데이터베이스
기말 과제

MovieLens DB 구축 및
DB 검색프로그램

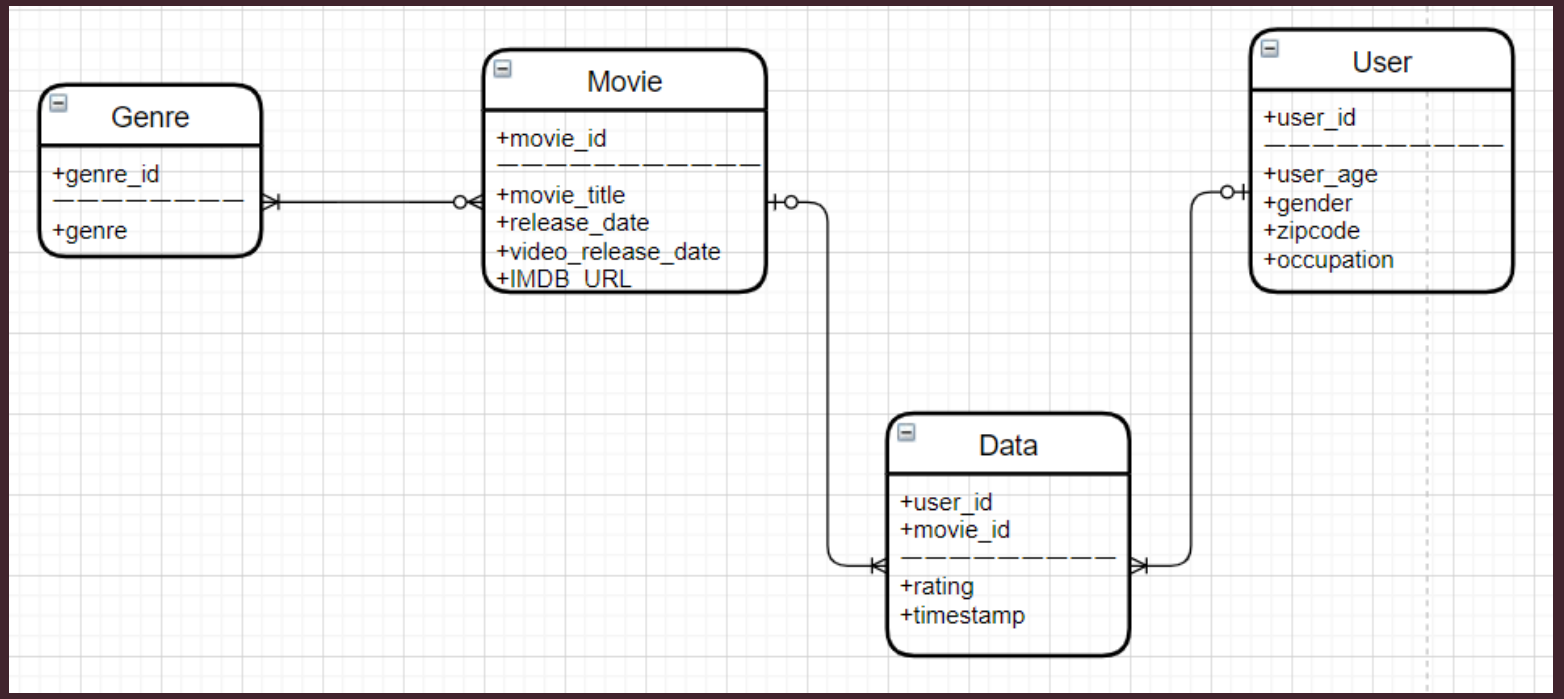
201511291 장유준

1. E/R Modeling

- Entities :
  - Genre (Weak Entity & ID-Dependent)
    multi-valued relationship
  - Movie (Strong Entity)
  - User (Strong Entity)
  - Data (Weak Entity & ID-Dependent)
  - ~~Occupation~~

