

# 실전 프로젝트1





9

# SQLite(2)

# SQLite 설치

- 데이터베이스 연결 또는 생성
  - `sqlite3` [데이터베이스 파일 명]
- 명령어
  - `.help`
  - `.tables`
  - `.header on`
  - `.schema`
  - `.exit or .quit`

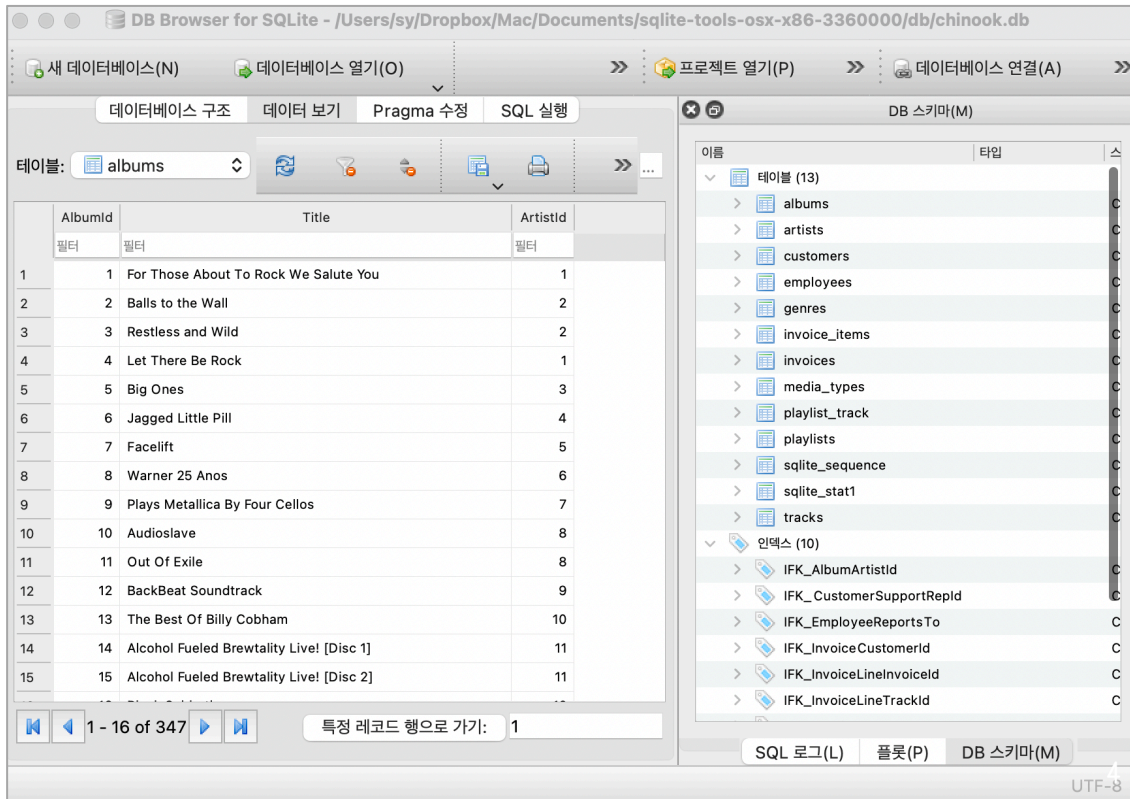
```
% sqlite3 ./db/chinook.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> .tables
albums          employees       invoices        playlists
artists         genres         media_types     tracks
customers       invoice_items  playlist_track
sqlite> .exit
```

# DB Browser for SQLite 설치

- DB관리를 위한 GUI tool
- <https://sqlitebrowser.org/>
- 버전 선택 후 다운로드 및 설치
  - Version 3.12.2 released
- chinook.db 열기



<https://sqlitebrowser.org/>



# 데이터베이스 용어

Database

테이블 (13)
albums
artists
customers
employees
genres
invoice_items
invoices
media_types
playlist_track
playlists
sqlite_sequence
sqlite_stat1
tracks

