

파이썬 고급 과정

3 4 .파이썬에서 DB사용하기

학습 내용

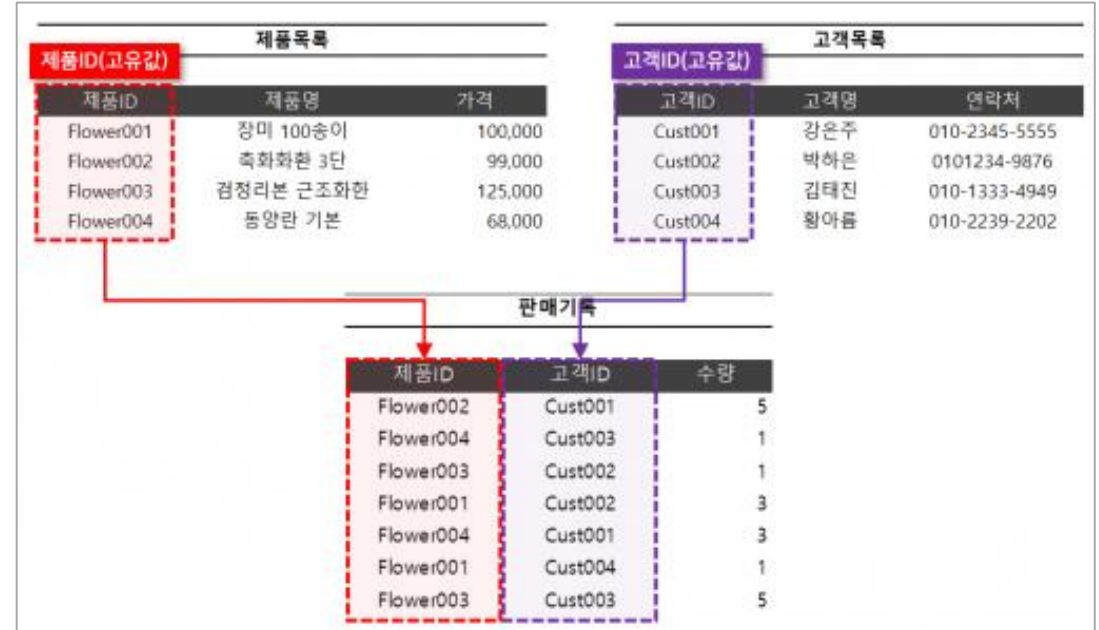
- 파이썬에서 Sqlite 사용하기

관계형 DB

- 거의 모든 관계형 DB는 SQL로 소통한다.
- SQL(Structured Query Language)을 알면 거의 모든 관계형 DB를 다 다룰 수 있다.
- RDBMS: 테이블이 key로 관계되어 관리됨



관계형 DB (RDBMS)



