프로젝트 4: DBMS 개발

2018007965 김산

모든 단계의 메뉴 화면에서 주어진 메뉴 번호 이외의 입력은 프로그램을 종료시킨다.

1. def main()

```
0. Exit
1. Administrator Menu
2. User Menu
Input:
```

DB에 관리자로 접속을 할지 사용자로 접속을 할지 정하는 함수이다.

0번을 입력하면 프로그램이 종료되고, 1번을 입력하면 administratorMenu()를 호출하고, 2번을 입력하면 userMenu()를 호출한다.

2. def administratorMenu()

```
0: Return to Previous Menu
1: Enroll Music
2: Delete Music
3: Manage Users
4: Search Music
5. Show all Music
Input:
```

관리자로 접속을 했을 때 수행할 동작을 정하는 함수이다.

0번을 입력하면 전 단계인 main()으로 돌아가게 되고 1번을 입력하면 음악을 DB에 등록하는 기능을 할 수 있는 함수를 호출, 2번을 입력하면 음악을 DB에서 삭제하는 기능을 할 수 있는 함수를 호출, 3번을 입력하면 DB의 사용자를 관리할 수 있는 함수를 호출, 4번을 입력하면 검색한 음악의 정보를 출력하는 함수를, 5번을 입력하면 DB상의 모든 음악을 보여주는 함수를 호출한다.

2-1. def enrollMusic()

```
Name: Payphone
Info: good
ID: 4
Lyrics: Adam Levine
Compose: Adam Levine
Album ID: 1
Enroll ID: 1
Delete ID: 1
Artist ID: 1
Payphone enroll completed!
```

```
Name: a
Info: a
ID: a
Lyrics: a
Compose: a
Album ID: a
Enroll ID: a
Delete ID: a
Artist ID: a
Error occurred while enrolling music (1366, "Incorrect integer value: 'a' for column `mu
sic_app`.`music`.`ID` at row 1")
```

관리자 메뉴에서 1번을 입력할 경우 호출되는 함수로 음악을 DB에 등록하는 역할을 한다.

```
INSERT INTO music
(NAME, INFO, ID, LYRICS, COMPOSE, ALBUMID, ENROLL_ID, DELETE_ID, ARTISTID)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
SELECT NAME FROM music where id=%s
```

입력 받은 music tuple의 정보들을 INSERT문의 인자로 넘겨 음악을 추가한다. 또 입력 받은 인자 중 이름을 SELECT문을 통해 출력시켜 제대로 추가되었음을 보여준다. except문을 통해 정상적이지 않은 입력은 예외처리를 해 에러 메시지를 출력시킨다.

2-2. def deleteMusic()

```
ID: 4
Payphone delete completed!
```

```
ID: 123
Error occurred while deleting music
```

관리자 메뉴에서 2번을 입력할 경우 호출되는 함수로 음악을 DB에서 삭제하는 역할을 한다.

```
SELECT NAME FROM music WHERE ID=%s
DELETE FROM belong_to_playlist_music WHERE MID=%s
DELETE FROM has_music_genre WHERE MID=%s
DELETE FROM heard WHERE MID=%s
DELETE FROM preferred_music WHERE MID=%s
DELETE FROM music WHERE ID=%s
```

삭제할 음악의 ID를 입력 받으면 그 ID에 맞는 유효한 tuple이 있을 경우 이름을 SELECT문을 통해 출력시켜 삭제를 하겠다는 출력을 한다. 잘못된 입력일 경우 에러 메시지를 출력시키고 함수를 종료시킨다. 정상적인 입력일 경우, DELETE문을 통해 음악이 연결 되어있는 테이블인 belong_to_playlist_music, has_music_genre, heard, preferred_music에서 music의 ID와 같은 tuple들을 지워서 음악의 foreign key 연결관계를 전부 끊어준 후 music 테이블에서 해당 tuple을 지워 DB 상에서 원하는 음악을 지운다.