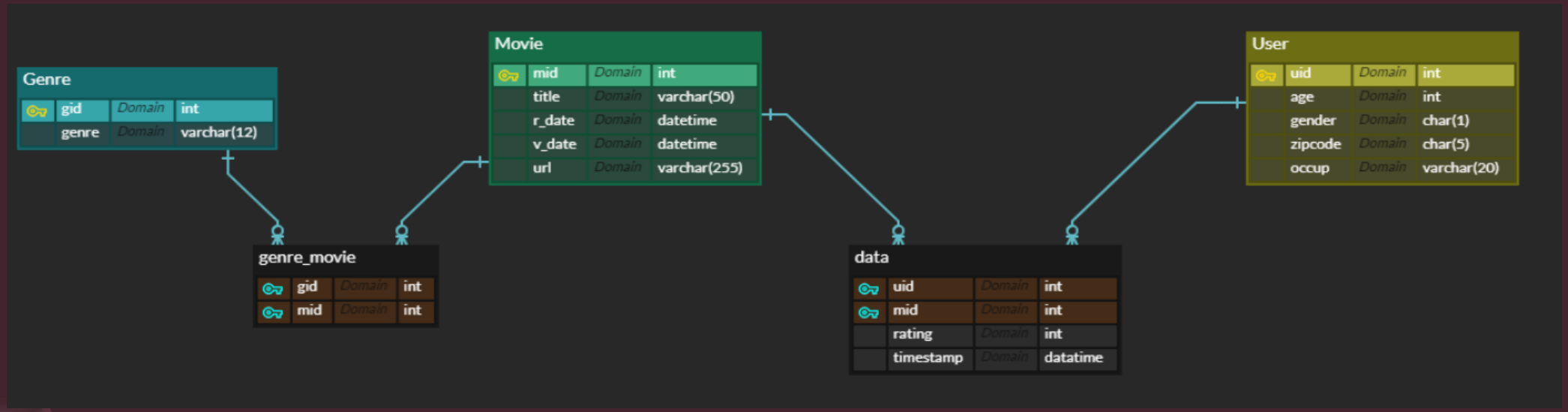
1. E/R Modeling

- Entity Relationship Diagram



2. Relational Database 설계

- Relational Table
  - Genre table과 Movie table의 multi-valued attribute 를 표현하기 위해 relation-table을 생성 , genre_movie table은 weak & id-dependent 하도록 설계



2. Relational Database 설계

- Genre table & movie table create

```
7  create table genre(
8  gid int not null,
9  genre_name varchar(12) not null unique,
10 primary key (gid)
11 );
```

```
13 create table movie(
14 mid int not null,
15 title varchar(100) not null,
16 r_date varchar(20),
17 v_date varchar(20),
18 url varchar(255),
19 primary key (mid)
20 );
```

2. Relational Database 설계

- Genre_Movie table & User table create

```
22  create table genre_movie(
23  gid int not null,
24  mid int not null,
25  primary key (gid, mid),
26  foreign key (gid) references genre(gid),
27  foreign key (mid) references movie(mid)
28  );
```

```
30  create table user(
31  uid int not null,
32  age int not null,
33  gender char(1) not null,
34  zipcode char(5) not null,
35  occup varchar(20),
36  primary key (uid)
37  );
```

2. Relational Database 설계

- Data table create

```
39  create table data(
40  uid int not null,
41  mid int not null,
42  rating int not null,
43  timestamp int not null,
44
45  primary key (uid, mid),
46  foreign key (uid) references user(uid),
47  foreign key (mid) references movie(mid)
48  );
```

3. Making Database

- User tuple insert
- Pymysql library import

```
conn = pymysql.connect(host='localhost', user='db2020', password='db2020', db='db_201511291')
curs = conn.cursor(pymysql.cursors.DictCursor)
```

Mysql db_201511291 데이터베이스에 접근

- Pandas 라이브러리를 이용해 tsv 파일을 open한 뒤 delimiter parameter 를 통해 문자열 파싱

```
user = pd.read_csv(r"C:\Users\oooo1\Desktop\수업\20년1학기\데이터베이스\기말 과제\movielens dataset\u.user.tsv", engine='python',
                   delimiter='|', header=None)

sql = "insert into user (uid, age, gender, occup, zipcode) values (%s, %s, %s, %s, %s)"  # user insert

tmp = []
for x in user.values:
    tmp.append(tuple(x))
curs.executemany(sql, tmp)
conn.commit()
```

