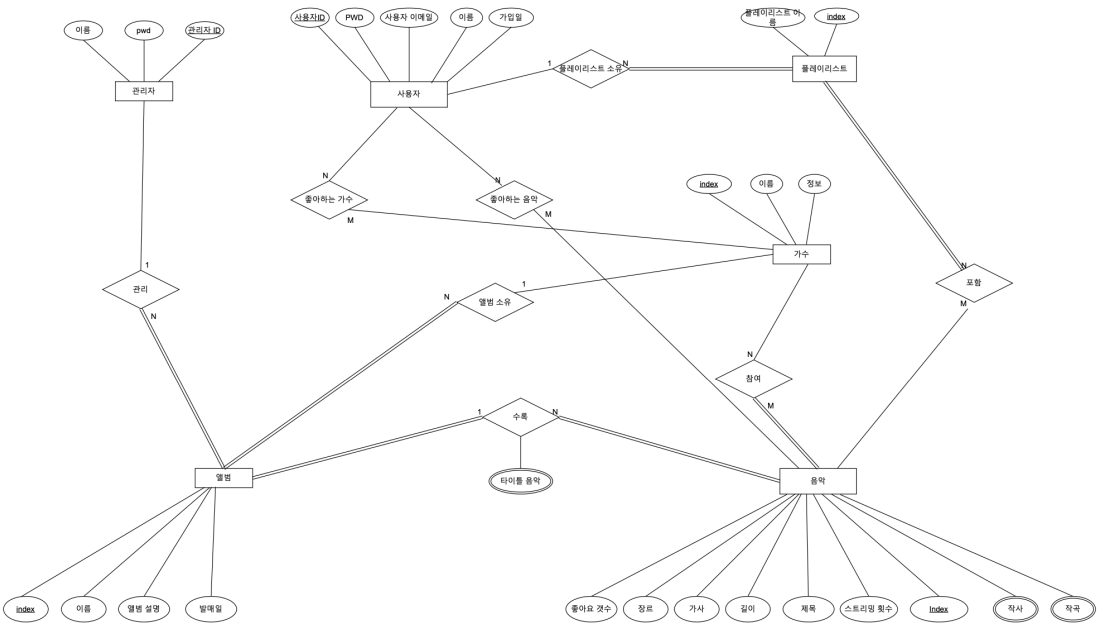
Project report

태그

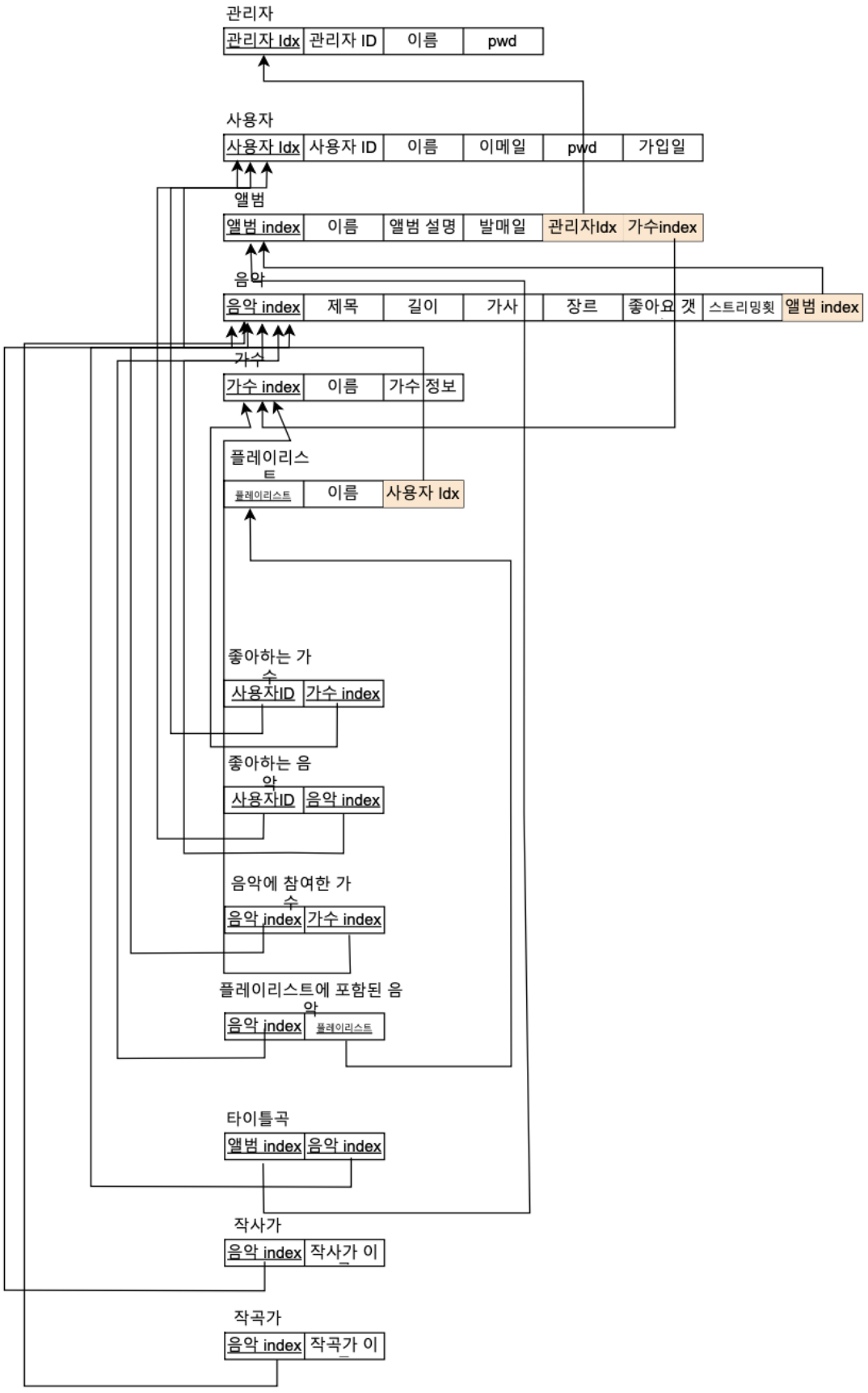
목표

- 요구사항을 기반으로 하는 어플리케이션 프로그램 개발
- 요구사항을 바탕으로 ER diagram 과 relational diagram 설계
- SQL DDL 을 이용하여 데이터베이스 스키마를 생성
- Python 을 이용하여 DBMS 응용 프로그램 개발

ER Diagram



Relational diagram


관리자
관리자 Idx
관리자 ID
이름
pwd
사용자
사용자 Idx
사용자 ID
이름
이메일
pwd
가입일
앨범
앨범 index
이름
앨범 설명
발매일
관리자Idx
가수index
음악
음악 index
제목
길이
가사
장르
좋아요 갯
스트리밍횟
앨범 index
가수
가수 index
이름
가수 정보
플레이리스트
플레이리스트
이름
사용자 Idx
좋아하는 가수
사용자ID
가수 index
좋아하는 음악
사용자ID
음악 index
음악에 참여한 가수
음악 index
가수 index
플레이리스트에 포함된 음악
음악 index
플레이리스트
타이틀곡
앨범 index
음악 index
작사가
음악 index
작사가 이
작곡가
음악 index
작곡가 이