2-3. def managingUsers()

```
0. Return to Previous Menu
1. Enroll User
2. Delete User
3. View Users
Input:
```

관리자 메뉴에서 3번을 입력할 경우 호출되는 함수로 사용자를 관리하는 역할을 하는 함수들을 호출하기 위한 메뉴 역할을 하는 함수이다.

3-1. def enrollUser()

```
Name: DataBaseManager
ID: 6
Password: 111111
Nickname: DBM
Gender: M
Manage ID: 1
Sign ID: 1
DataBaseManager enrolled completed!
```

```
Name: adsf
ID: adfs
Password: asdf
Nickname: asdf
Gender: asfd
Manage ID: asfd
Sign ID: asdf
Error occurred while enrolling user (1366, "Incorrect integer value: 'adfs' for column `
music_app`.`user`.`ID` at row 1")
```

managingUser() 메뉴에서 1번을 입력할 경우 호출되는 함수로 사용자를 DB에 등록하는 역할을 한다.

```
INSERT INTO user(NAME, ID, PASSWORD, NICKNAME, GENDER, MNG_ID, SIGN_ID)
VALUES (%s, %s, %s, %s, %s, %s, %s)
SELECT NAME FROM user where ID=%s
```

입력 받은 user tuple의 정보들을 INSERT문의 인자로 넘겨 사용자를 추가한다. 또 입력 받은 인자 중 이름을 SELECT문을 통해 출력시켜 제대로 추가되었음을 보여준다. except문을 통해 정상적이

지 않은 입력은 예외처리를 해 에러 메시지를 출력시킨다.

3-2. def deleteUser()

```
ID: 6
DataBaseManager delete completed!
```

```
ID: asd
Error occurred while deleting user invalid literal for int() with base 10: 'asd'
```

managingUser() 메뉴에서 2번을 입력할 경우 호출되는 함수로 사용자를 DB에서 삭제하는 역할을 한다.

```
SELECT NAME FROM user WHERE ID=%s
DELETE FROM playlist WHERE UID=%s
DELETE FROM preferred_music WHERE UID=%s
DELETE FROM heard WHERE UID=%s
DELETE FROM preferred_artist WHERE UID=%s
DELETE FROM preferred_genre WHERE UID=%s
DELETE FROM user WHERE ID=%s
```

삭제할 사용자의 ID를 입력 받으면 그 ID에 맞는 유효한 tuple이 있을 경우 이름을 SELECT문을 통해 출력시켜 삭제를 하겠다는 출력을 한다. 잘못된 입력일 경우 에러 메시지를 출력시키고 함수를 종료시킨다. 정상적인 입력일 경우, DELETE문을 통해 사용자가 연결 되어있는 테이블인 playlist, preferred_music, heard, preferred_artist, preferred_genre, user에서 user의 ID와 같은 tuple들을 지워서 사용자의 foreign key 연결관계를 전부 끊어준 후 user 테이블에서 해당 tuple을 지워 DB 상에서 원하는 사용자를 지운다.

3-3. def viewUsers()

```
Name: ParkRoot ID: 1 Password: 111111 Nickname: root Gender: M ManageID: 1 SignID: 1
Name: KimJunYeop ID: 2 Password: 123456 Nickname: OnnuRi Gender: F ManageID: 1 SignID: 1

Name: KimJeongSoo ID: 3 Password: 123 Nickname: JS Gender: F ManageID: 1 SignID: 1
Name: LeeJongBeom ID: 4 Password: 100 Nickname: LJB Gender: M ManageID: 1 SignID: 1
Name: Andy ID: 5 Password: 123 Nickname: ANDY10 Gender: M ManageID: 1 SignID: 1
```

managingUser() 메뉴에서 3번을 입력할 경우 호출되는 함수로 모든 사용자를 보여주는 역할을 한다.

```
SELECT * FROM user
```

DB상의 모든 사용자를 SELECT문을 통해 출력시킨다.

4. def userMenu()

```
0. Return to Previous Menu
1. Sign Up
2. Log In
Input:
```

사용자로 접속을 했을 때 수행할 동작을 정하는 함수이다.