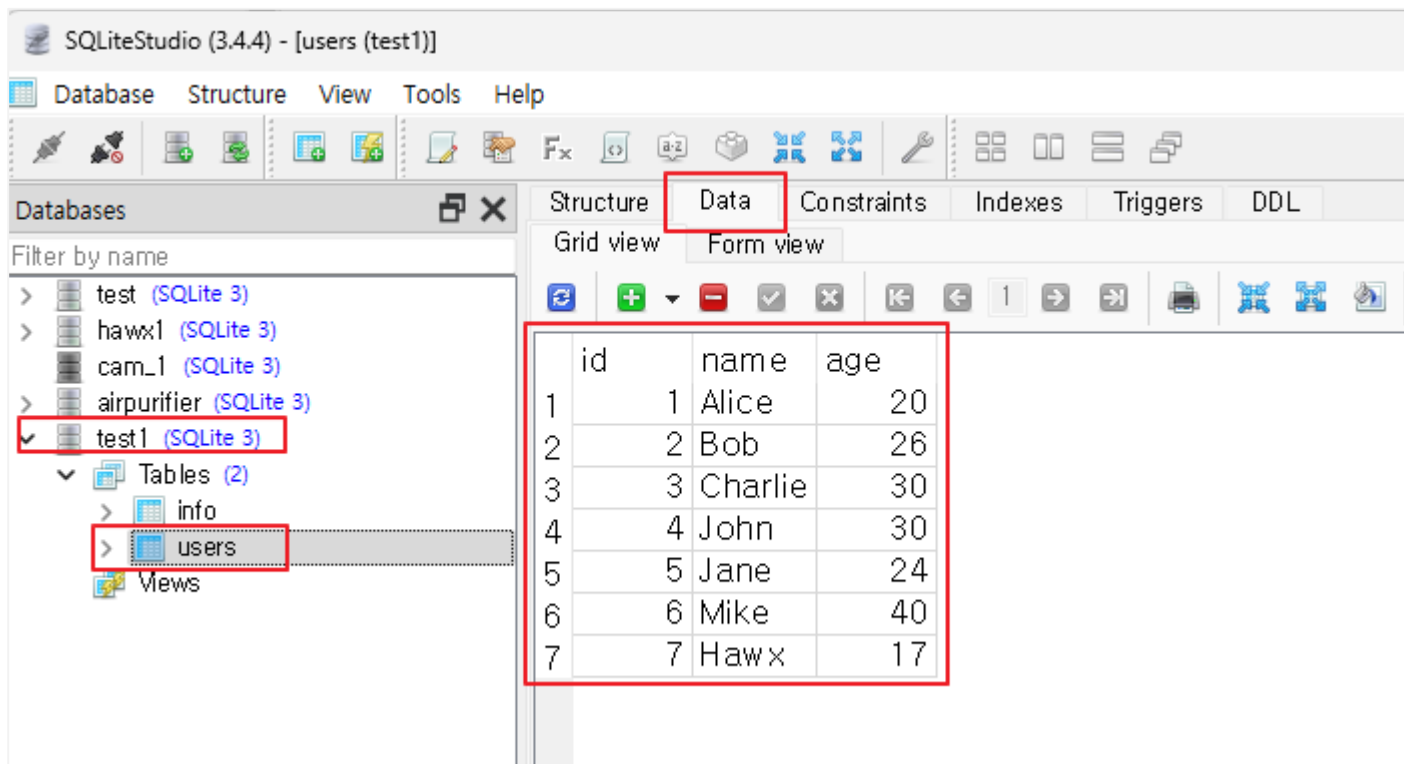
SQLITE3

- 경량화된 관계형 DB의 한 종류
- Sqlite3는 서버가 필요 없는 파일 기반 데이터베이스로, RDBMS의 핵심 기능을 제공하면서도 가벼움

SQLite3의 DB 구조 :

다른 정규 DB와는 달리 .db
(.sqlite) 파일 하나에 모든
테이블이 들어감

➔ 사용이 편하지만 대량
데이터에는 적합하지 않음.



SQLITE3 사용

1. SQLite 다운로드

<https://www.sqlite.org/download.html>

Precompiled Binaries for Windows

[sqlite-dll-win-x86-3470200.zip](#) 32-bit DLL (x86) for SQLite
(1.02 MiB) (SHA3-256: ab371612d180c43)

[sqlite-dll-win-x64-3470200.zip](#) 64-bit DLL (x64) for SQLite
(1.27 MiB) (SHA3-256: 2dc4f11f32e1efa0)

[sqlite-tools-win-x64-3470200.zip](#) A bundle of command-line tools: `sqlite3`, `sqlite3_an`
(6.09 MiB) (SHA3-256: 7e88369b0d37e18)

2. 압축해제 후 sqlite3.exe 실행

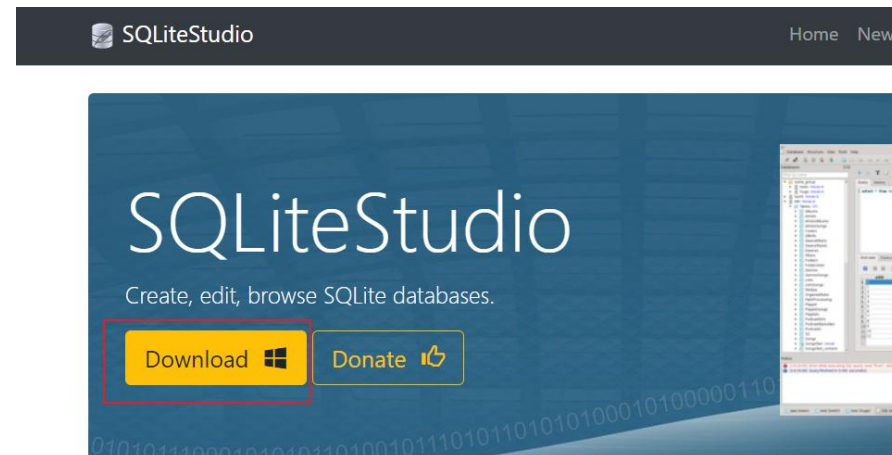
Python만 설치되어 있다면 SQLite를 파이썬 코드에서 바로 쓸 수 명령어를 터미널에서 직접 사용하고 싶을 때 다운로드해서 사용