Table



테이블 (13)
albums
AlbumId
Title
ArtistId

Field

Column

Name/type/length/null

Data or Record

테이블:  albums 

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼


▼

# 데이터베이스 생성 및 연결

- 기존 데이터베이스 연결
  - `sqlite3` [기존 데이터베이스 파일명]
- 새 데이터베이스 생성
  - `sqlite3` [새 데이터베이스 파일명]

```
% ls
chinook.db
% sqlite3 chinook.db
SQLite version 3.28.0 2019-04-15 14:49:49
Enter ".help" for usage hints.
sqlite> .tables
albums          employees       invoices        playlists
artists         genres         media_types    tracks
customers       invoice_items  playlist_track
sqlite> .quit
% sqlite3 myfirst.db
SQLite version 3.28.0 2019-04-15 14:49:49
Enter ".help" for usage hints.
sqlite>
```

# 테이블 생성

- 필드정의
  - 필드명/필드타입/Null 여부 등
- SQLite 필드타입 
  - NULL / TEXT / BLOB ← NVARCHAR = 문자열 → 이진식 (Binary large Object : 바이너리 오브젝트)
  - INTEGER(int) signed 1,2,3,4,5,6,8 byte number.
  - REAL: floating point value. 8 byte.
- 테이블 생성 SQL

테이블: artists

	ArtistId	Name
	필터	필터
1	1	AC/DC
2	2	Accept
3	3	Aerosmith
4	4	Alanis Morissette
5	5	Alice In Chains

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    ....  
);
```

```
CREATE TABLE artists  
(  
    ArtistId INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    Name NVARCHAR(120)  
);
```

→ unique 한 키로 사용할 field (기본키)  
→ 자동으로 1씩 증가해서 할당함 (이 필드값을 직접 넣어서는 안됨)  
→ NULL값이 들어오면 안됨

# 테이블 생성 예제

- 테이블명 : g\_artists
- 필드구성
  - 가수명 - 텍스트
  - 활동유형 - 텍스트
  - 활동연대 - 텍스트
  - 데뷔 - 텍스트
  - 일련번호 - 정수
  - 등록일자 - 텍스트
- 테이블 생성 SQL 작성

## 아이유 (IU)

활동유형	여성
활동연대	2000년대, 2010년대, 2020년대
데뷔	2008년 / 미아
국적	한국

```
CREATE TABLE g_artists (  
  id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
  name text,  
  a_type text,  
  a_year text,  
  debut text,  
  regdate text NOT NULL  
);
```



# CRUD SQL

## 테이블에 데이터 추가 SQL

▷ 어떤 field에  
데이터를 추가할 것인지

```
INSERT INTO tablename (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

번호	필드명	타입	설명
1	id	정수	일련번호
2	name	텍스트	이름
3	a_type	텍스트	활동유형
4	a_year	텍스트	활동연대
5	debut	텍스트	데뷔
6	regdate	텍스트	등록일자

### 아이유 (IU)

활동유형	여성
활동연대	2000년대, 2010년대, 2020년대
데뷔	2008년 / 미아
국적	한국

```
sqlite> .tables  
artists      g_artists  
sqlite> insert into g_artists (name, a_type, a_year,  
debut, regdate) values ('아이유', '여성', '2000년대', '2008  
년', '2021-09-28 00:09:51');  
sqlite> insert into g_artists (name, a_type, a_year,  
debut, regdate) values ('거미', '여성', '2000년대, 2010년',  
'2003년', datetime('now', 'localtime'));  
sqlite> select * from g_artists;  
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21
```

문자열은 꼭 single 따옴표로!  
(double 따옴표는 안됨)  
현재 시간을 넣어주는 함수

# CRUD SQL

## ○ 데이터 조회 SQL

```
SELECT * FROM tablename;  
SELECT * FROM tablename order by column1;  
SELECT * FROM tablename order by column1 desc;  
SELECT * FROM tablename where column1 = 'keyword';  
SELECT column1, column2 FROM tablename where column1 like 'keyword%';
```

