4. SQL문 조작하기 (Cloud9에서 연습했던 내용들)

4-1. CREATE

1. 터미널을 통해 sqlite3 접근하기

```
$ sqlite3
```

```
SQLite version 3.8.2 2013-12-06 14:53:30
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

2. DB 스키마 정의하기

```
sqlite> CREATE TABLE movies (
   ...>     영화코드 INTEGER PRIMARY KEY,
   ...>     영화이름 TEXT,
   ...>     관람등급 TEXT,
   ...>     감독 TEXT,
   ...>     개봉연도 INTEGER,
   ...>     누적관객수 INTEGER,
   ...>     상영시간 INTEGER,
   ...>     제작국가 TEXT,
   ...>     장르 TEXT
   ...> );
```

3. CSV 파일을 가져와서 DB로 만들기

CSV 파일 가져오기

```
sqlite> .mode csv
sqlite> .import boxoffice.csv movies
```

예쁘게 보기

```
sqlite> .headers on
sqlite> .mode column
```

4. movies 테이블의 전체 데이터 출력하기

```
sqlite> SELECT * FROM movies;
```

```
20010291,"해리포터와 마법사의 돌","전체관람가","크리스 콜럼버스",20011213,259733,152,"미국","가족"
20020186,"헤드윅","15세관람가","존 카메론 밋첼",20020809,31740,91,"미국","뮤지컬"
20040521,"클레멘타인","15세관람가","김두영",20040521,67000,100,"한국","액션
...
```

4-2. CRUD

1. `INSERT`: 새로운 행(레코드) 삽입 (데이터 추가)

- 원하는 열에만 데이터를 넣고 싶으면 Column을 명시하자

```
sqlite> INSERT INTO classmates(name, age)
   ...> VALUES ('홍길동', 23)
```

- 모든 열에 데이터를 넣을 땐 Column을 명시할 필요가 없다.

```
sqlite> INSERT INTO movies
   ...> VALUES (20182530, '극한직업', '15세관람가', '이병헌', 20190123, 3138467, 111, '한국', '코 미디');
```

2. `DELETE`: 영화코드가 20185124인 행(레코드) 출력

```
sqlite> SELECT * FROM movies
     ...> WHERE 영화코드 = 20185124;
```

3. `UPDATE`: 영화코드가 20185124인 행(레코드)의 감독 값을 '없음'으로 변경

- 특정 Table의 특정한 레코드를 수정한다.
  - movies 테이블에서
  - 영화코드가 20185124인 행(레코드)을 불러와서
  - '감독' column의 value 값을 '없음'으로 수정한다.

```
sqlite> UPDATE movies
   ...> SET 감독 = '없음'
   ...> WHERE 영화코드 = 20185124;
```

4. `DELETE`: 영화코드가 20040521인 행(레코드) 삭제

- 특정 Table의 특정한 레코드를 삭제한다.
- 중복이 불가능한(Unique) 값인 id를 기준으로 삭제하는게 좋다.

```
sqlite> DELETE FROM movies
   ...> WHERE 영화코드 = 20040521;
```

4-3. SELECT

1. `>=`: 상영시간 150분 이상인 레코드의 영화이름 출력

```
sqlite> SELECT 영화이름 FROM movies
   ...> WHERE 상영시간 >= 150;
```

```
"영화이름"
"해리포터와 마법사의 돌"
"다크 나이트"
"아바타"
"트랜스포머: 최후의 기사"
"블레이드 러너 2049"
"스타워즈: 라스트 제다이"
"당갈"
```

2. `=`: 장르가 애니메이션인 레코드의 영화코드와 영화이름 출력

```
sqlite> SELECT 영화코드, 영화이름 FROM movies
   ...> WHERE 장르 = '애니메이션';
```

```
"영화코드","영화이름"
20078561,"명탐정 코난:감벽의 관"
20135305,"극장판 포켓몬스터DP - 디아루가 VS 펄기아 VS 다크라이"
20161665,"스머프: 비밀의 숲"
20161744,"트롤"
20162607,"슈퍼 버드"
```

3. `AND`: 누적관객 10만 이상 및 관람등급 청불인 레코드의 영화이름, 누적관객수 출력

```
sqlite> SELECT 영화이름, 누적관객수 FROM movies
   ...> WHERE 누적관객수 >= 100000 AND 관람등급 = '청소년관람불가';
```

```
"영화이름","누적관객수"
"바람 바람 바람",1191776
"로건",2155751
"프리즌",2920530
"악녀",1192888
"브이아이피",1364953
```

4. `LIKE`: 개봉연도가 2000~2009년인 레코드의 영화이름 출력

- 정확한 값에 대한 비교가 아닌, **패턴을 확인하여 해당하는 값을 반환**한다.

%	2%	2로 시작하는 값
	%2	2로 끝나는 값
	%2%	2가 들어가는 값
_	_2%	아무값이나 들어가고 두번째가 2로 시작하는 값
	1___	1로 시작하고 4자리인 값
	2_%_%	2로 시작하고 적어도 3자리인 값

