2022-1 데이터베이스(01)

이플리: Ewha Playlist

6조 캐주얼 사용자들

김민서, 김서연, 김현민, 안채연, 최한비

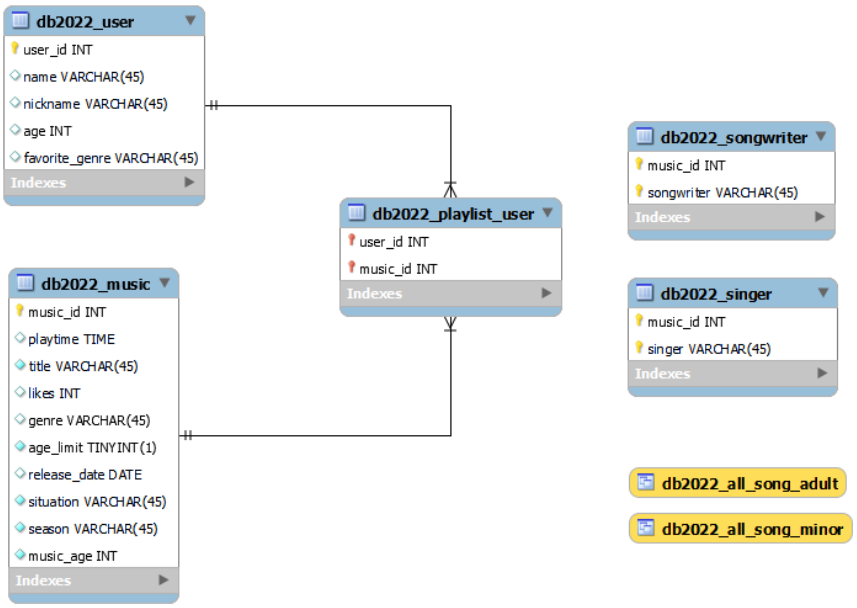
1. 데이터베이스 필요성 및 요구분석

음악 서비스 이용자들의 비용과 시간 투자는 지속적으로 상승하는 추세이다. 웹과 앱의 콘텐츠 이용자들은 시간이 흐를수록 음악 콘텐츠에 많은 비용과 시간을 할애하고 있다. 특히 스마트 기기의 보급이 확산되면서 다운로드와 스트리밍 서비스가 결합된 복합상품으로 음악을 감상하는 환경이 보편화된 상황이다. 더나아가 최근에는 영화, 음악, 연예인, 방송 등과 같은 엔터테인먼트(Entertainment) 사업에서도 음악이 점차 대중 중심 문화로서 자리 잡고 있다. 음악의 경우에는 그 나라의 문화수준을 측정하는 잣대로서의 역할을 하고 있으며, 하루에도 몇 천곡씩의 음악이 대중들 앞으로 흘러나오고 있다.

또한 아직까지도 국내 음악서비스는 연결이 잘못된 음원 DB 와 메타 DB 가 서비스되는 경우가 종종 발견된다. 해당 음악의 제작 환경과 저작권자에 대한 정확한 정보 제공이 부족하며, 실연자에 대한 정보는 아예 DB 체계조차 존재하지 않는것이 현실이다.

이처럼, 방대한 시중 음악 데이터를 정확하게 관리할 수 있는 음악 DBMS 시스템이 절실히 필요한 상황에서 이플리가 그 역할을 톡톡히 해낼 수 있다. 이뿐만 아니라, 이플리는 사용자들의 요구사항에 맞는 여러가지 기능들을 포함시킴으로서 만족스러운 사용자 경험이 될 수 있도록 한다. 첫째로, 광범위한 음악 데이터 속 나만의 맞춤 플레이리스트를 생성 및 관리하고자 하는 사용자들을 위한 나만의 플레이리스트 기능을 포함하고 있다. 둘째, Youtube [공부할 때 듣기 좋은 플레이리스트] 같이 특정 기준에 따른 선별된 음원 목록만을 감상하고 싶어하는 이용자들을 위해 공부,운동,출퇴근,여행 등 상황별로 적합한 음원을 선정하고 나이대별로 선호되는 음악을 골라서 감상할 수 있도록 하는 추천 플레이리스트 기능을 제공하고 있다. 셋째, 광범위한 음악 데이터 중 취향에 맞는 곡 정보를 자동으로 추천받고 싶은 사용자를 위해 사용자의 개인 데이터(선호장르)를 토대로 사용자가 좋아할 만한 곡들을 자동 추천해주는 기능을 포함하고 있다. 마지막으로, 무분별한 음악 접근성이 문제가 되는 상황에서 미성년자를 위한 정제된 음악 서비스가 필요한데, 이플리는 사용자의 나이 정보를 기반으로 음악들을 조회하고 추천받을 수 있도록 하는 정제된 필터링 서비스로 미성년자를 보호한다.

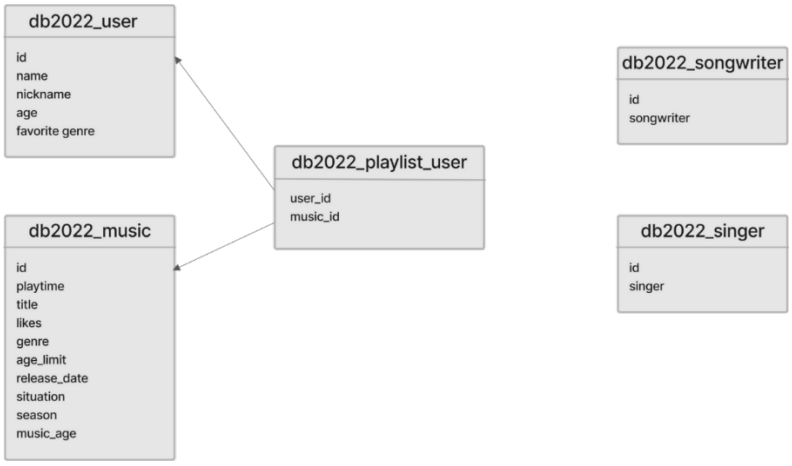
2. ER 다이어그램



음원관리 DBMS 이플리에는 총 5 개의 스키마가 존재한다.

- db2022_user: 유저정보를 저장한다.
- db2022_music: 음원 정보를 저장한다.
- db2022_songwriter: 음원의 작사작곡가를 저장한다. 하나의 음악에 작사작곡가는 multi-valued attribute 이므로 스키마를 따로 두었다.
- db2022_singer: 음원의 작사작곡가 정보를 저장한다. 하나의 음악에 가수는 multi-valued attribute 이므로 스키마를 따로 두었다.
- db2022_playlist_user: 사용자의 개인 플레이리스트 정보를 저장한다. playlist_user 에는 두개의 외래키가 존재한다. user_id 는 user 테이블의 user_id 를 참조하며, music_id 컬럼은 music 테이블의 music_id 를 참조한다.

3. 데이터베이스 스키마 다이어그램



4. 자바 코드의 클래스(Class)와 메소드(Method)에 대한 설명

1) code structure

(1) 메인 화면

a. MainFrame

i. 반복문을 통해 각 세부 페이지로 이동하는 8 개의 버튼을 생성하고 각각에 ActionListener 를 부착한다.

ii. ActionListener 는 switch-case 문에서 btnNum 을 기준으로 분기하여 이에 알맞은 클래스가 인스턴스가 생성되도록 하였다.

iii. 하나의 메인프레임만을 사용하였으며 메인프레임의 컨텐트팬 위에 부착되는 Panel 을 교체함으로서 페이지를 전환하였다.

(2) 전체 음악 목록

a. MusicList

i. 반복문을 통해 장르별 음원 목록을 출력해주는 7 개의 버튼 (전체음악목록조회,K-POP,발라드,랩/힙합,록/메탈,POP,트로트)을 생성하고 각각에 ActionListener 를 부착한다.

ii. ActionListener 는 switch-case 문에서 btnNum 을 기준으로 분기하여 SelectMusicList (String genre) 에 알맞은 파라미터를 넣어 호출되도록 하였다.

iii. SelectMusicList 내부에서는 파라미터로 입력된 genre 를 사용해서 해당 장르에 맞는 음원 만을 선택하는 select 쿼리를 만들고 실행시킨다.

(3) Top 10

a. TOP10

i. order by likes desc 구문을 이용한 select 쿼리를 통해 전체음원 목록 중 좋아요가 높은순으로 10 위까지 출력시킨다.

(4) 나만의 플레이리스트

a. NickName

i. Nickname(Container contentpane, MainFrame mainframe, JPanel panel) 생성자에서 JTextField()를 통해 사용자의 닉네임을 입력받는다.

ii. select 쿼리을 이용해 user 테이블에서 nickname 정보를 불러오고, getString 으로 가져온 유저 정보와 JTextfield 에 입력된 닉네임이 일치하는지 확인한다

iii. 입력한 닉네임이 유저 정보와 일치하지 않으면 회원 정보가 없다는 JLabel, 회원가입 JButton()을 부착하여 회원가입을 유도한다

iv. 입력한 닉네임이 유저 정보와 일치하면 nickname 을 playlist_main()에 인자로 넘겨 마이플레이리스트 메뉴 화면으로 전환한다

b. playlist_main

i. 음악추가, 음악삭제, 나만의 플레이리스트, 전체 음악 목록 4 가지 메뉴를 JButton()으로 부착하고 각각 addActionListener 를 통해 세부 페이지로 이동한다
ii. 음악 추가 버튼을 누르면 입력한 nickname 을 인자로 music_Info_insert()에 넘겨주어 음악 추가 화면으로 전환한다.
iii. 음악 삭제 버튼을 누르면 입력한 nickname 을 인자로 music_Info_delete()에 넘겨주어 음악 삭제 화면으로 전환한다.
iv. 나만의 플레이리스트 조회하기 버튼을 누르면 nickname 을 user_musicPlaylist(String nickname)에 인자로 넘겨주어 플레이리스트를 조회한다
    0. select 쿼리를 이용해 playlist_user 테이블에서 nickname 이 일치하는 user_id 를 찾고, music 테이블과 user 테이블에서 title, singer 정보를 불러온다
    1. 인자로 넘겨받은 nickname 은 사용자가 입력하는 것이므로 PreparedStatement 를 사용한다
v. 전체 음악 목록 조회를 선택하면 nickname 을 Total_musicPlaylist(String nickname)에 인자로 넘겨주어 전체 플레이리스트를 조회한다
    0. select 쿼리를 이용하여 user 테이블에서 nickname 과 일치하는 사용자의 나이 정보를 불러온다
    1. 성인용 음악 목록(db2022_all_song_adult) / 미성년자용 음악 목록(db2022_all_song_minor) 두 개의 view 를 생성하여 사용자의 나이에 맞게 출력한다

c. music_info_insert / music_info_delete
i. JTextField()를 이용해 추가/삭제할 곡의 title, singer 를 입력받는다
ii. 추가 버튼을 누르면 nickname, title, singer 를 insert(String nickname, String title, String singer)에, 삭제버튼을 누르면 delete(String nickname, String title, String singer) 함수에 인자로 넘겨준다
iii. insert(String nickname, String title, String singer) / delete(String nickname, String title, String singer)에서 select subquery 를 통해 music, singer 테이블에서 music_id 에 대한 정보를 불러오고 user 테이블에서 user_id 정보를 가져온다.
iv. insert()에서는 insert 쿼리를 통해 user_id 와 music_id 를 playlist_user 테이블에 추가하고 delete()에서는 delete 쿼리를 통해 user_id 와 music_id 를 playlist_user 테이블에서 삭제한다
v. select 쿼리로 music 테이블에서 곡의 age_limit 정보를 불러오고, user 테이블에서 사용자의 age 정보를 불러온다. If 문을 통해 미성년자의 경우 연령 제한이 있는 곡을 플레이리스트에 추가/삭제하지 못하도록 하였다

vi. nickname, title, singer 는 사용자가 입력하는 것이므로 PreparedStatement 를 사용한다

(5) 추천 플레이리스트

a. Playlist_recommended

i. initJpanel_topic(Container contentpane, MainFrame mainframe, JPanel panel)

