

## Homework 3: SQL - IMDb data

### 개요

인터넷 영화 데이터베이스 (Internet Movie Database)에서 SQL 쿼리를 작성하는 과제입니다. 이 과제를 통해 (1) 여러 가지 SQL 기능을 공부할 수 있고, (2) SQLite와 같은 DBMS의 사용에 익숙해지는 기회를 가질 수 있습니다.

- 제출기한: 2025년 4월 14일(월) 23:59 (KST)

### 사양

과제는 총 6개의 문제로 구성되어 있으며 100점 만점으로 채점됩니다. 각 문제마다, SQLite DBMS에서 원하는 데이터를 가져오는 SQL 쿼리를 작성해야 합니다. 문제를 완료하는 데에는 약 1-5 시간이 소요될 수 있습니다.

### 제출 형식

1번 문제의 정답은 q1.sql, 2번 문제의 정답은 q2.sql, ..., 6번 문제의 정답은 q6.sql로 저장한 후 모든 sql 파일을 학번.zip으로 압축하여 제출하세요. (예: 2020203000.zip) 자동으로 채점하므로 형식에 맞게 제출하지 않은 경우 0점 처리될 수 있습니다.

### 준비하기

#### 데이터베이스 불러오기

1. 다음의 파일을 다운로드 합니다:  
<https://15445.courses.cs.cmu.edu/fall2022/files/imdb-cmudb2022.db.gz>
2. 다운로드한 파일의 압축을 풀어 용량이 800 MB 이상인지 확인합니다: 876,220,416 bytes.
3. 다음의 명령어로 sqlite를 실행합니다:

```
$ sqlite3 imdb.db
```

4. **.tables** 명령어로 다음과 같이 6개의 테이블이 있는지 확인합니다:

```
$ sqlite3 imdb.db
SQLite version 3.41.1 2023-03-10 12:13:52
Enter ".help" for usage hints.
sqlite> .tables
akas      crew      episodes  people    ratings   titles
```

5. 다음의 명령어로 인덱스를 만듭니다:  
(인덱스 생성 전 후로 **.index** 명령어 실행하여 비교가능)

```
CREATE INDEX ix_people_name ON people (name);
CREATE INDEX ix_titles_type ON titles (type);
CREATE INDEX ix_titles_primary_title ON titles (primary_title);
CREATE INDEX ix_titles_original_title ON titles (original_title);
CREATE INDEX ix_akas_title_id ON akas (title_id);
CREATE INDEX ix_akas_title ON akas (title);
CREATE INDEX ix_crew_title_id ON crew (title_id);
CREATE INDEX ix_crew_person_id ON crew (person_id);
```

## 스키마 확인하기

각 테이블의 스키마를 통해 어떤 속성들을 포함하고 있는지, 기본 키와 외래 키는 무엇인지 살펴 보세요. 각 테이블에 대해 **sqlite3** 터미널에서 **.schema \$TABLE\_NAME** 명령을 실행하세요. 각 테이블마다 아래 예시와 같은 출력이 나와야 합니다:

### PEOPLE

```
sqlite> .schema people
CREATE TABLE people (
  person_id VARCHAR PRIMARY KEY,
  name VARCHAR,
  born INTEGER,
  died INTEGER
);
CREATE INDEX ix_people_name ON people (name);
```

people 테이블은 다음과 같이 인물에 관한 정보를 담고 있습니다:

person_id	name	born	died
nm0000004	John Belushi	1949	1982
nm0000005	Ingmar Bergman	1918	2007
nm0000006	Ingrid Bergman	1915	1982
nm0000011	Gary Cooper	1901	1961
nm0000018	Kirk Douglas	1916	2020

**person\_id**는 인물 ID, **name**은 이름, **born**은 출생 연도, **died**는 사망 연도를 의미합니다. 이름이 같아도 **person\_id**가 다르면 다른 사람으로 간주합니다.

## TITLES

```
sqlite> .schema titles
CREATE TABLE titles (
  title_id VARCHAR PRIMARY KEY,
  type VARCHAR,
  primary_title VARCHAR,
  original_title VARCHAR,
  is_adult INTEGER,
  premiered INTEGER,
  ended INTEGER,
  runtime_minutes INTEGER,
  genres VARCHAR
);
CREATE INDEX ix_titles_type ON titles (type);
CREATE INDEX ix_titles_primary_title ON titles (primary_title);
CREATE INDEX ix_titles_original_title ON titles (original_title);
```

titles 테이블은 작품에 관한 정보를 담고 있습니다:

title_id	type	primary_title	premiered	genres
tt0000010	short	Leaving the Factory	1895	Documentary,Short
tt0000033	short	Horse Trick Riders	1895	Comedy,Documentary,Short
tt0000037	short	Sea Bathing	1896	Short
tt0000050	short	Bebe et fillettes	1896	Documentary,Short
tt0000062	short	Danse serpentine	1896	Short

여러 어트리뷰트 중 과제에서는 **title\_id** (작품 ID), **type** (작품 유형), **primary\_title** (제목), **premiered** (공개 연도), **genres** (장르), **runtime\_minutes** (상영 시간)을 사용합니다.

## AKAS

```
sqlite> .schema akas
CREATE TABLE akas (
  title_id VARCHAR, -- REFERENCES titles (title_id),
  title VARCHAR,
  region VARCHAR,
  language VARCHAR,
  types VARCHAR,
  attributes VARCHAR,
  is_original_title INTEGER
);
CREATE INDEX ix_akas_title_id ON akas (title_id);
CREATE INDEX ix_akas_title ON akas (title);
```

akas 테이블은 영화가 외국어로 더빙이 된 경우 더빙 버전의 영화 제목에 대한 정보를 아래와 같이 담고 있습니다:

```
sqlite> SELECT title_id, title, language
...> FROM akas
...> WHERE language IS NOT NULL
...> LIMIT 5;
```

title_id	title	language
tt0000010	Lumiere-fabriken's arbetare	sv
tt0000010	La sortie de l'usine Lumiere a Lyon	en
tt0000010	工場の出口	ja
tt0000013	The Photographical Congress Arrives in Lyon	en
tt0000016	Baten lamnar hamnet	sv

이 번 과제에서는 akas 테이블을 사용하지 않습니다.