- 이후 insert문을 작성 후 db201511291 데이터베이스의 User table에 튜플 삽입

3. Making Database

- Genre tuple insert

```
genre = pd.read_csv(r"C:\Users\oooo1\Desktop\수업\20년1학기\데이터베이스\기말 과제\movielens dataset\u.genre.tsv", engine='python',
                    delimiter='|', header=None)

sql = "insert into genre (genre_name, gid) values (%s, %s)"  # genre insert

tmp = []
for x in genre.values:
    tmp.append(tuple(x))
print(tmp)

curs.executemany(sql, tmp)
conn.commit()
```

- Genre table에 튜플 삽입

3. Making Database

- Item(movie) tuple & Genre_movie tuple insert

```
movie = pd.read_csv(r"C:\Users\oooo1\Desktop\수업\20년1학기\데이터베이스\기말 과제\movielens dataset\u.item.tsv", engine='python',
                    delimiter='|', header=None)

sql = "insert into movie (mid, title, r_date, v_date, url) values (%s, %s, %s, %s, %s)"  # movie insert

tmp = []
genre_list = []
genre_tuple = ()
count = 0
movie = movie.fillna('')

for x in movie.values:
    tmp.append(tuple(x[:5]))

    for genre_bit in tuple(x[5:]):
        if genre_bit == 1:
            genre_tuple = (count, x[0])
            genre_list.append(genre_tuple)  # genre_movie에 들어갈 튜플을 만든다
        count = count + 1
    count = 0
print(tmp)

curs.executemany(sql, tmp)
conn.commit()

sql = "insert into genre_movie (gid, mid) values (%s, %s)"  # genre_movie insert
curs.executemany(sql, genre_list)
conn.commit()
```

- Movie table, Genre_Movie table에 튜플 삽입

3. Making Database

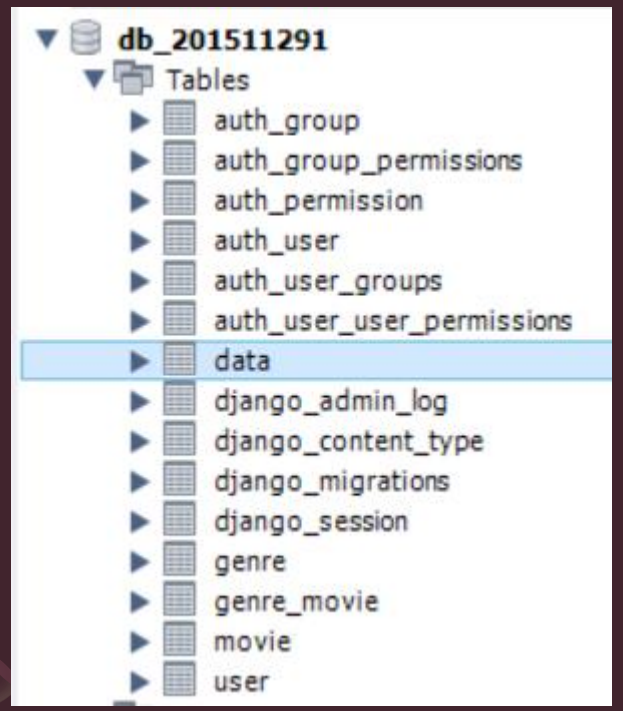
- Item(movie) tuple & Genre_movie tuple insert

```
data = pd.read_csv(r"C:\Users\oooo1\Desktop\수업\20년1학기\데이터베이스\기말 과제\movielens dataset\u.data.tsv", engine='python',
                   delimiter='\t', header=None)

sql = "insert into data (uid, mid, rating, timestamp) values (%s, %s, %s, %s)"  # data insert

tmp = []

for x in data.values:
    uid = int(x[0])
    mid = int(x[1])
    rating = int(x[2])
    timestamp = int(x[3])
    tmp.append((uid, mid, rating, timestamp))
print(tmp)

curs.executemany(sql, tmp)
conn.commit()
```

Data table에 튜플 삽입

3. Making Database

Workbench에 들어가 있는 DB의 목록 (파이썬 장고와 연동해 auth와 Django_ 와 같은 DB 들이 추가 삽입됨)



