ITE 2038 – Database Application:
Final Project – Music Streaming Service

2019014266 Lim Kyu Min

Contents:

I. Changes Applied on Current Project

1. 기능상의 변경점

I. 하나의 유저는 여러 개의 계정을 가질 수 있습니다 → 하나의 유저는 **단 한 개의 계정만** 가질 수 있습니다.

- 여러 개의 계정을 생성해도 얻을 수 있는 이점이 많지 않다고 판단했습니다. 다른 기능을 완성도 있게 만드는 것이 더욱 낫다고 판단하여 변경했습니다.

II. 신규가입시 요구되는 유저 정보로는 주민등록번호, 이름, 나이, 이메일과 주소가 있습니다 → 신규가입시 유저 정보로는 **주민등록번호, 이름, 나이와 이메일이** 있습니다

- 가입 시 너무 많은 정보를 요구한다는 생각이 들어, 해당 스트리밍 서비스에서 가장 사용이 적은 정보인 주소를 요구 정보에서 제거하기로 결정했습니다.

2. 데이터베이스 상의 변경점

I. Profile, Music_mostplayed와 같은 View가 추가되었습니다.

- 해당 View들을 통해 더욱 유연한 DB쿼리가 가능해졌습니다.

II. Database 테이블들의 이름이 변경되었습니다.

- 자세한 변경사항은 II. Database Description을 참고하시길 바랍니다.

III. ALBUM 테이블과 Ab_listed 테이블을 제거했습니다.

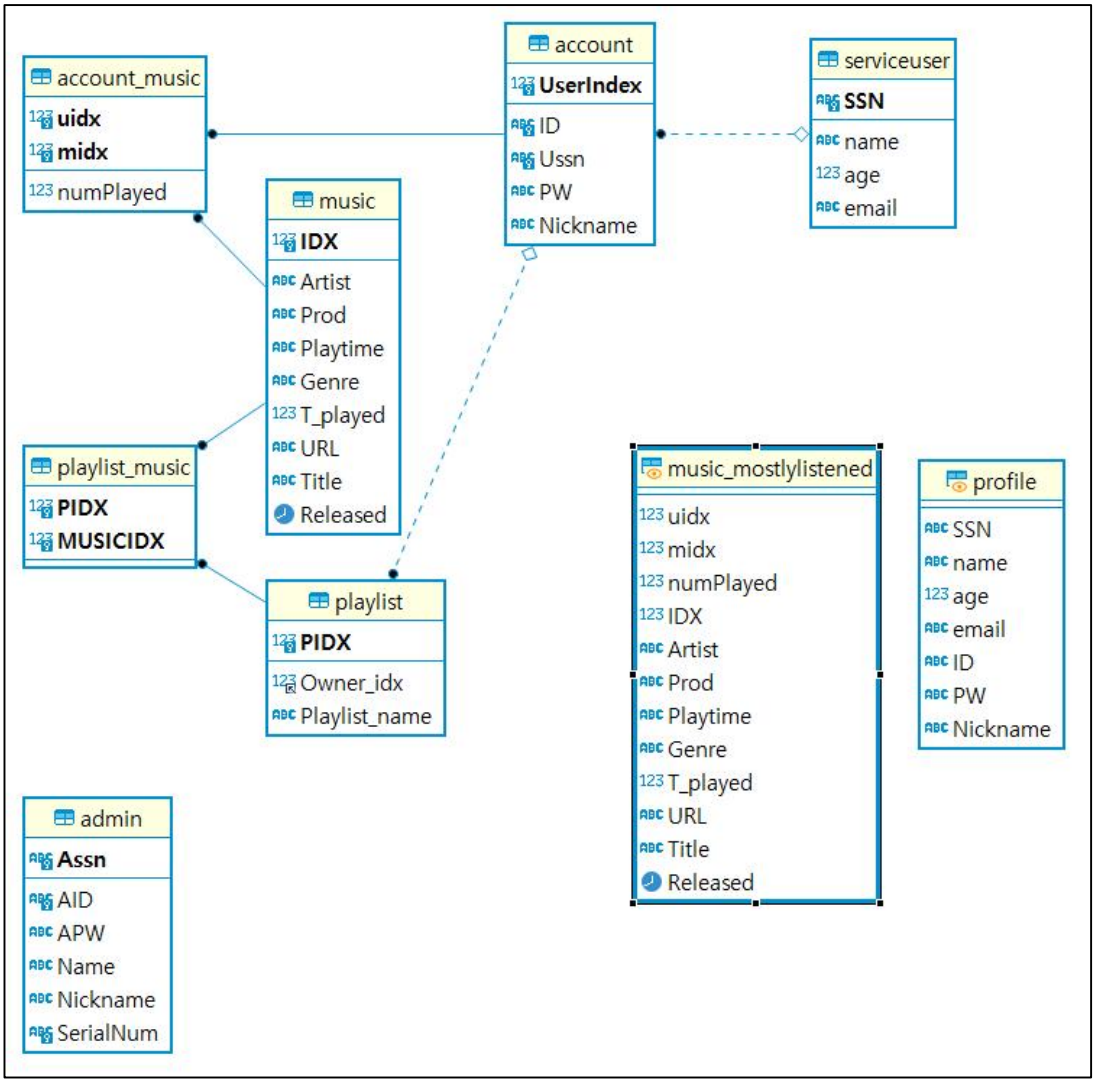
- 앨범은 Music 테이블에서 관리가 가능하다고 판단하여 제거했습니다.

IV. MGR_MUSIC과 MGR_USER 테이블을 제거했습니다.

- 메니저가 음악과 유저를 관리하는 것은 기능의 일이지, 데이터베이스로 관리할 일이 아니라고 판단하여 제거했습니다.

II. Database Description (DBProj)

Entity – Relationship Diagram



Database Analysis

servicesuser (Table)

```
CREATE TABLE `serviceuser` (
  `SSN` char(14) NOT NULL,
  `name` varchar(30) NOT NULL,
  `age` int(11) NOT NULL,
  `email` varchar(100) NOT NULL DEFAULT
'UNIDENTIFIED',
  PRIMARY KEY (`SSN`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Primary Key: SSN

> 유저 정보를 저장하는 테이블입니다. 유저의 주민등록번호, 이름, 나이와 이메일 주소를 저장합니다. 스트리밍 서비스의 원활한 기능 작동과 유연한 쿼리를 위해 계정과 분리하여 따로 테이블에 저장했습니다.

account (Table)

```
CREATE TABLE `account` (
  `UserIndex` int(11) NOT NULL
```

Primary Key: UserIndex