변경 사항

- 앨범 : 앨범 유형, 앨범 장르, 자켓 사진 삭제

  → 음악에 장르가 이미 존재하기 때문에 앨범 장르는 삭제함

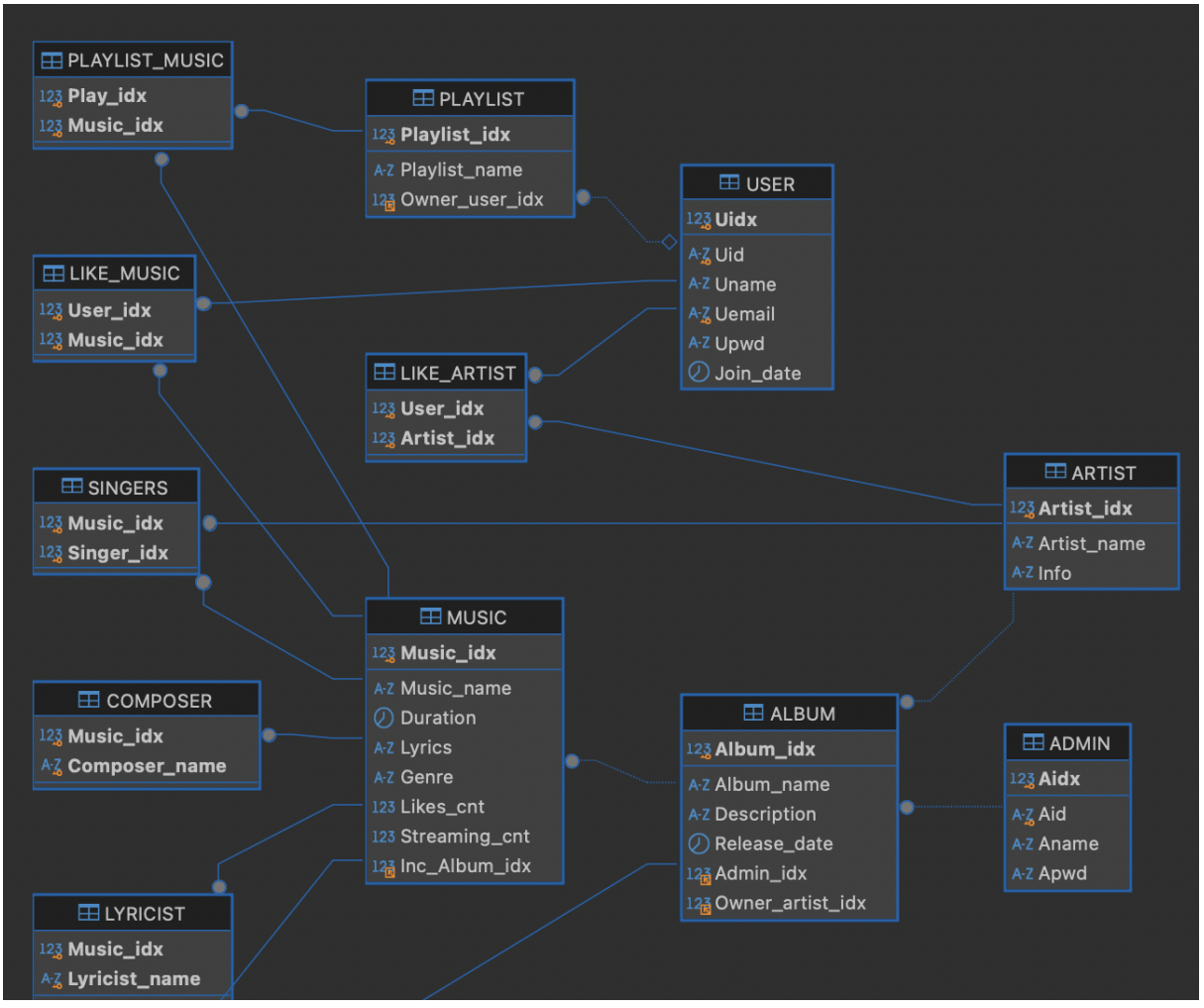
- 가수 : 프로필 사진 삭제

  → cli 프로그램 특성 상 이미지 삭제

- 관리자 : 관리자 Idx 추가
- 사용자 : 사용자 Idx 추가

  → 고정 길이 (혹은 짧은) Int 로 사용하기 위해 Idx 추가 (Id 는 사용자에게 입력받으므로 고정 길이가 아님)

Schema



필수 기능 목록

💡 **요구사항**

- 관리자는 서비스에 음악을 등록할 수 있다.
- 사용자는 서비스에 가입하여 음악을 들을 수 있다.
- 사용자는 좋아하는 음악에 표시를 해둘 수 있다.
- 사용자는 본인의 다양한 플레이리스트를 만들 수 있으며, 공유 가능하다.

💡 기능

관리자

1. 관리자 회원가입 - id 중복 검사
2. 관리자 로그인
3. 가수 등록
4. 앨범 및 음악 등록
5. 앨범 삭제
6. 음악 삭제
7. 관리 중인 앨범 확인

사용자

1. 사용자 회원가입 - id 중복 검사
2. 사용자 로그인
3. 노래 검색
    a. 가수 이름으로 검색
    b. 노래 제목으로 검색
    c. 장르로 검색

→ 검색 후

- 노래 재생 → 노래의 streaming cnt 업데이트 (1 증가)
- 좋아하는 노래에 추가 → 노래의 likes cnt 업데이트 (1증가)

4. 플레이리스트 생성
5. 보유 중인 플레이리스트 확인
6. 플레이리스트 수정
    a. 노래 추가
    b. 노래 삭제
7. 플레이리스트 삭제
8. 다른 사람의 플레이리스트 보기

9. 가수 검색

   → 좋아하는 가수 추가 가능

10. 좋아요 한 음악 보기
11. 좋아요 한 가수 보기
12. 가장 많이 스트리밍 된 노래 top 5 확인
13. 가장 많은 좋아요를 받은 음악 top 5 확인

코드 설명

파일

- `db_connection.py` : mysql 연결을 위한 코드
- `music_streaming.py` : 프로그램 시작 페이지
- `admin.py` : 관리자 페이지
- `user.py` : 사용자 페이지

music_streaming

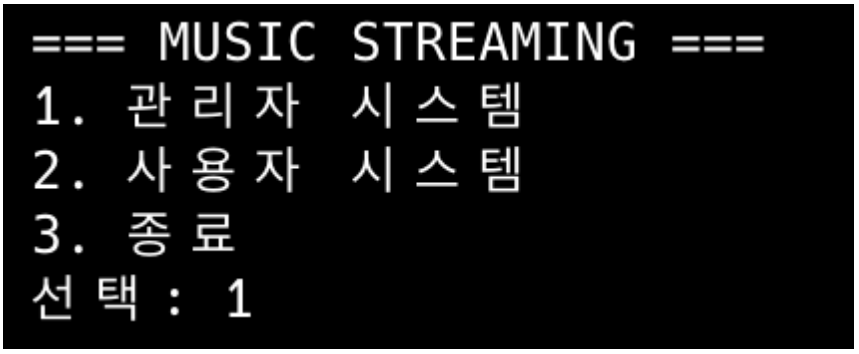
```
from admin import admin_page
from user import user_page

def main():
    while True:
        print("=" * 50)
        print("music streaming".center(45))
        print("=" * 50)
        print("1. 관리자 시스템")
        print("2. 사용자 시스템")
        print("3. 종료")
        choice = input("선택: ")
        if choice == "1":
            admin_page()
        elif choice == "2":
```

```
            user_page()
        elif choice == "3":
            print("프로그램을 종료합니다.")
            break
        else:
            print("잘못된 입력입니다. 다시 시도하세요.")

if __name__ == "__main__":
    main()
```

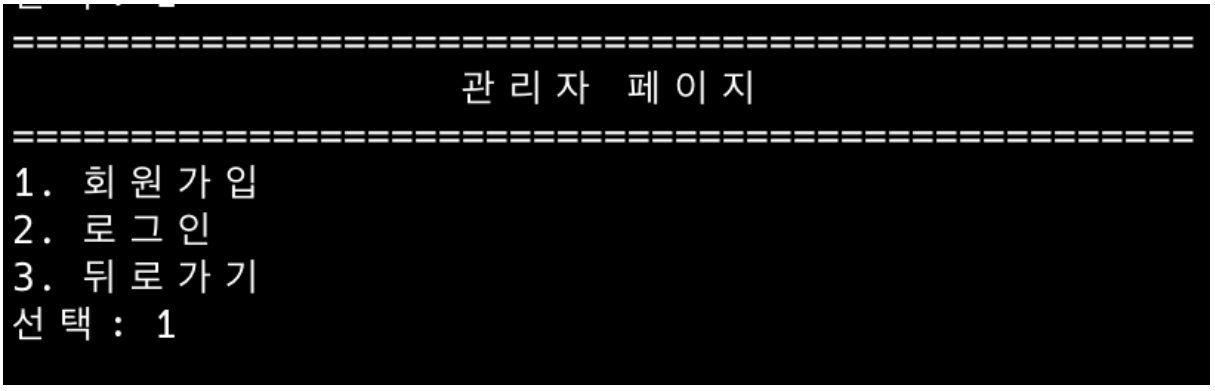
→ 사용자는 관리자/사용자로 분리된다.

수행 기능 스크린 샷

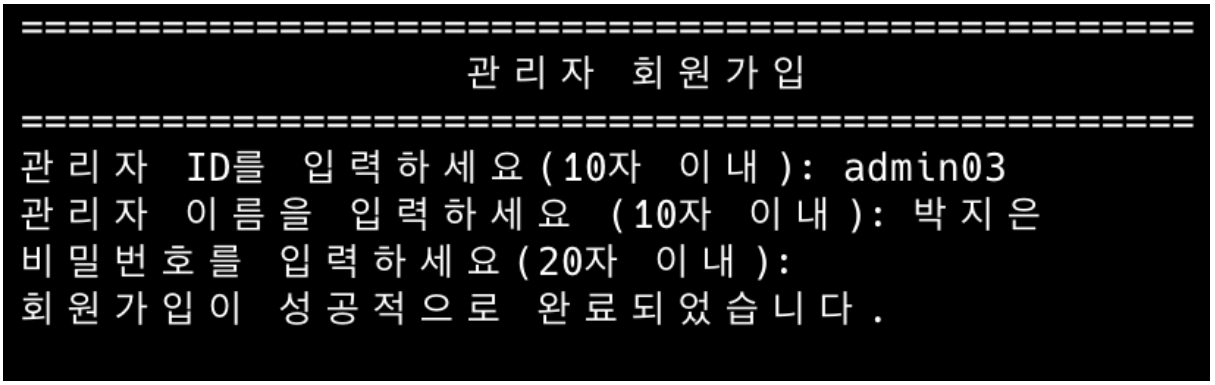
시작 화면



관리자

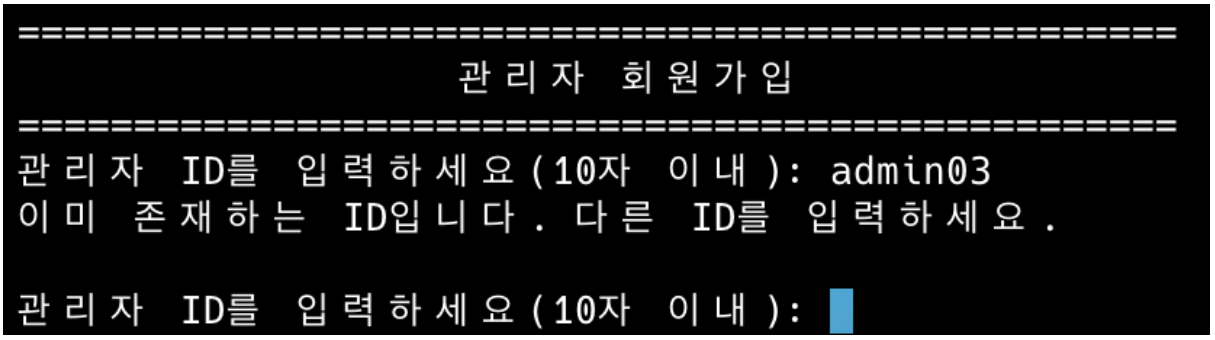


회원가입




```
query "INSERT INTO ADMIN (Aid, Aname, Apwd) VALUES (%s, %s,
%s)"
cursor.execute(query, (adminid, Aname, Apwd))
```