0. '상황별', '계절별', '나이별' 주제 버튼을 생성하여 새로운 JPanel 에 부착하고, 이를 parameter 로 전달 받은 Container contentpane 에 부착한다.
1. 각각의 버튼을 클릭하면 세부 조건 페이지로 넘어가게 하는 ActionListener 를 부착한다.

ii. initJpanel_detail(JPanel panel_detail, int topicNum)

0. 상황별 주제는 '운동~출퇴근', 계절별 주제는 '봄~겨울', 나이별 주제는 '10 대~60 대 이상'의 세부 조건 버튼을 생성하여 새로운 JPanel 에 부착한다.

   이때 parameter topicNum 을 통해 주제의 타입을 전달 받고, topicNum 을 이용하여 세부 조건 2 차원 배열에서 주제에 알맞는 조건을 보여준다.

1. 각 버튼을 클릭하면 해당 세부 조건에 따라 show_playlist 메소드가 실행된다.

iii. show_playlist(int topicNum, String detail)

0. parameter 로 선택한 주제(계절 등)와 세부 조건(여름 등)을 입력받는다.
1. db2022_singer 와 db2022_songwriter 의 경우, 하나의 음악마다 여러 인물이 중복으로 참여한 경우가 있으므로, 각각을 music_id 로 GROUP_CONCAT 처리 한 후에 사용한다. 이를 통해 하나의 music_id 당 singer 정보와 songwriter 정보가 하나의 tuple 로 묶이게 된다.
2. 해당 조건에 알맞는 플레이리스트는 음악을 제목, 가수, 작사작곡가, 재생시간, 좋아요 수, 장르, 발매일의 형식으로 콘솔창에 출력하며 제공한다.

(6) 음악 검색

a. Search

i. show_title(String title)

0. parameter 로 사용자가 입력한 제목 검색어 title 을 받는다.

1. PreparedStatement 를 이용한 select 쿼리문에서 음악의 title 부분을 ?으로 지정한 후, 입력 받은 title 검색어를 이용하여 해당 음악을 찾는다.
2. 음악이 존재하면 제목, 가수, 작사작곡가, 재생시간, 좋아요 수, 장르, 발매일의 형식으로 콘솔창에 출력한다. 만약 입력 받은 제목을 가진 음악이 존재하지 않으면, '[검색어] 제목의 음원이 존재하지 않는다'라고 출력한다.

ii. show_singer(String singer)

0. parameter 로 사용자가 입력한 가수명 검색어 singer 를 받는다.
1. PreparedStatement 를 이용한 select 쿼리문에서 음악의 singer 부분을 ?으로 지정한 후, 입력 받은 title 검색어를 이용하여 해당 음악을 찾는다.
2. 음악이 존재하면 제목, 가수, 작사작곡가, 재생시간, 좋아요 수, 장르, 발매일의 형식으로 콘솔창에 출력한다. 만약 입력 받은 가수명을 가진 음악이 존재하지 않으면, '[검색어] 가수명의 음원이 존재하지 않는다'라고 출력한다.

(7) 음악 추천

a. Recommend

i. Recommend(Container contentpane, MainFrame mainframe, JPanel panel) 생성자에서 MainFrame 의 Container, MainFrame 객체, JPanel 을 넘겨 받아 새로운 JPanel 을 생성하고 넘겨 받은 Container 에 부착한다.

ii. 생성한 JPanel 에 메인화면으로 돌아가는 JButton, 닉네임 입력 안내 JLabel, 닉네임 입력 JTextField, 입력 후 정보를 전송하는 JButton 4 개 컴포넌트를 부착한다.

iii. 사용자가 textfield 에 닉네임을 입력하고 입력 버튼을 누르면 입력 받은 닉네임 정보를 인자로 넘겨 model 객체를 통해 Recommend_Model.java 파일에 있는 recommend(nickname) 함수를 호출한다.

b. Recommend_Model - recommend(String nickname)

i. MySQL Database 와 connection 을 맺는다. 연결에 실패한 경우 fail 메시지와 함께 구체적인 에러 내용을 출력한다.

ii. 작성해둔 쿼리를 이용해 PreparedStatement 객체를 생성한다.

iii. 사용자로부터 입력 받은 nickname 을 쿼리 1 과 쿼리 4 에 setString() 함수를 통해 매개변수로 전달하고 쿼리를 실행하여 각각 rs1, rs4 ResultSet 에 실행 결과를 저장한다.

iv. rs1 에는 유저의 선호 장르와 유저가 마이 플레이리스트에 담은 곡의 장르 전체에 해당하는 곡 목록이 저장되어 있으며, 이를 do while 루프를 이용해 한 튜플씩 정보를 출력한다.

0. r1 에서 출력하려는 곡의 age_limit 이 true 고 rs4 에 저장된 유저의 연령이 20 세 미만이라면 해당 튜플은 출력하지 않고 건너뛴다.

1. 만약 위의 if 문 조건에 해당하지 않는다면 해당 튜플의 제목, 가수, 작사작곡가, 재생시간, 좋아요 수, 장르, 발매일 정보를 출력한다.
2. 이때 가수 정보는 현재 출력하려는 rs1 튜플의 music_id 를 쿼리에 넣고 실행하여 rs2 에 저장한 후 do while 루프로 출력한다.
3. 마찬가지로 작사작곡가 정보도 현재 출력하려는 rs1 튜플의 music_id 를 쿼리에 넣고 실행하여 rs3 에 저장한 후 do while 루프로 출력한다. 나머지 정보는 rs1 에서 가져와 출력한다.

(8) 음원 관리

a. Management1
   i. 비밀번호 입력창과 비밀번호를 입력했을 때 나오는 관리자 전용 창을 구현한다.
   ii. Management1(Container contentpane, MainFrame mainframe, JPanel prevPanel, int i) 생성자에서 받아온 정수 i 값을 통해 창을 변경한다.
   iii. 만약 i 가 1 일 경우
      0. 메인 프레임에서 받아온 container 에 manaPa1 이라는 새로운 Panel 을 부착한다.
      1. 생성자에서 받아온 prevPanel 을 통해 뒤로가기 버튼을 구현한다.
      2. TextField 를 통해 비밀번호를 입력받는다. 비밀번호는 'DB2022Team06'이다. 비밀번호가 틀리면 OptionPane 을 통해 비밀번호가 틀렸음을 알린다. 비밀번호가 맞으면 Management1 을 통해 새로운 창을 부착한다. 이때 i 를 2 로 조정한다.
   iv. 만약 i 가 2 일 경우
      0. 사용자가 입력한 값과 비밀번호가 일치하면 보여지는 창으로, 이곳에서 음악을 추가할지, 수정할지, 삭제할지를 결정할 수 있다.
      1. 메인 프레임에서 받아온 container 에 manaPa2 이라는 새로운 Panel 을 부착한다.
      2. 반복문을 통해 3 개의 버튼 ("새로운 음악 추가", "기존 음악 수정", "기존 음악 삭제")을 생성하고 따로 정의한 MyListener 를 부착한다.
      3. MyListener 은 Management2 객체를 통해 MainFrame 의 Container, MainFrame 객체, 현재 창의 Panel 과 사용자가 어떤 버튼을 눌렀는지 알리는 btnNum 을 인자로 넘겨준다.

b. Management2
   i. 음악 추가, 수정, 삭제를 돕기 위해 해당 음악의 정보를 받아오는 창이다.
   ii. Management2(Container contentpane, MainFrame mainframe, JPanel prevPanel, int btnNum) 생성자에서 받아온 btnNum 을 통해 창을 변경한다.
   iii. btnNum 이 0 인 경우

0. 음악 추가 버튼을 누른 경우다. 추가할 음악의 정보를 받는 창을 띄운다.
1. TextField, ComboBox, RadioButton 으로 정보를 입력하고, 확인 버튼을 눌러 입력받은 정보를 바탕으로 Management_Model.java 파일에 있는 insert(Date playtime, String title, int likes, String genre, boolean age_limit, Date release_date, String situation, String season, int music_age, String songwriter, String singer) 객체를 호출한다.
2. try-catch 문을 활용하여 특정 형식이 지켜지지 않거나, 정보에 맞지 않는 값이 들어오면 오류 메시지를 띄운다.

iv. btnNum 이 1 인 경우
0. 음악 수정하기를 누른 경우다. 수정할 곡을 특정하기 위해 가수와 제목을 입력 받는다.
1. Management_Moder.java 파일에 있는 findMusic(String singer, String title) 함수를 호출하여 입력한 곡의 music_id 값을 받아온다.
2. 만약 해당하는 곡이 없으면 해당하는 곡이 없다는 메시지 창을 띄운다.
3. 해당하는 곡을 찾으면 현재 창은 보이지 않게 하고 Management2 함수를 통해 새 panel 을 부착하도록 한다.이때 btnNum 은 3 으로 전달한다.

v. btnNum 이 2 인 경우
0. 음악 삭제하기를 누른 경우다. 수정할 곡을 특정하기 위해 가수와 제목을 입력 받는다.
1. model 객체를 통해 Management_Moder.java 파일에 있는 delete(String singer, String title) 함수를 호출하여 음악을 삭제한다.

