



# 데이터베이스(sql/nosql)

# 시간계획

- 오늘도 파이팅 입니다.^ ^

시간	학습내용
09:00~10:00	Django 프로젝트 deploy 실습
10:20~11:20	Sql 실습 환경 셋팅
11:40~12:40	SQLite DB를 만들고 sql 기본 익히기
12:40~14:00	즐거운 점심 시간
14:00~15:00	DB DML 실습하기
15:20~16:20	DB 연동 python 프로그래밍-Create, Read
16:40~17:50	DB 연동 python 프로그래밍-Update, Delete

# 머신러닝 기반 데이터 분석, 예측 파트 진행 순서



## 1) 분석 및 예측

시각화, 머신러닝:

- python
- numpy, Pandas
- Matplotlib, Seaborn
- 기초통계
- Scikit learn

- 탐색적 데이터 분석 방법으로 데이터를 분석함  
- 분석 데이터를 시각화 하는 방법을 익힘  
- 머신러닝 이해 하고 사이킷런 활용해 다양한 머신러닝 모델을 만들고 평가하는 방법 적용



## 2) Web pgm 기본

Front end side :

- HTML5
- CSS3
- Javascript
- jQuery

- 웹에 산재되어 있는 데이터를 수집, 분석하기 위한 웹 문서 표현 기술인 웹 표준 활용 능력 익힘.  
- 웹 데이터의 구조 이해



## 3) 데이터 저장

Back end side:

- Django : python 기반 web server 프레임워크
- Mysql - CRUD
- MongoDB(js기반)

- 데이터를 구조화 하여 저장하는 방법 적용  
- 정형, 비정형 데이터 유형을 이해하고, 저장하는 방법 적용  
- 클라우드 서비스 이해, 프리티어 서비스를 활용해 웹서비스 구현



## 4) 데이터 수집가공

웹 크롤링 & 스크래핑:

- Python 기반
- BeautifulSoup
- Selenium
- 머신러닝 통합 예제
- Linux shell pg

- 웹크롤링 및 스크래핑 기술을 적용  
- 웹에 산재되어 있는 데이터를 수집, 가공, 파일로 저장하는 방법 활용  
- 데이터 분석을 위해 전처리 방법을 익힘



## 5) 팀 협업 프로젝트



- 의미 있는 도출을 위한 팀 주제 정하기
- 웹크롤링, 오픈데이터
- 데이터 DB 저장
- 데이터 분석, 시각화
- 머신러닝 예측
- 웹 서비스로 구현하기

클라우드 서비스  
AWS

# DB 수업내용



- DB, RDBMS 이해
- SQL실습환경 셋팅
- SQLite, SQL 기본 익히기
- SQLite DB연동, python 프로그래밍
- mysql 설치, 설정, 사용자 생성
- SQL(with mysql)
  - CML(CRUD), DDL
  - Join, SQL function
- Nosql(mongoDB)

# 실습 내용



- 실습 환경 구성
  - 주피터 노트북, VS code, 파이썬 가상환경, Dbeaver
- SQLite로 SQL 기본 문법 실습
- SQLite 연동 python 프로그래밍 실습



## 학습목표

Ubuntu Linux에 DB실습환경을 셋팅 한다.

실습을 통해 RDBMS, DB, Table의 구조를 이해한다.

SQL DML의 CRUD 기본 명령을 알고 코딩한다.

파이썬과 SQLite DB를 연동하여 CRUD 코딩을 한다.

DB 클라이언트인 Dbeaver를 설치하여 DB와 연결하여 활용한다.

# 데이터 베이스 소개

# 빅데이터 전문가-데이터 사이언티스트

보통 외국에서는 빅데이터 전문가를 데이터 사이언티스트(data scientist)라고 하여 세 종류를 이야기 한다. (**Forbes**발췌)

## 1. **Hacking Skill**은 시스템 전문가

빅데이터 시스템은 기본적으로 분산 시스템이기 때문에 Hadoop이나 기타 전문적인 분산시스템의 구조 및 설치, 운영 등의 전문 기술이 필요하다.

## 2. **Math & Statistics Knowledge**는 분석 기술

요즘 이야기하는 BI(Business Intelligence)전문가나 분석 전문가를 이야기 한다.

빅데이터에서는 대표적으로 R이라는 툴을 사용하여 데이터 분석 알고리즘을 만들거나 하는 경우가 많다.

## 3. 실무 전문가도 데이터 사이언티스트

각 분야별 실무를 알아야만 정형, 비정형 데이터 속에 의미 있는 데이터를 추출 할 수 있다.