0번을 입력하면 전 단계인 main()으로 돌아가게 되고 1번을 입력하면 사용자가 회원가입을 하여서 DB에 user 하나를 등록하는 함수 호출, 2번을 입력하면 기존에 있던 user 중 하나의 사용자가 로그인을 하는 함수를 호출한다.

4-1. def signUp()

```
Name: Andy
ID: 5
Password: 123
Nickname: ANDY10
Gender: M
Manage ID: 1
Sign ID: 1
Andy signed up completed!
```

```
Name: a
ID: a
Password: a
Nickname: a
Gender: a
Manage ID: a
Sign ID: a
Error occurred while signing up (1366, "Incorrect integer value: 'a' for column `music_app`.`user`.`ID` at row 1")
```

사용자 화면에서 1번을 입력하면 나오는 회원가입을 위한 함수이다.

```
INSERT INTO user(NAME, ID, PASSWORD, NICKNAME, GENDER, MNG_ID, SIGN_ID)
VALUES (%s, %s, %s, %s, %s, %s, %s)
SELECT NAME FROM user where ID=%s
```

user에 추가를 하기 위해 7가지의 정보를 입력을 받고 INSERT문을 활용해서 테이블에 추가를 한다. 그리고 추가가 되었다는 결과가 출력되게 하기 위해 SELECT문을 활용하여 입력한 ID와 동일한 ID를 갖는 tuple의 이름을 활용하여 성공적으로 회원가입이 되었다는 것을 출력한다. except문을 통해 잘못된 입력 등, 정상적이지 못한 경우가 있다면 예외처리를 해준다.

4-2. def logIn()

```
ID: 5
Password: 123
Name: Andy ID: 5 logged in!
```

```
ID: a
Password: a
ID or Password Error!
```

사용자 화면에서 2번을 입력하면 나오는 로그인을 위한 함수이다.

```
SELECT NAME, ID FROM user WHERE ID=%s AND PASSWORD=%s
```

로그인을 하기 위해 ID와 Password를 입력 받고 이에 해당하는 tuple이 있을 경우 SELECT문을 활용하여 해당 ID를 가진 user tuple의 NAME과 ID를 출력시킨다. 그리고 추가가 되었다는 결과가 출력되게 하기 위해 SELECT문을 활용하여 입력한 ID와 동일한 ID를 갖는 tuple의 이름을 활용하여 성공적으로 회원가입이 되었다는 것을 출력한다. 그리고 입력 받았던 ID를 인자로 하는 로그인 후의 역할을 할 함수를 호출한다. 해당 ID와 Password를 가진 tuple이 없을 경우 에러 메시지를 출력하고 사용자 메뉴로 돌아가게 된다.

5. realUserMenu(a)

```
0. Return to Previous Menu
1. Playlist Menu
2. Preference Menu
3. View Music Information
Input:
```

로그인 후에 실제로 사용자로서의 역할을 할 함수들을 호출하기 위한 함수이다. 1번과 2번 메뉴에서 인자를 활용하기 위해 전 단계에서 사용자의 ID를 인자로 받아온다.

0번을 입력하면 전 단계인 main()으로 돌아가게 되고 1번을 입력하면 playlist 테이블을 관리할 역할을 하는 함수를 호출, 2번을 입력하면 사용자가 선호하는 음악, 가수, 장르에 대해서 추가, 삭제 등의 역할을 하는 기능을 할 수 있는 함수를 호출, 3번을 입력하면 음악에 대한 정보를 볼 기능을 할 함수를 호출한다.

6. def playlistMenu(a)

```
0. Return to Previous Menu
1. Create Playlist
2. Delete Playlist
3. Add Music to Playlist
4. Delete Music from Playlist
5. View my Playlists
6. View all Playlists
7. View Details of Playlists
Input:
```

Playlist에 관한 동작을 수행하기 위한 함수들을 호출하기 위한 함수이다. 이 함수를 통해 호출되는 함수에서 ID를 활용하기 위해 인자로 받아온다.