vi. btnNum 이 3 인 경우
0. 음악 수정 버튼을 눌렀을 때 그 음악을 특정한 후에 띄워지는 창이다. 형식은 위와 동일하다.
1. TextField, ComboBox, RadioButton 으로 정보를 입력하고, 확인 버튼을 눌러 입력받은 정보를 통해 Management_Model.java 파일에 있는 modify(int music_id, Date playtime, String title, int likes, String genre, boolean age_limit, Date release_date, String situation, String season, int music_age, String songwriter, String singer) 객체를 호출한다.
2. 마찬가지로 try-catch 문을 활용하여 특정 형식이 지켜지지 않거나, 정보에 맞지 않는 값이 들어오면 오류 메시지를 띄운다.

c. Management_Model

i. MySQL Database 와 연결이 필요한 함수들을 모아둔 창이다.

ii. MySQL Database 와 connection 을 맺는다. 연결에 실패한 경우 fail 메시지가 뜬다.
iii. findMusic() 함수는 가수와 제목을 입력받아 해당 노래의 music_id 를 반환한다. 음악 수정하기, 음악 삭제하기 기능에서 활용된다.
iv. insert() 함수는 인자로 받은 정보를 PreparedStatement 를 통해 Database 에 insert 한다. 삽입에 성공하면 console 창에 성공적으로 추가됐다는 메시지를 띄운다. tuple 이 추가되는 table 은 db2022_music, db2022_songwriter, db2022_singer 가 있다.
v. delete() 함수는 인자로 받은 정보를 findMusic()함수로 넘겨 music_id 를 찾는다. 만약 해당곡을 찾지 못한다면 console 창에 해당하는 곡이 없다는 정보를 띄운다. PreparedStatement 를 통해 Database 에 있는 정보를 delete 한다. tuple 이 삭제되는 table 은 db2022_music, db2022_singer, db2022_songwriter, db2022_playlist_user 가 있다.
vi. modify() 함수는 인자로 받은 정보를 PreparedStatement 를 통해 Database 에 update 한다. 이미 music_id 를 받아왔기 때문에 findMusic()함수를 호출할 필요는 없다. 대상이 되는 table 은 db2022_music, db2022_songwriter, db2022_singer 가 있다. db2022_songwriter 와 db2022_singer table 에 경우는 여러명을 포함하고 있기 때문에 전부 삭제하고 새롭게 삽입하는 방식을 취한다.
vii. insert, delete, modify 함수 모두 정보의 왜곡이 없도록 transaction 을 사용했다.

(9) 마이페이지

: 아직 회원가입을 하지 않은 상태라면 SignPage 로, 회원가입을 완료한 상태라면 MyPage 로 이동한다.

a. USERINFO : 유저 정보 저장해두는 전역객체
   i. public static int user_id
   ii. public static String nickname
   iii. public static String name
   iv. public static int age
   v. public static String favorite_genre
b. SigninPage
   i. 사용자의 이름,닉네임,나이,선호장르 를 입력받는 4 개의 textfield 를 생성한다.
   ii. 정보 제출시 누를 JButton 을 하나 생성하고 Signin(String name,String nickname,int age,String favorite_genre) 생성자에 적합한 파라미터를 넣어서 호출시키는 ActionListener 를 부착한다.

iii. Signin 생성자 내부에서는 앞에서 들어온 파라미터를 활용해 insert 쿼리를 만들어 유저디비에 새로운 유저를 삽입한다.

c. MyPage

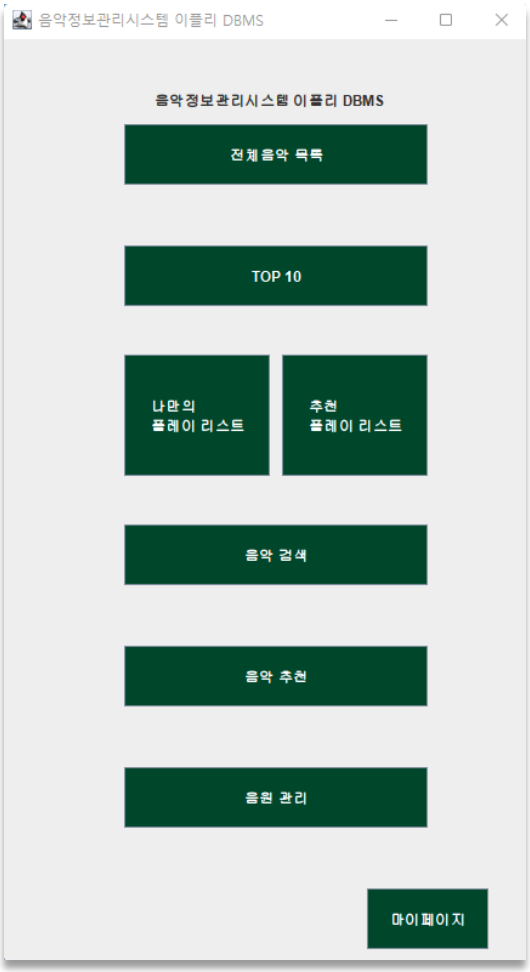
i. 유저 정보 저장해두는 전역객체 USERINFO 에 저장된 값을 활용해 현재 접속중인 유저의 정보를 화면에 출력시킨다.

ii. 선호장르 수정이 가능하도록 1 개의 textfield 를 생성하며, 제출 버튼에 UpdateGenre(String favorite_genre) 생성자를 호출시키는 ActionListener 를 부착한다.

iii. UpdateGenre 생성자 내부에서는 전역객체에 저장된 user_id 정보와 앞에서 들어온 파라미터를 활용해 Update 쿼리를 만들어 해당 유저의 선호장르 정보를 수정한다.

2) 실행화면 캡쳐

(1) 메인화면



(2) 전체음악목록

```
<전체 음원 목록>

1 | ZOOM | 00:02:54 | 제시 | bayb,bobblehead,싸이,염따,유건형,제시 | 34498 | 랩/힙합 | 2022-04-13
2 | MY BAG | 00:02:40 | (여자)아이들 | 네이슨,소연 | 51204 | 랩/힙합 | 2022-03-14
3 | WAP | 00:03:07 | Cardi B (feat. Megan Thee Stallion) | Austin Owens,,Belcalis Almanzar,Frank Rodriguez,James Foye III,Jorden Thorpe,Megan Pete | 28958 | 랩/힙합 | 2020-08-07
4 | Freaky Deaky | 00:03:35 | Doja Cat,Tyga | B Ham,Doja Cat & Tyga,Lourdiz,Mike Crook,Ryan OG | 3695 | 랩/힙합 | 2022-02-25
5 | MONTERO(Call Me By Your Name) | 00:02:17 | Lil Nas X | David Biral,Denzel Baptiste,Montero Hill,Omer Fedi,Rosario Lenzo | 23000 | 랩/힙합 | 2021-04-23
6 | abcdefu | 00:02:48 | GAYLE | David Pittenger,GAYLE,Sara Davis | 92667 | POP | 2021-08-13
7 | Let Me Down Slowly | 00:02:49 | Alec Benjamin | Alec Benjamin,Michael Pollack,Nolan Lambroza | 38428 | POP | 2018-11-16
8 | Santa Tell Me | 00:03:24 | Ariana Grande | Ariana Grande,Ilya,Savan Kotecha | 215428 | POP | null
9 | Peaches | 00:03:18 | Justin Bieber (Feat. Daniel Caesar & Giveon) | Ashton Simmonds,Felisha King),Giveon Evans,Justin Bieber,Louis Bell,Miles Ale | 213596 | POP | 2021-10-08
10 | Dance Monkey | 00:03:29 | Tones And I | Toni Watson | 169935 | POP | null
11 | 아모르 파티 | 00:03:38 | 김연자 | 신철,윤일상,이건우 | 72806 | 트로트 | 2013-07-23
12 | 이젠 나만 믿어요 | 00:04:02 | 임영웅 | 김이나,조영수 | 154066 | 트로트 | 2020-04-03
13 | 사랑의 배터리 | 00:03:26 | 홍진영 | 강은경,조영수 | 29114 | 트로트 | 2009-06-19
14 | 단발머리 | 00:03:48 | 조승필 | 박건호,조승필 | 7292 | 트로트 | 1980-03-20
15 | 트로트가 나는 좋아요 | 00:03:57 | 송가인 | 최고야,최비룡 | 8436 | 트로트 | 2020-12-26
16 | Hello Future | 00:03:40 | NCT DREAM | Adrian Mckinnon,KENZIE,Moonshine | 99591 | KPOP | 2021-06-28
17 | View | 00:03:12 | SHINee | Adrian Mckinnon,LDN Noise,Ryan S. Jhun,종현 | 136749 | KPOP | 2015-05-18
18 | LOVE DIVE | 00:02:57 | IVE | Elle Campbell,Nick Hahn,Sophia Brennan,서지음 | 118304 | KPOP | 2022-04-05
19 | Feel Special | 00:03:26 | TWICE | Hayley Aitken,J.Y.Park,Ollipop | 106520 | KPOP | 2019-09-23
```

```
<장르가 트로트인곡 목록>
id|타이틀|재생시간|장르|좋아요|발매일
11 | 아모르 파티 | 00:03:38 | 트로트 | 72806 | 2013-07-23
12 | 이젠 나만 믿어요 | 00:04:02 | 트로트 | 154066 | 2020-04-03
13 | 사랑의 배터리 | 00:03:26 | 트로트 | 29114 | 2009-06-19
14 | 단발머리 | 00:03:48 | 트로트 | 7292 | 1980-03-20
15 | 트로트가 나는 좋아요 | 00:03:57 | 트로트 | 8436 | 2020-12-26
```

