_id BIGSERIAL PRIMARY KEY NOT NULL,
   user_id INTEGER NOT NULL,
   playlist_name VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS playlist (
   playlist_id INTEGER NOT NULL,
   song_id INTEGER NOT NULL,
   FOREIGN KEY (playlist_id) REFERENCES user_playlist(playlist_id) ON DELETE
CASCADE,
   PRIMARY KEY (playlist_id, song_id)
);
```

Purchase.sql

```
CREATE TABLE IF NOT EXISTS purchase_history (
    purchase_id BIGSERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    purchase_date TIMESTAMP NOT NULL,
    FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS cart (
    user_id INTEGER NOT NULL,
    song_id INTEGER NOT NULL,
    PRIMARY KEY (user_id, song_id),
    FOREIGN KEY (user_id) REFERENCES user_db(user_id) ON DELETE CASCADE,
    FOREIGN KEY (song_id) REFERENCES song(song_id) ON DELETE CASCADE
);
```

Reviews.sql

```
CREATE TABLE IF NOT EXISTS reviews (
    review_id BIGSERIAL PRIMARY KEY NOT NULL,
    user_id INTEGER REFERENCES user_db(user_id) ON DELETE CASCADE NOT NULL,
    song_id INTEGER REFERENCES song(song_id) ON DELETE CASCADE NOT NULL,
    review_text TEXT,
    rating INTEGER NOT NULL,
    review_date TIMESTAMP NOT NULL
);
```

Song_synopsis.sql

```
CREATE TABLE IF NOT EXISTS song_synopsis(
    song_id INTEGER NOT NULL,
    synopsis TEXT NOT NULL
);
```

Song.sql

```
CREATE TABLE IF NOT EXISTS song (
   song_id BIGSERIAL PRIMARY KEY NOT NULL,
   artist_id INTEGER NOT NULL,
   name VARCHAR(255) NOT NULL,
   -- Specify the maximum length for VARCHAR
   album_id INTEGER NOT NULL,
   song_length INTERVAL,
   age_rating INTEGER,
   popularity INTEGER CHECK (
      popularity BETWEEN 0
      AND 10
   ),
   price DECIMAL,
   genre_id INTEGER,
   CONSTRAINT fk_artist FOREIGN KEY (artist_id) REFERENCES artist (artist_id) ON
DELETE CASCADE,
   CONSTRAINT fk_album FOREIGN KEY (album_id) REFERENCES album (album_id) ON
DELETE CASCADE,
   CONSTRAINT fk_genre FOREIGN KEY (genre_id) REFERENCES genre (genre_id) ON
DELETE CASCADE -- Specify the referenced table
);
```

User_db.sql

```
CREATE TABLE user_db (
   user_id SERIAL PRIMARY KEY NOT NULL,
   username VARCHAR(50) UNIQUE NOT NULL,
   password TEXT NOT NULL,
   email VARCHAR(50) UNIQUE NOT NULL,
   phone_number VARCHAR(50) UNIQUE NOT NULL,
   created_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   last_login TIMESTAMP,
   last_logout TIMESTAMP,
   last_updated TIMESTAMP
);
```

Prepared by:

1. 2105027
2. 2105028