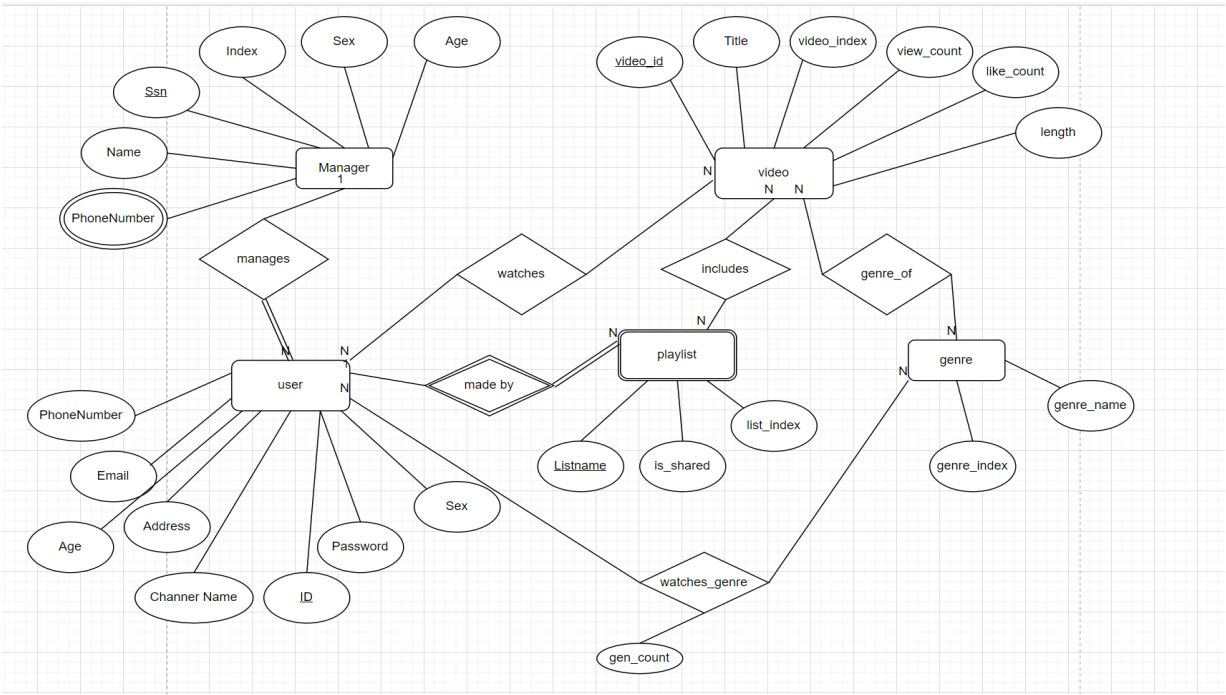
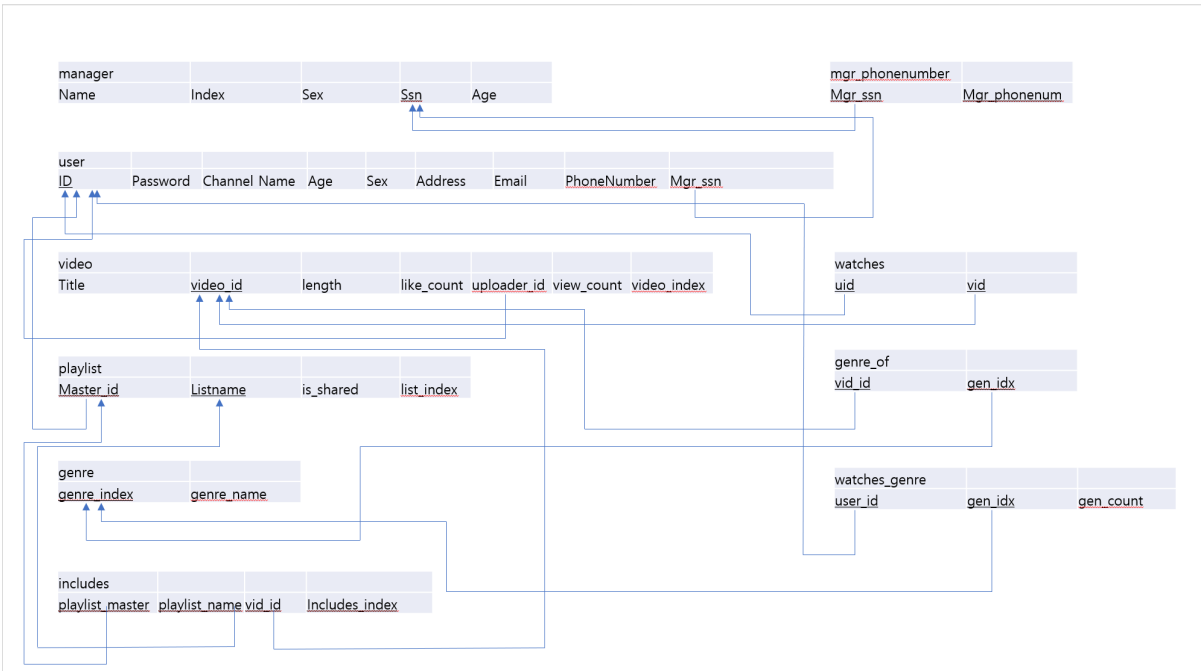
프로젝트 2: DBMS 프로그램 개발

컴퓨터 소프트웨어학부

2019049716 안재국

0. ER-Diagram과 Relational model





1. manager

매니저 entity들에 대한 정보를 저장하는 테이블입니다.

2. user

사용자 entity들에 대한 정보를 저장하는 테이블입니다.

3. video

영상 entity들에 대한 정보를 저장하는 테이블입니다.

4. playlist

플레이리스트 entity들에 대한 정보를 저장하는 테이블입니다.

5. genre

장르 entity들에 대한 정보를 저장하는 테이블입니다.

6. mgr_phonenumber

매니저들은 여러개의 phonenumber를 가질 수 있기 때문에 새로운 테이블을 만들어 매니저의 전화번호를 저장했습니다. 이때 매니저 테이블의 PK인 Ssn을 새로운 테이블의 FK로 가져왔습니다.

7. watches

유저와 비디오 사이의 watches라는 관계를 저장하기 위한 테이블입니다. user의 PK인 ID와 video의 PK인 video_id를 FK로 가져와서 PK로 사용합니다.

8. genre_of

비디오와 장르 사이의 genre_of 관계를 저장하기 위한 테이블입니다. genre의 PK인 genre_index와 video의 PK인 video_id를 FK로 가져와서 사용합니다.

9. watches_genre

유저와 장르 사이의 watches_genre 관계를 저장하기 위한 테이블입니다. user의 PK인 ID와 genre의 PK인 genre_index를 FK로 가져와 사용하며 relation attribute인 gen_count를 테이블의 일반 attribute로 가져옵니다.

10. includes

플레이 리스트와 비디오 사이의 includes 관계를 저장하기 위한 테이블입니다. playlist의 PK인 {master_id, Listname}과 video의 PK인 video_id를 FK로 가져와서 사용합니다.

프로젝트 1. 요구사항 분석에 비해 추가된 점.

1. 인터페이스에서 원하는 영상이나 리스트 등을 선택할 때, 0부터 시작하는 인덱스를 보고 고를 수 있도록 video와 playlist에 index attribute를 추가하였습니다.

Video에서의 index는 모든 영상을 구분 가능한 candidate key가 되며, playlist에서는 weak key가 됩니다.

2. manager table에 있던 '관리하는 이용자의 수' attribute는 굳이 따로 기록하지 않아도 추론될 수 있는 값이기 때문에 삭제했습니다.

3. user table에 있던 '업로드한 영상의 수' attribute 역시 relation을 통해 추론될 수 있는 값이기 때문에 삭제했습니다.

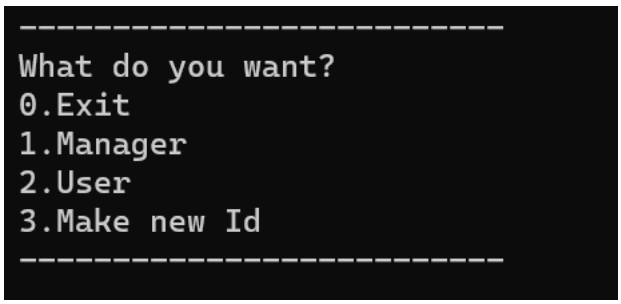
4. video table에 있던 '업로드 날짜' attribute는 사용하지 않게 되어 삭제했습니다.

5. user table에 있던 '구독자 수' attribute는 구독 기능을 구현하지 않게 되어 삭제했습니다.

6. playlist table에서 어떤 영상을 담고 있는지 나타내기 위해 넣어두었던 '영상 고유번호(video_id)' attribute는 관계에 의해 표현된다고 생각하여 삭제했습니다.

1. 프로그램의 기능과 인터페이스 소개

- 매니저 모드



먼저 처음 실행 시 유저가 하고싶은 작업을 물어봅니다.