• SQLite3 사용을 위한 SQLiteStudio

- SQLite 데이터베이스를 관리하고 탐색하기 위한 **GUI도구**
- SQLite 데이터베이스를 쉽게 생성, 관리, 및 쿼리할 수 있는 환경을 제공하여, 초보자와 전문가 모두에게 유용

스튜디오(GUI) 설치

<https://sqlitestudio.pl/>




3.4.4 released!

Mostly important bugfixes, but also few new things.


[Read More →](#)

NULL / NOT NULL이란?

NAVER 회원 정보는 어디에 저장되지?

실명 인증된 아이디로 가입 


아이디 @naver.com

비밀번호 

☒ 이메일주소 (비밀번호 찾기 등 본인 확인용)

이름

생년월일 8자리

통신사 선택 


남자

여자

내국인

외국인

휴대전화번호

☒ [필수] 인증 약관 전체동의 

필수 정보 (Not Null)

선택 정보 (Null)

- Null : '공란 허용'의 의미임.
'0'이 아님! (파이썬의 None의 의미)

(예)

```
SELECT * FROM users WHERE age IS NULL;  
-- age 컬럼이 null인 모든 행을 선택
```

- Not Null : '공란 불허'의 의미임.
(= 데이터가 없으면 에러임)

(예)

```
id INTEGER PRIMARY KEY NOT NULL  
--> id는 숫자형이고 식별 키고 필수 정보임
```

SQLITE3에서 허용하는 데이터 유형

https://www.sqlite.org/datatype3.html#storage_classes_and_datatypes

데이터 타입	설명
--------	----

NULL	NULL 값
------	--------

INTEGER	부호있는 정수. 1, 2, 3, 4, 6, or 8 바이트로 저장
---------	--------------------------------------

REAL	부동 소수점 숫자. 8 바이트로 저장
------	----------------------

TEXT	텍스트. UTF-8, UTF-16BE or UTF-16-LE 중 하나에 저장
------	--

BLOB	Binary Large Object. 입력 데이터를 그대로 저장
------	-------------------------------------