## CREW

```
sqlite> .schema crew
CREATE TABLE crew (
  title_id VARCHAR, -- REFERENCES titles (title_id),
  person_id VARCHAR, -- REFERENCES people (person_id),
  category VARCHAR,
  job VARCHAR,
  characters VARCHAR
);
CREATE INDEX ix_crew_title_id ON crew (title_id);
CREATE INDEX ix_crew_person_id ON crew (person_id);
```

crew 테이블은 작품 제작에 참가한 사람들의 정보를 담고 있습니다:

title_id	person_id	category
-----	-----	-----
tt0000003	nm5442200	editor
tt0000005	nm0249379	producer
tt0000007	nm0249379	producer
tt0000009	nm0085156	director
tt0000010	nm0525910	director

여러 어트리뷰트 중 과제에서는 **title\_id** (작품 ID), **person\_id** (인물 ID), **category** (역할) 만 사용합니다. **title\_id**는 titles의 **title\_id**를 참조하고, **person\_id**는 people의 **person\_id**를 참조합니다.

## RATINGS

```
sqlite> .schema ratings
CREATE TABLE ratings (
  title_id VARCHAR PRIMARY KEY, -- REFERENCES titles (title_id),
  rating FLOAT,
  votes INTEGER
);
```

ratings 테이블은 작품에 대한 평점 정보를 지니고 있습니다:

```
sqlite> .schema ratings
CREATE TABLE ratings (
  title_id VARCHAR PRIMARY KEY, -- REFERENCES titles (title_id),
  rating FLOAT,
  votes INTEGER
);
```

**title\_id**는 titles의 **title\_id**를 참조합니다.

## 무결성 검사

**titles** 테이블이 갖고 있는 튜플의 개수는 다음과 같습니다:

```
sqlite> SELECT COUNT(*) FROM titles;
COUNT(*)
-----
1375462
```



## SQL 질의문 작성하기

각 답안은 **하나의 질의문**으로 작성해야 합니다. 하나의 질문에 대해 여러 개의 질의문을 작성한 경우 가장 처음 질의문만 실행되며 **모든 문제에서 부분점수는 없습니다.**

### 문제 1: (10점) q1.sql

crew 테이블의 category를 중복 없이 사전순으로 출력하세요.

### 문제 2: (10점) q2.sql

Documentary 작품 중 상영시간이 가장 짧은 작품 10개를 **상영시간이 짧은 순서**대로 출력하세요.

**세부사항:** 작품의 primary\_title, premiered, runtime\_minutes를 출력해야 합니다. 작품의 genres 어트리뷰트에 Documentary가 포함되어 있으면 Documentary 영화로 취급해야 하며 **runtime\_minutes가 동일한 경우 primary\_title을 사전순으로 정렬**하여 출력합니다.

출력의 첫 줄은 다음과 같습니다: **Les nouveaux barbares|1995|0**

### 문제 3: (20점) q3.sql

세상을 떠난 (타계한, 죽은) 작품 감독 중 가장 오래 산 (장수한, 나이가 많은) 감독 20인을 **나이가 많은 순서**대로 출력하세요.

**세부사항:** 하나 이상의 작품에서 director 역할을 맡은 경우 해당 인물을 감독으로 간주하고 나이는 간단하게 사망 연도 - 출생 연도로 계산합니다. **이름|나이** 형태로 출력해야 하며 출력

중 첫 줄은 다음과 같습니다: **Manoel de Oliveira|107**. 나이가 많은 순서로 출력하고 **나이가 동일한 경우 이름을 사전순으로 정렬**하여 출력합니다.

#### 문제 4: (20점) q4.sql

배우로서 (actor or actress) crew 테이블에 가장 많이 등장하는 20인을 출력하세요.

**세부사항:** 배우로서 crew 테이블에 가장 많이 등장하는 20인을 **등장 횟수가 많은 순으로 이름|등장횟수**와 같이 출력합니다. 배우가 아닌 역할로 작품 제작에 참여한 경우 해당 튜플은 등장 횟수에 더하지 않습니다. 출력 중 첫 줄은 다음과 같습니다: **Vic Sotto|10509**

#### 문제 5: (20점) q5.sql

2010년부터 (2010년 포함) 매 년 가장 높은 평점을 받은 비디오 게임을 출력하세요.

**세부사항:** **발매 연도순**으로 (2010, 2011, ...) **발매연도|제목|평점**의 형식으로 출력합니다. 비디오 게임의 type은 videoGame입니다. 한 해에 가장 높은 평점을 받은 작품이 **여러 개 있다면 이들을 모두 제목의 사전순**으로 출력합니다. 출력 중 첫 줄은 다음과 같습니다: **2010|Red Dead Redemption|9.5**

#### 문제 6: (20점) q6.sql

1974년에 태어나고 이름에 “Leonardo”를 포함한 사람들이 참여한 작품 중 가장 인기 있는 작품들을 10개 출력하세요.

**세부사항:** 1974년에 태어나고 이름에 “Leonardo”를 포함한 사람들이 참여한 작품 중 ratings 테이블에서 가장 많은 votes를 받은 10개를 **votes가 많은 순**으로 출력합니다. **제목|votes**의 형식으로 출력하며 출력 중 첫 줄은 다음과 같습니다: **Inception|2306926**