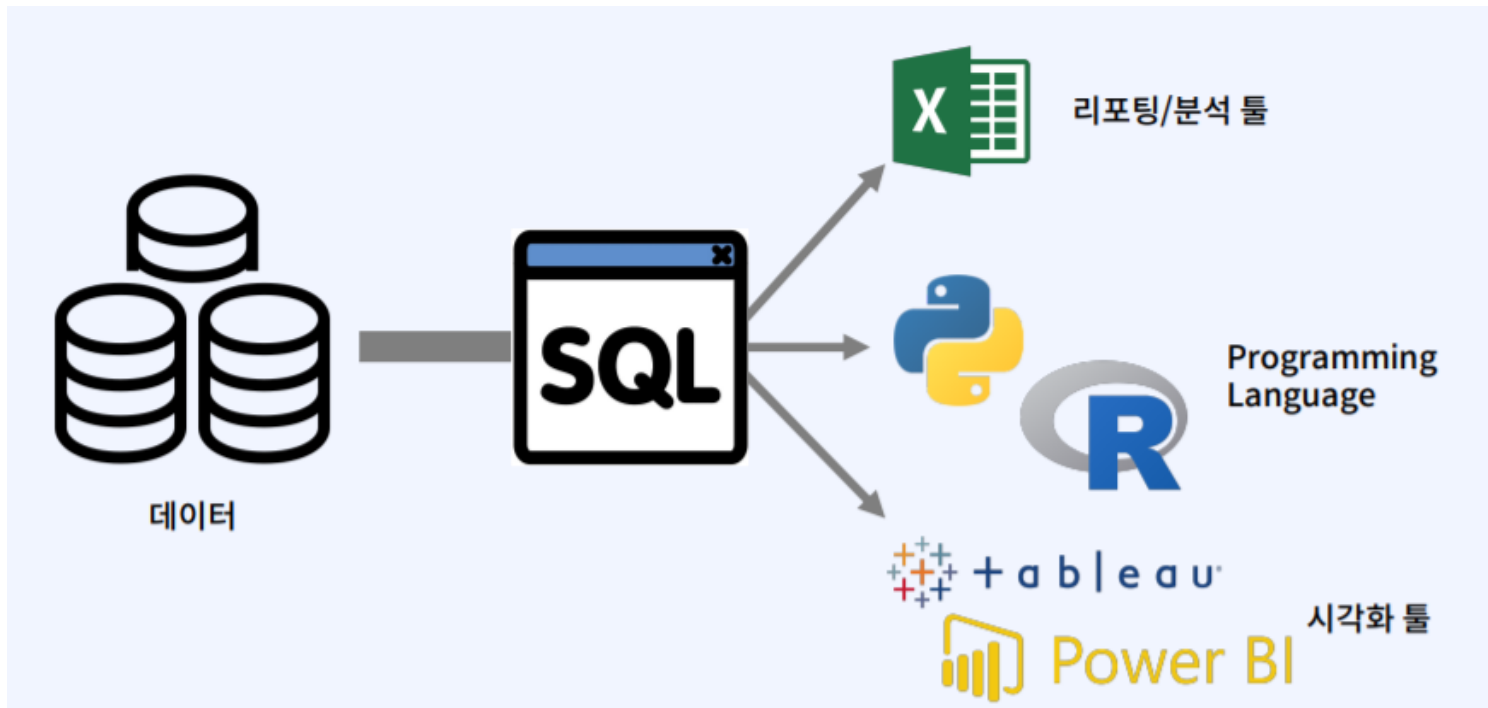
# 데이터베이스 소개

- 데이터과학에 중심에 데이터베이스가 있음.
- 프로그래밍은 결국은 데이터를 처리하기 위함



# 데이터베이스 개요

- 데이터 분석 분야의 가장 핵심



# 데이터베이스 개요



데이터베이스(DB)란?

- **체계화된 데이터의 집합**
- 여러 응용 시스템들의 **통합된 정보를 저장하여, 운영할 수 있는** 공용 데이터의 묶음
- 논리적으로 연관된 하나 이상의 **자료 모음**으로, 데이터를 고도로 구조화 함으로써 **검색/갱신등의 데이터 관리를 효율화**함
- DBMS: **데이터베이스를 관리하는 시스템**

# 데이터베이스 개요

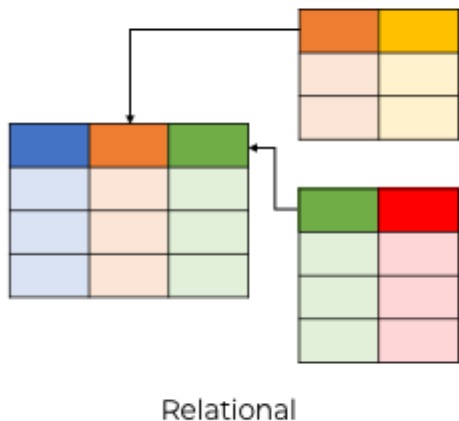
## 데이터베이스 장단점

- 데이터베이스 장점
  - 데이터 중복 최소화
  - 데이터 공유
  - 일관성, 무결성, 보안성 유지
  - 최신의 데이터 유지
  - 데이터의 표준화 가능
  - 데이터의 논리적, 물리적 독립성
  - 용이한 데이터 접근
  - 데이터 저장 공간 절약
- 단점
  - 데이터베이스 전문가 필요
  - 시스템의 복잡함
  - 많은 비용 부담

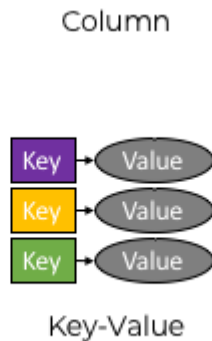
# DB 분류

- 정형DB vs 비정형DB

## SQL DATABASES



## NoSQL DATABASES



# 데이터베이스 랭킹

<https://db-engines.com/en/ranking>

378 systems in ranking, September 2021

Rank			DBMS	Database Model	Score		
Sep 2021	Aug 2021	Sep 2020			Sep 2021	Aug 2021	Sep 2020
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1271.55	+2.29	-97.82
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1212.52	-25.69	-51.72
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	970.85	-2.50	-91.91
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	577.50	+0.45	+35.22
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	496.50	-0.04	+50.02
6.	6.	↑ 7.	Redis +	Key-value, Multi-model ⓘ	171.94	+2.05	+20.08
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model ⓘ	166.56	+1.09	+5.32
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	160.24	+3.16	+9.74
9.	9.	9.	SQLite +	Relational	128.65	-1.16	+1.98
10.	↑ 11.	10.	Cassandra +	Wide column	118.99	+5.33	-0.18
11.	↓ 10.	11.	Microsoft Access	Relational	116.94	+2.10	-1.51
12.	12.	12.	MariaDB +	Relational, Multi-model ⓘ	100.70	+1.72	+9.09
13.	13.	13.	Splunk	Search engine	91.61	+1.01	+3.71
14.	14.	↑ 15.	Hive +	Relational	85.58	+1.64	+14.41
15.	15.	↑ 17.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	78.26	+3.11	+17.81
16.	16.	16.	Amazon DynamoDB +	Multi-model ⓘ	76.93	+2.03	+10.75