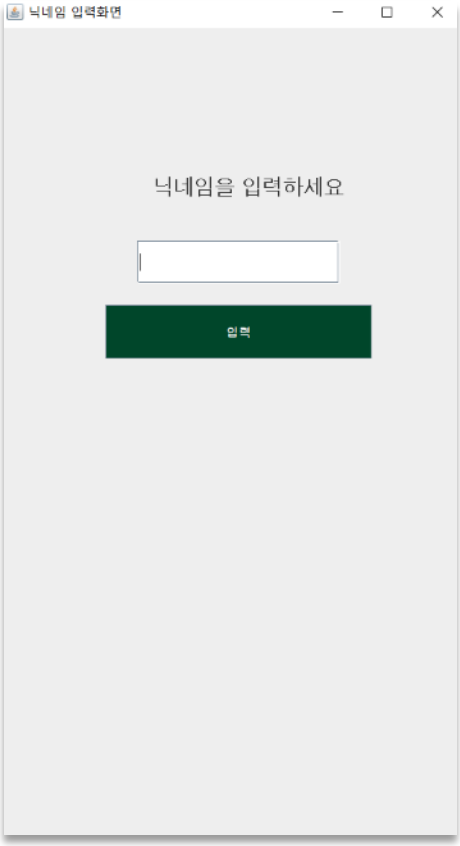
[출력 형식] 해당 장르의 곡 목록 콘솔창에 출력

(3) Top10

```
순위|제목|재생시간|가수|작사작곡가|likes|장르|발매일

1위: 어떻게 이별까지 사랑하겠어, 널 사랑하는 거지 | 00:04:50 | AKMU (악뮤) | 이찬혁 | 381043 | 발라드 | 2019-09-25
2위: Santa Tell Me | 00:03:24 | Ariana Grande | Ariana Grande,Ilya,Savan Kotecha | 215428 | POP | null
3위: Peaches | 00:03:18 | Justin Bieber (Feat. Daniel Caesar & Giveon) | Ashton Simmonds,Felisha King),Giveon Evans,Justin Bieber,Louis Bell,Miles Ale | 213596 | POP | 2021-10-08
4위: 스물다섯, 스물하나 | 00:04:45 | 자우림 | 김윤아 | 185117 | 록/메탈 | 2022-04-13
5위: Dance Monkey | 00:03:29 | Tones And I | Toni Watson | 169935 | POP | null
6위: 이젠 나만 믿어요 | 00:04:02 | 임영웅 | 김이나,조영수 | 154066 | 트로트 | 2020-04-03
7위: View | 00:03:12 | SHINee | Adrian Mckinnon,LDN Noise,Ryan S. Jhun,종현 | 136749 | KPOP | 2015-05-18
8위: LOVE DIVE | 00:02:57 | IVE | Elle Campbell,Nick Hahn,Sophia Brennan,서지음 | 118304 | KPOP | 2022-04-05
9위: 바람이 분다 | 00:03:56 | 이소라 | 이소라,이승환 | 109102 | 발라드 | 2004-12-10
10위: Feel Special | 00:03:26 | TWICE | Hayley Aitken,J.Y.Park,Ollipop | 106520 | KPOP | 2019-09-23
```

[출력 형식] 메인화면에서 TOP10 버튼 클릭 시 바로 콘솔창에 출력

(4) 나만의 플레이리스트



```
----------------------------

nickname: 하이
title: 집에
singer: 새소년

Successfully Add

----------------------------
```

나의 플레이리스트에 음악 추가 성공시

```
하이님의 플레이리스트 조회

1. 집에                    | 새소년
2. 스물다섯, 스물하나      | 자우림
```

내 플레이리스트 조회

[출력 형식] 닉네임 입력 후, 메뉴 선택 시 콘솔창에 수행결과 출력

(5) 추천 플레이리스트

```
**** [여름] 추천 플레이리스트 ****
-----------------------------------------------------------------------------------------------
    제목    |    가수    |    작사작곡    |    재생시간    |    좋아요 수    |    장르    |    발매일
-----------------------------------------------------------------------------------------------
ZOOM | 제시 | bayb,bobblehead,싸이,염따,유건형,제시 | 00:02:54 | 34498 | 랩/힙합 | 2022-04-13 |
MY BAG | (여자)아이들 | 네이슨,소연 | 00:02:40 | 51204 | 랩/힙합 | 2022-03-14 |
아모르 파티 | 김연자 | 신철,윤일상,이건우 | 00:03:38 | 72806 | 트로트 | 2013-07-23 |
사랑의 배터리 | 홍진영 | 강은경,조영수 | 00:03:26 | 29114 | 트로트 | 2009-06-19 |
트로트가 나는 좋아요 | 송가인 | 최고야,최비룡 | 00:03:57 | 8436 | 트로트 | 2020-12-26 |
Hello Future | NCT DREAM | Adrian Mckinnon,KENZIE,Moonshine | 00:03:40 | 99591 | KPOP | 2021-06-28 |
View | SHINee | Adrian Mckinnon,LDN Noise,Ryan S. Jhun,종현 | 00:03:12 | 136749 | KPOP | 2015-05-18 |
Hot Summer | f(x) | KENZIE,Mikkel Remee Sigvardt,Thomas Troelsen | 00:03:45 | 37872 | KPOP | 2011-06-14 |
```

```
**** [40대] 추천 플레이리스트 ****
-----------------------------------------------------------------------------------------------
    제목    |    가수    |    작사작곡    |    재생시간    |    좋아요 수    |    장르    |    발매일
-----------------------------------------------------------------------------------------------
사랑의 배터리 | 홍진영 | 강은경,조영수 | 00:03:26 | 29114 | 트로트 | 2009-06-19 |
스물다섯, 스물하나 | 자우림 | 김윤아 | 00:04:45 | 185117 | 록/메탈 | 2022-04-13 |
```

```
**** [출퇴근] 추천 플레이리스트 ****
-----------------------------------------------------------------------------------------------
    제목    |    가수    |    작사작곡    |    재생시간    |    좋아요 수    |    장르    |    발매일
-----------------------------------------------------------------------------------------------
MY BAG | (여자)아이들 | 네이슨,소연 | 00:02:40 | 51204 | 랩/힙합 | 2022-03-14 |
Dance Monkey | Tones And I | Toni Watson | 00:03:29 | 169935 | POP | null |
Perhaps Love (사랑인가요) | 제이,하울 | 김이나,박근철 | 00:04:35 | 73707 | 발라드 | 2006-01-26 |
비밀의 화원 | 이상은 | 이상은 | 00:04:11 | 26219 | 발라드 | 2003-03-10 |
충맞은것처럼 | 백지영 | 방시혁 | 00:03:59 | 40509 | 발라드 | 2008-11-13 |
집에 | 새소년 | 황소윤 | 00:03:48 | 10579 | 록/메탈 | 2019-10-04 |
```

[출력 형식] 해당 분류 기준에 맞는 플레이리스트 목록 콘솔창에 출력

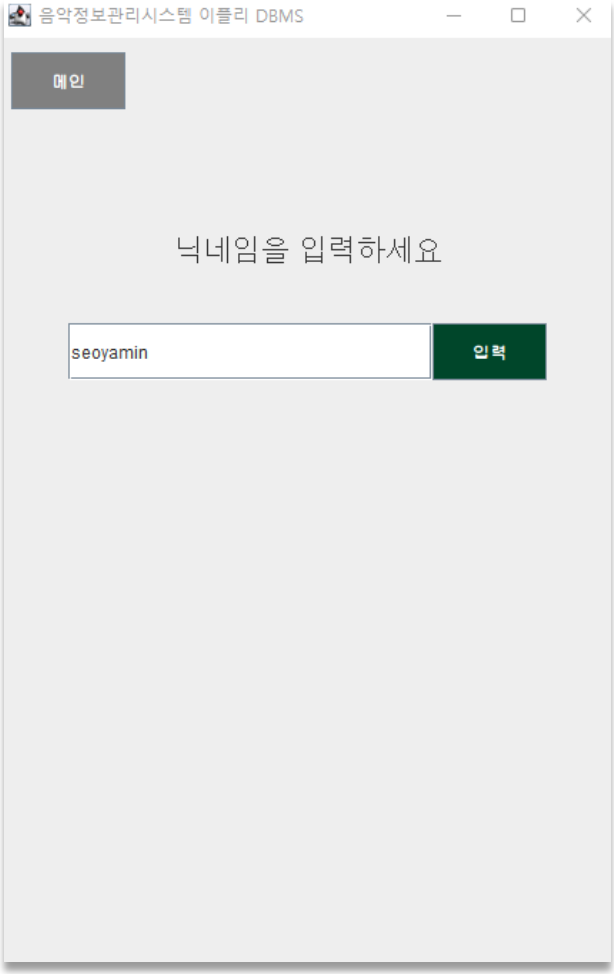
(6) 음악 검색

```
**** [김연자] 가수명 검색 결과 ****
-----------------------------------------------------------------------------------------------
    제목     |     가수     |     작사작곡     |     재생시간     |     좋아요 수     |     장르     |     발매일
-----------------------------------------------------------------------------------------------
아모르 파티 | 김연자 | 신철,윤일상,이건우 | 00:03:38 | 72806 | 트로트 | 2013-07-23 |
```

```
**** [아모르 파티] 제목 검색 결과 ****
-----------------------------------------------------------------------------------------------
    제목     |     가수     |     작사작곡     |     재생시간     |     좋아요 수     |     장르     |     발매일
-----------------------------------------------------------------------------------------------
아모르 파티 | 김연자 | 신철,윤일상,이건우 | 00:03:38 | 72806 | 트로트 | 2013-07-23 |
```