이름 keyword를 포함하는

```
sqlite> select * from g_artists;
```

```
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21
```

```
sqlite> .header on → 데이터를 가져왔을 때 첫 줄을 보이기
```

```
sqlite> select * from g_artists order by name;  
id|name|a_type|a_year|debut|regdate  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21  
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51
```

```
sqlite> select * from g_artists order by name desc;  
id|name|a_type|a_year|debut|regdate
```

```
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21
```

```
sqlite> select name, debut from g_artists where name = '거미';  
name|debut
```

```
거미|2003년
```

```
sqlite> select * from g_artists where name like '%이%';  
id|name|a_type|a_year|debut|regdate  
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51
```

# CRUD SQL

- 데이터 수정 SQL

```
UPDATE tablename  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

- 예시) 필드 : **debut**, 데이터 : **2008년 -> 2008년 / 미아** 변경

```
sqlite> select * from g_artists where name='아이유';  
id|name|a_type|a_year|debut|regdate  
1|아이유|여성|2000년대|2008년|2021-09-28 00:09:51  
sqlite> update g_artists set debut='2008년 / 미아' where name='아이유';  
sqlite> select * from g_artists where name = '아이유';  
id|name|a_type|a_year|debut|regdate  
1|아이유|여성|2000년대|2008년 / 미아|2021-09-28 00:09:51
```

# CRUD SQL

- 데이터 삭제 SQL

```
DELETE FROM tablename WHERE condition;
```

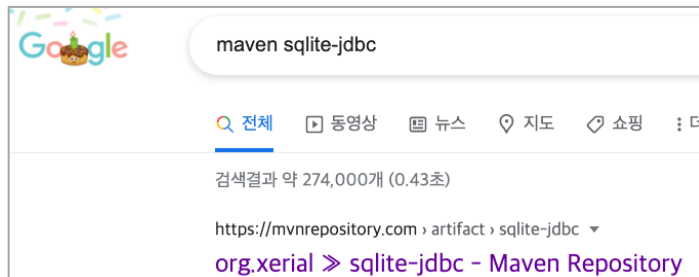
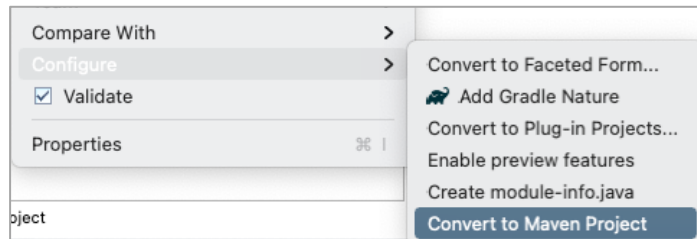
```
sqlite> insert into g_artists (name, a_type, regdate) values ('테스트 가수', '유형',  
'2021-09-28 10:00:00');  
sqlite> select * from g_artists;  
id|name|a_type|a_year|debut|regdate  
1|아이유|여성|2000년대|2008년 / 미아|2021-09-28 00:09:51  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21  
3|테스트 가수|유형|||2021-09-28 10:00:00  
sqlite> delete from g_artists where id = 3;  
sqlite> select * from g_artists;  
id|name|a_type|a_year|debut|regdate  
1|아이유|여성|2000년대|2008년 / 미아|2021-09-28 00:09:51  
2|거미|여성|2000년대, 2010년|2003년|2021-09-29 00:15:21
```

# Java 와 SQLite 연동

- 새 프로젝트 생성 : SQLiteProject
- Maven project로 변환 : Project 선택 > Configure > Convert to Maven Project
- 새 클래스 추가 : Main class
  - public static void main(String [] args) 체크
- “Hello World!!!” 출력 및 실행
- 라이브러리 추가 : 구글검색 “maven sqlite-jdbc”
  - pom.xml 에 dependency 추가

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/
  org.xerial/sqlite-jdbc -->
    <dependency>
      <groupId>org.xerial</groupId>
      <artifactId>sqlite-jdbc</artifactId>
      <version>3.34.0</version>
    </dependency>
</dependencies>
```

pom.xml



# Java 와 SQLite 연동

- java.sql.Connection : 데이터베이스 연결, 종료
- java.sql.Statement : SQL 구문 전달하여 실행
  - ResultSet executeQuery (String sql) : select 쿼리 실행, 결과를 ResultSet으로 가져옴
  - int executeUpdate (String sql) : insert into, update, delete 쿼리 실행