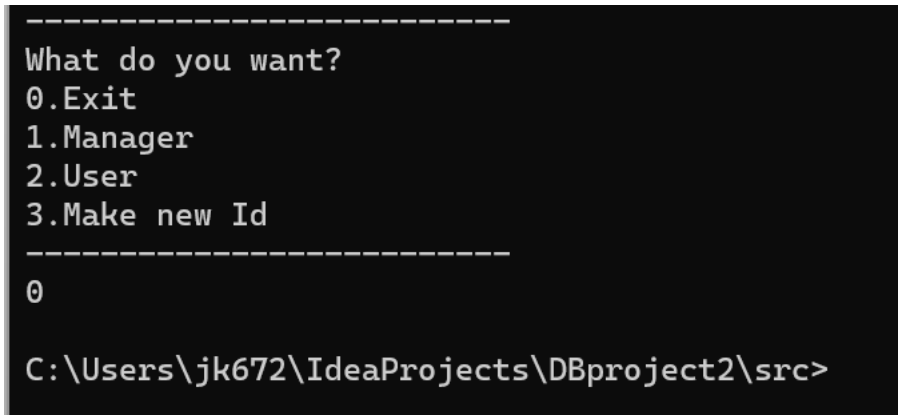
0. 종료

1. 매니저 모드로 로그인

2. 유저 모드로 로그인

3. 새로운 아이디 만들기(유저)

이때 매니저는 함부로 아이디를 만들 수 없다고 생각하여 유저용 아이디만 만들 수 있도록 하였습니다.



0을 입력 시 바로 종료하는 것을 볼 수 있습니다.

```
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
1
Tell me your name
Jaeguk
Please tell me your Ssn 'Jaeguk'
12345
Welcome 'Jaeguk'!
-----------------------------
0.delete video
1.view user's information
2.view user's viewing history
3.view videos uploaded by user
4.quit
-----------------------------
```

1번을 통해 매니저 모드로 로그인을 하면

매니저는 이름과 Ssn으로 로그인이 가능합니다.

매니저가 사용할 수 있는 기능은 4개로

0. 전체 영상 목록에서 원하는 영상을 하나 삭제합니다.

1. 자신이 관리하는 유저들에 대한 정보를 볼 수 있습니다.

2. 자신이 관리하는 유저들의 시청기록을 볼 수 있습니다.

3. 자신이 관리하는 유저가 업로드한 영상을 유저별로 볼 수 있습니다.

4. 로그아웃 합니다.

```
1
=============================================================================================================
<User List>
user 0. ID : jk6722, Age : 23, Sex : M, Address : Seoul, Email : jk6722@naver.com, Phone number : 01048036722, Mgr_ssn :
 12345
user 1. ID : John, Age : 21, Sex : M, Address : LA, Email : john@hanyang.ac.kr, Phone number : 01042032021, Mgr_ssn : 12
345
user 2. ID : movie, Age : 25, Sex : F, Address : Seoul, Email : movie@hanyang.ac.kr, Phone number : 01042123011, Mgr_ssn
 : 12345
=============================================================================================================
```

1번을 입력하면 해당 매니저가 관리하는 모든 유저들에 대한 정보를 출력하는 것을 볼 수 있습니다.

```
2
user 0. ID : jk6722
user 1. ID : John
user 2. ID : movie
enter user's ID you want to see history
John
=======================================
<history of user 'John'>
0. Video_id : 2, Title : Avengers, length : 1:22:58
=======================================
```

2번을 입력하면 현재 본인이 관리하고 있는 유저들의 ID를 보여줍니다.

그리고 시청기록을 보고싶은 유저의 ID를 입력하면 해당 유저의 시청목록을 보여줍니다.

```
3
=====================================
<List of user's ID you are managing>
0. jk6722
1. John
2. movie
=====================================
enter the ID of user you want to know
movie
------------------------------
<list of videos 'movie' uploaded>
0.  title : Titanic, length : 1:25:30, view : 0, genres : Romance,
1.  title : About time, length : 1:30:15, view : 0, genres : Romance,
2.  title : Avengers, length : 1:22:58, view : 1, genres : Action, Comedy,
3.  title : Spiderman, length : 1:15:43, view : 0, genres : Romance, Action, Comedy,
4.  title : Love Rosie, length : 1:40:01, view : 0, genres : Romance,
5.  title : Pororo, length : 48:20, view : 0, genres : Kids,
------------------------------
```

3번을 입력하면 마찬가지로 현재 본인이 관리하고 있는 유저들의 목록을 보여주며 업로드한 영상의 목록을 보고싶은 유저의 ID를 입력하면 해당 유저가 업로드한 영상의 목록을 보여줍니다.

```
-----------------------------
0.delete video
1.view user's information
2.view user's viewing history
3.view videos uploaded by user
4.quit
-----------------------------
4
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
```

4번을 입력하면 로그아웃하고 처음으로 돌아갑니다.

- 유저 모드

```
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
2
Please enter your id(Ex) 'jk6722')
jk6722
please enter your password
qmffhdn33!
-----------------------------
```

2번 유저모드로 로그인 시도 시

회원가입 시 입력했던 아이디와 비밀번호로 로그인을 하게 됩니다.

```
-----------------------------
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
-----------------------------
```

유저가 사용할 수 있는 기능은 로그아웃을 제외하고 9개가 있습니다.

0. 다른 사람이 업로드한 영상을 시청합니다.

1. 영상을 업로드합니다.

2. 자신만의 플레이 리스트를 만듭니다.

3. 자신의 플레이 리스트 안에 있는 영상을 시청합니다.

4. 자신이 가지고 있는 플레이 리스트의 목록을 확인합니다.

5. 플레이 리스트를 삭제합니다.

6. 특정 플레이 리스트에 영상을 추가하거나 삭제합니다.

7. 본인이 업로드한 영상을 삭제합니다.

8. 자신이 많이 시청한 장르의 순위를 확인합니다.

```
0
0. Title : Titanic, Channel : GoodMovie, view : 0 genre : Romance,
1. Title : About time, Channel : GoodMovie, view : 0 genre : Romance,
2. Title : Avengers, Channel : GoodMovie, view : 1 genre : Action, Comedy,
3. Title : Spiderman, Channel : GoodMovie, view : 0 genre : Romance, Action, Comedy,
4. Title : Love Rosie, Channel : GoodMovie, view : 0 genre : Romance,
5. Title : Pororo, Channel : GoodMovie, view : 0 genre : Kids,
6. Title : Chicken Mukbang, Channel : JohnTV, view : 0 genre : Mukbang,
What video do you want to see? tell me a number
1
You are watching video now...
Do you want to see more video?
0. yes
1. no
0
What video do you want to see? tell me a number
3
You are watching video now...
Do you want to see more video?
0. yes
1. no
1
```

0번을 입력하면 현재 업로드 되어 있는 모든 영상의 목록을 출력합니다.

그리고 자신이 보기를 원하는 영상의 인덱스 번호를 입력해서 영상을 볼 수 있습니다.

시청이 끝나면 영상을 더 볼 것인지 아니면 그만 볼 것인지 물어봅니다.

0을 입력하면 계속해서 다른 영상을 볼 수 있고

1을 입력하면 영상 시청을 종료합니다.

```
------------------------------
1
you should determine your channel name first
enter your own channel name to make
JaegukTV
Channel 'JaegukTV' is opened!
enter the title of your video(Don't use symbol ' for your title!)
MyFirstVideo
enter the length of your video(Ex) 1:25:05
45:50
enter all genres of your video(Ex) 1, 3, 5
------------------------------
0. action
1. romance
2. comedy
3. horror
4. eating show(mukbang)
5. talk show
6. kids
------------------------------
0, 1, 2
New video was uploaded.
------------------------------
```

1번을 입력하면 새로운 영상을 업로드할 수 있습니다.

만약 채널명 attribute가 null인 유저라면 영상을 올리기 전에 먼저 채널명을 정해야합니다.

채널이 개설되면 영상의 제목과 길이, 장르를 순서대로 입력하면 영상이 업로드 됩니다.

```
------------------------------
0
0. Title : Titanic, Channel : GoodMovie, view : 1 genre : Romance,
1. Title : About time, Channel : GoodMovie, view : 1 genre : Romance,
2. Title : Spiderman, Channel : GoodMovie, view : 3 genre : Romance, Action, Comedy,
3. Title : Love Rosie, Channel : GoodMovie, view : 0 genre : Romance,
4. Title : Pororo, Channel : GoodMovie, view : 0 genre : Kids,
5. Title : Chicken Mukbang, Channel : JohnTV, view : 0 genre : Mukbang,
6. Title : MyFirstVideo, Channel : JaegukTV, view : 0 genre : Romance, Action, Comedy,
What video do you want to see? tell me a number
```

영상을 업로드한 후 0번 명령을 통해 영상의 목록을 확인해 보면 새로 업로드한 영상이 올라와있는 것을 확인할 수 있습니다.

```
------------------------------
2
enter the Name of your new playlist(Don't use ' symbol)
Jaeguk List
Do you want to share your playlist to other people?
0. yes
1. no
0
New playlist was created
------------------------------
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
------------------------------
4
========================
0. Jaeguk List
========================
------------------------------
```

2번을 입력하면 새로운 플레이 리스트를 만들 수 있습니다.

먼저 만들 플레이 리스트의 이름을 입력합니다.