```
AUTO_INCREMENT,
  `ID` varchar(30) NOT NULL,
  `Ussn` char(14) NOT NULL,
  `PW` varchar(50) NOT NULL,
  `Nickname` varchar(30) NOT NULL,
  PRIMARY KEY (`UserIndex`),
  UNIQUE KEY `ID` (`ID`),
  UNIQUE KEY `UC_USSN` (`Ussn`),
  KEY `USSN_FK` (`Ussn`),
  CONSTRAINT `USSN_FK` FOREIGN KEY
(`Ussn`) REFERENCES `serviceuser`
(`SSN`) ON DELETE CASCADE ON UPDATE
CASCADE
    ) ENGINE=InnoDB AUTO_INCREMENT=8
          DEFAULT CHARSET=utf8
```

Foreign Key: Ussn → serviceuser(SSN)

> 유저의 계정 정보를 저장하는 테이블입니다. 유저의 아이디와 비밀번호, 그리고 닉네임을 저장합니다.
> 각 유저들은 신규유저마다 1씩 증가하는 index번호를 부여받으며, 스트리밍 서비스와 DB는 유저의 계정들을 해당 인덱스 번호로 관리합니다.

music (Table)

```
CREATE TABLE `music` (
  `IDX` int(11) NOT NULL
AUTO_INCREMENT,
  `Artist` varchar(50) NOT NULL DEFAULT
'ANONYMOUS',
  `Prod` varchar(50) NOT NULL DEFAULT
'ANONYNOUS',
  `Playtime` varchar(10) NOT NULL,
  `Genre` varchar(20) NOT NULL DEFAULT
'UNDEFINED',
  `T_played` int(11) NOT NULL DEFAULT
0,
  `URL` varchar(100) NOT NULL,
  `Title` varchar(50) NOT NULL,
  `Released` date NOT NULL,
  PRIMARY KEY (`IDX`)
   ) ENGINE=InnoDB AUTO_INCREMENT=16
          DEFAULT CHARSET=utf8
```

Primary Key: IDX

> 음악의 정보를 저장하는 테이블입니다. 음악의 이름, 아티스트, 프로듀서, 플레이타임, 장르, 발매일과 같은 기본적인 정보를 저장합니다.
> music 테이블은 총 플레이 횟수 또한 저장합니다(T_Played). 이 수치는 이후에 가장 많이 플레이된 곡들을 순위별로 나열하거나, 가장 인기있는 곡을 선정할 때 사용됩니다.
> 모든 음악은 신규 음악이 추가될 때마다 1씩 증가하는 index번호를 받으며, 스트리밍 서비스와 DB는 음악을 해당 인덱스 번호로 관리합니다.

account_music (Table)

```
CREATE TABLE `account_music` (
  `uidx` int(11) NOT NULL,
  `midx` int(11) NOT NULL,
  `numPlayed` int(11) NOT NULL DEFAULT
0,
  PRIMARY KEY (`uidx`,`midx`),
  KEY `AMMIDX_FK` (`midx`),
  CONSTRAINT `AMMIDX_FK` FOREIGN KEY
(`midx`) REFERENCES `music` (`IDX`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `AMUIDX_FK` FOREIGN KEY
(`uidx`) REFERENCES `account`
(`UserIndex`) ON DELETE CASCADE ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Primary Key: uidx, midx
Foreign Key: uidx → account(UserIndex), midx → music(IDX)

> 어떤 계정이 어떤 음악을 얼마나 많이 들었는지 그 횟수를 저장합니다(numPlayed).
> 해당 계정이 새로운 음악을 들을 때마다 그 정보가 tuple로 추가되며, 같은 음악을 들을 때마다 해당 Tuple의 numPlayed 값이 증가합니다.
> 해당 정보는 각 계정이 가장 많이 들은 노래의 장르를 분석해 같은 장르를 추천하는, 일명 '음악 추천 기능'에 사용이 됩니다.

playlist (Table)

```
CREATE TABLE `playlist` (
  `PIDX` int(11) NOT NULL
AUTO_INCREMENT,
  `Owner_idx` int(11) NOT NULL,
  `Playlist_name` varchar(50) NOT NULL,
  PRIMARY KEY (`PIDX`),
```

Primary Key: PIDX
Foreign Key: Owner_idx → account(UserIndex)

> 플레이리스트의 정보를 저장하는 테이블입니다. 플레이리스트의 이름과 해당 플레이리스트를 소유하고 있는 계정 등의 정보를 저장하고 있습니다.

`KEY PLIST_FK (Owner_idx),`
`CONSTRAINT PLIST_FK FOREIGN KEY (Owner_idx) REFERENCES account (UserIndex) ON DELETE CASCADE ON UPDATE CASCADE`
`) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8`
	> 기본적으로 플레이리스트의 이름은 중복을 허용하나, 스트리밍 서비스에서 같은 계정에서 같은 플레이리스트 이름을 사용하는 것은 금지하고 있습니다.
	> 모든 플레이리스트는 플레이리스트가 추가될 때마다 1씩 증가하는 index번호를 받으며, 스트리밍 서비스와 DB는 플레이리스트들을 해당 인덱스 번호로 관리합니다.

playlist_music(Table)	

`CREATE TABLE playlist_music (`
`PIDX int(11) NOT NULL,`
`MUSICIDX int(11) NOT NULL,`
`PRIMARY KEY (PIDX,MUSICIDX),`
`KEY MIDX_FK (MUSICIDX),`
`CONSTRAINT MIDX_FK FOREIGN KEY (MUSICIDX) REFERENCES music (IDX) ON DELETE CASCADE ON UPDATE CASCADE,`
`CONSTRAINT PIDX_FK FOREIGN KEY (PIDX) REFERENCES playlist (PIDX) ON DELETE CASCADE ON UPDATE CASCADE`
`) ENGINE=InnoDB DEFAULT CHARSET=utf8`
	Primary Key: PIDX, MUSICIDX
	Foreign Key: MUSICIDX → music(IDX), PIDX → playlist(PIDX)
	
	> 플레이리스트에 저장된 음악 정보를 저장하는 테이블입니다.
	> 플레이리스트에 음악이 추가될 때마다 그 음악의 인덱스 정보와 추가된 플레이리스트의 인덱스 정보가 함께 이 테이블에 저장이 됩니다.

admin(Table)	

`CREATE TABLE admin (`
`Assn char(14) NOT NULL,`
`AID varchar(30) NOT NULL,`
`APW varchar(50) NOT NULL,`
`Name varchar(30) NOT NULL,`
`Nickname varchar(30) NOT NULL,`
`SerialNum char(4) NOT NULL,`
`PRIMARY KEY (Assn),`
`UNIQUE KEY AID (AID),`
`UNIQUE KEY UNI_SERIAL (SerialNum)`
`) ENGINE=InnoDB DEFAULT CHARSET=utf8`
	Primary Key: Assn
	
	> 관리자의 정보를 저장하는 테이블입니다.
	> 관리자의 주민등록번호, 아이디, 비밀번호, 이름, 닉네임과 고유번호를 저장하여 관리자의 데이터를 관리합니다.