# 데이터베이스 분류

- 정형DB와 비정형DB

정형DB(SQL)	비정형DB(NoSQL)
정해진 규격 Schema, table – column	정해진 규격 없음
Join 가능	Join 불가능
트랜잭션 사용	트랜잭션 없음
분산처리 어려움	분산처리 용이함.

# RDBMS (SQL) 이해





# RDBMS 이해

- RDBMS; Relational Database Management System
- RDBMS – DB – table

## RDBMS

**DATABASE 1**

Table 1		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 2		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 3		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

**DATABASE 2**

Table 1		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 2		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 3		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

**DATABASE 3**

Table 1		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 2		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 3		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

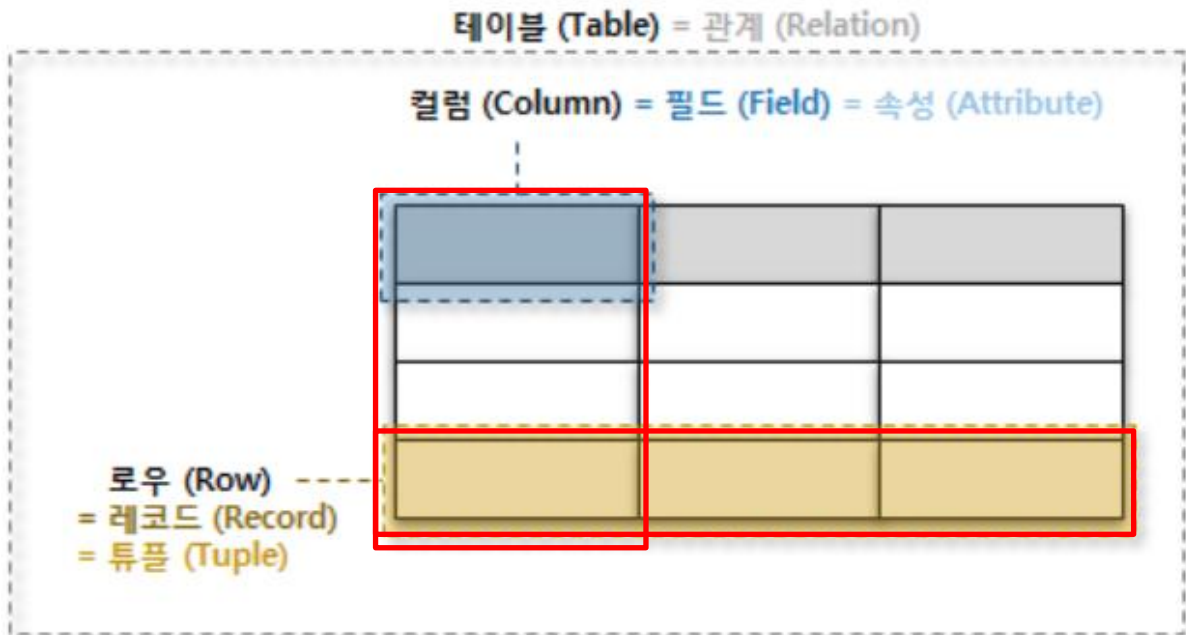
# RDBMS 이해



- **관계형 데이터베이스 시스템**
- 역사가 오래됐고, 가장 많이 사용됨
- 가장 신뢰성이 높고, 데이터 분류, 정렬, 탐색 속도가 빠름
- **관계형 데이터베이스 = 테이블!**
- **2차원 테이블(Table)** 형식을 이용하여 데이터를 정의하고 설명하는 **데이터 모델**
- 데이터를 **속성(Attribute)**과 **데이터 값(Attribute Value)**으로 구조화
- 데이터를 구조화 한다는 것은 속성과 데이터 값 사이에 관계를 찾아내고 이를 **테이블 모양의 구조로 도식화** 한다는 의미

# RDBMS 주요 용어

- Table, Row, Column



# RDBMS 주요 용어

- **Primary Key(기본키)**

- 한 테이블의 각 Row를 유일하게 식별해주는 **column**을 의미함
- 각 테이블마다 **Primary Key**가 존재해야 하며, **NULL** 값을 허용하지 **않음**
- 각 Row 마다 **유일한 값**이어야 함

- **Foreign Key(외래키 또는 외부키)**

- Foreign Key는 한 테이블의 필드(Attribute) 중 **다른 테이블의 행(Row)**을 식별할 수 있는 **키**
- 다른 테이블과 관계를 맺을 때 사용

# RDBMS 주요 용어

- Primary Key와 Foreign Key



데이터를 한 테이블에 넣지 않고 왜 나눠서 저장 할까요?