[출력 형식] 제목이나 가수명을 입력하고 '검색' 버튼 클릭 시 콘솔창에 음악 정보 출력

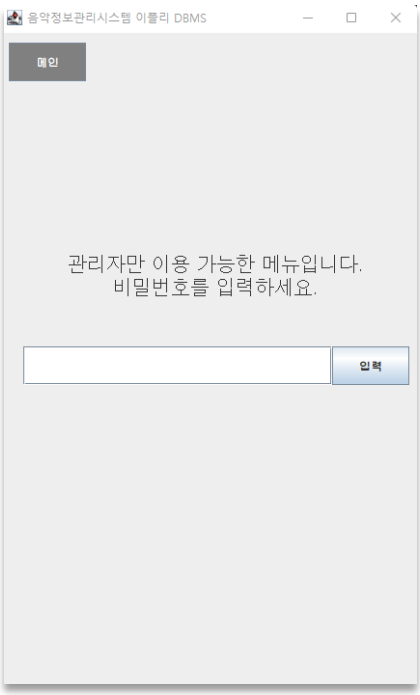
(7) 음악 추천

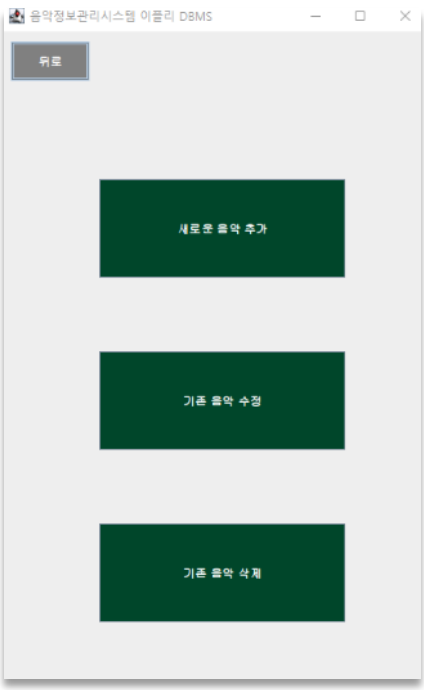


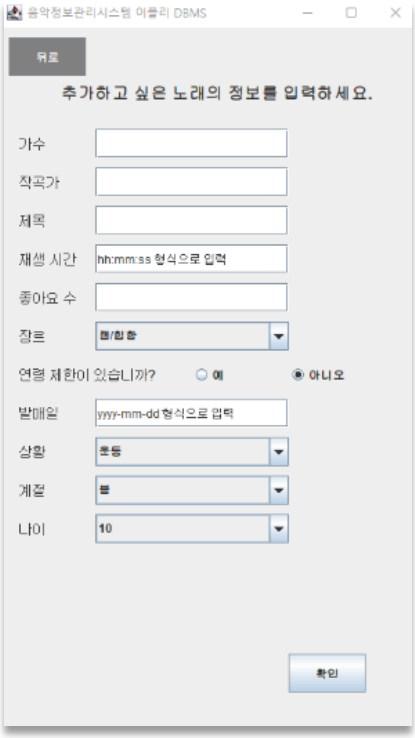

```
seoyamin님의 취향에 맞는 추천 음악 리스트 입니다.
-----------------------------------------------------------------------------------------------
    제목     |     가수     |     작사작곡     |     재생시간     |     좋아요 수     |     장르     |     발매일
-----------------------------------------------------------------------------------------------
Perhaps Love (사랑인가요) | 제이 하울 | 김이나 박근철 | 00:04:35 | 73707 | 발라드 | 2006-01-26
비밀의 화원 | 이상은 | 이상은 | 00:04:11 | 26219 | 발라드 | 2003-03-10
흙맞은것처럼 | 백지영 | 방시혁 | 00:03:59 | 40509 | 발라드 | 2008-11-13
바람이 분다 | 이소라 | 이소라 이승환 | 00:03:56 | 109102 | 발라드 | 2004-12-10
어떻게 이별까지 사랑하겠어, 널 사랑하는 거지 | AKMU (악뮤) | 이찬혁 | 00:04:50 | 381043 | 발라드 | 2019-09-25
```

[출력 형식] 닉네임을 입력하고 '검색' 버튼 클릭 시 콘솔창에 추천 음악 정보 출력

(8) 음원 관리







```
-------------------------------------------------------------

title: 테스트
singer: 테스트

Successfully Add

-------------------------------------------------------------
```

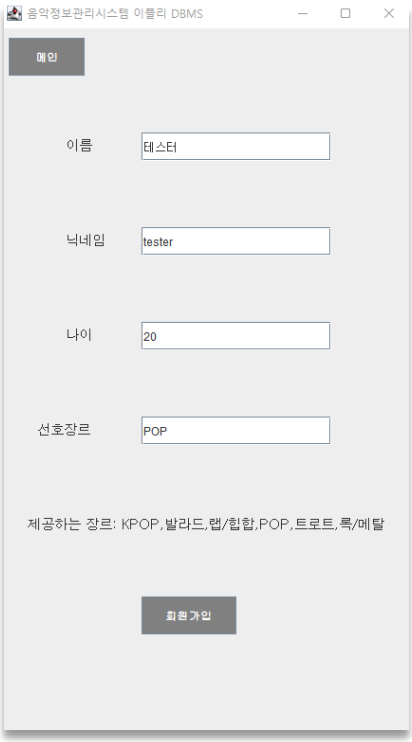
```
------------------------------------------------------------

Successfully UPDATE

------------------------------------------------------------
```

```
------------------------------------------------------------

Successfully DELETE

------------------------------------------------------------
```

[출력 형식] 새로운 곡을 추가 한 경우 제목, 가수와 함께 콘솔창에 성공 메시지 출력 / 기존 곡을 수정, 삭제한 경우 콘솔창에 성공 메시지 출력

(9) 마이페이지

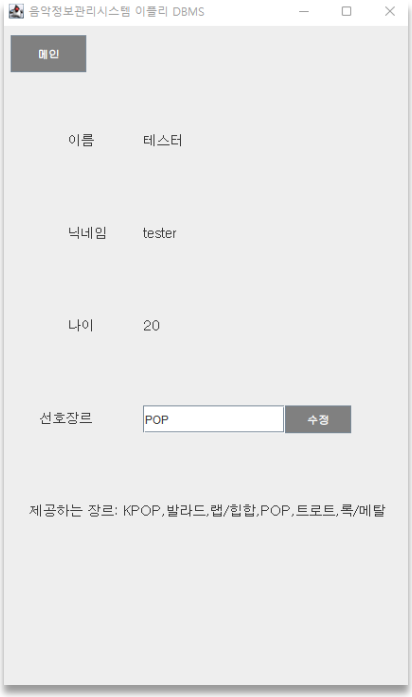
a. 회원가입




```
[회원가입 완료]tester유저가 등록되었습니다
```

[출력 형식] 유저 정보 입력 후 회원가입 버튼을 누르면 유저명과 함께 등록 메시지 콘솔창에 출력

b. 내정보조회




```
선호장르가 정상적으로 변경되었습니다.
```

[출력 형식] 선호 장르를 바꾸고 수정 버튼을 누르면 콘솔창에 성공 메시지 출력

5. 응용 프로그램 설치 조건

1) JDBC Driver 설치 및 연결

- MySQL Connector/J 8.0.29 설치 (https://dev.mysql.com/downloads/connector/j/)
- Eclipse 에서 프로젝트 선택 후 Project - Properties - Java Build Path - Libraries - Modulepath - Add External JARs... - mysql-connector-java-8.0.29.jar 선택

2) Database 구성

```
create database DB2022Team06 default character set utf8 collate utf8_unicode_ci;
create user DB2022Team06@localhost identified by 'DB2022Team06';
grant all privileges on DB2022Team06.* to DB2022Team06@localhost;
mysql -uDB2022Team06 -p DB2022Team06 < 로컬 저장 경로₩create.sql
```

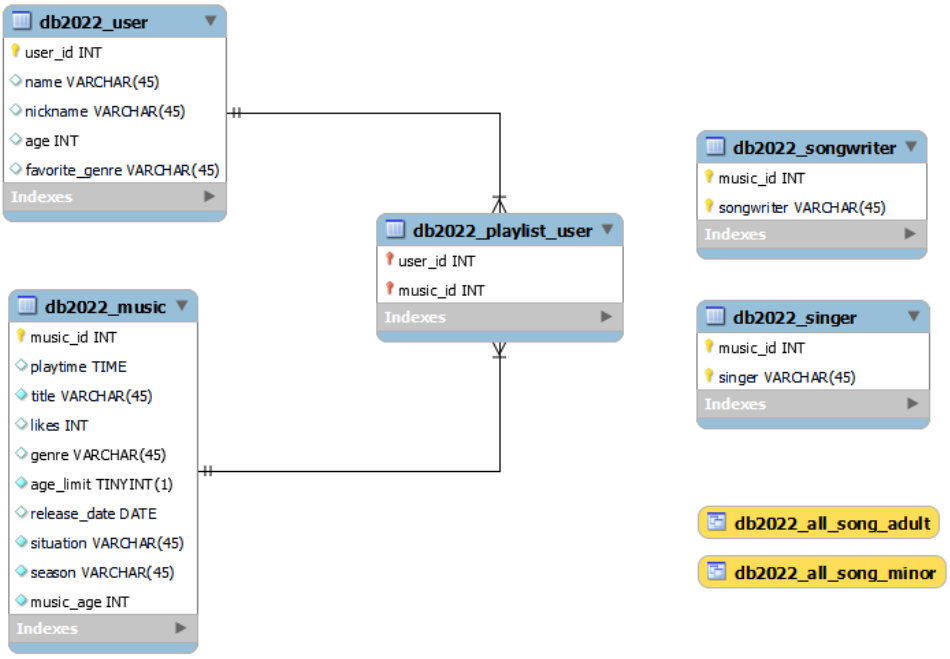
3) Java 프로젝트 실행

- open projects - DB2022Team06
- package DB2022Team06 – MainFrame.java 선택 후 run

6. 요구사항분석

1) 5개 이상의 테이블을 가지고 있어야 하고 각 테이블들의 컬럼(Attribute)의 수를 합하면 20개 이상이어야 한다.

: 총 5 개의 테이블 (db2022_user, db2022_playlist_user, db2022_music, db2022_singer, db2022_songwriter)을 지니고 있으며, 총 21 개의 컬럼(Attribute)을 지니고 있다.



2) 초기화를 위해 적어도 30개의 레코드(투플)를 가지고 있어야 한다. (모든 테이블들의 레코드 수의 총합이 30개 이상)

```sql
# db2022_music 테이블 튜플 삽입
INSERT INTO db2022_music(playtime,title,likes,genre,age_limit,release_date,situation,season,music_age) VALUES
("00:02:54","ZOOM",34498,"랩/힙합",false,"2022-04-13","운동","여름",10 ),
("00:02:40","MY BAG",51204,"랩/힙합",false,"2022-03-14","출퇴근","여름",10 ),
("00:03:07","WAP",28958,"랩/힙합",true,"2020-08-07","운동","겨울",30 ),
("00:03:35","Freaky Deaky",3695,"랩/힙합",true,"2022-02-25","공부","가을",20 ),
("00:02:17","MONTERO(Call Me By Your Name)",23000,"랩/힙합",true,"2021-04-23","여행","봄",20 ),
("00:02:48","abcdefu",92667,"POP",true,"2021-08-13","공부","가을",20 ),
("00:02:49","Let Me Down Slowly",38428,"POP",false,"2018-11-16","공부","봄",10 ),
("00:03:24","Santa Tell Me",215428,"POP",false,"2014-11-24","여행","겨울",30 ),
("00:03:18","Peaches",213596,"POP",true,"2021-10-08","여행","봄",20 ),
("00:03:29","Dance Monkey",169935,"POP",false,"2019-05-10","출퇴근","가을",30 ),
("00:03:38","아모르 파티",72806,"트로트",false,"2013-07-23","운동","여름",50 ),
("00:04:02","이젠 나만 믿어요",154066,"트로트",false,"2020-04-03","여행","겨울",50 ),
("00:03:26","사랑의 배터리",29114,"트로트",false,"2009-06-19","운동","여름",40 ),
("00:03:48","단발머리",7292,"트로트",false,"1980-03-20","여행","겨울",60 ),
("00:03:57","트로트가 나는 좋아요",8436,"트로트",false,"2020-12-26","운동","여름",60 ),
("00:03:40","Hello Future",99591,"KPOP",false,"2021-06-28","여행","여름",10 ),
("00:03:12","View",136749,"KPOP",false,"2015-05-18","운동","여름",20 ),
("00:02:57","LOVE DIVE",118304,"KPOP",false,"2022-04-05","공부","봄",10 ),
("00:03:26","Feel Special",106520,"KPOP",false,"2019-09-23","공부","가을",10 ),
("00:03:45","Hot Summer",37872,"KPOP",false,"2011-06-14","운동","여름",20 ),
("00:04:35","Perhaps Love (사랑인가요)",73707,"발라드",false,"2006-01-26","출퇴근","봄",30 ),
("00:04:11","비밀의 화원",26219,"발라드",false,"2003-03-10","출퇴근","봄",30 ),
("00:03:59","총맞은것처럼",40509,"발라드",false,"2008-11-13","출퇴근","봄",30 ),
("00:03:56","바람이 분다",109102,"발라드",false,"2004-12-10","공부","겨울",30 ),
("00:04:50","어떻게 이별까지 사랑하겠어, 널 사랑하는 거지",381043,"발라드",false,"2019-09-25","여행","겨울",30 ),
("00:03:48","비밀번호 486",57584,"록/메탈",false,"2007-03-15","여행","겨울",30 ),
("00:03:48","집에",10579,"록/메탈",false,"2019-10-04","출퇴근","가을",20 ),
("00:02:54","American Idiot",6010,"록/메탈",true,"2004-09-21","운동","봄",30 ),
("00:02:01","We Will Rock You",31157,"록/메탈",false,"1977-10-07","운동","가을",20 ),
("00:04:45","스물다섯, 스물하나",185117,"록/메탈",false,"2013-10-14","여행","가을",40 );
```