0번을 입력하면 전 단계인 main()으로 돌아가게 되고 1번을 입력하면 playlist tuple 하나를 생성할 역할을 하는 함수를 호출, 2번을 입력하면 playlist tuple 하나를 삭제할 역할을 하는 함수를 호

출, 3번을 입력하면 음악을 플레이리스트에 추가할 역할을 하는 함수를 호출, 4번을 입력하면 음악을 플레이리스트에 삭제할 역할을 할 함수를 호출, 5번을 입력하면 로그인한 사용자가 가지고 있는 플레이리스트를 보여주는 역할을 하는 함수를 호출, 6번을 입력하면 모든 플레이리스트를 보여주는 역할을 할 함수를 호출, 7번을 입력하면 원하는 플레이리스트의 자세한 정보를 보여주는 역할을 할 함수를 호출한다.

6-1. def createPlaylist(a)

```
Name: BestOfBest
Playlist ID:5
BestOfBest create completed!
```

```
Name: a
Playlist ID:a
Error occurred while creating playlist (1366, "Incorrect integer value: 'a' for column `
music_app`.`playlist`.`ID` at row 1")
```

플레이리스트 메인 화면에서 1번을 입력하면 나오는 함수이다.

```
INSERT INTO playlist(UID, NAME, ID) VALUES (%s,%s,%s)
SELECT NAME FROM playlist where UID=%s AND ID=%s
```

새로운 Playlist를 만들기 위해 플레이리스트 이름과 ID를 입력 받고 INSERT문을 활용하여 새롭게 tuple을 만든다. 그리고 추가가 되었다는 결과가 출력되게 하기 위해 SELECT문을 활용하여 입력한 ID와 동일한 ID를 갖는 tuple의 이름을 활용하여 성공적으로 추가가 되었다는 것을 출력한다. except문을 활용하여 잘못된 입력이 들어올 경우 에러 메시지를 출력하고 이전 메뉴로 돌아가게 된다.

6-2. def deletePlaylist()

```
Playlist ID: 5
BestOfBest delete completed!
```

```
Playlist ID: a
Error occurred while deleting playlist (1064, "You have an error in your SQL syntax; che
ck the manual that corresponds to your MariaDB server version for the right syntax to us
e near 'UID='4'' at line 1")
```

플레이리스트 메인 화면에서 2번을 입력하면 나오는 함수이다.

```
SELECT NAME FROM playlist WHERE ID=%s AND UID=%s
DELETE FROM belong_to_playlist_music WHERE PID=%s
DELETE FROM playlist WHERE ID=%s
```

플레이리스트를 삭제하기 위해 플레이리스트 ID를 입력 받고 SELECT문을 활용하여 삭제할 playlist tuple을 출력한다. 그리고 삭제를 하기 위해 DELETE문을 활용하여 입력한 ID와 동일한 ID를 갖는 tuple을 belong_to_music 테이블에서 삭제하면서 플레이리스트와 연결된 음악들의 연결을 해제하고 또 해당 playlist tuple도 삭제한다. except문을 활용하여 잘못된 입력이 들어올 경우

에러 메시지를 출력하고 이전 메뉴로 돌아가게 된다.

6-3. def addToPlaylist(a)

```
Music ID: 1
Playlist ID: 6
Adding Lucky to Best1 completed!
```

```
Music ID: 1
Playlist ID:4
Error occurred while adding music to playlist (1062, "Duplicate entry '1-4' for key 'PRI
MARY'")
```

플레이리스트 메인 화면에서 3번을 입력하면 나오는 함수이다.

```
INSERT INTO belong_to_playlist_music(MID, PID) VALUES (%s,%s)
SELECT music.NAME, playlist.NAME
FROM music, playlist where music.ID=%s AND playlist.ID=%s
```

플레이리스트에 음악을 추가하기 위해 만들기 위해 음악과 플레이리스트의 ID를 입력 받고 INSERT문을 활용하여 새롭게 tuple을 만든다. 그리고 추가가 되었다는 결과가 출력되게 하기 위해 SELECT문을 활용하여 입력한 ID와 동일한 ID를 갖는 belong_to_playlist_music tuple의 이름을 활용하여 성공적으로 추가가 되었다는 것을 출력한다. except문을 활용하여 잘못된 입력이 들어올 경우 에러 메시지를 출력하고 이전 메뉴로 돌아가게 된다.