# RDBMS - DB 스키마(Schema)

데이터베이스의 테이블 구조 및 형식, 관계 등의 정보를 형식 언어(formal language)로 기술한 것

- 일종의 **데이터베이스 설계도**로 이해하면 됨
- 관계형 데이터베이스를 사용하여 데이터를 저장할 때 가장 먼저 할 일은 **데이터의 공통 속성을 식별하여 컬럼(Column)으로 정의하고, 테이블(Table)을 만드는 것**
- 통상적으로 하나의 테이블이 아닌 여러 개의 테이블로 만들고, **각 테이블 구조, 형식, 관계를 정의함**
- 데이터베이스마다 스키마를 만드는 언어가 존재하며, 해당 스키마만 있으면 동일한 구조의 데이터베이스를 만들 수 있음

# RDBMS - DB 스키마(Schema)

## • DB 스키마(Schema) 예시) 학생 관리 DB

### 1) 외부 스키마 :

개개 사용자 관점에서  
정의한 DB 스키마 전체  
DB의 한 논리적인  
부분 Subschema  
(부 스키마)

외부 스키마 1  
(교무처)

ST	
S <sub>n</sub>	INT
Name	CHAR(10)
Grade	INT
Dept	CHAR(5)

외부 스키마 2  
(학생처)

STUDENT	
S <sub>no</sub>	PIC 9(4)
S <sub>name</sub>	PIC X(10)
Y <sub>ear</sub>	PIC 9(2)
Addr	PIC X(44)

### 2) 개념 스키마

(conceptual schema)

: 전체 범 기관적인 관점에서  
정의한 DB 스키마 모든 응용에  
대한 전체적인 통합된 데이터  
구조 schema

개념 스키마

STUDENT	
S <sub>number</sub>	INTEGER
Name	CAHR(10)
Year	SMALLINT
Grade	SMALLINT
Dept	CHAR(5)
Address	CHAR(44)

### (3) 내부 스키마(internal schema)

: 전체 저장장치 관점에서의 정의한  
DB 스키마 개념 스키마에 대한  
저장구조(storage structure)를 정의

내부 스키마

STORED-STUDENT	LENGTH = 71	
prefix	BYTE(4)	OFFSET = 0
S <sub>no</sub>	BYTE(4)	OFFSET = 4 INDEX = STINDX
S <sub>name</sub>	BYTE(10)	OFFSET = 8
S <sub>year</sub>	BYTE(2)	OFFSET = 18
S <sub>grade</sub>	BYTE(2)	OFFSET = 20
S <sub>dept</sub>	BYTE(5)	OFFSET = 22
S <sub>addr</sub>	BYTE(44)	OFFSET = 27

# RDBMS – SQL



- **SQL; Structured Query Language**
- **관계형 데이터베이스 관리 시스템에서 데이터를 관리하기 위해 사용되는 표준 프로그래밍 언어(Language)**
- **데이터베이스 스키마 생성 및 수정, 테이블 관리, 데이터 추가, 수정, 삭제, 조회 등, 데이터베이스와 관련된 거의 모든 작업을 위해 사용되는 언어(CRUD; Create, Read, Update, Delete)**
- **데이터베이스마다 문법에 약간의 차이가 있지만, 표준 SQL을 기본으로 하므로, 관계형 데이터베이스를 다루기 위해서는 필수적으로 알아야 함**



# RDBMS – SQL



SQL은 크게 세 가지 종류로 나뉨

- 데이터 정의 언어(**DDL**, Data Definition Language)
- 데이터 처리 언어(**DML**, Data Manipulation Language)
- 데이터 제어 언어(**DCL**, Data Control Language)

# RDBMS – SQL – DDL



- **DDL; Data Definition Language**
- 데이터 구조 정의 언어
- 테이블(TABLE), 인덱스(INDEX) 등의 객체를 만들고 관리하는데 사용되는 명령
- CREATE, ALTER, DROP 등이 있음

# RDBMS – SQL – DML



- **DML; Data Manipulation Language**
- 테이블 데이터 조작 언어
- 데이터 **CRUD**  
[**C**reate(생성), **R**ead(읽기), **U**ppdate(갱신), **D**eleate(삭제)]
- INSERT 테이블(Table)에 하나 이상의 데이터 추가
- UPDATE 테이블(Table)에 저장된 하나 이상의 데이터 수정
- DELETE 테이블(Table)의 데이터 삭제
- SELECT 테이블(Table)에 저장된 데이터 조회

- DCL; Data Control Language
- 데이터 핸들링 권한 설정, 데이터 무결성 처리 등 수행
- **GRANT** 데이터베이스 개체(테이블, 인덱스 등)에 대한 사용 권한 설정
- BEGIN 트랜잭션(Transaction) 시작
- COMMIT 트랜잭션(Transaction) 내의 실행 결과 적용
- ROLLBACK 트랜잭션(Transaction)의 실행 취소

# SQLite를 활용한 DB 익히기

DB 생성 / 삭제

DB 테이블 생성/삭제(DDL)

# 데이터 베이스 만들기

## 1-1 테이블 생성

```
In [1]: import sqlite3
```

```
In [2]: print('sqlite3 version :', sqlite3.version)
```

sqlite3 version : 2.6.0

## 1-2 DB 연결

- sqlite3는 db연결시 filedb가 만들어짐(ex: database.db)

```
In [4]: #conn = sqlite3.connect("database.db")  
#conn = sqlite3.connect("database.db")  
  
# auto commit : 실행하면 db에 바로 반영  
conn = sqlite3.connect("database.db", isolation_level=None)
```

# 데이터 베이스 만들기

- jupyter notebook 폴더에서 sqlite database.db 생성 확인

<input type="checkbox"/>	 Untitled.ipynb	Running 41분 전	72 B
<input type="checkbox"/>	 database.db	한 달 전	8.19 kB
<input type="checkbox"/>	 dump.sql	한 달 전	747 B