3) 기본 키(primary key), 외래 키(foreign key), not null 제약조건(not null constraints)를 포함해야 한다.

```sql
# create.sql 참조

create table db2022_user (
    user_id int AUTO_INCREMENT,
    name varchar(45),
    nickname varchar(45) unique,
    age int,
    favorite_genre varchar(45),
    primary key(user_id)
);

create table db2022_music (
    music_id int AUTO_INCREMENT,
    playtime Time,
    title varchar(45) not null,
    likes int,
    genre varchar(45),
    age_limit boolean not null,
    release_date Date,
    situation varchar(45) not null,
    season varchar(45) not null,
    music_age int not null,
    primary key(music_id),
    check (genre in ('랩/힙합','POP','KPOP','트로트','발라드','록/메탈','없음'))
);
```

- db2022_playlist_user 의 user_id 와 music_id 는 기본키이자 외래키의 역할을 한다.
- db2022_music 의 title, age_linit, situation, season, music_age 컬럼은 not null 속성을 지닌다.

4) 적어도 2개의 뷰를 정의해야 한다. (레포트에 view를 사용한 이유를 설명)

```
# 성인에게 노출되는 전곡 뷰 (연령제한 있는 곡 포함)
create view db2022_all_song_adult as
    select title,age_limit, GROUP_CONCAT(singer SEPARATOR ',') as singer
    from db2022_music natural join db2022_singer group by music_id;

# 미성년자에게 노출되는 전곡 뷰 (연령제한 있는 곡 제외)
create view db2022_all_song_minor as
   select title,singer from db2022_all_song_adult where age_limit=false;
```

: 이플리에는 성인 전용 view 인 db2022_all_song_adult 와 미성년자 전용 view 인 db2022_all_song_minor 총 두 개의 뷰가 존재한다.

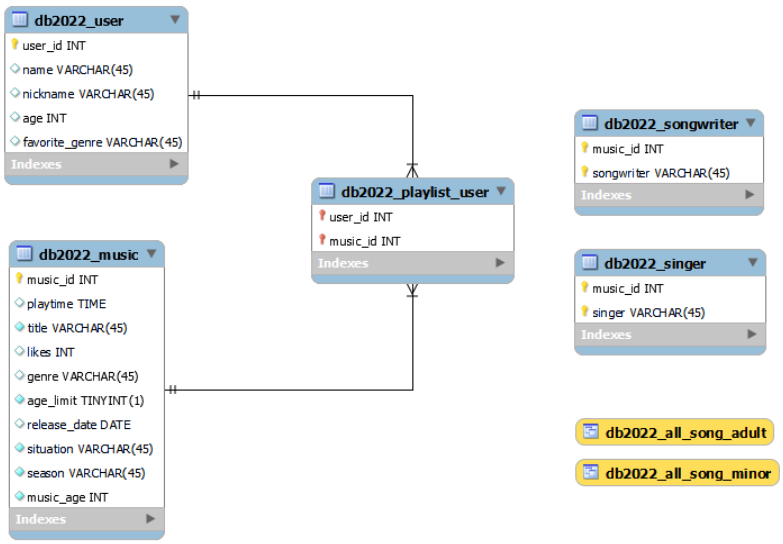
(1) db2022_music 테이블에는 사용자가 필요한 정보 이외에도 situation, season, music_age 등 관리자를 위해 특화된 정보들도 존재한다. 따라서 모든 데이터를 필요로 하지 않으므로, 특정 컬럼만 select 하는 db2022_all_song_adult view 를 통하여 유저가 필요한 정보(제목,가수)만을 보여줄 수 있도록 한다. 또한 서로 다른 테이블에 존재하는 title, singer 정보를 natural join 과 groub_concat 으로 묶어 하나의 테이블로 정의함으로써 음악목록 조회 시 매번 natural join 을 수행해야하는 불편함을 덜고 편의성을 높였다.

(2) db2022_all_song_minor 는 db2022_all_song_adult 에서 나이제한이 있는 곡들을 제외하고 접근할 수 있도록 하였다. 해당 뷰를 사용한 이유는 미성년자인 사용자의 특성에 따라 나이제한 (19 금)이 존재하는 곡들은 제외하고 조회할 수 있도록 하기 위해서이다.

5) 모든 테이블과 뷰의 이름들은 "DB2022_"라는 접두어를 가지고 있어야 한다.

예) DB2022_CONCERTINFO

```
# create.sql 참조

create table db2022_user (
    ...
);

create table db2022_playlist_user (
    ...
);

create table db2022_music (
    ...
);

create table db2022_singer (
    ...
);

create table db2022_songwriter (
    ...
);
```



6) 적어도 4개의 인덱스를 정의해야 한다.

```
# create.sql 참조

# 유저 정보가 담긴 db2022_user 테이블의 인덱스
create index index_music on db2022_user(user_id);

# 작사작곡가 정보가 담긴 db2022_songwriter 테이블의 인덱스
create index index_songwriter on db2022_songwriter(music_id, songwriter);

# 가수 정보가 담긴 db2022_singer 테이블의 인덱스
create index index_singer on db2022_singer(music_id, singer);

# 곡 정보가 담긴 db2022_music 테이블의 인덱스
create index index_music on db2022_music(music_id);
```

7) 인덱스를 사용하는 쿼리들을 포함해야 한다.

: music, singer, songwriter, user 테이블에서 select, update, delete를 수행하는 쿼리가 모두 해당된다.

- Sign 클래스의 생성자함수

```
//user_id 가져오기
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery(
        "select user_id from db2022_user where nickname=\""+nickname+"\"");

int user_id=-1;
while(rset.next()) {
    user_id=rset.getInt("user_id");
}

USERINFO.user_id=user_id;
```

- Music 리스트의 생성자 함수

```
if(genre=="전체") {
      rset = stmt.executeQuery(
             "with TMP AS( \r\n"
                     + "select music_id,title,playtime,likes,genre,age_limit,release_date, GROUP_CONCAT(songwriter SEPARATOR ',') as songwriter
                     + "from db2022_music natural join db2022_songwriter group by music_id) \r\n"
                     + "SELECT music_id,title,playtime,songwriter,likes,genre,age_limit,release_date,GROUP_CONCAT(singer SEPARATOR ',')\r\n"
                     + "from TMP natural join db2022_singer group by music_id "
       );
      System.out.println("<전체 음원 목록>\n");

       while (rset.next()) {

           System.out.println(rset.getString("music_id")+" | "+rset.getString("title")+" | "+rset.getTime("playtime")+" | "+rset.getString("GROUP
       }
}
```

8) 뷰를 사용하는 쿼리를 포함해야 한다.

: 추천 플레이리스트에서 전체 음원 목록을 조회할 때 playlist_main의 Total_musicPlaylist() 메소드에서 db2022_all_song_adult와 db2022_all_song_minor 뷰를 사용했다.

```java
/* 전체 음악 목록 sql*/
public void Total_musicPlaylist(String nickname) {
    try {
        int i=1;
            Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            //stmt.executeQuery("use DB2022Team06");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("select age from db2022_user where nickname="+"'"+nickname+"'");
            Statement stmt1 = conn.createStatement();
            ResultSet rs1 = stmt1.executeQuery("select * from db2022_all_song_adult");
            Statement stmt2 = conn.createStatement();
            ResultSet rs2 = stmt2.executeQuery("select * from db2022_all_song_minor");

            int user_age=0;
            System.out.println("_____________________________");
            System.out.println("전체 음악 목록 조회");
            System.out.println("_____________________________");

            while(rs.next()) {
                user_age=rs.getInt(1);
                if(user_age<=19) { //미성년자용 전체음악목록 조회
                    while(rs2.next()) {
                        System.out.printf("%d. %-20s| %-20s\n", i, rs2.getString(1), rs2.getString(2));
                        i++; }
                    }
                else{
                    while(rs1.next()) {
                        System.out.printf("%d. %-20s| %-20s\n", i, rs1.getString(1), rs1.getString(2));
                        i++; }
                    }
                }
            }
```

9) 트랜잭션(transaction)을 포함해야 한다.

: 음원 관리 탭(Management_Model)에서 관리자가 음원을 삽입, 수정, 삭제 처리 할 수 있도록 할 때 데이터베이스의 정보가 왜곡되지 않도록 트랜잭션을 사용하였다.

```java
/*정보를 받아 DB에 insert 하는 함수*/
public void insert(Date playtime, String title, int likes, String genre, boolean age_limit, Date release_date, String situation, String seas
    String query1 = "insert into db2022_music values(?,?,?,?,?,?,?,?,?,?)";
    String query2 = "insert into db2022_songwriter values(?,?)";
    String query3 = "insert into db2022_singer values(?,?)";

    //가수와 작곡가, 음악 정보를 따로 저장해두므로 세가지 쿼리를 작성했다.

    Connection conn = null;

    try {

        conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        Statement stmt = conn.createStatement();
        PreparedStatement pStmt1 = conn.prepareStatement(query1, Statement.RETURN_GENERATED_KEYS);
        PreparedStatement pStmt2 = conn.prepareStatement(query2);
        PreparedStatement pStmt3 = conn.prepareStatement(query3);

        /*트랜잭션*/
        conn.setAutoCommit(false);
        /*입력받은 곡의 정보를 insert문에 넣기*/
        pStmt1.setInt(1,0);
        pStmt1.setTimestamp(2, new java.sql.Timestamp(playtime.getTime()));
        pStmt1.setString(3, title);
        pStmt1.setInt(4, likes);
        pStmt1.setString(5, genre);
        pStmt1.setBoolean(6, age_limit);
        pStmt1.setDate(7, new java.sql.Date(release_date.getTime()));
        pStmt1.setString(8, situation);
        pStmt1.setString(9, season);
        pStmt1.setInt(10, music_age);
        pStmt1.executeUpdate();
```

```java
    //새로 입력한 id 구하기
    int autoIncrement = 0;
    ResultSet rs = pStmt1.getGeneratedKeys();
    if (rs.next()) {
        autoIncrement = rs.getInt(1);
    }

    /*그 id로 가수와 제목을 저장*/
    pStmt2.setInt(1, autoIncrement);
    pStmt2.setString(2, songwriter);
    pStmt2.executeUpdate();

    pStmt3.setInt(1, autoIncrement);
    pStmt3.setString(2, singer);
    pStmt3.executeUpdate();

    conn.commit();
    conn.setAutoCommit(true);

    System.out.println("----------------------------------------------------------------\n");
    System.out.println("title: " + title);
    System.out.println("singer: " + singer);
    System.out.println("\nSuccessfully Add\n");
    System.out.println("----------------------------------------------------------------\n");
}
catch (SQLException sqle) {
    if(conn!=null)
        try {
            conn.rollback();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            System.out.println("fail");
        }
}
```

10) 중첩된 쿼리(nested query)들을 가지는 쿼리들을 포함해야 한다.