```
sqlite> SELECT 영화이름 FROM movies
   ...> WHERE 개봉연도 LIKE '200%';
```

```
"영화이름","개봉연도"
"해리포터와 마법사의 돌",20011213
"헤드윅",20020809
"클레멘타인",20040521
"이프 온리",20041029
"원스",20070920
"다크 나이트",20080806
"아바타",20091217
```

```
sqlite> SELECT 영화이름 FROM movies
   ...> WHERE 영화이름 LIKE '해%';
```

```
"영화이름"
"해리포터와 마법사의 돌"
"해빙"
"해피 데스데이"
"해피 투게더
```

5. `DISTINCT`: 장르 값을 중복 없이 출력

```
sqlite> SELECT DISTINCT 장르 FROM movies;
```

```
"장르"
"가족"
"뮤지컬"
"액션"
"코미디"
"드라마"
```

6. `COUNT, DISTINCT`: 장르의 개수를 중복 없이 출력

```
sqlite> SELECT COUNT(DISTINCT 장르) FROM movies;
```

```
18
```

4-4. Expressions (표현식)

- AVG, SUM, MIN, MAX 표현식은 기본적으로 숫자(INTEGER)일 때만 가능하다.

1. `SUM`: 모든 영화의 총 누적관객수 출력

```
sqlite> SELECT SUM(누적관객수) FROM movies;
```

```
426433183
```

2. `MAX`: 가장 많은 누적관객수의 영화이름과 누적관객수 출력

```
sqlite> SELECT 영화이름, MAX(누적관객수) FROM movies;
```

```
"영화이름","MAX(누적관객수)"
"신과함께-죄와 벌",1439811
```

3. `MIN`: 상영시간이 가장 짧은 영화의 장르와 상영시간 출력

```
sqlite> SELECT 장르, MIN(상영시간) FROM movies;
```

```
"장르","MIN(상영시간)"
"애니메이션",5
```

4. `AVG`: 제작국가가 한국인 영화의 평균 누적관객수 출력

```
sqlite> SELECT AVG(누적관객수) FROM movies
   ...> WHERE 제작국가 = '한국';
```

```
"AVG(누적관객수)"
1615273.94615385
```

5. `COUNT`: 관람등급이 청소년관람불가인 영화의 개수 출력

```
sqlite> SELECT COUNT(*) FROM movies
   ...> WHERE 관람등급 = '청소년관람불가';
```

```
"COUNT(관람등급)"
27
```

6. `COUNT` : 상영시간이 100분 이상이고 장르가 애니메이션인 영화의 개수 출력

```
sqlite> SELECT COUNT(*) FROM movies
   ...> WHERE 상영시간 >= 100 AND 장르 = '애니메이션';
```

```
COUNT(*)
22
```

7. `LIMIT` : 상영시간이 100분 이상인 영화이름과 상영시간을 `위에서부터 3개` 만 출력

```
SELECT 영화이름, 상영시간 FROM movies
WHERE 상영시간 >= 100 LIMIT 3;
```

```
"영화이름","상영시간"
"해리포터와 마법사의 돌",152
"클레멘타인",100
"명탐정 코난:감벽의 관",106
```

8. `LIMIT` : 장르이름 중복없이 `위에서부터 5개` 만 출력

```
SELECT DISTINCT 장르 FROM movies LIMIT 5;
```

```
"가족" (1)
"뮤지컬" (2)
"액션" (3)
"코미디" (4)
"드라마" (5)
```

9. `LIMIT, OFFSET` : 장르이름 중복없이 `위에서 3번째 행부터` `3개` 만 출력

```
SELECT DISTINCT 장르 FROM movies LIMIT 3 OFFSET 2;
```

```
"액션" (3)
"코미디" (4)
"드라마" (5)
```

10. `ORDER BY(DESC)` : 상영시간으로 내림차순 정렬하여, 상위 10개만 출력

```
SELECT 영화이름, 상영시간 FROM movies
ORDER BY 상영시간 DESC LIMIT 10;
```

```
"블레이드 러너 2049",163
"아바타",161
"당갈",160
"해리포터와 마법사의 돌",152
"다크 나이트",152
"스타워즈: 라스트 제다이",151
"트랜스포머: 최후의 기사",150
"어벤져스: 인피니티 워",149
"버닝",147
"미션 임파서블: 폴아웃",147
```

11. `ORDER BY(ASC)` : 상영시간으로 오름차순 정렬하여, 상위 5개만 출력

```
SELECT 영화이름, 상영시간 FROM movies
ORDER BY 상영시간 ASC LIMIT 5;
```

```
"바다 탐험대 옥토넛 시즌4: 더 파이널",57
"극장판 파워레인저: 애니멀포스 VS 닌자포스 미래에서 온 메시지",57
"리틀 프린세스 소피아 : 신비한 섬",63
"극장판 숲의 요정 페어리루 ~크리스마스의 기적: 마법의 날개~",64
"극장판 프리파라 모두의 동경♪ 렛츠고☆프리파",65
```