이때 제목에 '(따옴표)가 들어가게 되면 쿼리문에 영향을 주기 때문에 따옴표는 사용하지 않아야 합니다.

제목을 입력하면 그 다음 해당 플레이 리스트를 다른 사람도 볼 수 있도록 할 것인지의 여부를 물어봅니다.

이번 프로그램에서는 구현하지 않았지만 다른 사람의 플레이 리스트를 퍼오는 기능을 만들 때 사용할 수 있습니다.

그리고 4번 명령을 통해 현재 내 리스트를 확인해보면 새로 만든 플레이 리스트가 들어있는 것을 볼 수 있습니다.

```
3
============================
<your playlists>
0. Jaeguk List
============================
enter the number of list to enter
0
sorry there is no video in this playlist
```

플레이 리스트를 만들었다면 3번 명령을 통해 리스트 안에 있는 영상을 시청할 수 있습니다.

하지만 방금 만든 리스트이기 때문에 안에 영상이 없어서 이때는 리스트 안에 영상이 없다는 문구를 출력합니다.

```
------------------------------
6
========================
0. Jaeguk List
========================
enter number of list you want to modify
0
========================
<List of the video in Jaeguk List>
========================
0. add video to playlist
1. delete video from playlist
0
-------------------------
0.  title : Titanic, length : 1:25:30, view : 0, genres : Romance,
1.  title : About time, length : 1:30:15, view : 1, genres : Romance,
2.  title : Avengers, length : 1:22:58, view : 1, genres : Action, Comedy,
3.  title : Spiderman, length : 1:15:43, view : 1, genres : Romance, Action, Comedy,
4.  title : Love Rosie, length : 1:40:01, view : 0, genres : Romance,
5.  title : Pororo, length : 48:20, view : 0, genres : Kids,
6.  title : Chicken Mukbang, length : 20:15, view : 0, genres : Mukbang,
-------------------------
enter number of the video to add
3
New video was inserted into your playlist
```

그렇다면 먼저 리스트에 영상을 추가해보겠습니다.

6번 기능을 사용하면 리스트에 영상을 추가하거나 제거할 수 있습니다.

먼저 건드리고 싶은 리스트의 번호를 입력받습니다.

그리고 리스트에 영상을 추가하고 싶은 것인지 아니면 리스트에서 영상을 삭제하고 싶은 것인지를 물어봅니다. 추가하는 것이라면 0번을, 삭제하는 것이라면 1번을 입력 받습니다.

0번을 입력해서 영상을 추가하게 되면 현재 업로드 되어있는 모든 영상들을 출력합니다.

자신이 시청했던 시청목록에서 추가하는 것으로 처음 생각했으나 그렇게 하면 시청하지 않은 영

상은 리스트에 담을 수 없다는 단점이 있어서 전체 영상 중에서 선택하는 것으로 구현했습니다.

리스트에 담고싶은 영상의 번호를 입력하면 새로운 영상이 추가되었다는 메세지를 출력합니다.

```
------------------------------
3
==========================
<your playlists>
0. Jaeguk List
==========================
enter the number of list to enter
0
==========================
<videos in your playlist>
0. Title : Spiderman, length : 1:15:43, genre : Romance, Action, Comedy,
1. Title : Love Rosie, length : 1:40:01, genre : Romance,
==========================
enter the number of video to watch
0
you are watching video now...
```

리스트에 영상을 담은 다음 다시 3번 명령을 통해 리스트에 있는 영상을 시청해보면,

먼저 보고싶은 리스트를 선택합니다. 그럼 해당 리스트에 있는 영상의 목록들을 보여주고 시청할 영상의 번호를 입력받습니다. 그러면 영상을 시청할 수 있습니다.

```
------------------------------
5
==========================
0. Jaeguk List
==========================
enter number of the list you want to delete
0
That playlist was deleted from your playlists
------------------------------
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
------------------------------
4
you don't have any playlist now...
how about making your own playlist?
------------------------------
```

리스트가 더이상 필요없게 되면 5번 명령을 통해 리스트를 삭제할 수 있습니다.

지우고 싶은 플레이 리스트의 번호를 입력하면 해당 리스트를 삭제하게 됩니다.

다시 4번 명령을 통해 리스트의 목록을 확인해보면 아무 리스트도 없다는 메세지를 출력하는 것을 볼 수 있습니다.

```
--------------------------------
7
you didn't upload any video
--------------------------------
```

7번은 자산이 업로드한 영상을 삭제하는 기능입니다.

영상을 업로드하지 않은 유저가 7번 기능을 사용하면 아무런 영상을 업로드하지 않았다는 메세지를 출력합니다.

```
7
-------------------------
<list of videos you uploaded>
0.  title : Titanic, length : 1:25:30, view : 0, genres : Romance,
1.  title : About time, length : 1:30:15, view : 1, genres : Romance,
2.  title : Avengers, length : 1:22:58, view : 1, genres : Action, Comedy,
3.  title : Spiderman, length : 1:15:43, view : 3, genres : Romance, Action, Comedy,
4.  title : Love Rosie, length : 1:40:01, view : 0, genres : Romance,
5.  title : Pororo, length : 48:20, view : 0, genres : Kids,
-------------------------
enter the number of video to delete
2
------------------------------
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
------------------------------
0
0. Title : Titanic, Channel : GoodMovie, view : 0 genre : Romance,
1. Title : About time, Channel : GoodMovie, view : 1 genre : Romance,
2. Title : Spiderman, Channel : GoodMovie, view : 3 genre : Romance, Action, Comedy,
3. Title : Love Rosie, Channel : GoodMovie, view : 0 genre : Romance,
4. Title : Pororo, Channel : GoodMovie, view : 0 genre : Kids,
5. Title : Chicken Mukbang, Channel : JohnTV, view : 0 genre : Mukbang,
What video do you want to see? tell me a number
```

만약 업로드한 영상이 있는 유저라면 자신이 업로드했던 영상의 리스트를 보여주고 삭제하고 싶은 영상의 번호를 입력받습니다.

그리고 0번을 통해 영상의 목록을 보면 해당 영상이 지워진 것을 확인할 수 있습니다.

```
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
------------------------------
8
1st. Romance 4 times.
2nd. Action 3 times.
3rd. Comedy 3 times.
------------------------------
```

유저의 마지막 기능은 자신이 시청한 장르를 많이 시청한 장르부터 순서대로 확인할 수 있습니다.

8번 명령을 사용하여 시청한 장르와 장르별 시청 횟수를 볼 수 있습니다.

- 계정 생성

```
--------------------------
3
--------------------------
enter your new id
Database
enter your password
thank
enter your new channel name. If you don't want to make channel just press enter

enter your age
23
enter your sex(F or M)
M

enter your Address(Ex) Seoul)
Seoul
enter your Email(Ex) "jk6722@naver.com" )
Database@hanyang.ac.kr
enter your phone number(Ex)01012345678)
01012345678
Congratulate to join us!
You can login with your new id
--------------------------
```

처음 화면에서 3번을 입력하면 새로운 계정을 생성할 수 있습니다.

아이디, 비밀번호, 채널명, 나이, 성별, 거주지, 이메일, 전화번호를 입력하면 계정이 생성됩니다.

이때 아이디는 다른 유저와 겹치지 않아야 하며, 채널명은 원하지 않으면 입력하지 않아도 됩니

다.

```
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
2
Please enter your id(Ex) 'jk6722')
Database
please enter your password
thank
-----------------------------
0.watch other's video
1.upload your own video
2.make a playlist
3.watch video in your playlist
4.view list of your playlists
5.delete the playlists
6.add or delete video from your playlist
7.delete video you uploaded
8.view the ranking of genre
9.quit
-----------------------------
```

새로 생성한 계정으로 로그인을 해보면 로그인이 되는 것을 확인할 수 있습니다.

```
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
3
-------------------------
enter your new id
Database
Another user is using that ID!!
Please enter again.
enter your new id
jk6722
Another user is using that ID!!
Please enter again.
enter your new id
```

회원가입시 이미 다른 유저가 사용하고 있는 아이디를 생성하려고 하면, 다른 사람이 이미 사용하고 있다는 메세지를 출력하고 다시 입력받습니다.

2. 코드 설명

```
try {
    String url = "jdbc:mysql://localhost:3307/youtube";
    String user = "root";
    String password = "qmffhdn33!";

    Connection connection = DriverManager.getConnection(url, user, password);
    Statement statement = connection.createStatement();
    Statement update_stat = connection.createStatement();
    Statement temp_stat = connection.createStatement();
    ResultSet result, temp_set;
```

먼저 jdbc connector를 설치하여 외부라이브러리에 추가해서 mysql과 연동할 수 있도록 합니다.

그리고 url에는 접소갈 sql의 주소를 넣어주고, user에는 아이디, password에는 비밀번호를 문자열로 입력해줍니다.