: 음악 추천 탭에서 사용자의 선호 장르와 사용자가 마이 플레이리스트에 넣은 곡의 장르를 찾을 때 subquery를 사용하였다.

```java
public class Recommend_Model {

    private static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    private static final String USERNAME = "DB2022Team06"; //DBMS 접속 시 아이디
    private static final String PASSWORD = "DB2022Team06"; //DBMS 접속 시 비밀번호
    private static final String URL = "jdbc:mysql://localhost:3306/DB2022Team06"; //DBMS 접속할 DB

    public void recommend(String nickname) {
        //유저의 선호 장르와 나만의 플레이리스트에 담은 곡의 장르 전체를 fav_genre(genre)에 저장하고 이를 이용해 music 테이블에서 해당하는 장르의 곡 정보를 모두 가져오는 쿼리
        String query1 = "with fav_genre(genre) as ((select distinct favorite_genre as genre from db2022_user where nickname = ?) "
                + "union (select distinct genre from db2022_playlist_user natural join db2022_user natural join db2022_music "
                + "where db2022_music.music_id = db2022_playlist_user.music_id and db2022_playlist_user.user_id = db2022_user.user_id and nickname = ?)) "
                + "select * from db2022_music where genre in (select * from fav_genre)";
        String query2 = "select singer from db2022_singer where music_id = ?"; //추천 목록에 들어간 곡의 가수 정보를 가져오기 위한 쿼리
        String query3 = "select songwriter from db2022_songwriter where music_id = ?"; //추천 목록에 들어간 곡의 작사작곡 정보를 가져오기 위한 쿼리
        String query4 = "select age from db2022_user where nickname = ?"; //유저의 연령을 확인하기 위한 쿼리

        try (
            Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            Statement stmt = conn.createStatement();
            PreparedStatement pStmt1 = conn.prepareStatement(query1);
            PreparedStatement pStmt2 = conn.prepareStatement(query2);
            PreparedStatement pStmt3 = conn.prepareStatement(query3);
            PreparedStatement pStmt4 = conn.prepareStatement(query4);
            )
        {
            //입력된 nickname 쿼리에 넣어 쿼리 실행하기
            pStmt1.setString(1, nickname);
            pStmt1.setString(2, nickname);
            pStmt4.setString(1, nickname);
            ResultSet rs1 = pStmt1.executeQuery(); //추천할 곡 목록 및 정보 가져오기
            ResultSet rs4 = pStmt4.executeQuery(); //유저의 나이 정보 가져오기
```

11) 조인 쿼리 (join query)들을 가지는 쿼리들은 포함해야 한다.

: 음악 추천을 위해 사용자가 나만의 플레이 리스트에 담은 곡의 장르 정보를 가져오기 위해서

user 테이블, playlist 테이블, music 테이블을 natural join 하였다.

```java
public class Recommend_Model {

    private static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    private static final String USERNAME = "DB2022Team06"; //DBMS 접속 시 아이디
    private static final String PASSWORD = "DB2022Team06"; //DBMS 접속 시 비밀번호
    private static final String URL = "jdbc:mysql://localhost:3306/DB2022Team06"; //DBMS 접속할 DB

    public void recommend(String nickname) {
        //유저의 선호 장르와 나만의 플레이리스트에 담은 곡의 장르 전체를 fav_genre(genre)에 저장하고 이를 이용해 music 테이블에서 해당하는 장르의 곡 정보를 모두 가져오는 쿼리
        String query1 = "with fav_genre(genre) as ((select distinct favorite_genre as genre from db2022_user where nickname = ?) "
                + "union (select distinct genre from db2022_playlist_user natural join db2022_user natural join db2022_music where db2022_music.m
                + "select * from db2022_music where genre in (select * from fav_genre)";

        ...
    }
}
```

12) 매개 변수를 가지면서 동적으로 만드는 쿼리를 포함해야 한다. 다시 말해, 사용자로부터 입력 값을 받고 사용자가 입력한 값으로 쿼리를 생성한다.

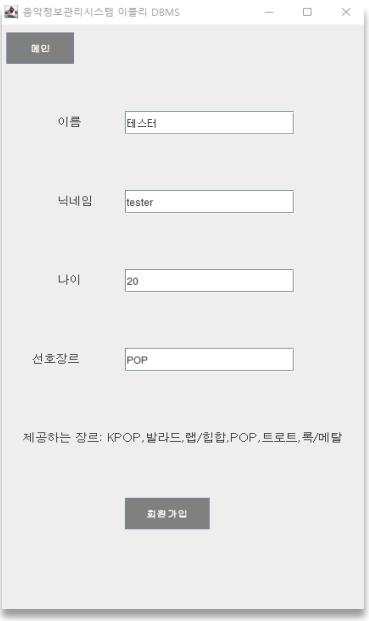
```java
for(int i=0;i<btnList.length;i++) {
    // 입력 텍스트 필드
    JTextField tf = new JTextField(20);
    tf.setLocation(150, OFFSET+i*100+10);
    tf.setSize(200, 30);
    mypagePannel.add(tf);
    textfieldList[i]=tf;

}

Font font = new Font("돋움", Font.PLAIN, 15);
JLabel la = new JLabel("제공하는 장르: KPOP,발라드,랩/힙합,POP,트로트,록/메탈");
la.setFont(font);
la.setLocation(30,500);
la.setSize(450,50);
mypagePannel.add(la);

JButton inputBtn = new JButton("회원가입");
inputBtn.setLocation(150, 600);
inputBtn.setSize(100, 40);
inputBtn.setForeground(Color.WHITE);
inputBtn.setBackground(Color.GRAY);
mypagePannel.add(inputBtn);

inputBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //데이터 입력에 따라 처리

        new Signin(textfieldList[0].getText(),textfieldList[1].getText()
                ,Integer.parseInt(textfieldList[2].getText()),textfieldList[3].getText());

    }
});
```

: (SigninPage 클래스 내부) 마이페이지에서 회원가입(유저등록)을 처리할 때 유저의 이름, 별명, 나이, 선호장르 정보를 사용자가 동적으로 입력한다. 이후 사용자가 입력한 값을 Signin 생성자의 파라미터로 전달한다.

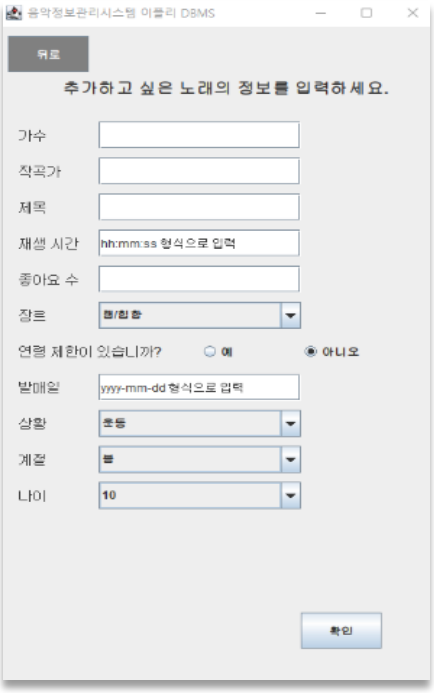
```java
public Signin(String name,String nickname,int age,String favorite_genre) {


    List<String> list = new ArrayList<>(Arrays.asList("KPOP","발라드","랩/힙합","POP","트로트","록/메탈"));

    if(!list.contains(favorite_genre)) { //제공하는 장르에 없다면
        System.out.println("유효한 장르를 입력해주세요 :(");
        return;
    }

    try {
        Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
       // System.out.println("connection success");

        try {

        PreparedStatement pStmt = conn.prepareStatement(
                "INSERT INTO db2022_user(name,nickname,age,favorite_genre) VALUES(?,?,?,?)");

        pStmt.setString(1, name);
        pStmt.setString(2, nickname);
        pStmt.setInt(3, age);
        pStmt.setString(4, favorite_genre);
        pStmt.executeUpdate();
        pStmt.close();
```

: Signin 생성자 내부에서 전달받은 유저 정보를 바탕으로 PreparedStatement를 활용해 동적으로 쿼리를 생성하고 실행한다.

13) 그래픽 또는 문자 기반의 사용자 인터페이스를 사용해야 한다. 사용하기 쉽고 사용하기에 도움이 되는 정보를 가지고 있는 메뉴를 제공해야 한다.