profile(View)	

`create view dbproj.profile as`
`select`
`s.SSN as SSN,`
`s.name as name,`
`s.age as age,`
`s.email as email,`
`a.ID as ID,`
`a.PW as PW,`
`a.Nickname as Nickname`
`from`
`(dbproj.serviceuser s`
`join dbproj.account a on`
`(s.SSN = a.Ussn))`
	View: serviceuser + account
	
	> serviceuser와 account의 테이블을 조인하여 필요한 column만을 가져온 View입니다.
	> 유저의 프로필 전체를 쿼리하거나, 정보를 화면에 보여줘야할 때 사용됩니다.

music_mostlylistened(View)	

`Create view dbproj.music_mostlylistened as (`
`Select *`
`From (dbproj.account_music am`
`join dbproj.music m on`
`(am.midx = m.IDX)))`
	View: music + account_music
	
	> music와 account_music의 테이블을 조인하여 필요한 column만을 가져온 View입니다.
	> 추천 음악 정보를 가져오거나, 가장 인기 있는 음악 정보를 효율적으로 쿼리할 때 사용됩니다.

III. Code Analysis

Project: MusicStreamingService.java

* 함수에 사용되는 자세한 쿼리문은 V. Used SQL Queries를 참고하시길 바랍니다.

I. Main.java (src/com/company/Main.java)	
com/company package에 존재하는 클래스입니다. 해당 서비스의 메인 화면을 담당합니다. 유저들이 가장 처음 맞이하는 화면이며, 해당 화면에서는 로그인, 가입, 그리고 어드민 메뉴로의 접속이 가능합니다. 0을 입력하면 스트리밍 서비스를 종료합니다.	
`Connection connection = DatabaseHandler.getConnect();`	해당 함수로 기존에 존재하는 MariaDB의 지정된 데이터베이스로 연결됩니다. 자세한 함수 설명은 DatabaseHandler.java를 참고하시길 바랍니다.
`UserFunction.signIn(connection);`	유저가 1번을 선택했을 때, 로그인 화면으로 넘어가게 됩니다. 자세한 것은 UserFunction.java를 참고하시길 바랍니다.
`UserFunction.register(connection);`	유저가 2번을 선택했을 때, 가입 화면으로 넘어가게 됩니다. 자세한 것은 UserFunction.java를 참고하시길 바랍니다.
`AdminFunc.AdminAuth(connection)`	유저가 3번을 선택했을 때, 관리자 계정으로 로그인 하기 위한 화면으로 넘어가게 됩니다. 자세한 것은 AdminFunc.java를 참고하시길 바랍니다.

II. DatabaseHandler.java (src/dbpackage/ DatabaseHandler.java)	
dbpackage package에 존재하는 클래스입니다. 데이터베이스의 접근과 연결 (Connection), 그리고 이름 정보를 저장하기 위해 사용됩니다. DatabaseHandler는 데이터베이스 접근을 위한 기본 정보를 제공하며, 테이블과 뷰의 이름과 같은 쿼리를 위한 필수 정보를 저장합니다.	
`public static Connection getConnect()`	MariaDB로의 접근을 위한 함수입니다. 해당 함수를 통해 데이터베이스와의 연결이 시작됩니다.
`DriverManager.getConnection ("jdbc:mysql://"+SERVER_HOST+"/"+ DATABASE_NAME,USERNAME,PASSWORD)`	해당 MariaDB로 연결을 하고, 성공 시 그에 접근할 수 있는 Connection을 return합니다.
`public final static String…`	테이블과 뷰의 이름과 같은 쿼리를 위한 필수 정보들을 나타냅니다.

III. GeneralQuery.java (src/dbpackage/ GeneralQuery.java)	
dbpackage package에 존재하는 클래스입니다. 스트리밍 서비스의 대부분의 쿼리를 담당합니다. General Select, General Insert와 같이 범용적으로 사용할 수 있는 쿼리문들을 저장하고 있습니다.	

`public static ResultSet generalCheck(Connection con, String colNames, String from)`
 	
`public static ResultSet generalCheck(Connection con, String colNames, String from, String condition)`
	Select 쿼리문을 통해 데이터의 유무를 체크하고, 쿼리된 데이터를 가져올 때 범용적으로 사용되는 함수입니다. 쿼리 결과를 ResultSet으로 반환합니다.

`public static ResultSet generalCheckByOrder(Connection con, String colNames, String from, String orderCondition, String limit)`	데이터의 순서를 정하고, 그 중 상위 Tuple들을 가져올 때 범용적으로 사용되는 함수입니다. 주로 순위를 정할 때 사용됩니다. 쿼리 결과를 ResultSet으로 반환합니다.

`public static boolean generalUpdate(Connection con, String tableName, String setValue, String condition)`	Update 쿼리문을 통해 데이터를 변경할 때 범용적으로 사용되는 함수입니다. 주로 유저나 관리자가 데이터를 변경할 때 사용되며, 결과가 성공적이면 true를 반환합니다.

`public static boolean generalInsert(Connection con, String tableName, String values)`
 	
`public static boolean generalInsert(Connection con, String tableName, String columns, String values)`
	Insert 쿼리문을 통해 데이터를 추가할 때 범용적으로 사용되는 함수입니다. 주로 유저나 (신규 가입 혹은 플레이 리스트 음악 추가 등) 관리자가 데이터를 추가할 때 사용되며, 결과가 성공적이면 true를 반환합니다.

`public static void generalDelete(Connection con, String tableName, String condition)`	Delete 쿼리문을 통해 데이터를 삭제할 때 범용적으로 사용되는 함수입니다. 주로 유저나 (회원 탈퇴 및 플레이리스트 음악 삭제 등) 관리자가 데이터를 삭제할 때 사용됩니다.

IV. DatabaseQuery.java (src/dbpackage/ DatabaseQuery.java)	
dbpackage package에 존재하는 클래스입니다. 스트리밍 서비스에 필요한 쿼리 중, 세부적으로 추가 옵션이 필요하거나, 민감한 작업으로 인해 따로 쿼리를 진행해야 하는 함수들을 저장하고 있습니다.	
`public static ResultSet findUser (Connection con, String ssn)`	각 유저를 구분하는 주민등록번호 (SSN)으로 유저유 찾을 때 사용됩니다. 유저 그 자체에 대한 프로필을 쿼리하거나, 해당 유저의 중복 가입 여부를 체크하는데에 사용이 됩니다.

`public static ResultSet findAccount (Connection con, String colNames, String ssn)`
 	
`public static ResultSet findAccount (Connection con, String colNames, String ID, String PW, String SerialNum)`
	계정 정보를 찾을 때 사용이 됩니다. 유저와 관리자의 계정 모두를 담당하고 있으며, 관리자의 경우 추가로 관리자 고유의 Serial Number를 요구합니다.