6-4. def deleteFromPlaylist(a)

```
Music ID: 1
Playlist ID:6
Deleting Lucky from Best1 completed!
```

```
Music ID: a
Playlist ID:a
Error occurred while deleting music from playlist
```

플레이리스트 메인 화면에서 4번을 입력하면 나오는 함수이다.

```
SELECT music.NAME, playlist.NAME
FROM music, playlist, belong_to_playlist_music
where music.ID=belong_to_playlist_music.MID
AND playlist.ID=belong_to_playlist_music.PID
AND belong_to_playlist_music.MID=%s
AND belong_to_playlist_music.PID=%s
DELETE FROM belong_to_playlist_music WHERE MID=%s AND PID=%s
```

플레이리스트에서 음악을 삭제하기 위해 음악과 플레이리스트의 ID를 입력 받고 SELECT문을 활용하여 belong_to_playlist_music 테이블에서 그 관계에 있는 음악과 플레이리스트의 이름을 보여준다. 그리고 DELETE문을 활용하여 입력한 ID와 동일한 ID를 갖는 belong_to_playlist_music tuple

을 삭제한다. except문을 활용하여 잘못된 입력이 들어올 경우 에러 메시지를 출력하고 이전 메뉴로 돌아가게 된다.

6-5. def viewMyPlaylists(a)

```
FUNSONG ID: 2
Best Songs ID: 4
Best1 ID: 6
```

플레이리스트 메인 화면에서 5번을 입력하면 나오는 함수이다.

```
SELECT playlist.NAME, playlist.ID FROM playlist WHERE playlist.UID=%s
```

현재 사용자가 가지고 있는 플레이리스트 목록을 보여준다. 앞서 호출되는 함수에서 로그인한 사용자의 ID를 인자로 넘겨받아 현재 함수가 호출되면 SELECT문을 활용하여 현재 사용자의 ID와 playlist의 UID가 같은 플레이리스트와 플레이리스트의 ID만 출력시킨다.

6-6. def viewPlaylists()

```
ParkRoot : RPlayList ID: 1
LeeJongBeom : FUNSONG ID: 2
Andy : MAROON5 ID: 3
LeeJongBeom : Best Songs ID: 4
LeeJongBeom : Best1 ID: 6
```

플레이리스트 메인 화면에서 6번을 입력하면 나오는 함수이다.

```
SELECT user.NAME, playlist.NAME, playlist.ID
FROM user, playlist where user.ID=playlist.UID
```

모든 사용자가 가지고 있는 플레이리스트를 볼 수 있는 함수로, SELECT문을 이용하여 사용자의 이름과 플레이리스트의 이름 그리고 플레이리스트의 ID를 볼 수 있다.

6-7. def viewDetailPlaylists()

```
Playlist ID: 4
Lucky Strike
Memories
Knock Knock
```

플레이리스트 메인 화면에서 7번을 입력하면 나오는 함수이다.

```
SELECT music.NAME FROM music, playlist, belong_to_playlist_music
where music.ID=belong_to_playlist_music.MID
AND playlist.ID=belong_to_playlist_music.PID
AND belong_to_playlist_music.PID=%s
```

모든 사용자가 가지고 있는 플레이리스트 중 특정 플레이리스트를 검색하여 그 플레이리스트에 수록된 곡들의 목록을 SELECT문을 통해 볼 수 있다.