→ 입력받은 Id, name, pwd 로 튜플 삽입

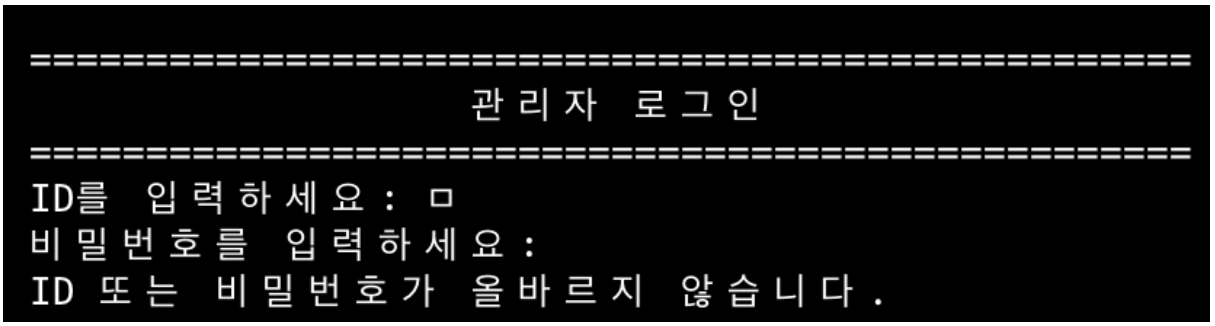


→ ID 중복 검사 시행

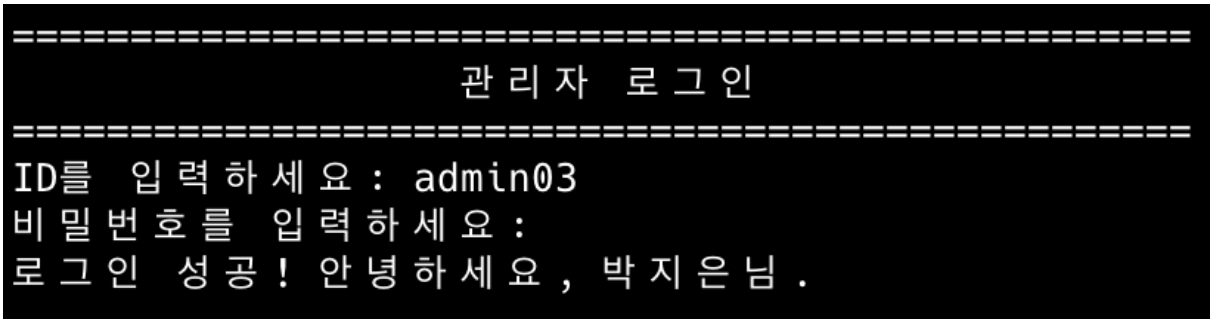
```
def is_adminid_duplicate(adminid):
        ...
        query = "SELECT COUNT(*) FROM ADMIN WHERE Aid = %s"
        cursor.execute(query, (adminid,))
        result = cursor.fetchone()

        if (result[0] == 0):
            return False
        return True
    ...
```

→ 이미 존재하는 Id 인지 확인

로그인



→ 로그인 실패



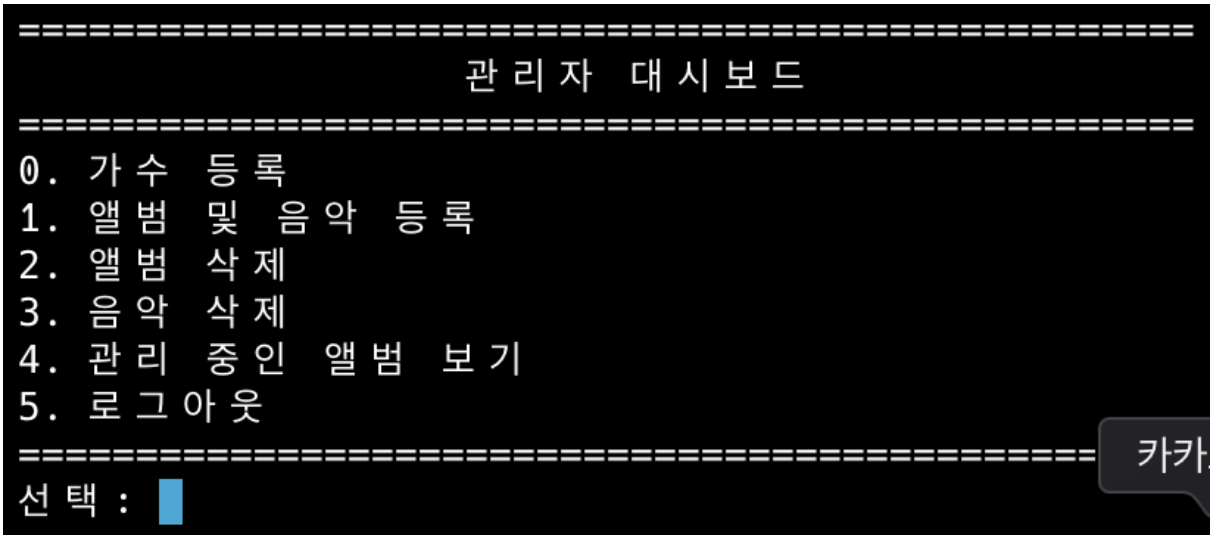
→ 로그인 성공

```
global Admin_idx
query = "SELECT * FROM ADMIN WHERE Aid = %s AND Apwd = %s"
        cursor.execute(query, (Aid, Apwd))
        result = cursor.fetchone()

        if result:
            print(f"로그인 성공! 안녕하세요, {result[2]}님.\n
\n")
            Admin_idx = result[0]
```

→ id와 pwd 로 select 한 결과가 있다면 로그인 성공

→ 이후 계속해서 사용될 admin idx 는 전역변수로 선언해두고 로그인 후 이곳에 저장한다.

관리자 대시보드



0. 가수 등록 : 관리자는 가수를 등록 할 수 있다
1. 앨범 및 음악 등록 : 음악을 등록 할 때에는 반드시 앨범을 통해 등록해야 한다.

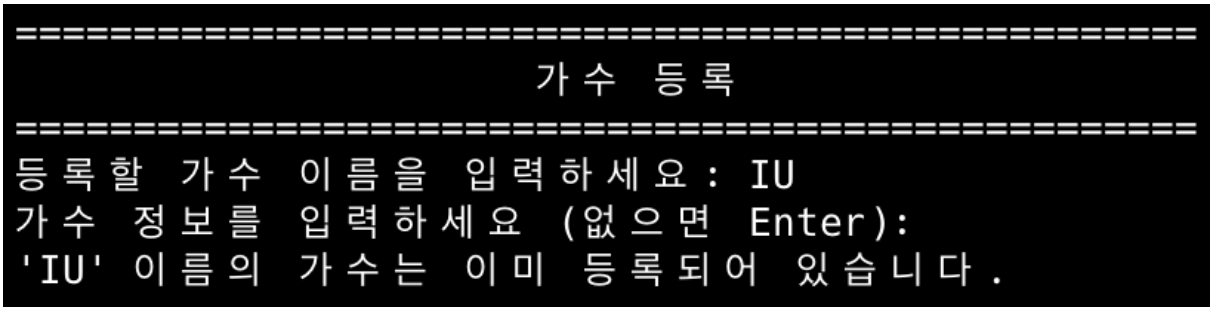
   → 앨범 안에 음악이 없는 것은 가능하나, 앨범 없이 음악을 추가할 수는 없다.

   → 앨범을 처음 등록할 때에는 적어도 한 개 이상의 음악을 추가해야한다. (하지만 이후 앨범 안에 있는 모든 음악을 삭제하더라도, 앨범은 유지한다.)
2. 관리자는 자신이 관리하는 앨범을 삭제할 수 있다.

   → 앨범이 삭제되는 경우, 앨범 안에 있는 음악 또한 전부 삭제된다. (cascade)
3. 관리자는 자신이 관리하는 앨범에서 음악을 삭제할 수 있다.

   → 이미 등록한 앨범에 음악을 추가할 수는 없다.
4. 관리자는 자신이 관리 중인 앨범을 볼 수 있다.