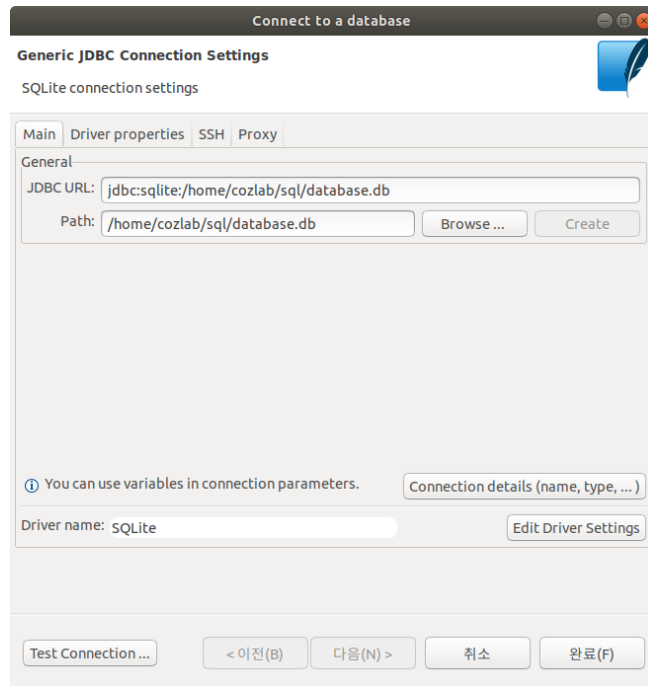
# SQLite를 활용한 python 코딩하기

- SQLite : 파이썬에 내장되어 있는 파일형 DB
- python 코딩으로 sqlite DB 파일 생성  
**sqlite3.connect('file\_path/database.db')** 실행하면 파일 생성됨



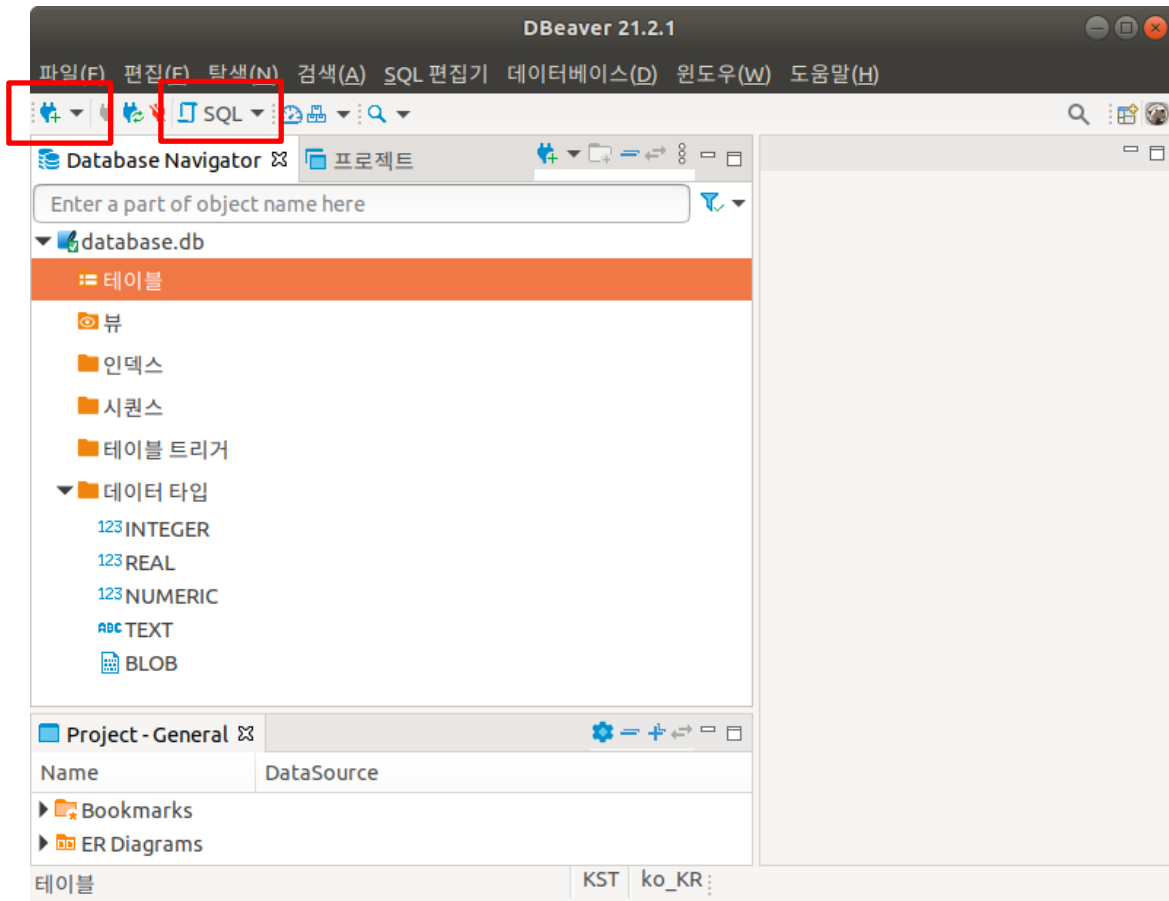
# DB 클라이언트 실행

- 아이콘 확인, 실행
- sqlite DB 연결하기



# DB 클라이언트 실행, DB 연결

- sqlite DB 연결
- sql편집 탭 실행



# DB 테이블 생성 – DDL

DBeaver 21.2.1 - <database.db> Script-5

파일(F) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 윈도우(W) 도움말(H)