↳ Connection class를 통해 데이터베이스와 연결  
**Statement** stmt = conn.createStatement();  
**stmt.executeUpdate**("insert into artists (name) values (" + name + " ');");

- java.sql.PreparedStatement : SQL 구문 전달하는 역할, Statement 객체 기능 향상

**PreparedStatement** pstmt = conn.prepareStatement("insert into artists (name) values(?)");  
pstmt.setString(1, name);  
**pstmt.executeUpdate()**;

- java.sql.ResultSet : select 쿼리 결과를 저장하는 객체

# 데이터조회

- 데이터베이스파일 프로젝트에 복사
- JDBC API 코딩 방법
- (SQLite) JDBC Driver 로드
- SQLite DB 파일에 연결
- Statement 객체 생성
- SQL 구문생성 및 실행(select 문)
- ResultSet로 결과 가져오기
- 데이터 화면에 출력
- Statement 객체 종료
- Connection 객체 종료



```
*** 데이터 조회 ***
1 아이유
2 거미
6 테스트 가수
```

```
public class Main {
    public static void main(String[] args) {
        Connection con = null;
        try {
            // SQLite JDBC 체크
            Class.forName("org.sqlite.JDBC");

            // SQLite 데이터베이스 파일에 연결
            String dbFile = "myfirst.db";
            con = DriverManager.getConnection("jdbc:sqlite:" + dbFile);

            // 데이터 조회
            System.out.println("\n*** 데이터 조회 ***");
            Statement stat1 = con.createStatement();
            String sql1 = "select * from g_artists";
            ResultSet rs1 = stat1.executeQuery(sql1);
            while (rs1.next()) {
                String id = rs1.getString("id");
                String name = rs1.getString("name");
                System.out.println(id + " " + name);
            }
            stat1.close();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (con != null) {
                try {
                    con.close();
                } catch (Exception e) {}
            }
        }
    }
}
```

# 데이터 추가

## ○ JDBC API 코딩 방법

- JDBC Driver 로드
- SQLite DB 파일에 연결
- Statement 객체 생성
- SQL 구문 생성 및 실행
  - insert into 문
- 실행결과를 가지고 성공 평가
- Statement 객체 종료
- Connection 객체 종료

```
System.out.println(id + " " + name);
}
stat1.close();
```

```
// 데이터 추가
System.out.println("\n*** 새 데이터 추가 ***");
Statement stat2 = con.createStatement();
String sql2 = "insert into g_artists (name, a_type, a_year, debut, regdate)"
    + " values ('방탄소년단', '남성', '2010년대', '2013년', datetime('now', 'localtime'))";
int cnt = stat2.executeUpdate(sql2);
if(cnt > 0)
    System.out.println("새로운 데이터가 추가되었습니다!");
else
    System.out.println("[Error] 데이터 추가 오류!");
stat2.close();
```

*※ cnt 이 0 이 아닌 1 이면 새로운 레코드가 추가 됨*

```
} catch (Exception e) {
```

테이블: g\_artists

	id	name	a_type	a_year	debut	regdate
...	필터	필터	필터	필터	필터	
1	1	아이유	여성	2000년대	2008년 / 미아	2021-09-28 00:09:51
2	2	거미	여성	2000년대, 2010년	2003년	2021-09-29 00:15:21
3	6	테스트 ...	유형	NULL	NULL	2021-09-28 10:00:00
4	7	방탄소년단	남성	2010년대	2013년	2021-09-29 16:31:16