User tuples

```
12 •  select * from user;
```

Result Grid | Filter Rows:

uid	age	gender	zipcode	occup
1	24	M	85711	technician
2	53	F	94043	other
3	23	M	32067	writer
4	24	M	43537	technician
5	33	F	15213	other
6	42	M	98101	executive
7	57	M	91344	administrator
8	36	M	05201	administrator
9	29	M	01002	student
10	53	M	90703	lawyer

3. Making Database

Movie tuples

```
13 •  select * from movie;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

mid	title	r_date	v_date	url
1	Toy Story (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Toy%20Story...
2	GoldenEye (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?GoldenEye%2...
3	Four Rooms (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Four%20Roo...
4	Get Shorty (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Get%20Short...
5	Copycat (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Copycat%20(...
6	Shanghai Triad (Yao a yao yao dao waipo qiao) ...	01-Jan-1995		http://us.imdb.com/Title?Yao+a+yao+yao+da...
7	Twelve Monkeys (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Twelve%20M...
8	Babe (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Babe%20(1995)
9	Dead Man Walking (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Dead%20Man...
10	Richard III (1995)	22-Jan-1996		http://us.imdb.com/M/title-exact?Richard%20II...

3. Making Database

Genre tuples

```
14 •  select * from genre;
```

Result Grid | Filter Rows:

gid	genre_name
1	Action
2	Adventure
3	Animation
4	Children's
5	Comedy
6	Crime
7	Documentary
8	Drama
9	Fantasy
10	Film-Noir

Genre_movie tuples