SQLITE STUDIO

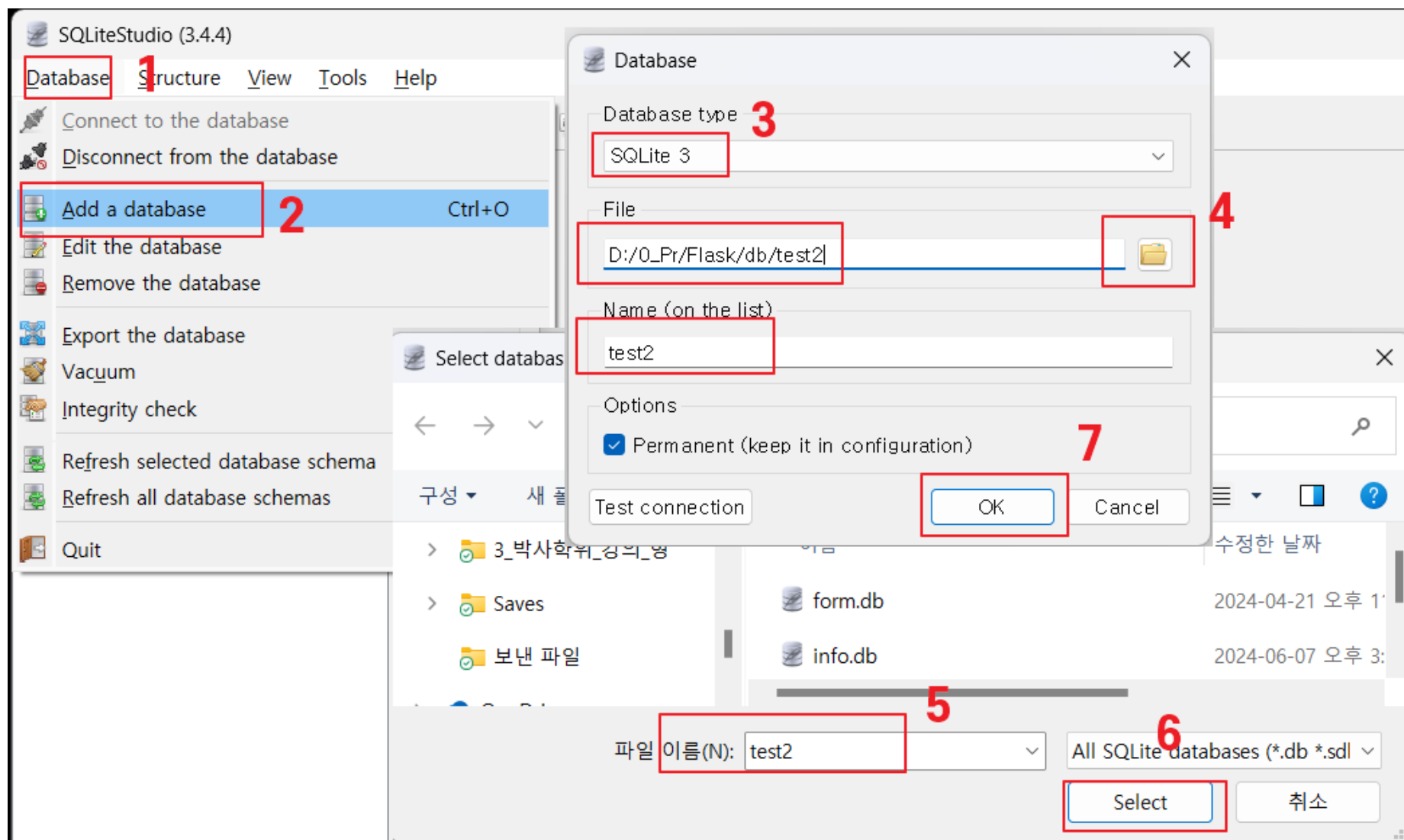
The screenshot shows the SQLiteStudio 3.4.4 interface. On the left, the 'Databases' pane shows a tree view with 'test (SQLite 3)' expanded, showing tables like 'Club', 'Student', 'StudentHistory', 'tt', and 'form'. The 'SQL editor 1' pane shows a query: `1 DELETE FROM INFO WHERE NO >= 6 AND NO <= 10;`. On the right, the 'users (test1)' table structure is displayed with columns: id (INTEGER, Primary Key), name (TEXT), and age (INTEGER). Below the table structure, there is a 'Details' section with 'Type' and 'Name' columns.

id	title	description	created
1	SQLite	SQLite is ..	2018-1-1
2	MySQL	MySQL is ..	2018-1-3

표 (table)