Database Navigator 프로젝트

Enter a part of object name here

▼ database.db

▼ 테이블

▼ users\_t

▶ 컬럼

▶ Unique 키

▶ 외부 키

▶ 인덱스

▶ 참조

▶ 트리거

뷰

인덱스

시퀀스

테이블 트리거

▼ 데이터 타입

123 INTEGER

Project - General

Name DataSource

```
CREATE TABLE users_t(id integer PRIMARY KEY,
username text, email text, phone text,
website text, regdate text)
```

Statistics 1

Enter a SQL expression to filter

Name	Value
Updated Rows	0
Query	CREATE TABLE users_t(id integer PRIMARY KEY, username text, email text, phone text, website text, regdate text)
Finish time	Sun Oct 24 10:41:28 KST 2021

CREATE TABLE users\_t(id INTEGER PRIMARY KEY,  
username TEXT, email TEXT, phone TEXT,  
website TEXT, regdate TEXT)

# DB 테이블 생성

DBeaver 21.2.1 - <database.db> Script-5

파일(E) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 윈도우(W) 도움말(H)

Database Navigator 프로젝트

Enter a part of object name here

database.db

- 테이블
  - users\_t
    - 컬럼
    - Unique 키
    - 외부 키
    - 인덱스
    - 참조
    - 트리거
  - users\_t2
    - 컬럼
    - Unique 키
    - 외부 키
    - 인덱스
    - 참조

Project - General

Name DataSource

- Bookmarks
- ER Diagrams
- Scripts

Statistics 1

CREATE TABLE users\_t2(id integer PRIMARY KEY, username text, email text, phone text, website text, regdate text)

Name	Value
Updated Rows	0
Query	-- 테이블 만들기 CREATE TABLE users_t2(id integer PRIMARY KEY, username text, email text, phone text, website text, regdate text)
Finish time	Sun Oct 24 10:50:36 KST 2021

Save Cancel Script 0 row(s) updated - 69ms

KST ko\_KR 쓰기 가능 스마트 삽입 2

-- 주석 표시

- users\_t2로 새로운 테이블 만들기

# DB 테이블(user\_t2) 삭제

The screenshot shows a database management interface with a left sidebar, a central SQL editor, and a bottom status bar.

**Left Sidebar:** Displays the database structure for 'database.db'. The '테이블' (Tables) folder is expanded, showing 'users\_t' and its columns: 'id' (INTEGER), 'username' (TEXT), 'email' (TEXT), 'phone' (TEXT), 'website' (TEXT), and 'regdate' (TEXT). Other folders like 'Unique 키', '외부 키', '인덱스', '참조', '트리거', and '뷰' are also visible.

**Central SQL Editor:** Contains a script for creating and dropping tables. The 'DROP TABLE users\_t2 ;' statement is highlighted with a red box.

```
-- 테이블 만들기
CREATE TABLE users_t(id integer PRIMARY KEY,
username text, email text, phone text,
website text, regdate text)

-- 테이블 만들기
CREATE TABLE users_t2(id integer PRIMARY KEY,
username text, email text, phone text,
website text);

-- 테이블 삭제 (drop)
DROP TABLE users_t2 ;
```

**Bottom Panel:** Shows the execution results of the SQL statement. The 'Query' section displays the executed command: 'DROP TABLE users\_t2'. The 'Finish time' is 'Sun Oct 24 11:13:36 KST 2021'. The status bar at the bottom indicates '1 row(s) updated - 69ms'.

Name	Value
Updated Rows	1
Query	-- 테이블 삭제 (drop) DROP TABLE users_t2
Finish time	Sun Oct 24 11:13:36 KST 2021

# DB 테이블(user\_t)에 데이터 입력(create)

The screenshot shows a database management tool interface. The left sidebar displays the database structure for 'database.db', showing a table 'users\_t' with columns: id (INTEGER), username (TEXT), email (TEXT), phone (TEXT), website (TEXT), and regdate (TEXT). The main window displays a SQL script for creating and inserting data into the 'users\_t' table. The script includes comments in Korean and SQL commands for creating the table, inserting data, and dropping the table. The bottom panel shows the execution results, indicating that 1 row was updated in 95ms.

```
-- 테이블 만들기
CREATE TABLE users_t(id integer PRIMARY KEY,
username text, email text, phone text,
website text, regdate text)

-- 테이블 만들기
CREATE TABLE users_t2(id integer PRIMARY KEY,
username text, email text, phone text,
website text);

-- 테이블 삭제 (drop)
DROP TABLE users_t2 ;

-- DB 테이블 다루기
-- CRUD(Create, Read, Update, Delete)
insert into users_t values(1, 'kim', 'kim@cozlab.com', '010-12345678', 'cozlab.com', '20201024');
insert into users_t values(2, 'lee', 'lee@naver.com', '010-12345678', 'naver.com', '20201024');
```

Statistics 1


insert into users\_t values(2, 'lee', 'lee@naver.com', '010-12345678', 'naver.com', '20201024')

Name	Value
Updated Rows	1
Query	insert into users_t values(2, 'lee', 'lee@naver.com', '010-12345678', 'naver.com', '20201024')
Finish time	Sun Oct 24 11:08:50 KST 2021