그리고 Connection을 만들어서 url, 아이디, 비밀번호를 매개변수로 넘겨주면 Connection이 형성됩니다. 그럼 이 connection을 이용해서 쿼리문을 요청할 때 사용할 Statement와 결과 테이블을 저장할 ResultSet 인스턴스를 만들어 줍니다.

```
while(true){
    System.out.println("-------------------------");
    System.out.println("What do you want?");
    System.out.println("0.Exit");
    System.out.println("1.Manager");
    System.out.println("2.User");
    System.out.println("3.Make new Id");
    System.out.println("-------------------------");
    int cmd = -1;
    Scanner scan = new Scanner(System.in);
    cmd = scan.nextInt();
    scan.nextLine();
    if(cmd == 0){ //exit program
        statement.close();
        update_stat.close();
        temp_stat.close();
        connection.close();
        return;
    }
```

전체 코드는 0번 exit를 입력받을 때 까지 동작합니다.

```java
public static int Manager(){
    Scanner scan = new Scanner(System.in);
    int cmd = -1;
    System.out.println("------------------------------");
    System.out.println("0.delete video");
    System.out.println("1.view user's information");
    System.out.println("2.view user's viewing history");
    System.out.println("3.view videos uploaded by user");
    System.out.println("4.quit");
    System.out.println("------------------------------");
    cmd = scan.nextInt();
    scan.nextLine();
    return cmd;
}
```

```java
public static int User(){
    Scanner scan = new Scanner(System.in);
    int cmd = -1;
    System.out.println("------------------------------");
    System.out.println("0.watch other's video");
    System.out.println("1.upload your own video");
    System.out.println("2.make a playlist");
    System.out.println("3.watch video in your playlist");
    System.out.println("4.view list of your playlists");
    System.out.println("5.delete the playlists");
    System.out.println("6.add or delete video from your playlist");
    System.out.println("7.delete video you uploaded");
    System.out.println("8.view the ranking of genre");
    System.out.println("9.quit");
    System.out.println("------------------------------");
    cmd = scan.nextInt();
    scan.nextLine();
    return cmd;
}
```

매너저 모드와 유저모드에서의 인터페이스 출력과 명령어 입력은 Manager, User 메소드를 통해 입력받습니다.

#Delete video

```
case 0: // delete video from data
    //show videos first
    result = statement.executeQuery( sql: "Select * from video");
    if(!result.next()){
        System.out.println("There is no video now");
        break;
    }
    result = statement.executeQuery( sql: "Select * from video");
    int v_idx = 0;
    System.out.println("==============================");
    while (result.next()) {
        String v_title = result.getString( columnLabel: "Title");
        int v_count = result.getInt( columnLabel: "view_count");
        int v_id = result.getInt( columnLabel: "video_id");
        ResultSet genres = temp_stat.executeQuery( sql: "Select genre_name from genre_of, video, genre where video_id = vid_id AND gen_idx = genre_index AND vid_id = " + v_id);
        //should show channel name
        System.out.print(v_idx + ". Title : " + v_title + ", Video_id : " + v_id + ", view : " + v_count + ", genre : ");
        while (genres.next()) {
            System.out.print(genres.getString( columnLabel: "genre_name") + ", ");
        }
        System.out.println();
        update_stat.executeUpdate( sql: "update video set video_index = " + v_idx + " where video_id = " + v_id);
        v_idx++;
    }
```

쿼리문을 사용해 전체 video 테이블의 터플을 불러옵니다.

만약 데이터베이스 내에 터플이 하나도 존재하지 않는다면, 아직 업로드된 영상이 하나도 없다는 뜻이므로 "There is no video now" 메세지를 출력하고 종료합니다.

영상이 존재한다면 모든 영상의 목록을 출력합니다. 이때 genre_of 테이블과 video 테이블을 join 하여 영상의 장르도 같이 출력할 수 있도록 합니다. join attribute는 genre_of table의 vid_id와 video 테이블의 video_id가 됩니다.

사용자가 원하는 영상의 번호를 입력받을 때 인덱스를 0부터 시작해서 편하게 볼 수 있도록 video_index를 출력시 마다 업데이트 해줍니다. 중간에 삭제된 영상이 있다면 0,1,2,6,...이렇게 여

```
System.out.println("==============================");
System.out.println("input the number of the video to delete");
int v_idx_to_delete = scan.nextInt();
result = statement.executeQuery( sql: "Select video_id from video where video_index = " + v_idx_to_delete);
if (!result.next()) {
    System.out.println("No such video...! try again");
    continue;
}
int vid_to_delete = result.getInt( columnLabel: "video_id");
System.out.println("Are you sure? please enter 'Yes' if you want to delete");
scan.nextLine();
String is_sure = scan.nextLine();
if (!is_sure.equals("Yes") && !is_sure.equals("YES") && !is_sure.equals("yes")) {
    System.out.println("Ok. I won't delete it");
    continue;
}
update_stat.executeUpdate( sql: "Delete from includes where vid_id = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from genre_of where vid_id = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from watches where vid = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from video where video_id = " + vid_to_delete);
System.out.println("video_id : " + vid_to_delete + " was deleted...");
break;
```

질 수 있기 때문입니다.

영상을 모두 출력했다면 매니저가 지우고 싶어하는 영상의 index를 입력받습니다.

잘못된 입력이 들어왔다면 "No such video" 라는 메세지를 출력하고 나갑니다.

정상적인 입력이 들어오더라도 삭제하기 전에 정말로 삭제할 것인지 한 번 더 물어본 후 확실하다면 Delete 구문을 사용해서 해당 영상과 관련된 모든 터플들을 삭제합니다.

영상이 지워지게 되면 건드려야 할 테이블은 video, genre_of, includes, watches 4개가 있습니다.

지우고자하는 영상의 video_id를 이용해서 터플들을 지웁니다.

#show user's information

```
case 1: // view user's inform
    result = statement.executeQuery( sql: "Select * from user where Mgr_ssn = " + Mgrssn);
    System.out.println("=============================================================================================================");
    System.out.println("<User List>");
    int user_idx = 0;
    while(result.next()){
        System.out.println("user " + user_idx + ". ID : " + result.getString( columnLabel: "Id") + ", Age : " + result.getInt( columnLabel: "Age") + ", Sex : " +
                result.getString( columnLabel: "Sex") + ", Address : " + result.getString( columnLabel: "Address") + ", Email : " + result.getString( columnLabel: "Email") +
                ", Phone number : " + result.getString( columnLabel: "PhoneNumber") + ", Mgr_ssn : " + result.getInt( columnLabel: "Mgr_ssn"));
        user_idx++;
    }
    System.out.println("=============================================================================================================");
    System.out.println("");
    break;
```

유저의 정보만을 출력하는 기능이기 때문에 현재 해당 기능을 이용중인 매니저의 Ssn과 user의 attribute인 Mgr_ssn을 join condition으로 하여 해당 매니저가 관리하는 유저들의 정보를 Select 합니다.

그리고 유저들의 정보를 차례로 출력합니다.

#show user's watching history

```
case 2: // view user's viewing history
    result = statement.executeQuery( sql: "Select Id from user where Mgr_ssn = " + Mgrssn);
    if(!result.next()){
        System.out.println("There is no user you are managing");
        break;
    }
    result = statement.executeQuery( sql: "Select Id from user where Mgr_ssn = " + Mgrssn);
    int user_count = 0;
    while(result.next()){
        String user_id = result.getString( columnLabel: "Id");
        //String Chan_name = result.getString("ChannelName");
        System.out.println("user " + user_count + ". ID : " + user_id);
        user_count++;
    }
    System.out.println("enter user's ID you want to see history");
    String id_want_to_see = scan.nextLine();
    id_want_to_see = "'" + id_want_to_see + "'";
    result = statement.executeQuery( sql: "Select video_id, Title, length from watches, video where video_id = vid AND uid = " + id_want_to_see);
    System.out.println("====================================");
    System.out.println("<history of user " + id_want_to_see + ">");
    user_count = 0;
    while(result.next()){
        String vid = result.getString( columnLabel: "video_id");
        String vTitle = result.getString( columnLabel: "Title");
        String vlength = result.getString( columnLabel: "length");
        System.out.println(user_count + ". " + "Video_id : " + vid + ", Title : " + vTitle + ", length : " + vlength);
        user_count++;
    }
```

Mgr_ssn과 Ssn을 join condition으로 해당 매니저가 관리하고 있는 유저들의 Id를 Select 합니다.

그리고 그 유저들의 ID를 모두 출력한 후, 시청기록을 보고싶은 유저의 ID를 입력받습니다.

유저의 시청 기록은 watches라는 테이블을 이용해서 가져올 수 있습니다. watches 테이블의 vid, uid를 FK로 해서 해당 유저가 시청한 영상과 관련된 터플을 불러옵니다.

불러온 터플들을 차례로 출력합니다.

String을 쿼리문에 사용할 때는 문자열 앞뒤에 ' (작은 따옴표)를 붙여주어야 합니다.