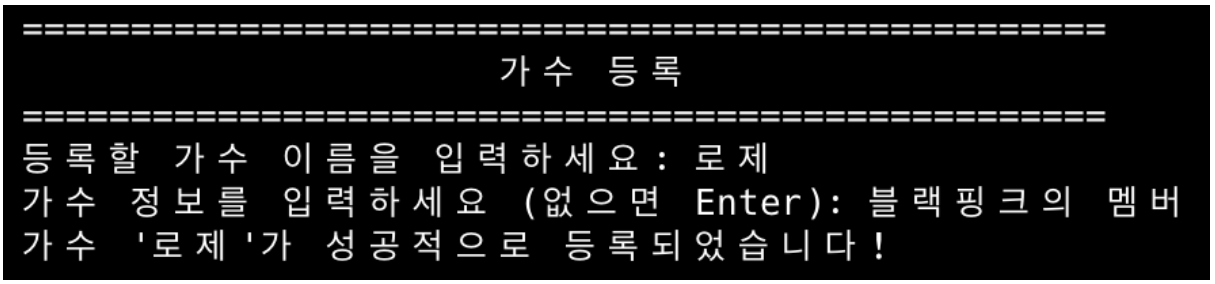
가수 등록



→ 이미 존재하는 가수를 추가할 수 없다 (간단한 프로그램 특성 상 가수를 구분하는 방법이 Idx 와 이름 뿐이고, 사용자 입장에서 구분할 수 있는 방법이 가수 이름 뿐이므로 중복된 이름을 갖는 가수는 존재할 수 없게 구현했다.)

```
query = "SELECT COUNT(*) FROM ARTIST WHERE Artist_name = %
s"
        cursor.execute(query, (artist_name,))
        result = cursor.fetchone()

        if result[0] > 0:
            print(f"'{artist_name}' 이름의 가수는 이미 등록되어
있습니다.")
            return
```

→ 이름으로 select 해서 결과가 존재한다면 이미 등록되어 있는 것




```
if result[0] > 0:
    print(f"'{artist_name}' 이름의 가수는 이미 등록되어 있습니
다.")
    return

query = """
    INSERT INTO ARTIST (Artist_name, Info)
    VALUES (%s, %s)
"""
cursor.execute(query, (artist_name, info if info else Non
e))
```

→ 가수는 `name` 과 `info` 를 입력 받아서 튜플 추가

앨범 및 음악 등록

→ 앨범을 먼저 등록

- 앨범 소유 가수는 단 한명이다.

```
Owner_artist_idx = get_artist_idx(Owner_artist_name)

def get_artist_idx(artist_name):
    ...
 query = "SELECT Artist_idx FROM ARTIST WHERE Artist_name =
%s"
    ...
```

→ 가수 이름을 받으면 가수의 idx 를 찾는다.

```
query = """
            INSERT INTO ALBUM (Album_name, Description, Rel
ease_date, Admin_idx, Owner_artist_idx)
            VALUES (%s, %s, %s, %s, %s)
        """
        cursor.execute(query, (Album_name, Description, Rel
ease_date, Admin_idx, Owner_artist_idx))

cursor.execute("SELECT LAST_INSERT_ID();")
Album_idx = cursor.fetchone()[0]
```

```
music_registration(Album_idx)
set_title(Album_idx)
```

→ 앨범 등록 후 마지막으로 등록한 (현재 등록 중인) 앨범 Idx 를 받아서 이후 앨범에 음악과 타이틀을 추가하는 함수에 넘겨준다.

→ 노래 등록 프로세스

→ 가사는 존재하지 않을 수 있다 (재생 시간과 장르 또한 존재하지 않을 수 있지만 입력할 것을 권장함)

→ 또한 작사가와 작곡가도 추가하지 않을 수 있다.

→ 가수는 적어도 한 명 이상이 존재해야 한다.

```
다른 노래를 추가하시겠습니까? (y/n): y
노래 제목: Lavender Haze
재생 시간 (HH:MM:SS): 00:03:23
가사 (가사가 없다면 Enter):
장르: Pop
노래 'Lavender Haze'이(가) 성공적으로 등록되었습니다.


==================================================
                    가수 등록
==================================================
가수 이름 (끝내려면 Enter): Taylor Swift
가수 'Taylor Swift'이(가) 등록되었습니다.

가수 이름 (끝내려면 Enter): IU
가수 'IU'이(가) 등록되었습니다.

가수 이름 (끝내려면 Enter):
==================================================
                    작곡가 등록
==================================================
작곡가 이름 (없다면 Enter): IU
작곡가 'IU'이(가) 등록되었습니다.

작곡가 이름 (없다면 Enter):
==================================================
                    작사가 등록
==================================================
작사가 이름 (없다면 Enter): Taylor Swift
작사가 'Taylor Swift'이(가) 등록되었습니다.

작사가 이름 (없다면 Enter):
다른 노래를 추가하시겠습니까? (y/n): n
==================================================
                  타이틀 곡 등록
==================================================
타이틀 곡 이름 (없으면 Enter): Anti-Hero
타이틀 곡 'Anti-Hero'이(가) 성공적으로 등록되었습니다.
```

→ 한 앨범에 한 개 이상의 노래가 들어갈 수 있다

→ 앨범 등록 마지막에는 타이틀 곡을 등록 할 수 있으나 이 역시 등록하지 않을 수도 있다.

```
query = """
INSERT INTO MUSIC (Music_name, Duration, Lyrics, Genre, Lik
es_cnt, Streaming_cnt, Inc_Album_idx)
VALUES (%s, %s, %s, %s, %s, %s, %s)
"""
```

→ 처음 음악을 추가할 때는 Likes_cnt 와 Streaming_cnt 는 0 으로 추가해 주었다.

```
  cursor.execute("SELECT LAST_INSERT_ID();")
  Music_idx = cursor.fetchone()[0]

  query_singer = """
      INSERT INTO SINGERS (Music_idx, Singer_idx)
      VALUES (%s, %s)
  """
```

→ 가수는 SINGERS 테이블에 입력받은 모든 가수를 추가해 주었다.

→ 직전에 추가했던 (현재 추가 중인) 음악 idx 를 받아서 가수를 추가해 주었다.

```
query_composer =
"""
    INSERT INTO COMPOSER (Music_idx, Composer_name)
    VALUES (%s, %s)
"""
```

```
query_lyricist =
"""
    INSERT INTO LYRICIST (Music_idx, Lyricist_name)
    VALUES (%s, %s)
"""
```

→ 작곡가와 작사가 또한 같은 로직으로 추가해 주었다.

음악 삭제

```
음악을 삭제할 앨범 이름을 입력하세요: Midnights
==================================================
                    음악 목록
==================================================
Anti-Hero
Lavender Haze

삭제할 음악 이름을 입력하세요: Lavender Haze
음악 'Lavender Haze'이(가) 성공적으로 삭제되었습니다.
```

```
query= """
    SELECT Album_idx
    FROM ALBUM
    WHERE Album_name = %s AND Admin_idx = %s
  """
query= """
    SELECT Music_idx, Music_name
    FROM MUSIC
    WHERE Inc_Album_idx = %s
"""
query_delete = """
    DELETE FROM MUSIC
    WHERE Music_name = %s AND Inc_Album_idx = %s
"""
```

→ 앨범을 먼저 찾고, 해당 앨범의 idx 를 이용하여 앨범에 속한 음악을 찾은 뒤 삭제할 음악의 이름을 입력받아서 DELETE 한다.

앨범 삭제

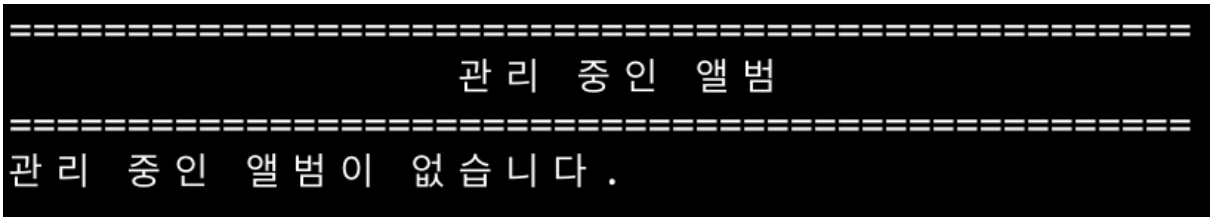
```
삭제할 앨범 이름을 입력하세요: test
test 앨범이 성공적으로 삭제되었습니다.
```

→ 앨범 안에 존재하던 노래 전부 삭제

```
query = """
    SELECT Album_idx
    FROM ALBUM
    WHERE Album_name LIKE %s AND Admin_idx = %s
"""
if not result:
    print(f"'{album_name}' 이름의 앨범을 찾을 수 없습니다.")
    return

album_idx = result[0]

query = "DELETE FROM ALBUM WHERE Album_idx = %s AND Admin_i
dx = %s"
```

→ 삭제할 앨범의 이름을 입력받은 뒤, 현재 본인이 관리 중인 앨범에 해당 앨범이 존재하는지 검증 후 삭제를 진행한다.