Save Cancel Script 200 1 row(s) updated - 95ms

KST ko\_KR 쓰기 가능 스마트 삽입 19:1:523

# DB 테이블(user\_t)에 데이터 Insert



```
insert into users_t values(1, 'kim', 'kim@cozlab.com', '010-1234-5678','cozlab.com','20201024');  
insert into users_t values(2, 'lee', 'lee@naver.com', '010-1234-5678','naver.com','20201024');  
insert into users_t values(3, 'hwang', 'hwang@cozlab.com', '010-2356-0978','cozlab.com','20201024');  
insert into users_t values(4, 'oh', 'oh@naver.com', '010-1234-5678','naver.com','20201024');
```

## DB 테이블(user\_t)에 데이터 Read

- 테이블 데이터 조회

```
SELECT *
```

```
FROM users_t ut ;
```


- 테이블의 username, phone 조회

```
SELECT username, phone
```

```
FROM users_t ;
```




## DB 테이블(user\_t)에 데이터 Update



```
update users_t set email ='kim@google.com', phone ='010-2356-3245'  
WHERE id=1;
```

# DB 테이블(user\_t)에 데이터 Delete



- 조건에 맞는 데이터 삭제

```
DELETE FROM users_t  
WHERE id=3;
```

- 테이블 데이터 모두 삭제

```
DELETE FROM users_t ;
```

## DB 테이블(user\_t) drop



- 테이블 삭제

```
DROP TABLE users_t ;
```

# SQL DML CRUD 기본 문법

## DML; Data Manipulation Language

# DB 테이블 만들기 기본 익히기

- 테이블 명 : post

필드명(컬럼명)	데이터 타입	제약조건
id	integer	Not null, primary key
title	text	Not null
content	text	Not null

- 테이블 명 : usersd

필드명(컬럼명)	데이터 타입	제약조건
id	integer	Not null, primary key
name	text	Not null
phone	text	Not null
address	text	-

# 테이블 만들기



- post 테이블

```
CREATE TABLE post(id INTEGER NOT NULL PRIMARY KEY,  
title TEXT NOT NULL, content TEXT NOT NULL);
```

- [문제] **usersd** 테이블을 만들어 보세요.

# SQL – DML CRUD – SQL INSERT

- 데이터(행) 추가 명령(Create)

```
INSERT INTO 테이블명 (컬럼명1, 컬럼명2) VALUES (값1, 값2);
```

```
INSERT INTO post (id, title, content) VALUES (1, '코딩', '재미있어요!');
```

- **[문제]** 5개의 데이터 set을 더 추가하세요.
  - 2, 'java', '객체지향 언어'
  - 3, 'html', '웹 표준 언어'
  - 4, 'javascript', '잘하면 좋겠네'
  - 5, 'django', '풀스택 개발 프레임워크'

# SQL – DML CRUD – SQL INSERT

- 데이터(행) 추가 명령(Create)

```
INSERT INTO 테이블명 (컬럼명1, 컬럼명2) VALUES (값1, 값2);
```

```
INSERT INTO usersd (id, name, phone, address)
```

```
VALUES (1, 'kim', '010-1111-1111', 'seoul');
```

```
2, 'lee', '010-2222-2222', 'seoul'
```

```
3, 'song', '010-3333-3333', 'daegu'
```

```
4, 'park', '010-4444-4444', 'pusan'
```

```
5, 'lee', '010-5555-5555', 'daegu'
```



# SQL – DML CRUD – SQL SELECT

- 데이터 조회 명령(Read)

```
SELECT 컬럼명1, 컬럼명2 FROM 테이블명;
```

```
SELECT title, content FROM post;
```

- [문제] 모든 데이터를 조회해 보세요.

# SQL – DML CRUD – SQL SELECT

- WHERE 추가하기 (조회 조건)

```
SELECT 컬럼명1, 컬럼명2 FROM 테이블명 WHERE 조건;
```

```
SELECT * FROM post WHERE id=2
```

- title 필드의 값 중에 'java'라는 문자열로 시작하는 것 모두 조회

```
SELECT * FROM post WHERE title like 'java%';
```

- **[문제]** post 테이블에서 title 값이 '코딩' 인 것 조회

# SQL – DML CRUD – SQL SELECT

- WHERE 추가하기 (조회 조건)

```
SELECT 컬럼명1, 컬럼명2 FROM 테이블명 WHERE 조건;
```

```
SELECT title, content FROM post WHERE id BETWEEN 1 and 3;
```

```
SELECT * FROM user WHERE address IN ('seoul', 'busan', 'deagu');
```

# SQL – DML CRUD – SQL SELECT

- ORDER BY 추가하기 (정렬)

```
SELECT * FROM 테이블명 ORDER BY 컬럼명 [ASC|DESC];
```

```
SELECT * FROM post ORDER BY title ASC
```

# SQL – DML CRUD – SQL UPDATE

- 데이터 수정 명령

```
UPDATE 테이블명 SET 컬럼명 = 값, ... WHERE 조건식;
```

```
UPDATE post SET title = '제목 수정 중',  
               content = '본문 수정 중'  
WHERE id=3;
```

# SQL – DML CRUD – SQL DELETE

- 데이터 삭제 명령

```
DELETE FROM 테이블명 WHERE 조건식;
```

```
DELETE FROM post WHERE id=3;
```

# Python 코딩 with SQLite(실습)

# SQL 기본 문법

- SQLite 데이터 타입

데이터 타입	설명
INTEGER	정수
REAL, NUMERIC	실수
TEXT	문자열
NULL	null 값 (데이터 없음)
BLOB	Binary Large Object. 입력 데이터를 그대로 저장