```
15 •  select * from genre_movie;
```

Result Grid | Filter Rows:

gid	mid
3	1
4	1
5	1
1	2
2	2
16	2
16	3
1	4
5	4
8	4

3. Making Database

Data tuples

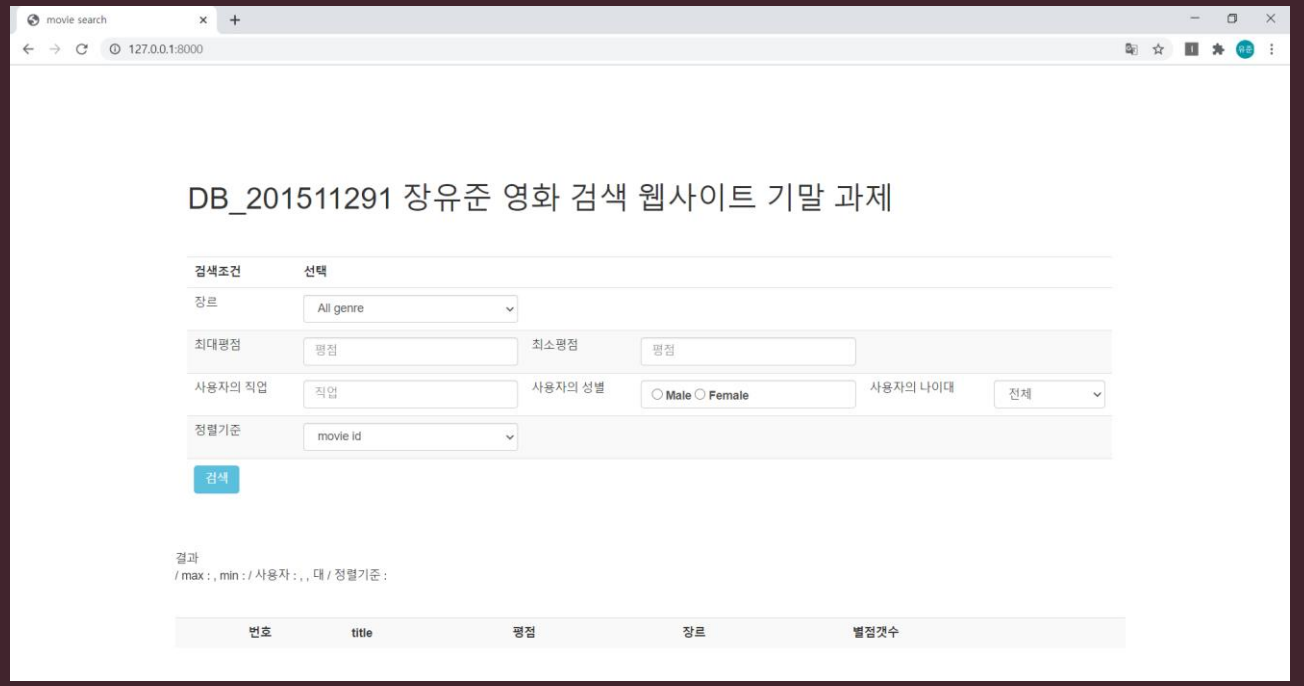
```
16 •  select * from data;
```

Result Grid | Filter Rows:

uid	mid	rating	timestamp
1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	6	5	887431973
1	7	4	875071561
1	8	1	875072484
1	9	5	878543541
1	10	3	875693118

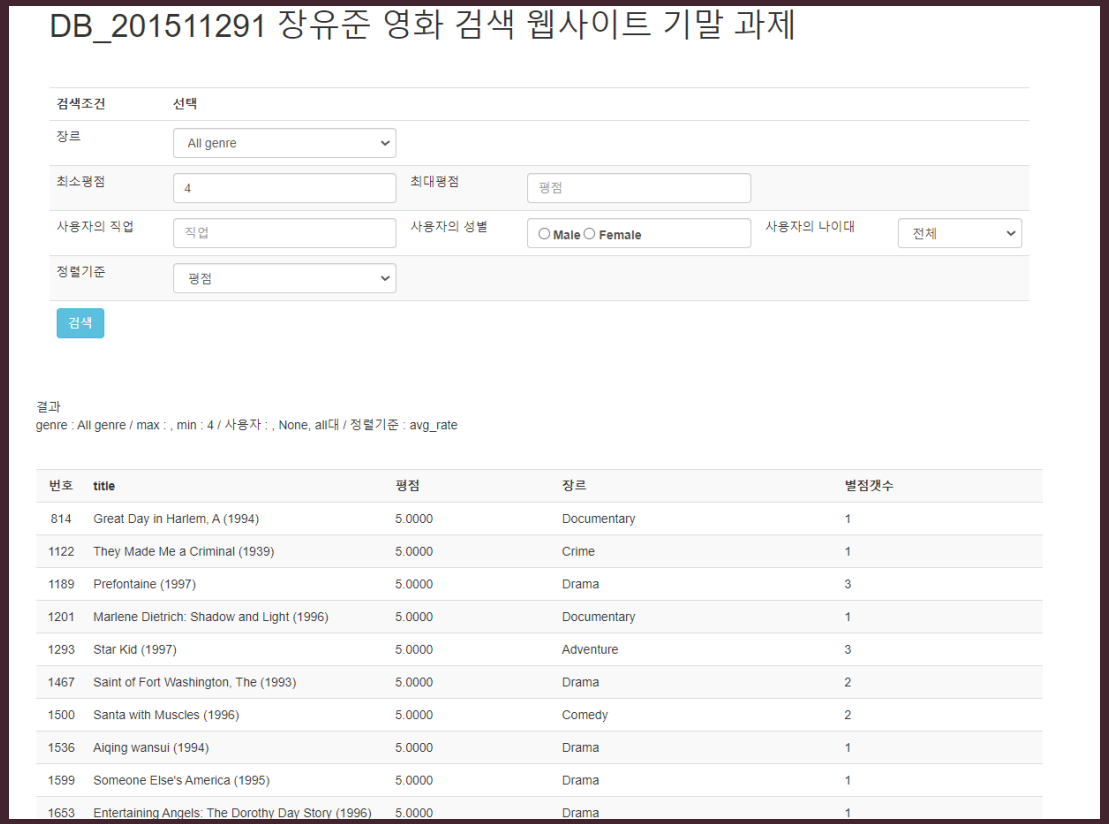
4. Searching program