`public static ResultSet findProfile (Connection con, String ID)`	유저의 프로필을 쿼리할 때 사용됩니다.

`public static ResultSet checkAccount(Connection con,String colNames, String id)`
 	
`public static ResultSet checkAccount(Connection con,String colNames, String id, String password)`
	유저의 계정을 찾을 때 사용됩니다. 혹은 유저의 새로 등록되는 계정이 중복되었는지 체크할 때에 사용됩니다. 기능별로 아이디만 찾거나, 아이디와 비밀번호 모두를 찾는 걸로 구분이 되어 있습니다.

`public static void insertNewUser(Connection con, ArrayList<String> userInfo)`
 	
`public static void insertNewAccount(Connection con, ArrayList<String> accountInfo, String ssn)`
	각각 새로운 유저와 계정을 추가할 때 사용이 됩니다.

`public static void updateProfile(Connection con, String from, String target, String info, String ssn)`
 	
`public static void updateProfile(Connection con, String from, String target, int info, String ssn)`
	유저가 자신의 정보 (이름, 아이디, 비밀번호 등등)을 변경할 때 사용됩니다. 범용적 Update함수를 사용합니다.

V. User.java (src/ServiceFunc/ User.java)
ServiceFunc Package에 존재하는 클래스입니다. 데이터베이스에서 불러온 유저 정보를 로컬에 보관하

는 역할을 하며, 유저 정보를 얻기 위한 getter들이 모여있습니다.	
public String getSSN() public String getName() public int getAge() …	각 유저의 필요한 정보를 제공해주는 getter들입니다.

VI. MusicAttribute.java (src/ServiceFunc/ MusicAttribute.java)	
ServiceFunc Package에 존재하는 클래스입니다. 데이터베이스에서 불러온 음악 정보를 로컬에 보관하는 역할을 하며, 음악 정보를 얻기 위한 getter들이 모여있습니다.	
public String getInfoType() public int getIndex() public String getAttribute()	각 음악의 필요한 정보를 제공해주는 getter들입니다.
public static List<MusicAttribute> getGeneralTypes() public static List<MusicAttribute> getInputTypes() public static List<MusicAttribute> getAllTypes()	Enumeration으로 이루어져있는 음악 정보들 중, 필요한 정보만을 가져와 Database에서 쿼리할 수 있도록 해주는 getter들입니다.

VII. UserPage.java (src/dbpackage/ UserPage.java)	
ServiceFunc Package에 존재하는 클래스입니다. 유저의 화면을 담당하며, 음악 스트리밍 서비스의 대부분의 기능을 담고 있고, 코드 역시 600줄 이상의 방대한 분량을 보여줍니다. 유저는 여기서 음악을 듣고, 자신의 계정을 관리하고, 플레이리스트를 관리하며, 계정 탈퇴를 할 수 있습니다. * 너무 함수가 많은 관계로, 주요한 함수 몇 가지만 소개합니다.	
private static void inputUserPageMenu(Connection con, ArrayList<String> userInfo) throws SQLException	유저의 메인 화면입니다. 유저는 여기서 다음과 같은 선택을 할 수 있습니다. 1. 음악 찾기 2. 플레이리스트 보기 3. 프로필/계정 관리 0. 로그아웃
private static void searchMusic(Connection con, Integer userIndex) throws SQLException	유저가 1. 음악 찾기를 선택하면 음악 선택창이 나옵니다. 유저는 여기서 다음과 같은 선택을 할 수 있습니다. 1. 모든 음악 리스트 보기 2. 추천 음악 보기 3. 제목으로 음악 찾기 4. 아티스트 이름으로 음악 찾기 5. 장르별 음악 찾기 6. 인덱스로 직접 음악 찾기 0. 뒤로 돌아가기
private static void playMusic(Connection con, ResultSet rs, Integer userIndex) throws SQLException private static void playMusic(Connection con, Integer userIndex) throws SQLException	음악을 재생할 때 사용되는 함수입니다.
private static void recommendationMenu(Connection con,int userIndex)	유저가 1 - 2. 추천 음악 보기를 선택하면 나오는 화면입니다. 유저는 다른 플레이어들에게 가장 인기있는 음악을 보거나, 아님 자신이 가장 많이 들

	었던 장르의 음악을 추천 받을 수 있습니다.
`private static void playListMenu(Connection con, int userIndex) throws SQLException`	유저가 2. 플레이 리스트 보기를 선택하면 나오는 화면입니다. 유저는 여기서 다음과 같은 선택을 할 수 있습니다. 1. 플레이 리스트 생성 2. 플레이 리스트 선택 0. 뒤로 돌아가기
`private static void playListOptions(Connection con, int playlistIndex, int userIndex)`	유저가 2-2 플레이 리스트 선택을 하면 나오는 화면입니다. 유저는 여기서 다음과 같은 선택을 할 수 있습니다. 1. 플레이리스트에 음악 추가하기 2. 플레이리스트에 있는 음악 재생하기 3. 플레이리스트에 있는 음악 삭제하기 4. 플레이리스트 이름 바꾸기 5. 플레이리스트 삭제하기 0. 뒤로 돌아가기
`private static boolean userProfile(Connection con, ArrayList<String> userInfo) throws SQLException`	유저가 3. 계정/프로필 관리를 선택했을 때 나오는 화면입니다. 유저는 여기서 다음과 같은 선택을 할 수 있습니다. 1. 프로필 보기 2. 개인정보 수정하기 (이름과 나이 등) 3. 이메일 정보 수정하기 4. 비밀번호 변경하기 5. 닉네임 변경하기 6. 회원 탈퇴하기 0. 뒤로 돌아가기
`private static boolean signOut()`	로그아웃시 사용되는 함수입니다.

VIII. UserFunction.java (src/ServiceFunc/ UserFunction.java)	
ServiceFunc Package에 존재하는 클래스입니다. UserPage와 연계하여 유저가 할 수 있는 기능들을 모아 놓은 클래스입니다. * 너무 함수가 많은 관계로, 주요한 함수 몇 가지만 소개합니다.	

`private static void registerNew(Connection con) throws SQLException`
 	
`private static void registerAccount(Connection con, String ssn)`
	유저의 신규 계정 가입에 사용되는 2개의 함수입니다. 각각 새로운 유저 정보와 새로운 유저의 계정을 담당합니다. 각각의 함수 모두 중복성 검사를 하여 계정이나 유저 정보가 중복될 시에 신규 가입을 막습니다.