#show list of the videos uploaded by user

```
case 3: // view list of the videos uploaded by user
    result = statement.executeQuery( sql: "Select * from user where Mgr_ssn = " + Mgrssn);
    if(!result.next()){
        System.out.println("you are not managing any user");
        break;
    }
    result = statement.executeQuery( sql: "Select * from user where Mgr_ssn = " + Mgrssn);
    System.out.println("====================================");
    System.out.println("<List of user's ID you are managing>");
    int temp_idx = 0;
    while(result.next()){
        System.out.println(temp_idx + ". " + result.getString( columnLabel: "ID"));
        temp_idx++;
    }
    System.out.println("====================================");
    System.out.println("enter the ID of user you want to know");
    String uid = scan.nextLine();
    uid = "'" + uid + "'";
    result = statement.executeQuery( sql: "Select * from video where uploader_id = " + uid);
    if(!result.next()){
        System.out.println(uid + " didn't upload any video");
        break;
    }
    result = statement.executeQuery( sql: "Select * from video where uploader_id = " + uid);
```

앞서와 마찬가지로 자신이 관리하고 있는 유저들의 아이디를 출력한 후 정보를 보고싶은 유저의 아이디를 입력받습니다.

video 테이블에 있는 uploader_id를 FK로 해서 유저 테이블의 ID와 join하면 해당 유저가 업로드한 영상에 대한 정보를 불러올 수 있습니다.

#user mode

```
else if(cmd == 2) { //user mode
    System.out.println("Please enter your id(Ex) 'jk6722')");
    String uid = scan.nextLine();
    uid = "'" + uid + "'";
    String answer = "";
    result = statement.executeQuery( sql: "Select Id, Password from user where Id = " + uid);
    if (!result.next()) {
        System.out.println("No such User...sorry");
        continue;
    }
    else
        answer = result.getString( columnLabel: "Password");
    System.out.println("please enter your password");
    String upass = scan.nextLine();
    if (!answer.equals(upass)) {
        System.out.println("wrong password!");
        continue;
    }
    System.out.println("Hello, " + uid + "!");
```

유저는 자신이 회원가입할 때 등록한 ID와 비밀번호로 로그인을 할 수 있습니다. 입력받은 ID가 존재하지 않는다면 "No such user" 메세지를 출력하고 , 비밀번호가 일치하지 않는다면 "wrong password!" 라는 메세지를 출력합니다.

데이터 베이스에 저장된 ID와 Password와 일치하면 성공적으로 로그인이 됩니다.

#watching video

```
case 0: // show video to user
    result = statement.executeQuery( sql: "Select * from video");
    if(!result.next()){
        System.out.println("There is no video now");
        break;
    }
    result = statement.executeQuery( sql: "Select * from video, user where uploader_id = ID order by video_index ASC");
    update_stat = connection.createStatement();
    temp_stat = connection.createStatement();
    int v_idx = 0;
    while (result.next()) {
        String v_title = result.getString( columnLabel: "Title");
        int v_count = result.getInt( columnLabel: "view_count");
        int v_id = result.getInt( columnLabel: "video_id");
        String channel = result.getString( columnLabel: "ChannelName");
        ResultSet genres = temp_stat.executeQuery( sql: "Select genre_name from genre_of, video, genre where video_id = vid_id AND gen_idx = genre_index AND vid_id = " + v_id);
        System.out.print(v_idx + ". Title : " + v_title + ", Channel : " + channel + ", view : " + v_count + " genre : ");
        while (genres.next()) {
            System.out.print(genres.getString( columnLabel: "genre_name") + ", ");
        }
        System.out.println();
        update_stat.executeUpdate( sql: "update video set video_index = " + v_idx + " where video_id = " + v_id);
        v_idx++;
    }
```

먼저 모든 video 테이블의 터플들을 Select하여 유저에게 보여줍니다.

이때 genre_of, genre, video 테이블을 join하여 영상에 대한 정보와, 영상과 관련된 장르명까지 알 수 있도록 보여줍니다.

join attribue는 genre_of의 vid_id, gen_idx, genre의 genre_index, video의 video_id가 됩니다.

그리고 유저가 보기를 원하는 영상의 index를 입력받습니다.

```
while (is_continue == 0) {
    System.out.println("What video do you want to see? tell me a number");
    int idx_to_watch = scan.nextInt();
    scan.nextLine();
    result = statement.executeQuery( sql: "Select video_id from video where video_index = " + idx_to_watch);
    if (!result.next()){
        System.out.println("wrong input!");
        continue;
    }
    result = statement.executeQuery( sql: "Select video_id from video where video_index = " + idx_to_watch);
    result.next();
    int v_id = result.getInt( columnLabel: "video_id");
    update_stat.executeUpdate( sql: "update video set view_count = view_count + 1 where video_index = " + idx_to_watch);
    ResultSet genres = statement.executeQuery( sql: "Select gen_idx from genre_of where vid_id = " + v_id);
    while (genres.next()) {
        temp_set = temp_stat.executeQuery( sql: "Select * from watches_genre where user_id = " + uid + " AND gen_idx = " + genres.getInt( columnLabel: "gen_idx"));
        if(temp_set.next())
            update_stat.executeUpdate( sql: "update watches_genre set gen_count = gen_count + 1 where user_id = " + uid + " AND gen_idx = "
                    + genres.getInt( columnLabel: "gen_idx"));
        else update_stat.executeUpdate( sql: "Insert into watches_genre values(" + uid + "," + genres.getInt( columnLabel: "gen_idx")+ "," + 1 + ")");
    }
    result = statement.executeQuery( sql: "Select * from watches where uid = " + uid +" AND vid = " + v_id);
    if(!result.next())
        update_stat.executeUpdate( sql: "Insert into watches(uid, vid) values (" + uid + "," + v_id + ")");
    //increase view count of the video
    System.out.println("You are watching video now...");
```

video와 관계를 맺고있는 테이블들은 video_id를 FK로 가지고 있기 때문에 index를 이용해서 video_id를 불러옵니다.

유저가 영상을 시청하게 되면, 영상의 조회수, 해당 유저가 시청한 장르별 시청횟수를 업데이트 하고 만약 해당 영상을 처음 시청한 것이라면 watches 테이블에 새로운 터플을 생성해서 시청 기록을 남겨야 합니다.

유저의 id인 uid와 video_id인 v_id를 사용해서 wataches_genre, video, watches 테이블에서 관련된 터플들을 업데이트 합니다.

#uploading new video

```
case 1: //upload video
    result = statement.executeQuery( sql: "Select * from user where Id = " + uid + " AND ChannelName IS NULL");
    if(result.next()){
        System.out.println("you should determine your channel name first");
        while(true){
            System.out.println("enter your own channel name to make");
            String new_channel = scan.nextLine();
            new_channel = "'" + new_channel + "'";
            temp_set = statement.executeQuery( sql: "Select * from user where ChannelName = " + new_channel);
            if(!temp_set.next()){
                update_stat.executeUpdate( sql: "update user set ChannelName = " + new_channel + " where Id = " + uid);
                System.out.println("Channel " + new_channel + " is opened!");
                break;
            }
            else System.out.println("There is alreay same name of channel...try again");
        }
    }
```

영상을 올리고 싶은 유저라면 ChannelName이 존재해야 합니다.

ChannelName attribute의 값이 null인 유저라면 새로운 Channel을 개설하도록 요청합니다.

update 구문을 사용해서 null로 되어있던 값을 새로 입력받은 값으로 변경합니다.

```
System.out.println("enter the title of your video(Don't use symbol ' for your title!)");
String new_title = scan.nextLine();
new_title = "'" + new_title + "'";
System.out.println("enter the length of your video(Ex) 1:25:05");
String new_time = scan.nextLine();
new_time = "'" + new_time + "'";
System.out.println("enter all genres of your video(Ex) 1, 3, 5");
String v_genre = check_genre();
String[] genres = v_genre.split( regex: ",");
result = statement.executeQuery( sql: "Select Max(video_id) from video");
int new_video_id = result.next() ? result.getInt( columnIndex: 1) + 1 : 0;
result = statement.executeQuery( sql: "Select Max(video_index) from video");
int new_video_index = result.next() ? result.getInt( columnIndex: 1) + 1 : 0;
statement.executeUpdate( sql: "Insert into video(Title,length,video_id,uploader_id,video_index) values ("
        + new_title + "," + new_time + "," + new_video_id + "," + uid + "," + new_video_index + ")");
for (String genre : genres)
    statement.executeUpdate( sql: "Insert into genre_of(vid_id, gen_idx) values(" + new_video_id + "," + Integer.parseInt(genre.trim()) + ")");
System.out.println("New video was uploaded.");
break;
```

유저로부터 업로드할 영상의 제목, 길이, 장르의 인덱스를 입력받습니다.

이때 관련된 장르는 여러 개 선택할 수 있도록 했습니다.

새로운 영상의 video_id는 기존에 저장되어있던 video_id 값 중 가장 큰 id에 + 1을 해서 사용했습니다. 가장 큰 값을 찾기 위해 Max aggregate function을 사용했습니다.

그리고 video 테이블에 새로운 터플을 생성하고, genre_of 테이블에 새로운 터플을 생성하여 해당 영상의 장르를 등록합니다.

#making new playlist

```
case 2: //make playlist
    System.out.println("enter the Name of your new playlist(Don't use ' symbol)");
    String new_name = scan.nextLine();
    new_name = "'" + new_name + "'";
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid + " AND Listname = " + new_name);
    if (result.next()) {
        System.out.println("You already have the same name list");
        break;
    }
    System.out.println("Do you want to share your playlist to other people?");
    System.out.println("0. yes");
    System.out.println("1. no");
    int Input = scan.nextInt();
    scan.nextLine();
    boolean is_share = (Input == 0);
    result = statement.executeQuery( sql: "Select Max(list_index) from playlist");
    int new_list_index = result.next() ? result.getInt( columnIndex: 1) + 1 : 0;
    statement.executeUpdate( sql: "Insert into playlist(Master_id, Listname, is_shared, list_index) values("
            + uid + "," + new_name + "," + is_share + "," + new_list_index + ")");
    System.out.println("New playlist was created");
    break;
```

유저로부터 만들고자하는 리스트의 이름을 입력받습니다.