관리 중인 앨범 보기



→ 관리 중인 앨범이 없는 경우

→ 관리 중인 앨범 확인 가능

- 앨범에 포함된 노래 전부 출력

```
query = """
  SELECT Album_idx, Album_name, Description, Release_date,
Owner_artist_idx
  FROM ALBUM
  WHERE Admin_idx = %s
"""
```

```
query = """
  SELECT Artist_name
  FROM ARTIST
  WHERE Artist_idx = %s
"""
query = """
    SELECT MUSIC.Music_idx, MUSIC.Music_name, MUSIC.Duratio
n, MUSIC.Genre
    FROM TITLESONG
    JOIN MUSIC ON TITLESONG.Music_idx = MUSIC.Music_idx
    WHERE TITLESONG.Album_idx = %s
"""
query = """
    SELECT MUSIC.Music_idx, MUSIC.Music_name, MUSIC.Duratio
n, MUSIC.Genre
    FROM MUSIC
    WHERE Inc_Album_idx = %s
"""

  query = """
      SELECT ARTIST.Artist_name
      FROM SINGERS
      JOIN ARTIST ON SINGERS.Singer_idx = ARTIST.Artist_idx
      WHERE SINGERS.Music_idx = %s
  """
```

→ 앨범 검색

→ 앨범 소유 가수 정보 가져오기

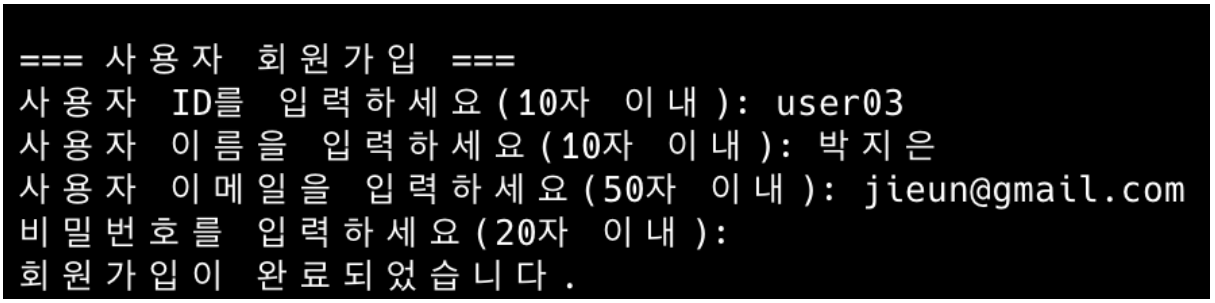
→ 타이틀 음악 정보 가져오기

→ 앨범 음악 가져오기

→ 가수 정보 가져오기

사용자

회원가입




```
query = "INSERT INTO USER (Uid, Uname, Uemail, Upwd, Join_d
ate) VALUES (%s, %s, %s, %s, CURRENT_DATE)"
```

로그인

```
class UserInfo:
    def __init__(self, Uidx, Uid, Uname, Uemail, Join_date):
        self.Uidx = Uidx
        self.Uid = Uid
        self.Uname = Uname
        self.Uemail = Uemail
        self.Join_date = Join_date
```

```
query = "SELECT * FROM USER WHERE Uid = %s AND Upwd = %s"
...
user_info = UserInfo(
    Uidx=result[0],
    Uid=result[1],
    Uname=result[2],
    Uemail=result[3],
    Join_date=result[5]
)
```

→ 사용자는 정보가 많기 때문에 구조체를 만들어주었고, 전역 변수로 선언한 뒤 로그인 시 정보를 저장한다.

사용자 대시보드

```
==================================================
                  사 용 자  로 그 인
==================================================
ID: user03
PWD:



로 그 인  성 공 !
환 영 합 니 다 , 박 지 은 님 .


==================================================
                  사 용 자  대 시 보 드
==================================================
1. 노 래  검 색
2. 플 레 이 리 스 트  생 성
3. 보 유  중 인  플 레 이 리 스 트  보 기
4. 플 레 이 리 스 트  수 정
5. 플 레 이 리 스 트  삭 제
6. 다 른  사 람 의  플 레 이 리 스 트  보 기
7. 가 수  검 색
8. 좋 아 요  한  음 악  보 기
9. 좋 아 요  한  가 수  보 기
0. 로 그 아 웃
==================================================
원 하 는  작 업 을  선 택 하 세 요 :
```

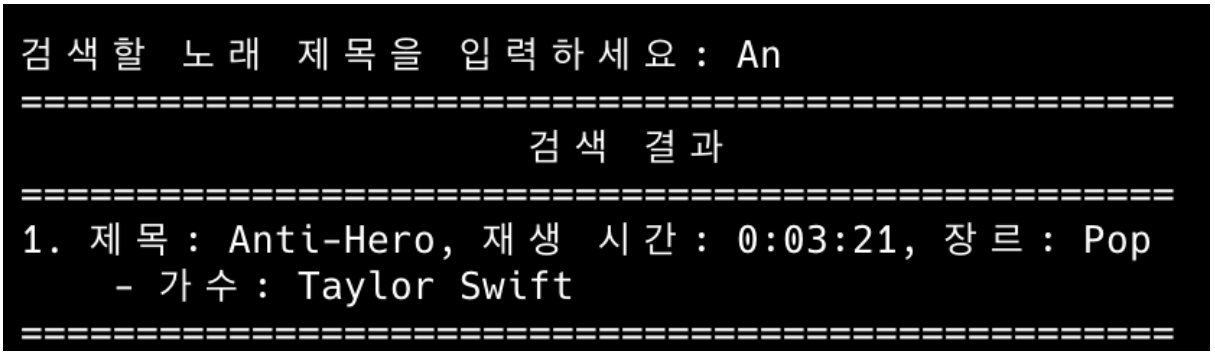
노래 검색

- 사용자는 노래 검색 후 재생/좋아하는 음악에 추가를 할 수 있다.

```
==================================================
                  노 래  검 색
==================================================
1. 노 래  제 목 으 로  검 색
2. 가 수 로  검 색
3. 장 르 로  검 색
4. 뒤 로 가 기
==================================================
```

사용자는 3가지 방식으로 검색을 할 수 있다.

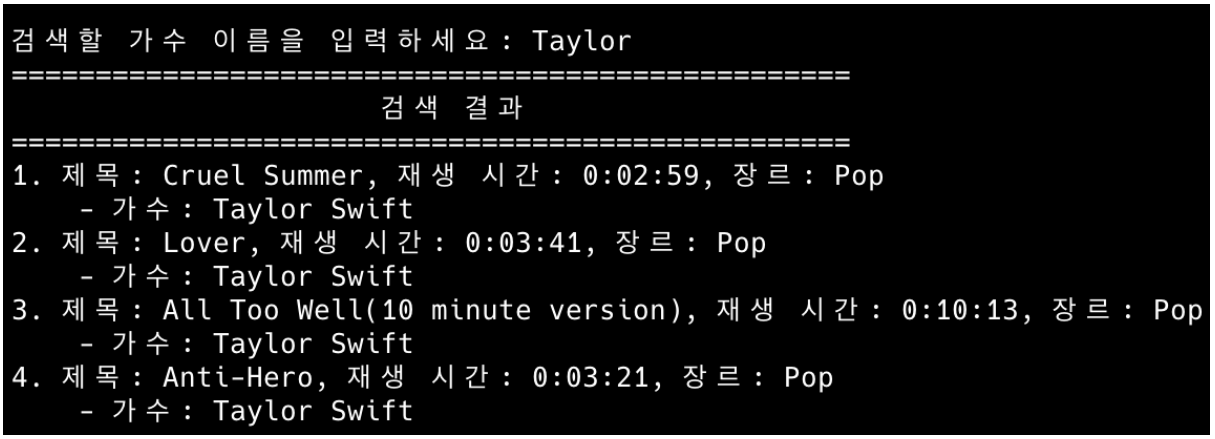
1. 노래 제목으로 검색




```
name = input("검색할 노래 제목을 입력하세요: ")
query = """
    SELECT Music_idx, Music_name, Duration, Genre, Lyrics
    FROM MUSIC
    WHERE Music_name LIKE %s
"""

cursor.execute(query, (f"%{name}%",))
```

→ substring 으로 검색이 가능하게 하도록 하기 위해 `LIKE` 를 사용했다.

2. 가수로 검색

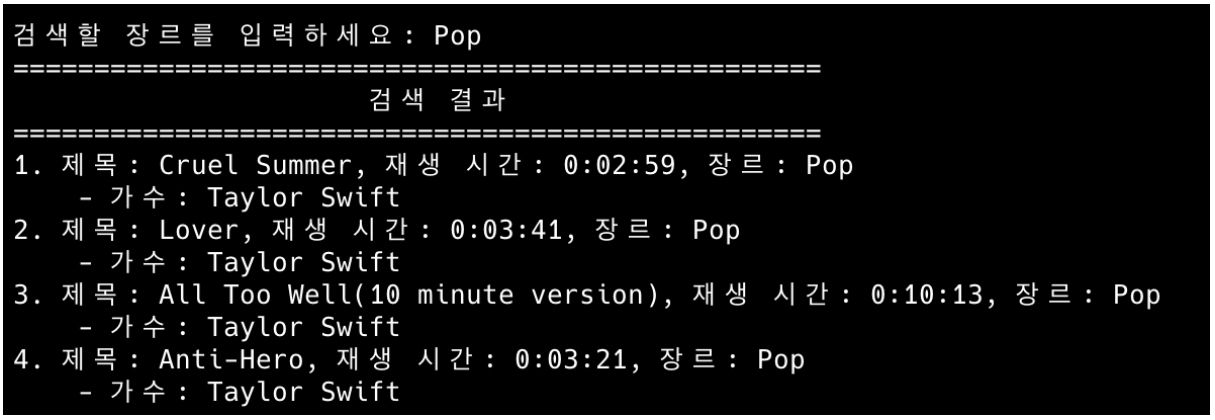



```
artist_name = input("\n검색할 가수 이름을 입력하세요: ")
query = """
```

```
  SELECT MUSIC.Music_idx, Music_name, Duration, Genre, Lyri
cs
  FROM ARTIST
  JOIN SINGERS ON Artist_idx = Singer_idx
  JOIN MUSIC ON SINGERS.Music_idx = MUSIC.Music_idx
  WHERE ARTIST.Artist_name LIKE %s;
"""
cursor.execute(query, (f"%{artist_name}%",))
```

→ 위와 동일한 로직으로 가수 이름을 입력 받아서 ARTIST 와 MUSIC 테이블을 JOIN 하여 음악 정보를 가져왔다.