`public static void signIn(Connection con)`	유저의 로그인을 담당하는 기능입니다. 유저는 다음과 같은 세부 선택을 할 수 있습니다. 1. 로그인하기 2. 계정 찾기 (아이디 혹은 비밀번호) 0. 돌아가기
`public static ArrayList<ResultSet> searchMusic(Connection con, String searchFor, String target)`	음악과 플레이리스트 관련된 함수들입니다. 음악을 찾거나, 플레이리스트 관련 기능에 필요한 기능들

public static ArrayList<ResultSet> searchPlaylist(Connection con, int idx) …	을 담당합니다.
public static void playMusic(Connection con, String musicIdx, Integer userIndex)	Javafx를 활용하여 로컬에 저장되어 있는 음악을 재생합니다. 로컬에 존재하는 URL을 데이터베이스에서 불러오며, 해당 노래를 javafx에서 재생합니다. 해당 javafx는 이후에 나오는 MusicPlayer.java에서 담당합니다.

IX. UserAccount.java (src/ServiceFunc/ UserAccount.java)	
ServiceFunc Package에 존재하는 클래스입니다. 유저의 계정을 찾는 데에 필요한 기능들이 나열되어 있습니다.	
public static void findAccount(Connection con)…	유저의 계정을 찾습니다. 유저는 아이디, 혹은 비밀번호를 일련의 과정을 거쳐 찾을 수 있습니다.

X. AdminFunc.java (src/AdminFunc/ AdminFunc.java)	
AdminFunc Package에 존재하는 클래스입니다. 관리자 기능을 실행하기에 앞서, 관리자인지 확인하는 인증기능과, 관리자의 대표 기능인 '모든 유저 보기' 기능을 가지고 있습니다.	
public static void AdminAuth(Connection con)	관리자인지를 확인하는 기능입니다. 관리자의 권한을 얻고 관리자 페이지로 들어가기 위해서는 다음 정보를 가지고 있어야 합니다. 1. 관리자 아이디 2. 관리자 비밀번호 3. 관리자 고유번호 (Serial Number)
public static ResultSet queryAllUser(Connection con)	현재 존재하는 모든 유저의 정보를 Query합니다.

XI. AdminPage.java (src/AdminFunc/ AdminPage.java)	
AdminFunc Package에 존재하는 클래스입니다. 관리자의 화면을 담당하며, 관리자가 스트리밍 서비스에서 할 수 있는 기능들을 보여주는 인터페이스를 담당합니다. 관리자는 여기서 유저를 관리하거나, 음악을 관리하는 등의 일을 할 수 있습니다. * 너무 함수가 많은 관계로, 주요한 함수 몇 가지만 소개합니다.	
public static void openingAdminPage(Connection con)	관리자의 메인 화면입니다. 관리자는 다음과 같은 옵션을 선택할 수 있습니다. 1. 유저 관리 메뉴 2. 음악 관리 메뉴 0. 돌아가기 (관리자 화면에서 나가기)
private static void openUserOption(Connection con)	관리자가 1. 유저 관리 메뉴를 선택했을 시 실행됩니다. 관리자는 유저 관리 메뉴에서 다음과 같은 기능을 실행할 수 있습니다. 1. 모든 유저 정보 가져오기 2. 특정 유저 정보 가져오기 3. 유저 닉네임 바꾸기 (부적절한 닉네임의 경우)

	4. 유저 삭제하기 0. 뒤로 돌아가기
`private static void openMusicOption(Connection con)`	관리자가 2. 음악 관리 메뉴를 선택했을 시 실행됩니다. 관리자는 음악 관리 메뉴에서 다음과 같은 기능을 실행할 수 있습니다. 1. 모든 음악 리스트 보기 2. 특정 음악 찾기 (제목으로) 3. 음악 추가하기 4. 음악 정보 수정하기 5. 음악 삭제하기 0. 돌아가기

XII. MusicPlayer.java (src/MediaFunc/ MusicPlayer.java)	
MediaFunc Package에 존재하는 클래스입니다. Javafx 라이브러리를 활용하였으며, 사용자가 재생하는 음악을 실제로 재생하여 음악이 나오게 하는 역할을 수행합니다.	
`public void initialize(String url)`	음악 재생을 위한 필수 정보들을 load하여 음악 재생 준비를 합니다.

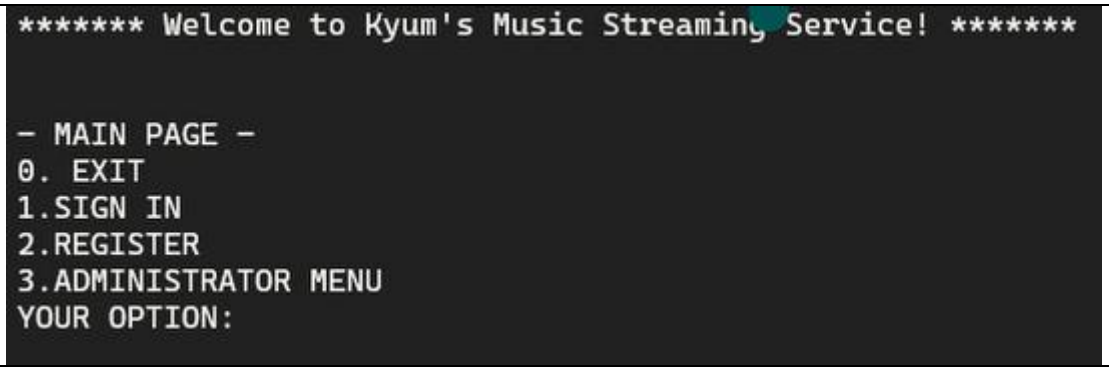
`public void play(boolean curState)`
`public void stop()`
`public void reload()`
	각각 음악을 재생 (다시 실행하면 일시정지), 중지, 처음부터 다시 재생 기능을 실행합니다.

XIII. StringUtils.java (src/Utils/ StringUtils.java)	
Utils Package에 존재하는 클래스입니다. 스트리밍 서비스는 CLI기반이어서 integer value, String value등에 민감하며, 각각의 입력 관련 예외에 약합니다. 이러한 단점을 효율적으로 보완하기 위해 만들어진 클래스입니다.	
`public static int parse(String input)`	입력을 받은 것을 숫자로 변환합니다. 대부분 유저나 관리자의 옵션값을 Integer value로 변환하는 역할을 하며, 수 이외의 정보가 입력 시, 오류 메세지와 함께 -1을 반환합니다.
`public static String stringTrim(String input)` `private static String stringReplace(String input)`	띄어쓰기 (일명 whitespace)와 ‘’나 “”같은 문자들은 SQL 쿼리시에 치명적입니다. 이러한 토큰들을 제거하는 역할을 합니다.