# Python 코딩 with SQL

- 파이썬에서 SQL 코딩 순서



`conn=db.connect()`   `cur=conn.cursor()`   `cur.execute(DML)`   `conn.commit()`   `conn.close()`  
`conn.rollback()`

# Python with SQL 연동 실습하기

---

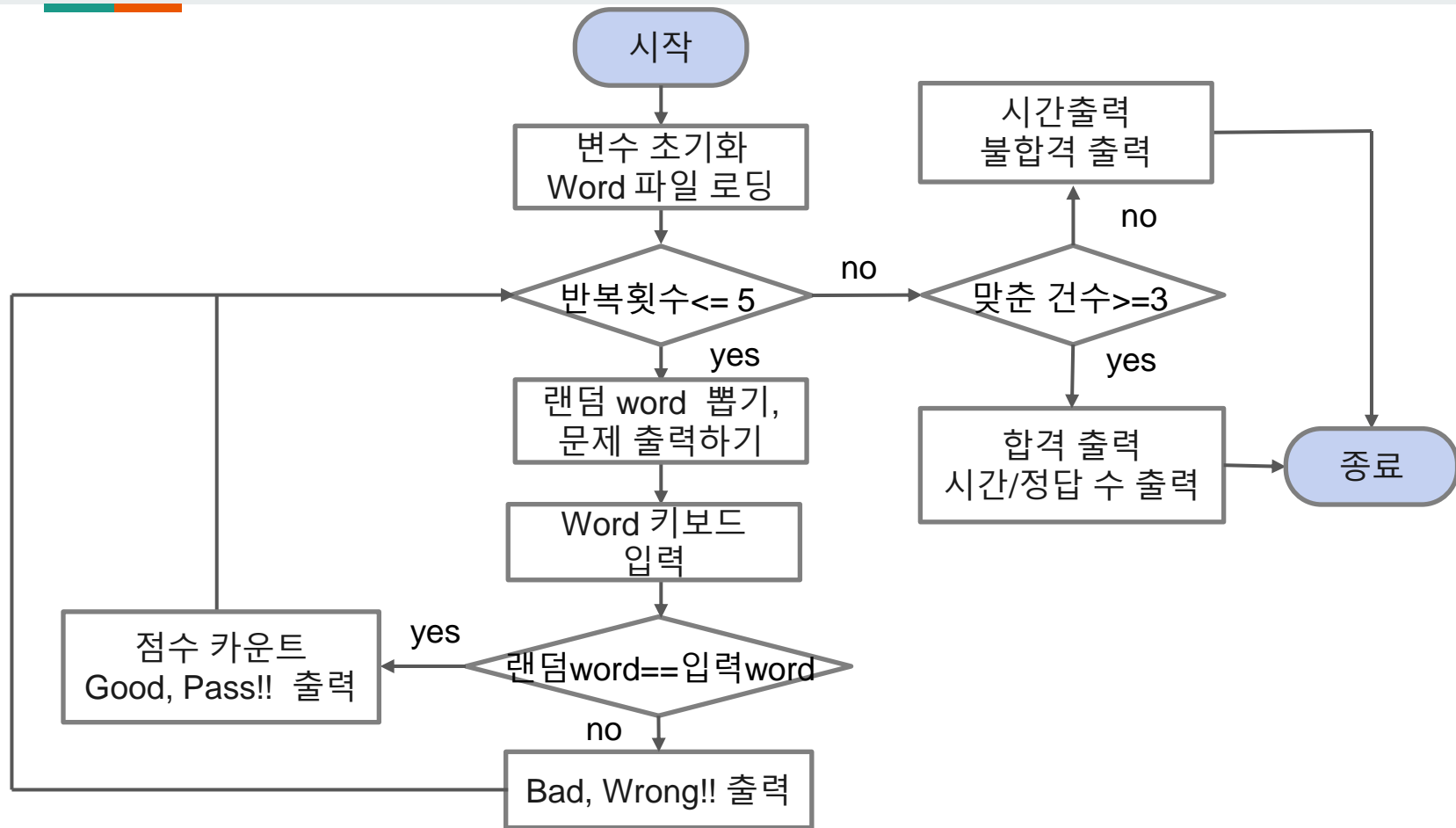
- 테이블 생성하고 basic 코딩 실습하기, jupyter notebook
- 첨부파일 ; 1-3sql\_basic\_sqlite\_with\_python.pdf

# DB 연동 프로그래밍 문제해결

# DB연동 python 프로그래밍 문제해결

- python word 맞추기 게임 문제 알고리즘 분석
  - word\_game1.py
- game score db에 기록하기
  - id : PK, 자동 증가
  - corr\_cnt : 정답 수
  - record : 걸린 시간
  - regdate : 기록 날짜
- 마지막 문재 소리재생 안되는 것 해결

# Word 타이핑 게임 알고리즘



# 리눅스, 소리출력 with python

- pip install pygame
- 사용 방법

```
import pygame
```

```
good_sound = pygame.resource.media(  
    'assets/good.wav',    #  
)  
good_sound.play()
```

# Word 타이핑 게임 - DB 연동하기

DBaiver 21.2.1 - records

파일(F) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 윈도우(W) 도움말(H)

SQL 커밋 롤백 Auto records.db <N/A>

records.db 테이블 records

테이블명: records 테이블 타입: TABLE

컬럼명	#	데이터 유형	Length	Not Null	Auto Increment
id	1	INTEGER		[ ]	[v]
corr_cnt	2	INTEGER		[ ]	[ ]
record	3	TEXT		[ ]	[ ]
regdate	4	TEXT		[ ]	[ ]

4 items

KST ko\_KR