ListName은 weak entitiy이기 때문에 같은 유저가 같은 이름의 플레이 리스트를 2개 이상 만들

수 없습니다. 이를 Select 구문을 사용하여 체크해줍니다.

만약 중복된 이름의 리스트가 존재하지 않는다면 Insert 구문을 사용해서 playlist 테이블에 새로운 터플을 생성합니다.

#watching video in the playlist

```
case 3: //view video in the play list
    int list_idx = 0;
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid);
    if(!result.next()){
        System.out.println("you don't have any playlist now");
        break;
    }
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid);
    System.out.println("==========================");
    System.out.println("<your playlists>");
    while(result.next()){
        String Lname = result.getString( columnLabel: "Listname");
        System.out.println(list_idx + ". " + Lname);
        Lname = "'" + Lname + "'";
        update_stat.executeUpdate( sql: "update playlist set list_index = " + list_idx + " where Listname = " + Lname + " AND Master_id = " + uid);
        list_idx++;
    }
    System.out.println("==========================");
    System.out.println("enter the number of list to enter");
    list_idx = scan.nextInt();
    scan.nextLine();
    result = statement.executeQuery( sql: "Select vid_id, Listname from includes, playlist where list_index = " + list_idx + " AND Listname = playlist_name AND Master_id = " + uid);
    if(!result.next()){
        System.out.println("sorry there is no video in this playlist");
        break;
    }
```

먼저 해당 유저가 플레이 리스트를 가지고 있는지를 먼저 체크합니다.

만약 플레이 리스트를 가지고 있지 않다면 해당 기능을 사용할 수 없으므로 “you don't have any playlist now”라는 메세지를 출력하고 종료합니다.

플레이 리스트를 가지고 있는 유저라면 해당 유저의 플레이 리스트 목록을 보여줍니다.

시청하고자 하는 영상이 들어있는 플레이 리스트의 인덱스를 입력받습니다.

리스트 인덱스와 유저의 ID를 이용해서 해당 플레이 리스트에 있는 영상의 목록을 불러옵니다.

플레이 리스트 안에 어떤 영상이 들어있는지는 includes 테이블을 이용하면 알 수 있습니다.

```
String Lname_to_watch = result.getString( columnLabel: "Listname");
Lname_to_watch = "'" + Lname_to_watch + "'";
result = statement.executeQuery( sql: "Select video_id, video_index, Title, length, includes_index from video, includes, playlist where Listname = " + Lname_to_watch +
        " AND playlist_master = Master_id AND Master_id = " + uid + " AND video_id = vid_id AND playlist_name = ListName ORDER BY includes_index ASC");
System.out.println("==========================");
System.out.println("<videos in your playlist>");
int inc_idx = 0;
while(result.next()){
    System.out.print(inc_idx + ". Title : " + result.getString( columnLabel: "Title") + ", length : " + result.getString( columnLabel: "length") +
            ", genre : ");
    int vid = result.getInt( columnLabel: "video_id");
    temp_set = temp_stat.executeQuery( sql: "Select genre_name from genre_of, genre where vid_id = " + vid + " AND gen_idx = genre_index");
    while(temp_set.next()){
        System.out.print(temp_set.getString( columnLabel: "genre_name") + ", ");
    }
    update_stat.executeUpdate( sql: "Update includes Set includes_index = " + inc_idx + " where playlist_master = " + uid +
            " AND playlist_name = " + Lname_to_watch + " AND vid_id = " + vid);
    inc_idx++;
    System.out.println();
}
System.out.println(inc_idx + ". quit");
System.out.println("==========================");
```

video, includes, playlist 를 join하여 플레이 리스트에 들어있는 영상들의 정보를 불러옵니다.

이때 유저의 id, video_id, playlis의 이름을 join attribute로 사용합니다.

그리고 불러온 ResultSet을 바탕으로 플레이 리스트 내의 영상들에 대한 정보를 유저에게 보여주고 시청하고싶은 영상의 index를 입력받습니다.

이후의 코드는 0번(영상 시청)에서의 코드와 동일합니다.

#show list of playlist

```
case 4: //view list
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid);
    if(!result.next()){
        System.out.println("you don't have any playlist now...");
        System.out.println("how about making your own playlist?");
        break;
    }
    result = statement.executeQuery( sql: "Select * from playlist where Master_id = " + uid);
    list_idx = 0;
    update_stat = connection.createStatement();
    while (result.next()) {
    System.out.println("========================");
        String name_of_list = result.getString( columnLabel: "Listname");
        System.out.println(list_idx + ". " + name_of_list);
        name_of_list = "'" + name_of_list + "'";
        update_stat.executeUpdate( sql: "update playlist set list_index = " + list_idx + " where Master_id = " + uid + " AND Listname = " + name_of_list);
        list_idx++;
    }
    System.out.println("========================");
    break;
```

현재 프로그램을 사용중인 유저의 ID를 join key로 하여 해당 유저의 플레이 리스트 이름을 불러옵니다.

만약 join한 결과로 나온 테이블에 터플이 존재하지 않는다면 해당 유저는 플레이 리스트가 없는 것이므로 “you don't have any playlist now...”라는 메세지를 출력하고 종료합니다.

터플이 존재한다면 해당 터플들의 attribute value를 이용해서 유저에게 플레이 리스트 목록을 보여줍니다.

#delete list

```
case 5: //delete list
    System.out.println("========================");
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid);
    if(!result.next()){
        System.out.println("there's no list to delete");
        break;
    }
    result = statement.executeQuery( sql: "Select * from playlist where Master_id = " + uid);
    list_idx = 0;
    update_stat = connection.createStatement();
    while (result.next()) {
        list_idx = result.getInt( columnLabel: "list_index");
        String name_of_list = result.getString( columnLabel: "Listname");
        System.out.println(list_idx + ". " + name_of_list);
        name_of_list = "'" + name_of_list + "'";
        update_stat.executeUpdate( sql: "update playlist set list_index = " + list_idx + " where Listname = " + name_of_list + " AND Master_id = " + uid);
        list_idx++;
    }
    System.out.println("========================");
    System.out.println("enter number of the list you want to delete");
    int list_to_delete = scan.nextInt();
    scan.nextLine();
    result = statement.executeQuery( sql: "Select Listname from playlist where list_index = " + list_to_delete + " AND Master_id = " + uid);
    if (!result.next()) {
        System.out.println("No such playlist!");
        break;
    }
```

유저의 ID를 join key로 해서 앞선 기능에서와 마찬가지로 해당 유저가 가지고 있는 플레이리스트를 불러옵니다.

list_index 역시 Listname과 마찬가지로 weak entity 이므로 유저의 ID와 list_index를 사용해서 지우고자 하는 플레이 리스트를 찾을 수 있습니다.

```
System.out.println("========================");
System.out.println("enter number of the list you want to delete");
int list_to_delete = scan.nextInt();
scan.nextLine();
result = statement.executeQuery( sql: "Select Listname from playlist where list_index = " + list_to_delete + " AND Master_id = " + uid);
if (!result.next()) {
    System.out.println("No such playlist!");
    break;
}
String name_to_delete = result.getString( columnLabel: "Listname");
name_to_delete = "'" + name_to_delete + "'";
statement.executeUpdate( sql: "Delete from includes where playlist_master = " + uid + " AND playlist_name = " + name_to_delete);
statement.executeUpdate( sql: "Delete from playlist where Master_id = " + uid + " AND list_index = " + list_to_delete);
System.out.println("That playlist was deleted from your playlists");
break;
```

플레이 리스트를 지우게 되면 플레이 리스트에 담긴 영상에 대한 정보를 기록하는 includes 테이블에서도 관련 터플들을 지워야 합니다.

Delete 구문을 사용해서 includes와 playlist 테이블에서 관련 터플들을 삭제합니다.

#add video to playlist or delete video from playlist

```
case 6: //delete or add video from,to playlist
    //view the list to user first
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid);
    if(!result.next()){
        System.out.println("you don't have playlist now. please make playlist first!");
        break;
    }
    result = statement.executeQuery( sql: "Select * from playlist where Master_id = " + uid);
    System.out.println("========================");
    list_idx = 0;
    update_stat = connection.createStatement();
    while (result.next()) {
        String name_of_list = result.getString( columnLabel: "Listname");
        System.out.println(list_idx + ". " + name_of_list);
        name_of_list = "'" + name_of_list + "'";
        update_stat.executeUpdate( sql: "update playlist set list_index = " + list_idx + " where Listname = " + name_of_list + " AND Master_id = " + uid);
        list_idx++;
    }
    System.out.println("========================");
    System.out.println("enter number of list you want to modify");
    int idx_to_modify = scan.nextInt(); //index of the playlist to modify
    scan.nextLine();
    result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid + " AND list_index = " + idx_to_modify);
    result.next(); //move cursor to first point
    String name_to_modify = result.getString( columnLabel: "Listname");
    System.out.println("========================");
    System.out.println("<List of the video in " + name_to_modify + ">");
    name_to_modify = "'" + name_to_modify + "'";
```

유저의 ID를 이용해서 해당 유저의 모든 playlist에 대한 정보를 불러와서 출력합니다.