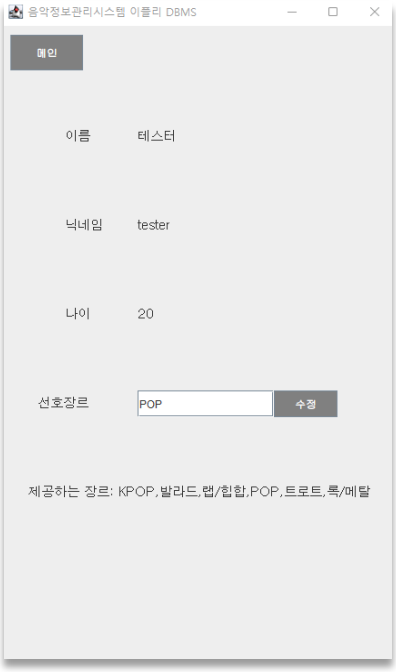
14) 데이터베이스에 삽입(insert)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.




```java
public void insert(String nickname, String title, String singer) {
    String query="insert into db2022_playlist_user(user_id, music_id) "
            + "values((select user_id from db2022_user where nickname=?), "
            + "(select T.music_id from db2022_music as T,db2022_singer as S where (T.title=?) and (S.singer=?)))";
    int age=0,bool=0;

    try {
        System.out.println("-----------------------------------------------------------------\n");
        System.out.println("nickname: "+nickname);
        System.out.println("title: "+title);
        System.out.println("singer: "+singer+"\n");

        Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        Statement stmt1 = conn.createStatement();
        Statement stmt2 = conn.createStatement();
        //stmt.executeQuery("use DB2022Team06");
        ResultSet rs1 = stmt1.executeQuery("select age from db2022_user where nickname="+"'"+nickname+"'");
        ResultSet rs = stmt2.executeQuery("select age_limit from db2022_music, db2022_singer "
                + "where (db2022_music.title="+"'"+title+"')"+"and (db2022_singer.singer="+"'"+singer+"')");

        conn.setAutoCommit(false);

        while(rs1.next()) {

            age=rs1.getInt(1);
            while(rs.next()) {
            bool=rs.getInt(1);

            if(bool==1 && age<=19) {
                System.out.print("denied: Age Limit");
                }
                else {
                    PreparedStatement pStmt = conn.prepareStatement(query);

                    pStmt.setString(1, nickname);
                    pStmt.setString(2, title);
                    pStmt.setString(3, singer);
                    pStmt.executeUpdate();

                    System.out.println("Successfully Add\n");
                    System.out.println("-----------------------------------------------------------------\n");
                    conn.commit();
                }
            }
        }
    }
```

: 음악 관리 탭에서 음원을 새로 추가하는 부분 (Management_Model 클래스의 insert 함수)

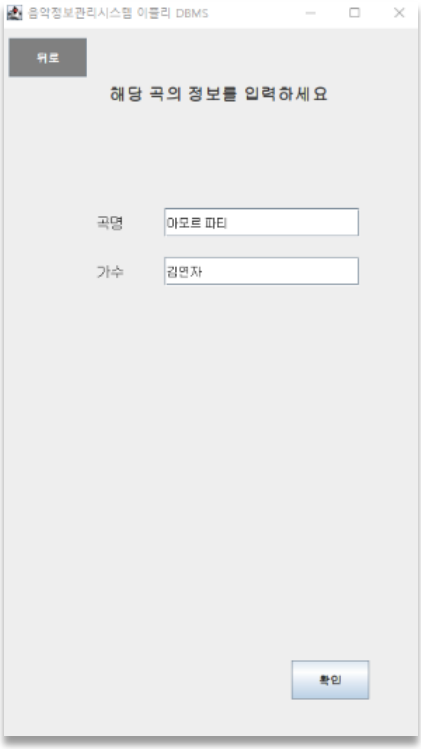
15) 데이터베이스에 갱신(update)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.




```java
public UpdateGenre(String favorite_genre) {


    List<String> list = new ArrayList<>(Arrays.asList("KPOP","발라드","랩/힙합","POP","트로트","록/메탈"));

    if(!list.contains(favorite_genre)) { //제공하는 장르에 없다면
        System.out.println("유효한 장르를 입력해주세요 :(");
        return;
    }

    try {
        Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
       // System.out.println("connection success");

        try {

        PreparedStatement pStmt = conn.prepareStatement(
                "UPDATE db2022_user SET favorite_genre=? where user_id=?;");

        pStmt.setString(1,favorite_genre );
        pStmt.setInt(2,USERINFO.user_id);

        pStmt.executeUpdate();

        USERINFO.favorite_genre=favorite_genre;
        pStmt.close();

        System.out.println("선호장르가 정상적으로 변경되었습니다." );
        }

        catch (SQLException sqle)
        {
        System.out.println("선호장르를 수정할 수 없습니다. (이유 : id존재하지 않음 등)" + sqle);
        }

        System.out.println("\n");
        conn.close();

    } catch (SQLException e1) {
        System.out.println("connection fail");
    }
}
```

: 마이페이지에서 선호장르를 수정(Update) 하는 부분

16) 데이터베이스에 삭제(delete)를 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.




```java
public class music_Info_delete extends JFrame {

    ...

    public void delete(String nickname, String title, String singer) {
        String query="delete from db2022_playlist_user where user_id=(select user_id from db2022_user where nickname=?) and"
                + " music_id=(select T.music_id from db2022_music as T,db2022_singer as S where (T.title=?) and (S.singer=?))";
        int age=0,bool=0;

        try {
            System.out.println("-------------------------------------------------------------\n");
            System.out.println("nickname: "+nickname);
            System.out.println("title: "+title);
            System.out.println("singer: "+singer+"\n");

            Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            Statement stmt1 = conn.createStatement();
            Statement stmt2 = conn.createStatement();
            //stmt.executeQuery("use DB2022Team06");
            ResultSet rs1 = stmt1.executeQuery("select age from db2022_user where nickname="+"'"+nickname+"'");
            ResultSet rs = stmt2.executeQuery("select age_limit from db2022_music, db2022_singer "
                    + "where (db2022_music.title="+"'"+title+"')"+"and (db2022_singer.singer="+"'"+singer+"')");

            conn.setAutoCommit(false);

            while(rs1.next()) {

                age=rs1.getInt(1);

                while(rs.next()) {

                bool=rs.getInt(1);

                if(bool==1 && age<=19) {
                    System.out.print("denied: Age Limit");
                    }
```

```
                    else {
                        PreparedStatement pStmt = conn.prepareStatement(query);

                        pStmt.setString(1, nickname);
                        pStmt.setString(2, title);
                        pStmt.setString(3, singer);
                        pStmt.executeUpdate();

                        System.out.println("Successfully DELETE\n");
                        System.out.println("-------------------------------------------------------------\n");
                        conn.commit();
                    }
                }
            }
        }

        catch (SQLException e1) {
            e1.printStackTrace();
            System.out.print("error");
        }

        }
```

17) 데이터베이스에 검색(select)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.

```
else {
    rset = stmt.executeQuery(
            "select * from db2022_music where genre=\""+genre+"\""
        );
    System.out.println("<장르가 "+genre+"인곡 목록>");
    System.out.println("id|타이틀|재생시간|장르|좋아요|발매일");
    while (rset.next()) {

        System.out.println(rset.getInt(1)+" | "+rset.getString("title")+" | "+rset.getTime("playtime")+
                " | "+rset.getString("genre")+" | "+rset.getInt("likes")+" | "+rset.getDate("release_date"));
        }

    }

 System.out.println("\n");
```

7. 팀만의 강점

1) 사용하기 쉬운 그래픽 기반 GUI를 가지고 있다.

: 이플리 DBMS 시스템은 자바 스윙의 여러가지 컴포넌트(JComboBox, JTextField,JRadioButton, JButton, JLabel) 들을 활용해 보다 사용자 친화적인 환경을 구성하고, 편리하게 값을 입력할 수 있도록 하였다.

2) 음악 추천 기능

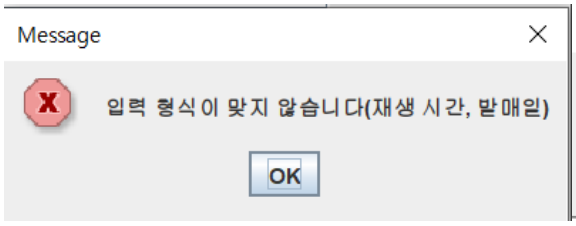
: 이플리에서는 사용자 개인 정보 (선호장르, 나이, 나만의 플레이리스트)를 기반으로 적합한 곡을 추천해주는 개인 맞춤형 서비스인 [음악 추천] 기능을 제공하고 있다. 이플리는 회원가입시 유저가 선택한 선호장르와 나만의 플레이리스트에 담겨있는 장르 등을 총체적으로 고려하여 추천할 음악들을 선별한다. 또한, 미성년자 유저에게는 연령 제한이 있는 곡을 추천하지 않는 기능을 포함하고 있다.

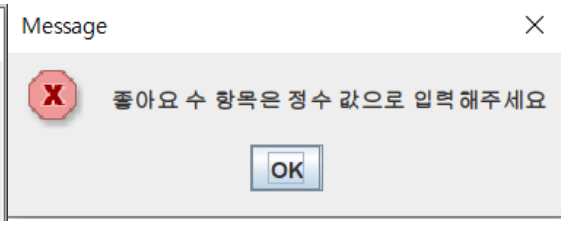
3) 유동적인 정보 수정 및 반영

: 이플리에서는 실시간 정보 수정 및 반영이 가능하다. 예를 들어 유저가 회원가입 후 마이페이지에서 선호장르를 수정하거나 나만의 플레이리스트에 담은 음악목록이 변화가 생기면, 이에 맞추어 추천 받을 수 있는 음악도 달라지고 또한 관리자가 새로운 음원을 추가함에 따라서 Top10 차트도 같이 변화한다.

4) 심플한 데이터베이스 관계 구조

: 이플리 DBMS 시스템은 개발 단계에서 발생할 수 있는 여러 이상현상들 (갱신이상, 삽입이상, 삭제이상)을 해결해내는 정규화 과정을 거쳤다. 이를 통해 불필요한 데이터 중복을 모두 제거하였고 참조관계가 깔끔하여 삽입, 삭제, 수정 요청 시 참조 무결성 조건에 위배되는 일이 드물기 때문에 사용이 쉽고 관리가 용이하다. 또한 사용자가 동적으로 입력한 값으로 쿼리를 생성할 때, 사용자가 적절한 입력 값을 주지 않았을 경우를 모두 고려하는 등의 예외처리도 잘 되어있어 테이블에 지정된 자료형 외에 다른 값이 들어가는 예외의 수를 사전에 차단한다.





8. 담당 파트

팀원이름	SQL	Java code	Report	발표	기타
김민서 2017008	Index 스크립트 작성	음악검색, 추천 플레이리스트	레포트 4번 문항 (자신이 담당한 기능파트 설명 작성), 실행화면 캡쳐 1	PPT 제작 및 레이아웃 짜기 1	jdbc 환경설정
김서연 2071008	Insert문 작성, Create.sql 작성 1	메인 화면, 전체음원목록, Top10,마이페이지	4번 문항 외 모두		깃허브 레포지토리 생성, 스키마 다이어그램 작성
김현민 2071020	Create.sql 작성 2	음원 관리	레포트 4번 문항 (자신이 담당한 기능파트 설명 작성), 실행화면 캡쳐 2	PPT 제작 및 레이아웃 짜기 2	노래 데이터 수집 1
안채연 2076232	View 생성	나만의 플레이리스트	레포트 4번 문항 (자신이 담당한 기능파트 설명 작성)	대본 작성 및 데모 비디오 녹화	ER 다이어그램 작성
최한비 2076429	Create.sql 주석 작성	음악 추천	레포트 4번 문항 (자신이 담당한 기능파트 설명 작성), 실행화면 캡쳐 3	발표 중 시연	노래 데이터 수집2, ReadMe 작성