3. 장르로 검색




```
genre = input("검색할 장르를 입력하세요: ")
query = """
  SELECT Music_idx, Music_name, Duration, Genre, Lyrics
  FROM MUSIC
  WHERE Genre = %s
"""
cursor.execute(query, (genre,))
```

→ 입력받은 장르를 통해 음악을 `SELECT` 해준다.

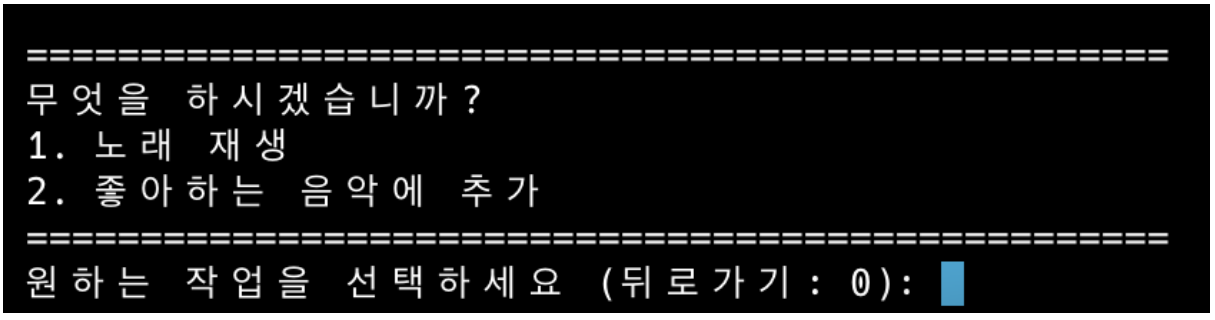
```
query = """
  SELECT ARTIST.Artist_name
  FROM SINGERS
  JOIN ARTIST ON SINGERS.Singer_idx = ARTIST.Artist_idx
  WHERE SINGERS.Music_idx = %s
```

```
"""
cursor.execute(query, (music[0],))
singers = cursor.fetchall()
singer_names = ", ".join(singer[0] for singer in singers)
except Exception as e:
singer_names = "알 수 없음"
print(f"가수 정보 로드 중 오류 발생: {e}")
finally:
cursor.close()
connection.close()

print(f"{idx}. 제목: {music[1]}, 재생 시간: {music[2]}, 장르:
{music[3]}")
print(f"    - 가수: {singer_names} ")
```

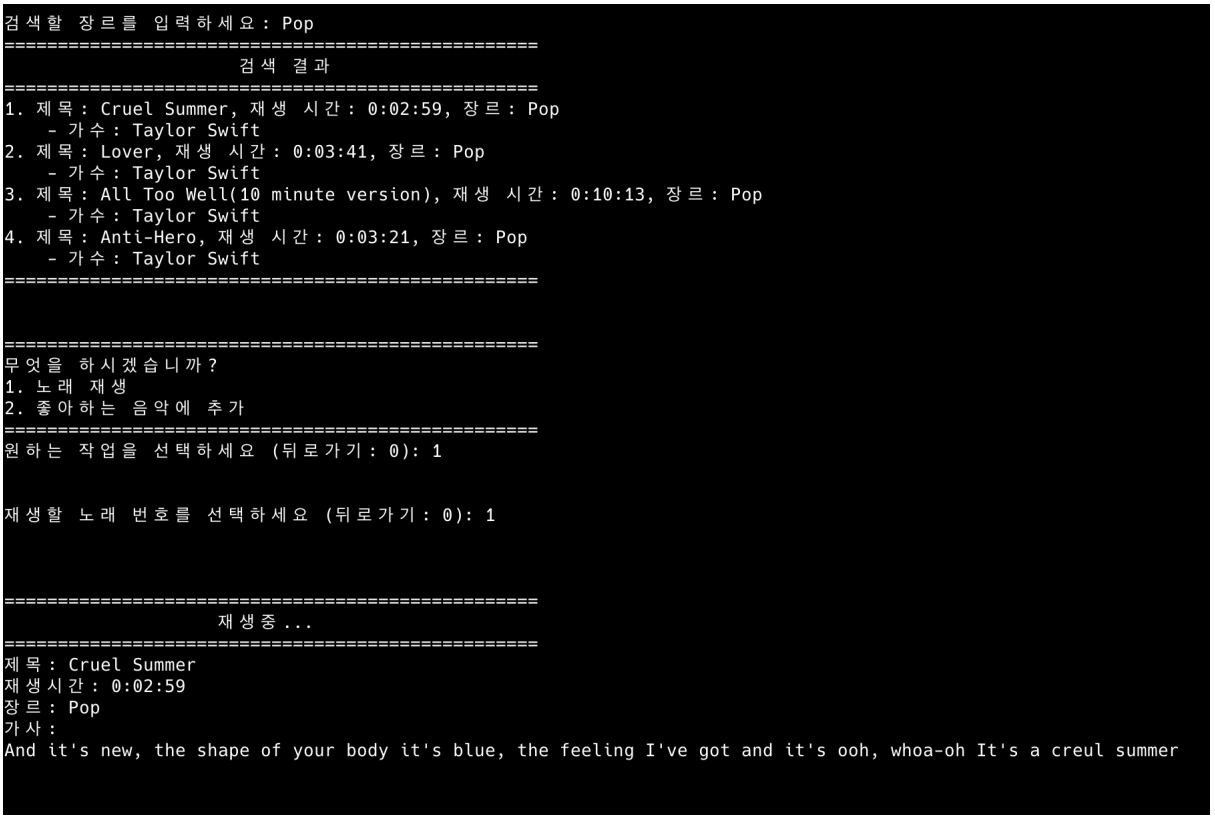
→ 이전에 찾아둔 music 을 가지고, 노래를 부른 가수를 `SINGERS` 테이블에서 `SELECT` 로 찾아온다.

→ 노래를 부른 가수는 한 명 이상일 수 있다. → (" , " 로 이어붙여서 출력해준다.)

검색 이후 사용자는 두가지 작업을 수행 가능하다



1. 노래 재생

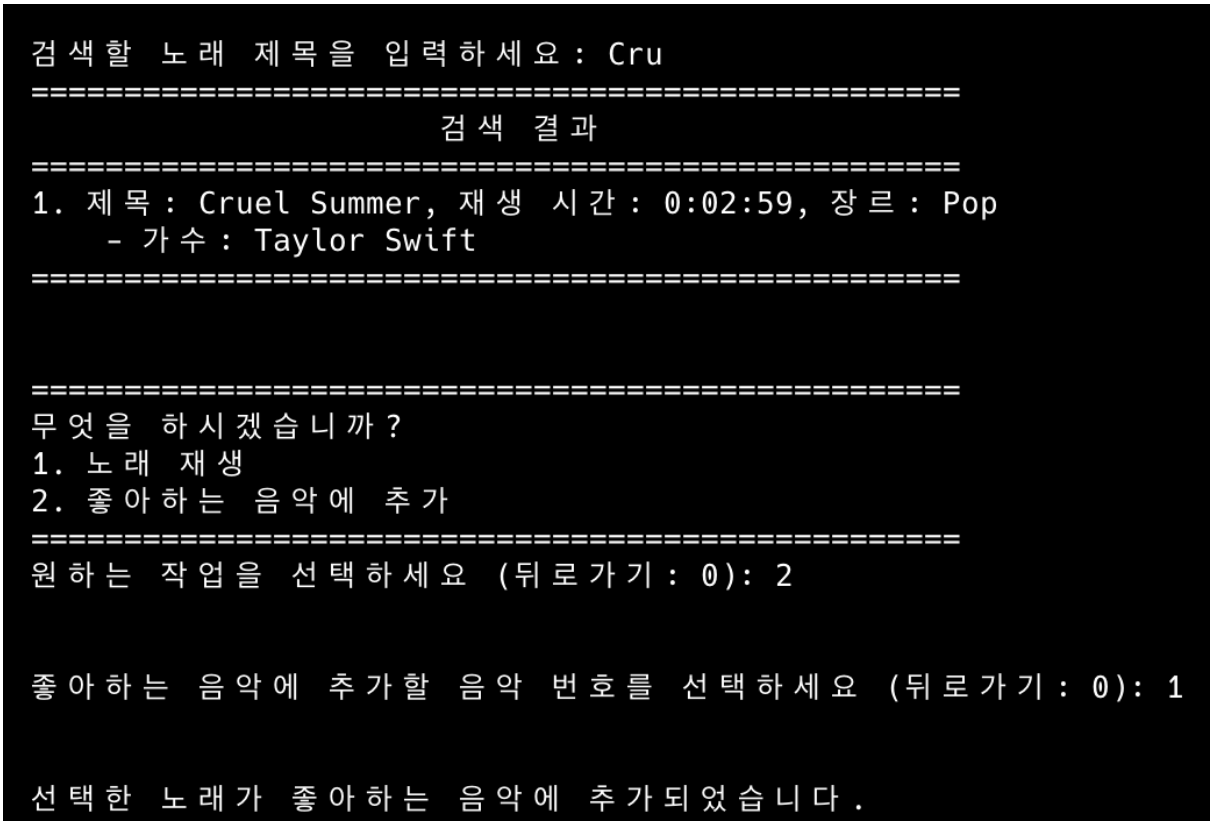


이전 검색에서 확인한 번호를 이용하여 노래를 재생하면, 노래 정보 및 가사가 나온다

```
query = """
    UPDATE MUSIC
    SET Streaming_cnt = Streaming_cnt + 1
    WHERE Music_idx = %s
"""
```

→ 노래를 재생하면 노래의 `streaming_cnt` 가 1 올라야한다.