그리고 변경하고자 하는 리스트의 index를 입력받습니다.

원래 플레이 리스트 테이블의 PK는 {유저의 ID, 리스트 이름} 이지만 리스트의 index역시 weak entity 이므로 유저의 ID와 리스트 인덱스를 이용해서 해당 플레이리스트를 불러올 수 있습니다.

includes와 video 테이블을 join하여 해당 플레이 리스트에 들어있는 영상들을 출력합니다.

```
System.out.println("========================");
System.out.println("enter number of list you want to modify");
int idx_to_modify = scan.nextInt(); //index of the playlist to modify
scan.nextLine();
result = statement.executeQuery( sql: "Select Listname from playlist where Master_id = " + uid + " AND list_index = " + idx_to_modify);
result.next(); //move cursor to first point
String name_to_modify = result.getString( columnLabel: "Listname");
System.out.println("========================");
System.out.println("<List of the video in " + name_to_modify + ">");
name_to_modify = "'" + name_to_modify + "'";
result = statement.executeQuery( sql: "Select Title, length, includes_index, video_id from includes, video where video_id = vid_id"
        + " AND playlist_master = " + uid + " AND playlist_name = " + name_to_modify + " ORDER BY includes_index ASC");
//table that user want to modify
inc_idx = 0;
while (result.next()) {
    String v_title = result.getString( columnLabel: "Title");
    String v_length = result.getString( columnLabel: "length");
    System.out.println(inc_idx + ". Title : " + v_title + ", length : " + v_length);
    update_stat.executeUpdate( sql: "Update includes Set includes_index = " + inc_idx + " Where playlist_master = " + uid +
            " AND playlist_name = " + name_to_modify + " AND vid_id = " + result.getInt( columnLabel: "video_id"));
    inc_idx++;
}
System.out.println("========================");
System.out.println("0. add video to playlist");
System.out.println("1. delete video from playlist");
int add_or_delete = scan.nextInt();
scan.nextLine();
```

이때 join attribute는 playlist의 이름, playlist의 주인, video_id가 됩니다.

이후 영상을 추가할 것인지 아니면 영상을 제거할 것인지를 입력받습니다.

#영상 추가

```
System.out.println("-------------------------");
System.out.println("enter number of the video to add");
int idx_to_add = scan.nextInt(); // index of the video to add
scan.nextLine();
result = statement.executeQuery( sql: "Select video_id from video where video_index = " + idx_to_add);
result.next();
int v_to_add = result.getInt( columnLabel: "video_id");
temp_set = temp_stat.executeQuery( sql: "Select vid_id from includes where playlist_master = " + uid + " AND playlist_name = " + name_to_modify +
        " AND vid_id = " + v_to_add);
if(temp_set.next()){
    System.out.println("there is a same video already");
    continue;
}
temp_set = temp_stat.executeQuery( sql: "Select Max(includes_index) from includes");
int includes_idx = temp_set.next() ? temp_set.getInt( columnIndex: 1) + 1 : 0;
statement.executeUpdate( sql: "Insert into includes(playlist_master, playlist_name, vid_id, includes_index) values(" +
        uid + "," + name_to_modify + "," + v_to_add + "," + includes_idx + ")");
System.out.println("New video was inserted into your playlist");
```

영상을 추가하는 것이라면 현재 업로드 되어있는 모든 영상들의 목록을 보여준 후, 추가하고자 하는 영상의 인덱스를 입력으로 받습니다. 인덱스로부터 video_id를 불러내어 해당 플레이 리스트에 이미 해당 영상이 들어있는지를 체크합니다.

만약 들어있지 않다면 Insert 구문을 이용해서 includes 테이블에 새로운 터플을 생성합니다.

#영상 삭제

```
else { //delete
   System.out.println("enter number of the video to delete");
   int idx_to_delete = scan.nextInt();
   scan.nextLine();
   result = statement.executeQuery( sql: "Select vid_id from includes where playlist_master = " + uid +
           " AND playlist_name = " + name_to_modify + " AND includes_index = " + idx_to_delete);
   result.next();
   int vid_to_delete = result.getInt( columnLabel: "vid_id");
   statement.executeUpdate( sql: "Delete from includes where playlist_master = " + uid + " AND playlist_name = " + name_to_modify + " AND vid_id = " + vid_to_delete);
   System.out.println("video was deleted from your playlist");

reak;
```

영상을 삭제하는 것이라면 삭제할 영상의 번호를 입력받은 후 유저의 아이디, 플레이 리스트 이름, 지울 영상의 번호를 이용해서 includes 테이블로부터 해당 영상과 관련된 터플을 삭제합니다.

#delete video uploaded

```
result = statement.executeQuery( sql: "Select * from video where uploader_id = " + uid);
System.out.println("--------------------------");
System.out.println("<list of videos you uploaded>");
temp_stat = connection.createStatement();
while (result.next()) {
    String v_title = result.getString( columnLabel: "Title");
    String v_length = result.getString( columnLabel: "length");
    int v_id = result.getInt( columnLabel: "video_id");
    v_idx = result.getInt( columnLabel: "video_index");
    int num_of_view = result.getInt( columnLabel: "view_count");
    ResultSet genre_names;
    genre_names = temp_stat.executeQuery( sql: "Select genre_name from genre_of, genre where vid_id = " + v_id + " AND gen_idx = genre_index");
    System.out.print(v_id + ". " + " title : " + v_title + ", length : " + v_length + ", view : " + num_of_view + ", genres : ");
    while (genre_names.next())
        System.out.print(genre_names.getString( columnLabel: "genre_name") + ", ");
    System.out.println();
}
System.out.println("--------------------------");
System.out.println("enter the number of video to delete");
int vid_to_delete = scan.nextInt();
scan.nextLine();
update_stat.executeUpdate( sql: "Delete from genre_of where vid_id = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from watches where vid = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from includes where vid_id = " + vid_to_delete);
update_stat.executeUpdate( sql: "Delete from video where video_id = " + vid_to_delete);
break;
```

먼저 해당 유저가 업로드한 모든 영상을 보여줍니다.

그리고 삭제하고자 하는 영상의 video_id를 입력으로 받아서 해당 영상을 삭제합니다.

영상이 삭제되면 영상의 장르를 저장하는 genre_of, 유저들의 시청 기록을 저장하는 watches, 플레이 리스트에 담긴 영상들을 저장하는 includes, 그리고 video 에서 관련된 터플들을 모두 삭제해야 합니다.

#show rank of genre user watched

```
case 8:
    result = statement.executeQuery( sql: "Select genre_index, genre_name, gen_count from watches_genre, genre where user_id = " + uid + " AND gen_idx = genre_index" + " order by gen_count DESC");
    String rank_str[] = {"1st", "2nd", "3rd", "4th", "5th", "6th", "7th"};
    int rank_idx = 0;
    while(result.next()) {
        System.out.println(rank_str[rank_idx] + ". " + result.getString( columnLabel: "genre_name") + " " + result.getString( columnLabel: "gen_count") + " times.");
        rank_idx++;
    }
    break;
```

watches_genre와 genre 테이블을 join하여 해당 유저의 장르별 시청 횟수와 그에 대응되는 장르의 이름을 불러옵니다.

이때 join attribute는 유저의 ID, genre_index가 되며 많이 본 순서대로 출력해야 하므로 시청횟수를 나타내는 gen_count를 기준으로 Order by를 사용하여 내림차순으로 추출합니다.

그리고 장르별 시청횟수를 유저에게 출력합니다.

4. 사용된 Query문 명세

a) 사용자에게 보여주어야 할 터플들이 담긴 테이블을 불러오기 위해서 Select Statement를 사용했습니다. 이때 특정 attribute를 기준으로 정렬된 테이블을 보여주기 위해서 Order by를 사용하기도 했습니다.

b) 새로운 영상을 업로드 한다거나, 회원가입을 한다거나, 새로운 플레이 리스트를 만드는 등 새로운 entity와 relation을 형성하기 위해서 Insert Into Statement를 사용했습니다.

c) 기존에 저장되어 있던 터플들을 삭제하기 위해서 Delete Statement를 사용했습니다.

d) 기존에 저장되어 있던 테이블에서 특정 attribute value를 바꾸기 위해서 Update Statement를 사용했습니다.