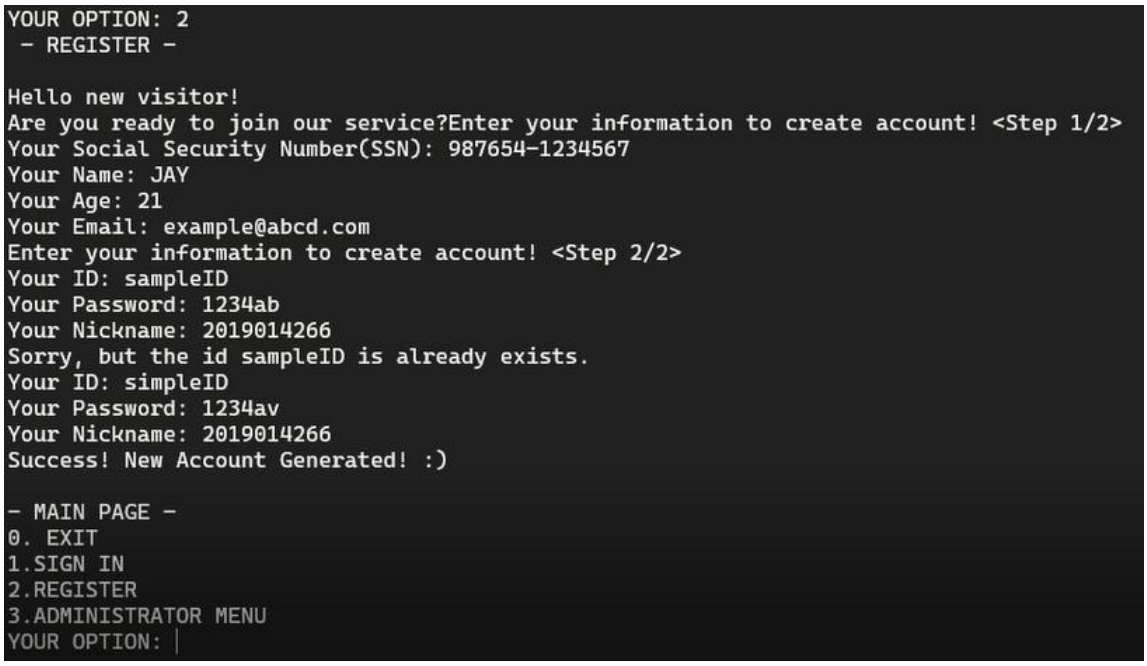
IV. Service Demonstration

* 구현된 서비스 중 일부를 스크린샷으로 시연합니다. 더욱 자세한 시연은 함께 첨부된 동영상을 참고해주시기를 바랍니다. *"MusicStreamingService 시연영상.mp4"*

Screenshots



Main Page



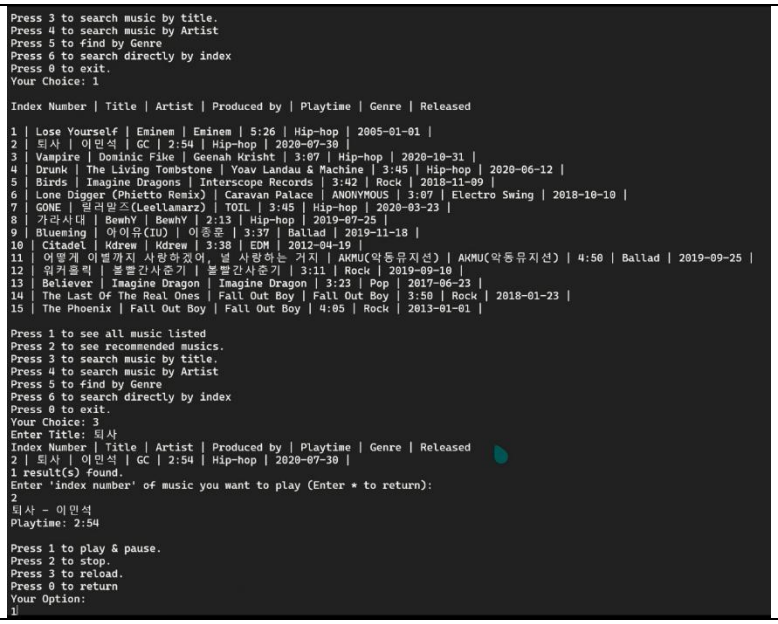
Register (Preventing Duplicate ID)

```
- MAIN PAGE -
0. EXIT
1.SIGN IN
2.REGISTER
3.ADMINISTRATOR MENU
YOUR OPTION: 1
 - SIGN IN -

Press 1 to sign in.
Press 2 to find my account
Press 0 to return (Go to main menu).
Your Option: 2

- FIND MY ACCOUNT -
0. RETURN
1.FIND MY ID
2.FIND MY PASSWORD
YOUR OPTION:
1
Find my ID- Enter your Social Security Number (SSN): 987654-1234567
ID Found: simpleID

- FIND MY ACCOUNT -
0. RETURN
1.FIND MY ID
2.FIND MY PASSWORD
YOUR OPTION:
2
Find my password- Enter your Social Security Number (SSN):
987654-1234567
Enter your ID:
simpleID
Password Found: 1234av
```

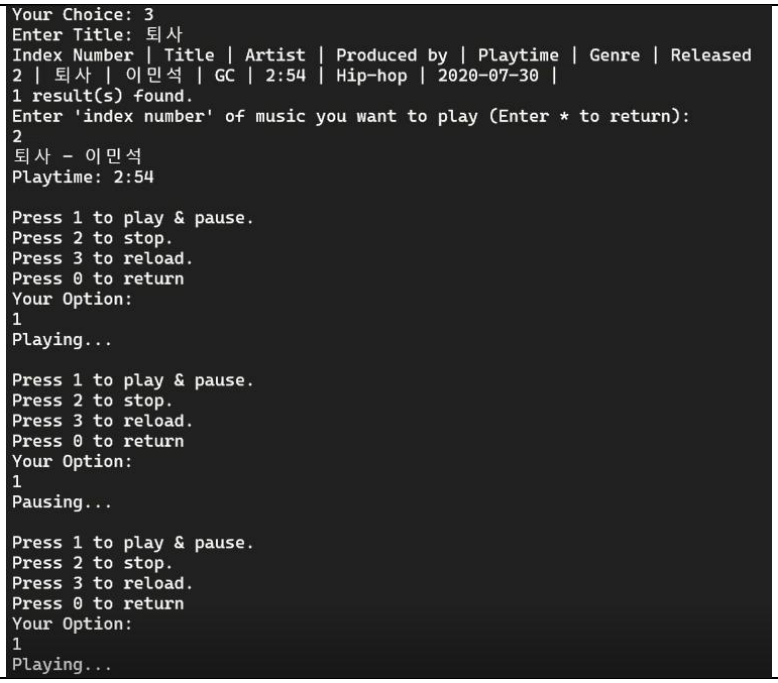
Finding Account

```
Press 1 to sign in.
Press 2 to find my account
Press 0 to return (Go to main menu).
Your Option: 1

Your ID (Enter * to return): simpleID
Your Password (Enter * to return): 1234av
Signing in...

WELCOME! 2019014266

-2019014266'(s) Page-
Press 1 to search music.
Press 2 to see my playlist
Press 3 to configure your profile.
press 0 to sign out
Your Option:
```