2. 좋아하는 음악에 추가




```
query = """
    INSERT INTO LIKE_MUSIC (User_idx, Music_idx)
    VALUES (%s, %s)
"""

query = """
    UPDATE MUSIC
    SET Likes_cnt = Likes_cnt + 1
    WHERE Music_idx = %s
"""
```

→ 좋아하는 음악에 추가하면 나의 LIKE_MUSIC 에 음악을 추가하고, 그 음악의 Likes_cnt 를 1 올려줘야한다.

플레이리스트 생성

```
==================================================
                  플레이리스트 생성
==================================================
플레이리스트 이름을 입력하세요: driving pop


'driving pop' 플레이리스트가 생성되었습니다.

노래를 플레이리스트에 추가합니다.
추가할 노래 제목을 입력하세요 (종료하려면 Enter): Cruel Summer
==================================================
                    검색 결과
==================================================
1. 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop
    - 가수: Taylor Swift
==================================================
추가할 노래 번호를 선택하세요 (취소: 0): 1


'Cruel Summer' 노래가 플레이리스트에 추가되었습니다.


추가할 노래 제목을 입력하세요 (종료하려면 Enter): L
==================================================
                    검색 결과
==================================================
1. 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop
    - 가수: Taylor Swift
2. 제목: Lover, 재생 시간: 0:03:41, 장르: Pop
    - 가수: Taylor Swift
3. 제목: All Too Well(10 minute version), 재생 시간: 0:10:13, 장르: Pop
    - 가수: Taylor Swift
==================================================
추가할 노래 번호를 선택하세요 (취소: 0): 2


'Lover' 노래가 플레이리스트에 추가되었습니다.


추가할 노래 제목을 입력하세요 (종료하려면 Enter):
노래 추가를 종료합니다.

플레이리스트 생성 및 노래 추가가 완료되었습니다.
```

```
query = """
    INSERT INTO PLAYLIST (Playlist_name, Owner_user_idx)
    VALUES (%s, %s)
"""
```

→ Playlist 추가

```
query = """
    SELECT Music_idx, Music_name, Duration, Genre
    FROM MUSIC
    WHERE Music_name LIKE %s
"""

query_singers = """
    SELECT ARTIST.Artist_name
    FROM SINGERS
    JOIN ARTIST ON SINGERS.Singer_idx = ARTIST.Artist_idx
    WHERE SINGERS.Music_idx = %s
"""
```

→ 사용자에게 음악 정보 출력

```
query= "INSERT INTO PLAYLIST_MUSIC (Play_idx, Music_idx) VA
LUES (%s, %s)"
```

→ 플레이리스트에 해당 음악 추가

플레이리스트 수정

- 노래추가 및 삭제 가능

```
==================================================
                  내 플레이리스트
==================================================
1. driving pop
==================================================
수정할 플레이리스트 번호를 선택하세요 (뒤로가기: 0): 1

'driving pop' 플레이리스트를 선택하셨습니다.

==================================================
'driving pop' 플레이리스트의 노래 목록
==================================================
1. 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop
2. 제목: Lover, 재생 시간: 0:03:41, 장르: Pop
==================================================



==================================================
                작업 선택
1. 노래 추가
2. 노래 삭제
3. 돌아가기
==================================================
선택: 1


추가할 노래 제목을 입력하세요 (종료하려면 Enter): R
==================================================
                    검색 결과
==================================================
1. 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop
    - 가수: Taylor Swift
2. 제목: Lover, 재생 시간: 0:03:41, 장르: Pop
    - 가수: Taylor Swift
3. 제목: All Too Well(10 minute version), 재생 시간: 0:10:13, 장르: Pop
    - 가수: Taylor Swift
4. 제목: Anti-Hero, 재생 시간: 0:03:21, 장르: Pop
    - 가수: Taylor Swift
==================================================
추가할 노래 번호를 선택하세요 (취소: 0): 3


'All Too Well(10 minute version)' 노래가 플레이리스트에 추가되었습니다.


추가할 노래 제목을 입력하세요 (종료하려면 Enter):
노래 추가를 종료합니다.
```

→ 노래 추가

```
==================================================
                  내 플레이리스트
==================================================
1. driving pop
==================================================
수정할 플레이리스트 번호를 선택하세요 (뒤로가기: 0): 1

'driving pop' 플레이리스트를 선택하셨습니다.

==================================================
'driving pop' 플레이리스트의 노래 목록
==================================================
1. 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop
2. 제목: Lover, 재생 시간: 0:03:41, 장르: Pop
3. 제목: All Too Well(10 minute version), 재생 시간: 0:10:13, 장르: Pop
==================================================



==================================================
                작업 선택
1. 노래 추가
2. 노래 삭제
3. 돌아가기
==================================================
선택: 2


삭제할 노래 제목을 입력하세요 (종료하려면 Enter): All Too Well(10 minute version
'All Too Well(10 minute version' 제목의 노래는 현재 플레이리스트에 없습니다.
삭제할 노래 제목을 입력하세요 (종료하려면 Enter): All Too Well(10 minute version)
'All Too Well(10 minute version)' 노래가 플레이리스트에서 삭제되었습니다.
삭제할 노래 제목을 입력하세요 (종료하려면 Enter):
```

→ 노래 삭제

```
query = """
    SELECT MUSIC.Music_idx
    FROM PLAYLIST_MUSIC
    JOIN MUSIC ON PLAYLIST_MUSIC.Music_idx = MUSIC.Music_id
x
WHERE PLAYLIST_MUSIC.Play_idx = %s AND MUSIC.Music_name = %
s
"""
query= "DELETE FROM PLAYLIST_MUSIC WHERE Play_idx = %s AND
Music_idx = %s"
```

→ 사용자가 입력한 이름의 music_idx 를 찾고 해당 음악을 플레이리스트에서 삭제

보유 중인 플레이리스트

```
=== 보유 중인 플레이리스트 ===

플레이리스트 이름: driving pop
포함된 노래:
  > 제목: Cruel Summer
  > 제목: Lover
  > 제목: All Too Well(10 minute version)

플레이리스트 이름: sleep
이 플레이리스트에는 노래가 없습니다.
```

** 플레이리스트에는 노래가 하나도 없어도 된다.

```
query = """
    SELECT Playlist_idx, Playlist_name
    FROM PLAYLIST
    WHERE Owner_user_idx = %s
"""
query = """
    SELECT MUSIC.Music_name, MUSIC.Duration, MUSIC.Genre
    FROM PLAYLIST_MUSIC
    JOIN MUSIC ON PLAYLIST_MUSIC.Music_idx = MUSIC.Music_idx
    WHERE PLAYLIST_MUSIC.Play_idx = %s
"""
```

→ 플레이리스트 이름과 함께 해당 플레이리스트에 속한 노래들의 정보를 가져온다.

플레이리스트 삭제

```
=== 플레이리스트 삭제 ===
삭제할 플레이리스트 이름을 입력하세요: ㄴ
'ㄴ' 이름의 플레이리스트가 존재하지 않습니다. 다시 입력하세요.
삭제할 플레이리스트 이름을 입력하세요: sleep
'sleep' 플레이리스트가 삭제되었습니다.
```

```
query = """
    SELECT Playlist_idx, Playlist_name
    FROM PLAYLIST
```

```
    WHERE Playlist_name = %s AND Owner_user_idx = %s
"""
query = "DELETE FROM PLAYLIST WHERE Playlist_idx = %s"
```

→ 사용자가 입력한 이름의 플레이리스트를 삭제한다.

→ cascade 로 플레이리스트에 있던 음악도 삭제된다.