7. def preferenceMenu(a)

```
0. Return to Previous Menu
1. Preferred Artist
2. Preferred Genre
3. Preferred Music
Input:
```

로그인 후 사용자 메인 화면에서 2번을 입력하면 나오는 함수이다. 각 입력에 맞추어 선호하는 아티스트, 장르, 음악을 추가, 삭제, 또는 볼 수 있는 역할을 하는 함수가 실행된다.

7-1. def preferredArtist(a)

```
0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Artist ID:2
Twice add completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 2
Artist ID: 3
BLACKPINK delete completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 3
Twice
```

```
0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Artist ID:100
Error occurred while adding artist (1452, 'Cannot add or update a child row: a foreign k
ey constraint fails (`music_app`.`preferred_artist`, CONSTRAINT `FK__artist_2` FOREIGN K
EY (`ARTISTID`) REFERENCES `artist` (`ID`))')
```

preferenceMenu 화면에서 1번을 입력하면 나오는 함수이다. 1, 2, 3번의 입력에서 잘못된 입력이

있을 경우 except문을 통한 예외처리로 에러 메시지를 출력한다.

```
INSERT INTO preferred_artist(UID, ARTISTID) VALUES (%s, %s)
SELECT NAME FROM artist where ID=%s
```

이 함수에서 1번을 입력하여 artist의 ID를 입력하면 처음 함수 호출 때 받았던 사용자의 ID 인자의 정보와 함께 선호하는 가수가 INSERT문을 통해 preferred_artist 테이블에 추가된다. 그리고 SELECT문을 통해 artist가 성공적으로 추가되었음을 출력한다.

```
SELECT NAME FROM artist WHERE ID=%s
DELETE FROM preferred_artist WHERE ARTISTID=%s
```

이 함수에서 2번을 입력하면 선호하는 가수를 preferred_artist 테이블에서 삭제하기 위해 가수의 ID를 입력받고 이것을 SELECT문을 활용하여 preferred_artist 테이블에서 그 관계에 있는 아티스트의 이름을 보여준다. 그리고 DELETE문을 활용하여 입력한 ID와 동일한 ARTISTID를 갖는 preferred_artist tuple을 삭제한다.

```
SELECT artist.NAME FROM artist, preferred_artist
WHERE preferred_artist.UID=%s AND artist.ID=preferred_artist.ARTISTID
```

이 함수에서 3번을 입력하면 함수 호출 때 받았던 인자를 사용하여 현재 사용자가 선호하는 가수를 모두 볼 수 있게 된다.

7-2. def preferredGenre(a)

```
0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Genre Name:Pop
Pop add completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 2
Genre Name: Pop
Pop delete completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 3
Hip-hop

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Genre Name:a
Error occurred while adding genre (1452, 'Cannot add or update a child row: a foreign ke
y constraint fails (`music_app`.`preferred_genre`, CONSTRAINT `FK__genre_3` FOREIGN KEY
(`GNAME`) REFERENCES `genre` (`NAME`))')
```

preferenceMenu 화면에서 2번을 입력하면 나오는 함수이다. 1, 2, 3번의 입력에서 잘못된 입력이 있을 경우 except문을 통한 예외처리로 에러 메시지를 출력한다.

```
INSERT INTO preferred_genre(UID, GNAME) VALUES (%s, %s)
SELECT NAME FROM genre where NAME=%s
```

이 함수에서 1번을 입력하여 genre의 NAME을 입력하면 처음 함수 호출 때 받았던 사용자의 ID 인자의 정보와 함께 선호하는 장르가 INSERT문을 통해 preferred_genre 테이블에 추가된다. 그리고 SELECT문을 통해 genre가 성공적으로 추가되었음을 출력한다.

```
SELECT NAME FROM genre WHERE NAME=%s
DELETE FROM preferred_genre WHERE GNAME=%s
```

이 함수에서 2번을 입력하면 선호하는 장르를 preferred_genre 테이블에서 삭제하기 위해 장르의 이름을 입력받고 이것을 SELECT문을 활용하여 preferred_genre 테이블에서 그 관계에 있는 장르의 이름을 보여준다. 그리고 DELETE문을 활용하여 입력한 NAME과 동일한 GNAME을 갖는 preferred_genre tuple을 삭제한다.

```
SELECT NAME FROM genre, preferred_genre WHERE preferred_genre.UID=%s
AND genre.NAME=preferred_genre.GNAME
```

이 함수에서 3번을 입력하면 함수 호출 때 받았던 인자를 사용하여 현재 사용자가 선호하는 장르를 모두 볼 수 있게 된다.