Login



Searching Music



Playing Music

See Top 1 ~ 5 popular Musics

```
Press 1 to see Mostly played music by other users
Press 2 to see Kyum's pick
Press 0 to return.
2
Kyum is analyzing your favorite....

A-ha! You like Hip-hop musics! Then, I will show some related musics....

Index Number | Title | Artist | Produced by | Playtime | Genre | Released
1 | Lose Yourself | Eminem | Eminem | 5:26 | Hip-hop | 2005-01-01 |
2 | 퇴사 | 이민석 | GC | 2:54 | Hip-hop | 2020-07-30 |
3 | Vampire | Dominic Fike | Geenah Krisht | 3:07 | Hip-hop | 2020-10-31 |
4 | Drunk | The Living Tombstone | Yoav Landau & Machine | 3:45 | Hip-hop | 2020-06-12 |
7 | GONE | 릴러말즈(Leellamarz) | TOIL | 3:45 | Hip-hop | 2020-03-23 |
8 | 가라사대 | BewhY | BewhY | 2:13 | Hip-hop | 2019-07-25 |
6 result(s) found.
Enter 'index number' of music you want to play (Enter * to return):
*
Return to music search page...
```

Get recommended by streaming service

```
Playlist 'my playlist'
Number of Music(s): 0


Options:
Press 1 to add music
Press 2 to play music
Press 3 to delete music
Press 4 to change playlist name
Press 5 to delete current playlist
Press 0 to Return
1
Write an index number of a music:
6
Added!

Playlist 'my playlist'
Number of Music(s): 1

6 | Lone Digger (Phietto Remix) | Caravan Palace | ANONYMOUS | 3:07 | Electro Swing | 2018-10-10 |

Options:
```

Playlist Options

```
-2019014266'(s) Page-
Press 1 to search music.
Press 2 to see my playlist
Press 3 to configure your profile.
press 0 to sign out
Your Option: 3

PROFILE OPTIONS:
Press 1 to see my profile.
Press 2 to change personal information(name,age).
Press 3 tp change email address.
Press 4 to change Password.
Press 5 to change Nickname.
Press 6 to Leave an account
Press 0 to return.
Your Choice: 1

- PROFILE -
SSN: 987654-1234567
name: JAY
age: 21
email: example@abcd.com
id: simpleID
password: 1234av
nickname: 2019014266
```

Profile Options

```
-2019014266'(s) Page-
Press 1 to search music.
Press 2 to see my playlist
Press 3 to configure your profile.
press 0 to sign out
Your Option: 3

PROFILE OPTIONS:
Press 1 to see my profile.
Press 2 to change personal information(name,age).
Press 3 tp change email address.
Press 4 to change Password.
Press 5 to change Nickname.
Press 6 to Leave an account
Press 0 to return.
Your Choice: 5
Your new Nickname: JAYJAY
Your new Nickname is JAYJAY. Am I right? (Enter y to say Yes): y

Change Applied!


-JAYJAY'(s) Page-
Press 1 to search music.
Press 2 to see my playlist
Press 3 to configure your profile.
press 0 to sign out
Your Option: 0
Are you Sure? (Enter Yes or y to confirm.)y
Bye!!
```

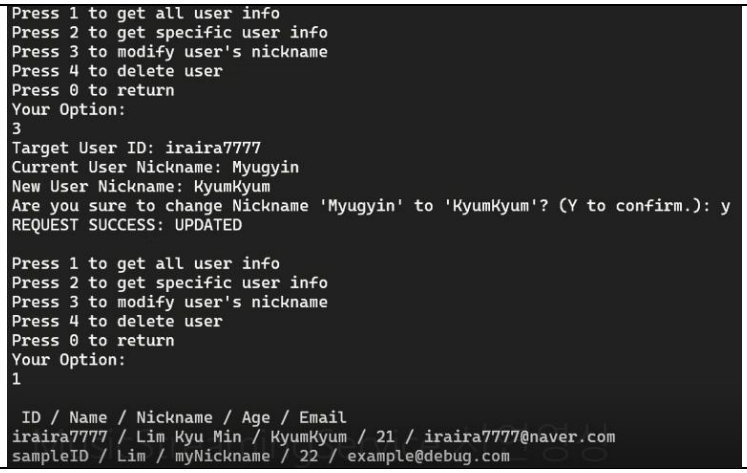
Profile Options – Change Nicknames

```
- MAIN PAGE -
0. EXIT
1.SIGN IN
2.REGISTER
3.ADMINISTRATOR MENU
YOUR OPTION: 3

- ADMINISTRATOR MENU -
AUTHENTICATION REQUIRED
Your ID (Enter * to return): adminDebug
Your Password (Enter * to return): 1234
Your Serial Code (Enter * to return): 3224

RESULT: AUTHORIZATION SUCCEED
OPENING THE ADMIN PAGE...

WELCOME MGR_JAY!

OPTIONS:
Press 1 to user menu
Press 2 to music menu
Press 0 to Close the Admin Page.
```

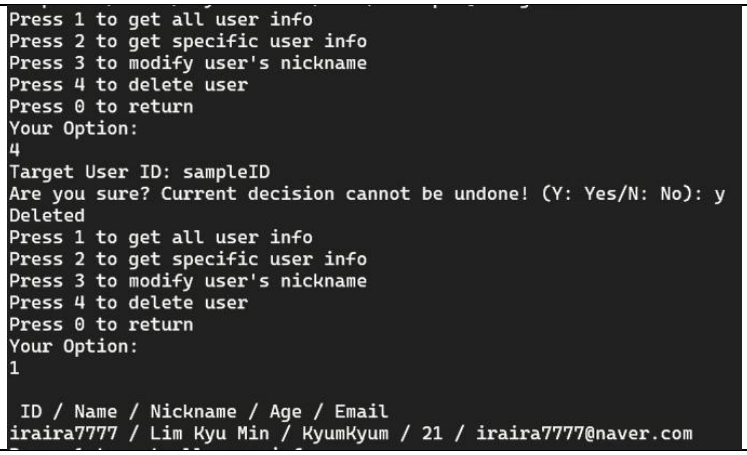
Administrator Page - Get Authorization

```
WELCOME MGR_JAY!

OPTIONS:
Press 1 to user menu
Press 2 to music menu
Press 0 to Close the Admin Page.
Your Option: 1
Press 1 to get all user info
Press 2 to get specific user info
Press 3 to modify user's nickname
Press 4 to delete user
Press 0 to return
Your Option:
```