다른 사람의 플레이리스트 보기

```
==================================================
다른 사용자의 플레이리스트 보기
==================================================

플레이리스트 목록:
==================================================
1. 이름: playlist01, 소유자 ID: user01
2. 이름: playlist02, 소유자 ID: user01
3. 이름: driving songs, 소유자 ID: user02
==================================================
보고 싶은 플레이리스트 번호를 선택하세요 (돌아가기: 0): 1

선택한 플레이리스트 음악 목록:
==================================================
노래 제목: Cruel Summer, 재생 시간: 0:02:59, 장르: Pop, 가수: Taylor Swift
노래 제목: Lover, 재생 시간: 0:03:41, 장르: Pop, 가수: Taylor Swift
==================================================
```

```
query = """
    SELECT Playlist_idx, Playlist_name, Uid
    FROM PLAYLIST
    JOIN USER ON Owner_user_idx = Uidx
    WHERE Owner_user_idx != %s
"""
query = """
      SELECT MUSIC.Music_name, MUSIC.Duration, MUSIC.Genre, A
      FROM PLAYLIST_MUSIC
      JOIN MUSIC ON PLAYLIST_MUSIC.Music_idx = MUSIC.Music_id
      JOIN SINGERS ON MUSIC.Music_idx = SINGERS.Music_idx
      JOIN ARTIST ON SINGERS.Singer_idx = ARTIST.Artist_idx
      WHERE PLAYLIST_MUSIC.Play_idx = %s
  """
```

→ 사용자 본인을 제외한 플레이리스트의 이름과 보유한 사용자 id 를 출력해준다.

→ 이후 사용자가 선택한 플레이 리스트에 속한 음악 정보 및 가수 정보를 출력해준다.

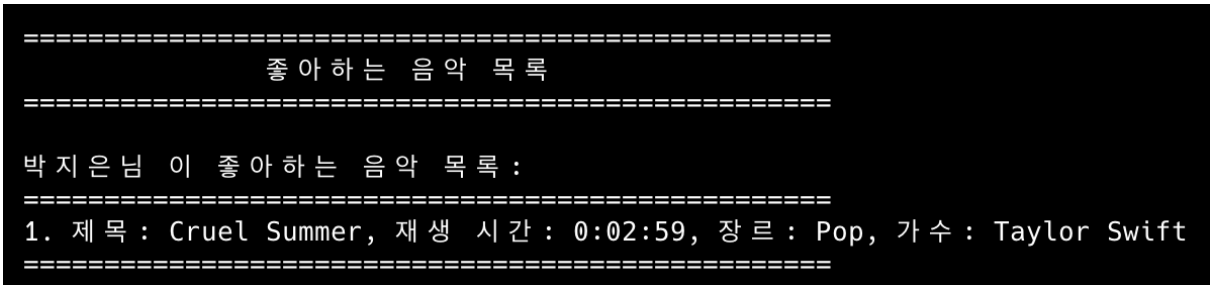
가수 검색

- 사용자는 가수를 검색하고, 좋아하는 가수에 추가할 수 있다


```
query = """
    SELECT Artist_idx, Artist_name, Info
    FROM ARTIST
    WHERE Artist_name LIKE %s
"""
query = """
    SELECT COUNT(*)
    FROM LIKE_ARTIST
    WHERE User_idx = %s AND Artist_idx = %s
"""
query= """
    INSERT INTO LIKE_ARTIST (User_idx, Artist_idx)
    VALUES (%s, %s)
"""
```

→ 가수 이름을 통해 `SELECT` 한 뒤, 좋아하는 가수에 이미 존재하는지 확인한 후 없다면 추가한다.

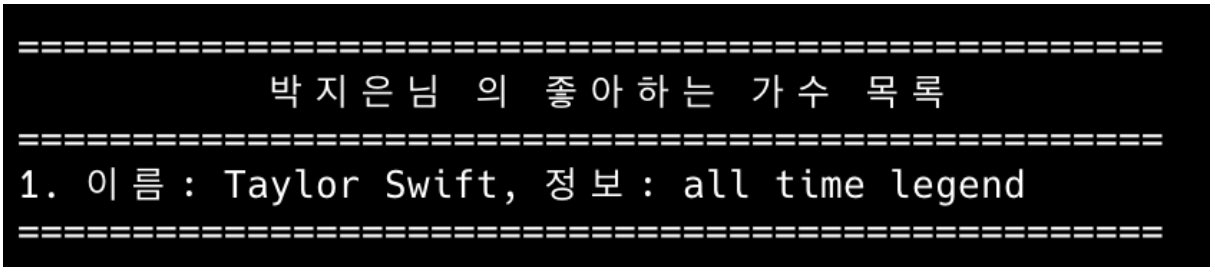
좋아요 한 음악 보기




```
    query = """
      SELECT MUSIC.Music_name, MUSIC.Duration, MUSIC.Genre,
GROUP_CONCAT(ARTIST.Artist_name SEPARATOR ', ') AS Artists
      FROM LIKE_MUSIC
      JOIN MUSIC ON LIKE_MUSIC.Music_idx = MUSIC.Music_idx
      JOIN SINGERS ON MUSIC.Music_idx = SINGERS.Music_idx
      JOIN ARTIST ON SINGERS.Singer_idx = ARTIST.Artist_idx
      WHERE LIKE_MUSIC.User_idx = %s
      GROUP BY MUSIC.Music_idx
    """
```

→ 좋아하는 음악에 있는 음악 정보와 가수 정보를 보여준다.

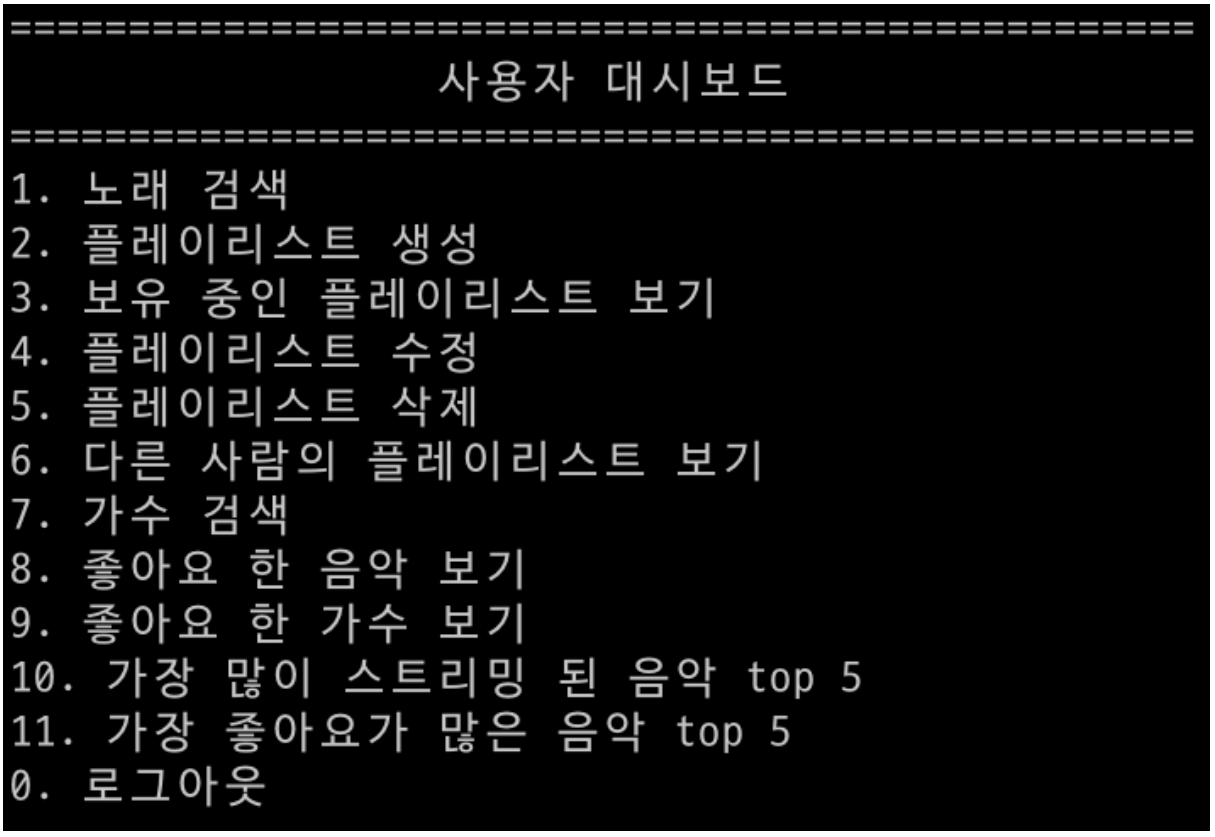
좋아요 한 가수 보기




```
query = """
    SELECT ARTIST.Artist_name, ARTIST.Info
    FROM LIKE_ARTIST
    JOIN ARTIST ON LIKE_ARTIST.Artist_idx = ARTIST.Artist_i
dx
    WHERE LIKE_ARTIST.User_idx = %s
"""
```

→ 좋아하는 가수에 추가한 가수 정보를 보여준다.

추가 기능



이후 top 5 기능을 추가하였다.

이전에 노래를 재생하거나 좋아하는 음악에 추가한 경우, streaming_cnt 와 like_cnt 를 높여줬으므로 각각 아래와 같은 sql 문을 이용하여 top 5 를 출력해주었다.

```
query = """
SELECT Music_name, Duration, Genre, Likes_cnt
FROM MUSIC
ORDER BY Likes_cnt DESC
LIMIT 5
"""
query = """
SELECT Music_name, Duration, Genre, Likes_cnt
FROM MUSIC
ORDER BY Likes_cnt DESC
```

```
LIMIT 5
"""
```