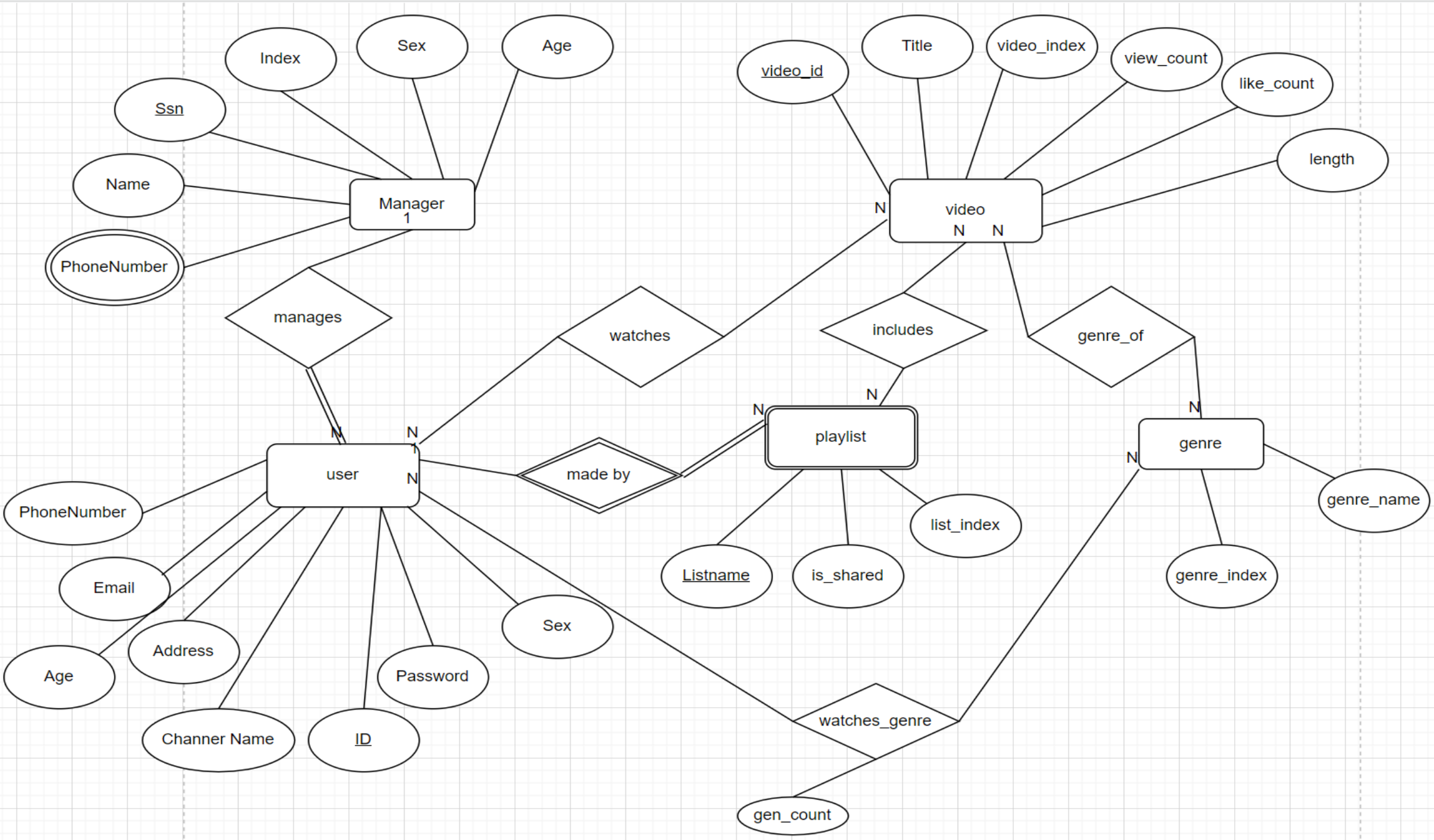
e) 영상을 업로드하기 위해서는 ChannelName이 null값이 되면 안되는데, 해당 조건을 체크하기 위해 Select Statement에서 where절 안에 Is null을 사용했습니다.

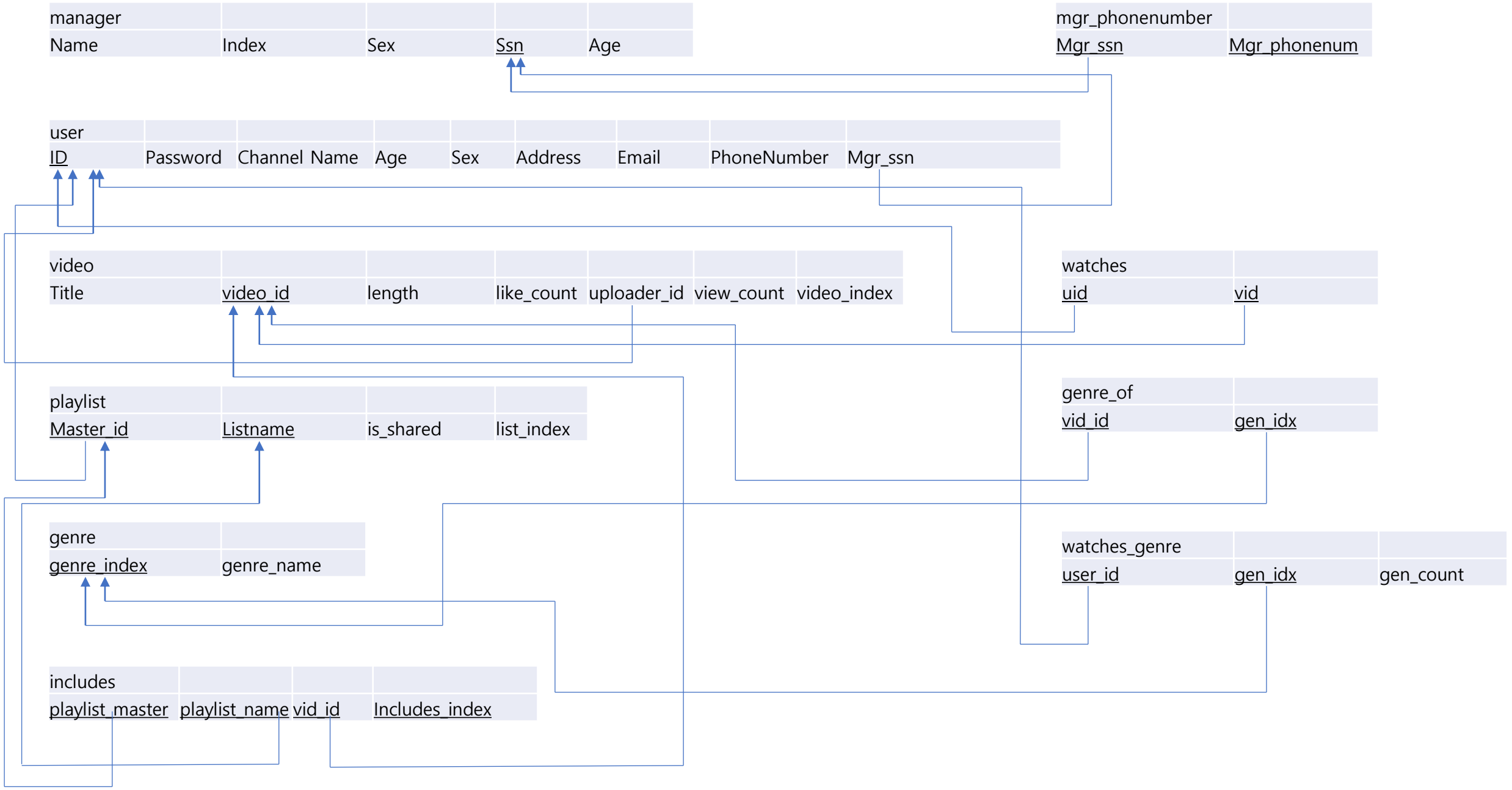
f) 영상이나, 관계 인스턴스를 새로 생성할 때 인덱스, 고유번호 값을 기존 값 중 가장 큰 값에서 +1 한 값을 사용했습니다. 이때 기존 값들 중 가장 큰 값을 찾기 위해 Select 와 Max aggregate function을 사용했습니다.

5. cmd창에서 실행 방법

```
C:\Users\jk672\IdeaProjects\DBproject2\src> java -cp "C:\Users\jk672\mysql-connector-j-8.0.31\mysql-connector-j-8.0.31.jar" test.
java
-------------------------
What do you want?
0.Exit
1.Manager
2.User
3.Make new Id
-------------------------
```

java 파일이 들어있는 디렉토리에 들어가서 java -cp 다음 connector 파일을 연결시켜주고 실행할 자바 파일의 이름을 쓰면 실행이 됩니다.


Index
Sex
Age
Ssn
Name
Manager
1
PhoneNumber
manages
video_id
Title
video_index
view_count
like_count
length
N
video
N N
watches
includes
genre_of
N
N
N
N
N
N
1
N
user
made by
playlist
genre
PhoneNumber
Email
Address
Age
Sex
Password
Channer Name
ID
Listname
is_shared
list_index
genre_name
genre_index
watches_genre
gen_count


manager
Name
Index
Sex
Ssn
Age
mgr_phonenumber
Mgr_ssn
Mgr_phonenum
user
ID
Password
Channel Name
Age
Sex
Address
Email
PhoneNumber
Mgr_ssn
video
Title
video_id
length
like_count
uploader_id
view_count
video_index
watches
uid
vid
playlist
Master_id
Listname
is_shared
list_index
genre_of
vid_id
gen_idx
genre
genre_index
genre_name
watches_genre
user_id
gen_idx
gen_count
includes
playlist_master
playlist_name
vid_id
Includes_index