## ○ 데이터베이스 파일 열기로 확인필요



# 데이터수정

- JDBC API 코딩 방법

- JDBC Driver 로드
- SQLite DB 파일에 연결
- Statement 객체 생성
- SQL 구문생성 및 실행
  - Update 문
- 실행결과를 가지고 성공 평가
- Statement 객체 종료
- Connection 객체 종료

- 데이터베이스 파일 열기로 확인필요

```
System.out.println("[Error] 데이터 수정 오류. ",  
stat2.close());
```

```
// 데이터 수정  
System.out.println("\n*** 데이터 수정 ***");  
Statement stat3 = con.createStatement();  
String sql3 = "update g_artists set a_year = '2000년대, 2010년대, 2020년대' "  
            + "where id=1 ";  
int cnt3 = stat3.executeUpdate(sql3);  
if(cnt3 > 0)  
    System.out.println("데이터가 수정되었습니다!");  
else  
    System.out.println("[Error] 데이터 수정 오류!");  
stat3.close();
```

\*\*\* 데이터 수정 \*\*\*  
데이터가 수정되었습니다!

1	1	아이유	여성	2000년대	2008년 / 미아	2021-09-28 00:09:51
---	---	-----	----	--------	------------	---------------------



1	1	아이유	여성	2000년대, 2010년대, 2020년대	2008년 / 미아	2021-09-28 00:09:51
---	---	-----	----	------------------------	------------	---------------------

# 데이터 삭제

- JDBC API 코딩 방법

- JDBC Driver 로드
- SQLite DB 파일에 연결
- Statement 객체 생성
- SQL 구문 생성 및 실행
  - delete 문
- 실행결과를 가지고 성공 평가
- Statement 객체 종료
- Connection 객체 종료

```
System.out.println("데이터 삭제 오류!");  
stat3.close();
```

```
// 데이터 삭제  
System.out.println("\n*** 데이터 삭제 ***");  
Statement stat4 = con.createStatement();  
String sql4 = "delete from g_artists where id=6 ;"  
int cnt4 = stat4.executeUpdate(sql4);  
if(cnt4 > 0)  
    System.out.println("데이터가 삭제되었습니다!");  
else  
    System.out.println("[Error] 데이터 삭제 오류!");  
stat4.close();
```

\*\*\* 데이터 삭제 \*\*\*  
데이터가 삭제되었습니다!

- 데이터베이스 파일 열기로 확인필요

	id	name	a_type	a_year	debut	regdate
...	필터	필터	필터	필터	필터	필터
1	1	아이유	여성	2000년대, 2010년대, 2020년대	2008년 / 미아	2021-09-28 00:09:51
2	2	거미	여성	2000년대, 2010년	2003년	2021-09-29 00:15:21
3	7	방탄소년단	남성	2010년대	2013년	2021-09-29 16:31:16

# 5주차 실습과제 제출

- Due date - 2021년 10월 3일 (일) 밤 11시 59분
- 제출방법 - LMS 과제로 PDF 파일 제출
- PDF 파일 내 포함 내용
  - 파일명 형식 : 이름(학번)\_5주차.pdf
  - 5주차 실습 수행을 통해 업데이트된 프로젝트 실행 결과
    - 프로젝트 최종 소스를 Commit & Push 된 자신의 Github URI 기입(새로운 Repo 필요)
    - SQLite Database 생성 및 테이블 생성에 필요한 실습 과정 이미지 첨부 및 간단 설명 입력
      - DB명 : myfirst.db, Table 2개 이름 : artists, g\_artists
    - 데이터를 5개 이상 등록하고, 조회, 수정, 삭제 연습 후 사용된 SQL문과 최종 조회 캡처 이미지 첨부
    - 새로운 프로젝트 생성, java와 SQLite 연동하여 CRUD 실습 후 각 소스와 결과 화면 캡처
  - 5주차 실습을 수행하면서 느낀 점