- Python Django를 이용해 웹 화면, 데이터베이스 구현
- Django ORM library를 사용해 필터링 하지 않고 Raw 쿼리를 이용해 필터링 하도록 구현



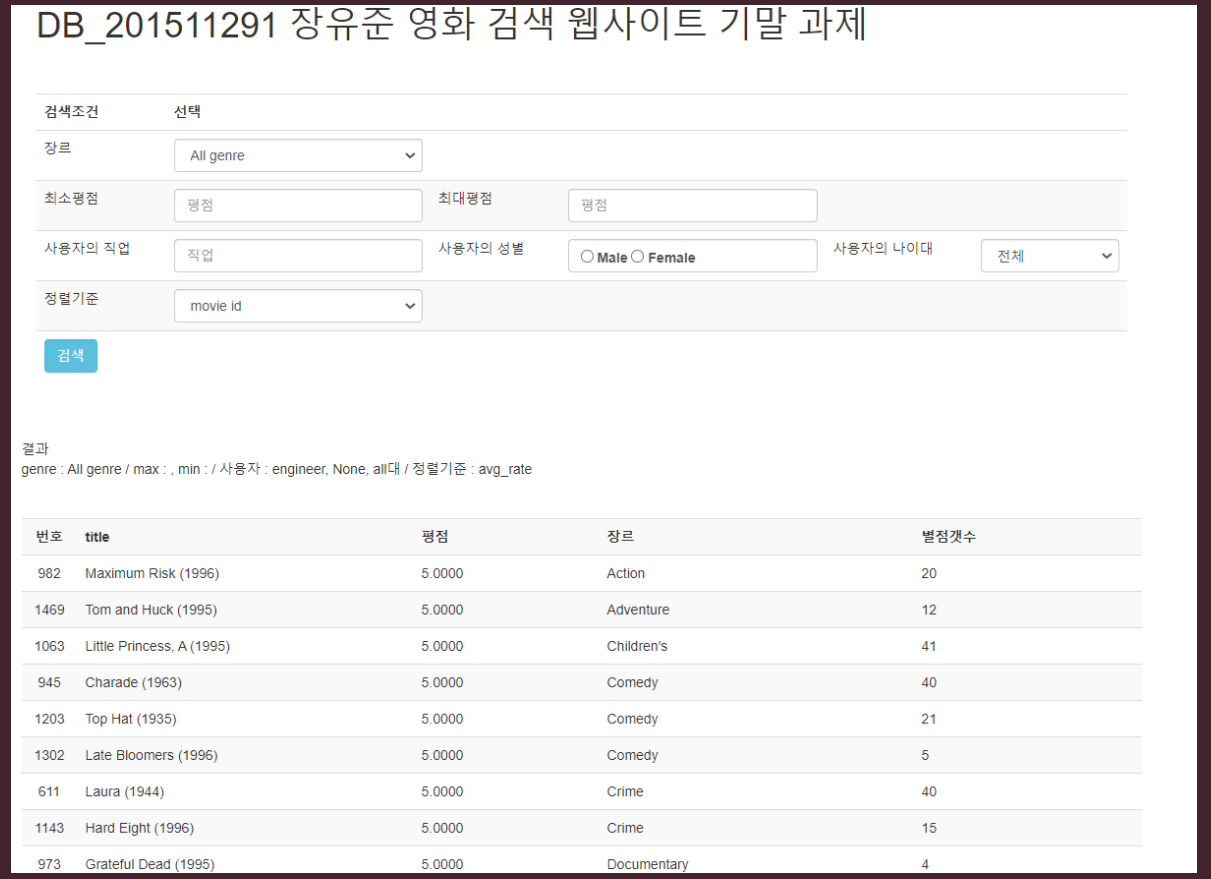
4. Searching program

검색조건 1 : 평균평점이 4.0이상인 영화의 정보를 평점순으로 검색하라.



4. Searching program

검색조건 2 : 직업이 engineer인 사용자들에 의한 평균평점이 높은 순으로 영화정보를 검색하라



4. Searching program

검색조건 3 : Comedy 장르영화 중에서 평균평점이 4.5 이상인 영화정보를 Vote가 많은 순으로 검색하라.

4. Searching program

검색조건 4 (임의설정) : 모든 장르 중에 사용자들의 직업이 writer이고 성별이 Male이며 나이대가 30대인 사람들의 평점이 3~4 사이에 있는 영화들을 영화 번호를 정렬기준으로 검색하라