7-3. def preferredMusic(a)

```
0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Music ID:2
Memories add completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 2
Music ID: 1
Lucky Strike delete completed!

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 3
Memories

0. Return to Previous Menu
1. Add
2. Delete
3. View
Input: 1
Music ID:2
Error occurred while adding music (1062, "Duplicate entry '3-2' for key 'PRIMARY'")
```

preferenceMenu 화면에서 3번을 입력하면 나오는 함수이다. 1, 2, 3번의 입력에서 잘못된 입력이 있을 경우 except문을 통한 예외처리로 에러 메시지를 출력한다.

```
INSERT INTO preferred_music(MID, UID) VALUES (%s, %s)
SELECT NAME FROM music where ID=%s
```

이 함수에서 1번을 입력하여 music의 ID를 입력하면 처음 함수 호출 때 받았던 사용자의 ID 인자의 정보와 함께 선호하는 음악이 INSERT문을 통해 preferred_music 테이블에 추가된다. 그리고 SELECT문을 통해 음악이 성공적으로 추가되었음을 출력한다.

```
SELECT NAME FROM music WHERE ID=%s
DELETE FROM preferred_music WHERE MID=%s
```

이 함수에서 2번을 입력하면 선호하는 음악을 preferred_music 테이블에서 삭제하기 위해 음악의 ID를 입력받고 이것을 SELECT문을 활용하여 preferred_music 테이블에서 그 관계에 있는 음악의 이름을 보여준다. 그리고 DELETE문을 활용하여 입력한 ID와 동일한 MID를 갖는 preferred_music tuple을 삭제한다.

```
SELECT NAME FROM music, preferred_music WHERE preferred_music.UID=%s
AND music.ID=preferred_music.MID
```

이 함수에서 3번을 입력하면 함수 호출 때 받았던 인자를 사용하여 현재 사용자가 선호하는 음악을 모두 볼 수 있게 된다.

8. def viewMusic()

```
0. Return to Previous Menu
1. Recent Heard
2. Mostly Heard
3. Listen to Music
4. Search Music
5. Show Music Details
Input:
```

음악에 관한 함수를 호출하기 위한 메뉴화면 역할을 하는 함수이다. realUserMenu()에서 3번을 입력했을 경우 호출된다.

0번을 입력하면 전 단계의 함수가 호출되고 1번을 입력하면 자신이 가장 최근에 들은 음악의 정보를 가져오는 함수를 호출, 2번을 입력하면 사용자들이 가장 많이 들은 음악을 호출, 3번을 입력하면 음악을 듣게 되는 함수를 호출, 4번을 입력하면 특정 음악을 검색하여 정보를 보고, 5번을 입력하면 모든 음악의 정보를 볼 수 있는 함수를 호출한다.

8-1. def recentHeard(a)

```
Lucky Strike Date: 2019-12-12
```

음악에 관한 메뉴 함수에서 1번을 입력할 경우 호출되는 함수로, DB상의 음악 중에 현재 사용자가 가장 최근에 들은 음악을 나타낸다.

```
SELECT music.name, heard.date
FROM music, heard where heard.UID=%s AND heard.MID=music.ID
order by heard.date DESC
```

함수 호출 때 받았던 사용자 ID 인자를 사용하여 사용자가 가장 최근 들은 음악을 SELECT문을 통해 heard 테이블의 UID 중 사용자의 ID와 같고, MID와 같은 ID를 가지고 있는 음악이 있는 heard tuple의 정보를 바탕으로 가장 최근 날짜의 음악을 order by를 통해 찾고 음악의 이름과 들은 날짜를 출력한다.