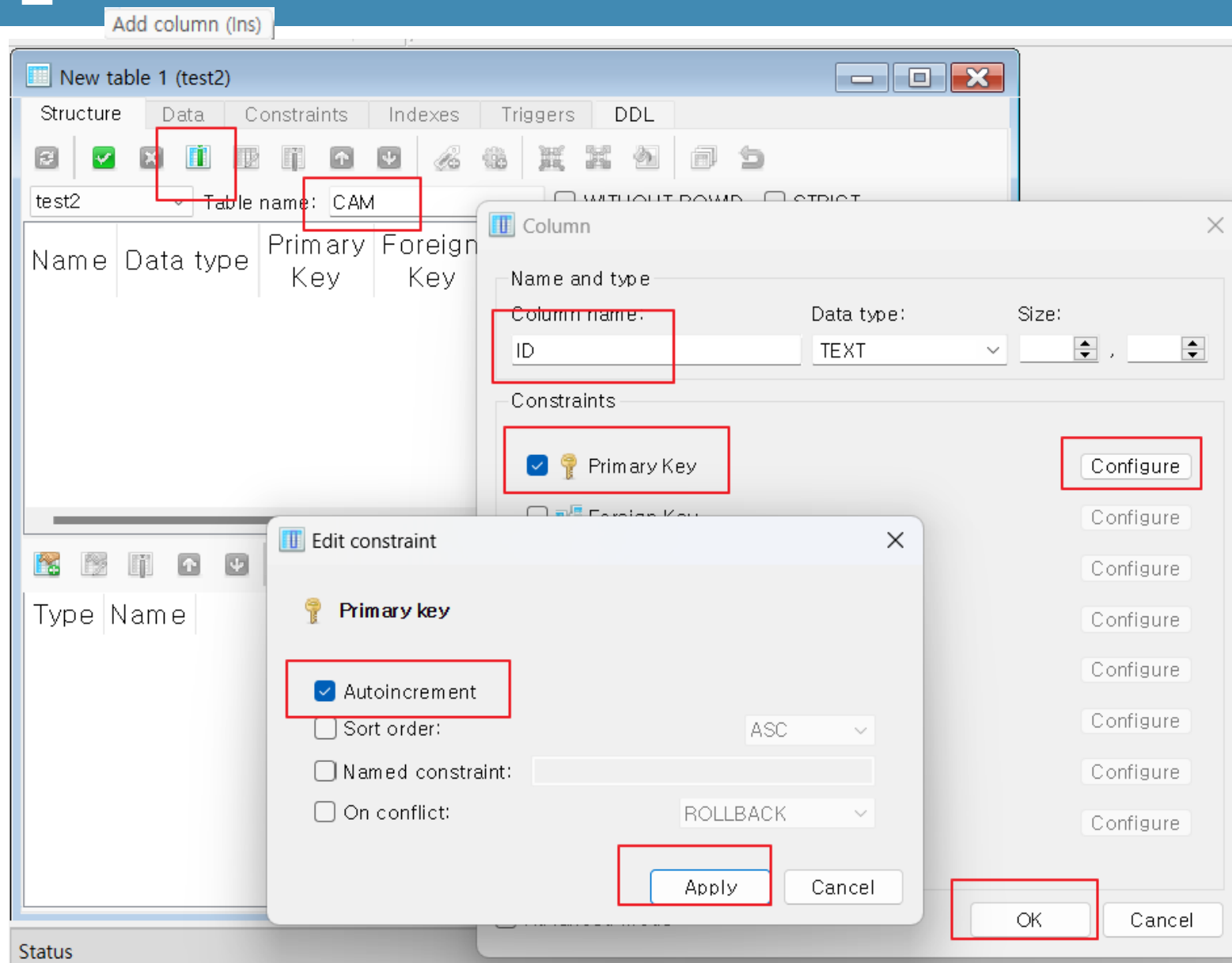
.....데이터베이스 (database, shema).....

SQLITE STUDIO 사용법(DB 생성 예시)