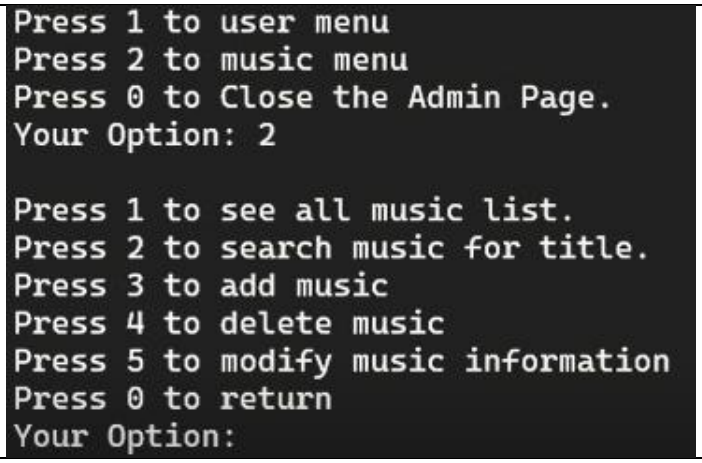
Administrator Page – User Menu



Administrator Page - User Menu: Change Nickname



Administrator Page - User Menu: Delete User



Administrator Page – Music Menu

V. Used SQL Queries

I. 범용적 SQL (60%의 SQL문은 해당 GenralQuery에서 해결이 됩니다.)

1. GeneralQuery.generalCheck()

```
SELECT [COLUMN NAME] FROM [TABLE NAME];

SELECT [COLUMN NAME] FROM [TABLE NAME] WHERE [CONDITIONS];
```

GeneralCheck에서 범용적 Select문의 구조입니다. 특정 정보를 쿼리할 때 많이 사용되며, 특히 중복성 여부를 체크할 때 가장 범용적으로 사용됩니다.

2. GeneralQuery.generalCheckByOrder()

```
SELECT [COLUMN NAME] FROM [TABLE NAME] ORDER BY [ORDERING CONDITION] LIMIT [QUERY LIMITS];

Ex) SELECT * FROM music ORDER BY T_played DESC, Title ASC LIMIT 0,5;
```

GeneralCheckByOrder에서 사용되는 쿼리문의 구조입니다. [ORDERING CONDITION]으로 쿼리 결과들을 정렬하고, [QUERY LIMITS]까지의 결과를 반환합니다.

3. GeneralQuery.generalUpdate()

```
UPDATE [TABLE NAME] SET [CHANGE VALUE] WHERE [TARGET CONDITION];
```

GeneralUpdate에서 사용되는 쿼리문의 구조입니다. 전형적인 Update문의 구조를 띄고 있습니다

4. GeneralQuery.generalInsert()

```
INSERT INTO [TABLE NAME] VALUES([VALUES]);

INSERT INTO [TABLE NAME]([SPECIFIED COLUMNS]) VALUES([VALUES]);
```

GeneralInsert에 사용되는 쿼리문의 구조입니다. 전형적인 INSERT문의 구조이며, 추가로 [SPECIFIED COLUMNS]을 통해 자신이 Insert하고자 하는 컬럼을 명시할 수 있습니다.

5. GeneralQuery.generalDelete()

```
DELETE FROM [TABLE NAME] WHERE [CONDITION];
```

GeneralDELETE에 사용되는 쿼리문의 구조입니다. 전형적인 DELETE문의 구조를 띄고 있습니다.

II. 특수 SQL (특별한 기능을 수행하기 위해서 만들어졌습니다. 설명이 존재합니다.)

6. DatabaseQuery.findUser()

```
SELECT * FROM serviceuser WHERE SSN = [TARGET SSN];
```

유저의 정보를 저장하고 있는 serviceuser 테이블에서 SSN을 통해 유저 정보 전체를 가져와야 할 때 사용됩니다.

7. DatabaseQuery.findAccount()

```
SELECT [TARGER COLUMNS] FROM account WHERE Ussn = [TARGET SSN];

SELECT [TARGER COLUMNS] FROM admin WHERE AID = [TARGET ADMIN ID] AND APW = [TARGET ADMIN
PW]  AND SerialNum =[TARGET ADMIN SERIAL NUMBER];
```

유저의 계정을 저장하고 있는 Account나 관리자의 계정을 저장하고 있는 Admin에서 계정계 쿼리할 때 사용됩니다. 관리자는 추가로 관리자 아이디와 비밀번호, 그리고 고유 번호를 요구합니다.

7. DatabaseQuery.findProfile()

```
SELECT * from profile where ID = [TARGET ID]
```

Serviceuser와 account를 join한 View인 profile에서 프로필 정보를 가져올 때 사용됩니다.

8. DatabaseQuery.checkAccount()

```
SELECT [TARGET COLUMNS] FROM account WHERE ID = [TARGET ID];
SELECT [TARGET COLUMNS]  FROM account WHERE ID = [TARGET ID] AND PW = [TARGET PW];
```

account에서 유저의 계정을 도로 찾거나, 계정이 존재하는지 유무를 판단하기 위해 사용됩니다.

9.DatabaseQuery. insertNewUser()

```
INSERT INTO serviceuser VALUES ([SSN],[NAME],[AGE],[EMAIL]);
```

10.DatabaseQuery. insertNewAccount()

```
INSERT INTO account(ID,PW,Nickname,Ussn) VALUES ([ID],[PW],[NICKNAME],[SSN])
```

유저가 새로 가입할 시, 그 데이터를 테이블에 넣기 위해 사용됩니다.

11.UserPage.listMostPlayed()

```
SELECT * FROM music WHERE T_Played = (SELECT MAX(T_played) FROM music);
```

전체적으로 가장 많이 플레이 된 음악의 정보를 쿼리합니다.

12.UserPage.recommendMusic()

```
SELECT Genre FROM music_mostlylistened WHERE uidx = [USER INDEX] AND numPlayed = (SELECT
MAX(numPlayed) FROM account_music where uidx = [USER INDEX]LIMIT 0,1);
```

유저가 가장 많이 들은 장르 정보를 쿼리할 때 사용됩니다. music_mostlylistened는 유저가 각 노래를 들은 수를 저장하는 account_music과 music 테이블을 join한 view입니다. music_mostlylistened에서 유저의 인덱스 정보로 쿼리하고 (Tuple중 해당 유저와 관련된 데이터만 쿼리), account_music에서 해당 유저가 가장 많이 들은 횟수의 노래 인덱스 정보를 같이 쿼리 하여 Genre정보를 쿼리합니다.

13.UserFunction.checkPLMusicNum()

```
SELECT COUNT(*) FROM playlist_music WHERE PIDX = [User Pidx];
```

14.UserFunction.checkPlaylistDuplicate ()

```
SELECT * FROM playlist WHERE PIDX = [User Pidx] AND Playlist_name = [PL NAME];
```

플레이리스트의 중복성 여부를 판단합니다. 기본적으로 플레이리스트의 이름은 중복될 수 있지만, 한 사람이 같은 이름의 플레이리스트를 2개 이상 가질 수는 없습니다.