8-2. def mostlyHeard()

```
Lucky Strike Heard Times: 2
```

음악에 관한 메뉴 함수에서 2번을 입력할 경우 호출되는 함수로, DB상의 음악 중에 가장 많이 들은 음악을 나타낸다.

```
SELECT heard.MID, COUNT(*) FROM heard GROUP BY heard.MID
SELECT music.NAME FROM music where music.ID=%s
```

먼저 SELECT문을 통해 count 함수를 이용해서 가장 가장 많이 들은 음악의 ID를 heard 테이블에서 구한 후, 별도의 변수를 선언해 가장 많이 들은 음악의 ID와 그 횟수를 담아둔다. 그리고 SELECT문을 한번 더 사용하여 count 함수를 통해 찾은 heard의 MID와 같은 music의 ID와 일치하는 music의 이름을 출력시키고 그 음악의 들은 횟수를 출력시킨다.

8-3. def listenToMusic(a)

```
Music ID: 2
Date:2019-12-11
Memories listen completed!
```

```
Music ID: 3
Date:2019-12-12
Error occurred while listening to music (1062, "Duplicate entry '4-3' for key 'PRIMARY'
")
```

음악에 관한 메뉴 함수에서 3번을 입력할 경우 호출되는 함수로, 원하는 음악을 들은 상태로 DB상에 기록된다.

```
INSERT INTO heard(UID, MID, Date) VALUES (%s,%s,%s)
SELECT NAME FROM music where music.ID=%s
```

인자로 사용자의 ID를 넘겨받고 입력으로 음악의 ID와 날짜를 받아서 INSERT문을 통해 heard tuple을 하나 생성한다. 그리고 SELECT문을 통해 입력 받았던 음악의 ID와 같은 ID를 가지는 tuple의 이름을 출력해서 성공적으로 들었음을 보여준다. except문을 통해 잘못된 입력은 예외를 출력한다.

8-4. def searchMusic()

```
Music Name:Lucky Strike
Name: Lucky Strike Info: sdf Music ID: 1 Artist: Maroon5 Lyrics: Kim Composer: Park Albu
m: V
```

```
Music Name:Lucky
No music found!
```

음악에 관한 메뉴 함수에서 4번을 입력할 경우 호출되는 함수로, DB상의 음악 중에 특정 음악에 대한 정보를 출력한다.

```
SELECT music.NAME, INFO, music.ID, artist.NAME, LYRICS, COMPOSE, album.name
FROM music, album, artist WHERE music.NAME=%s
AND music.ALBUMID=album.ID AND music.ARTISTID=artist.ID
```

입력으로 음악의 이름을 넘겨받으며, SELECT문을 통해 해당 이름을 가지고 있는 음악의 기본 정보와, artist와 album의 ID 중 music이 foreign key로 가지고 있는 ID의 이름들을 출력시킨다.

8-5. def musicDetails()

```
Name: Lucky Strike Info: sdf Music ID: 1 Artist: Maroon5 Lyrics: Kim Composer: Park Albu
m: V
Name: Memories Info: Maroon Music ID: 2 Artist: Maroon5 Lyrics: Adam Levine Composer: Ad
am Levine Album: V
Name: Knock Knock Info: Dancing Music ID: 3 Artist: Twice Lyrics: Twice Composer: Twice
Album: V
```

음악에 관한 메뉴 함수에서 5번을 입력할 경우 호출되는 함수로, DB상의 모든 음악의 정보를 출력시킨다.

```
SELECT music.NAME, INFO, music.ID, artist.NAME, LYRICS, COMPOSE, album.name
FROM music, album, artist WHERE music.ALBUMID=album.ID
AND music.ARTISTID=artist.ID
```

DB 상의 모든 음악들의 정보를 SELECT문을 artist와 album의 ID 중 music이 foreign key로 가지고 있는 ID의 이름들을 포함해서 출력시킨다.