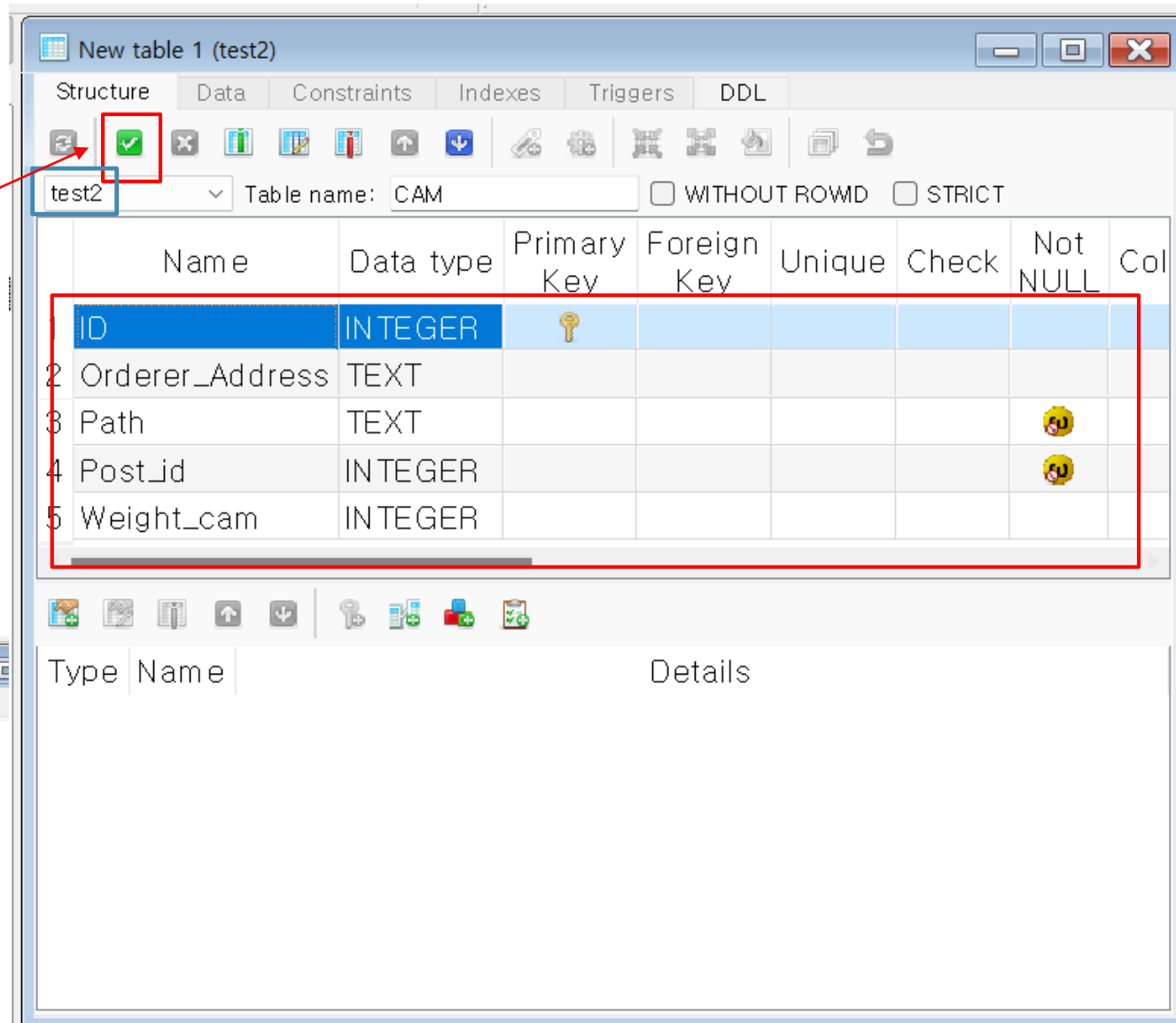
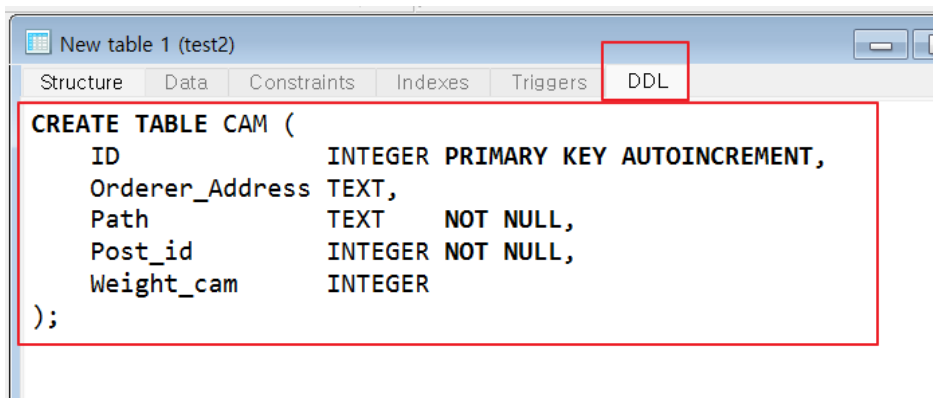
SQLITE STUDIO 사용법

(테이블/칼럼 생성 예시)



SQLITE STUDIO 사용법

(코딩 내용 커밋 예시)



SQLITE STUDIO

The screenshot displays the SQLite Studio interface with two table structure windows open: 'cam_table (cam_1)' and 'users (cam_1)'. The 'cam_table' window has a red box around the 'Structure' tab and a green checkmark icon. The 'users' window also has a green checkmark icon. A dialog box titled 'Queries to be executed' is in the foreground, containing the following SQL code:

```
PRAGMA foreign_keys = 0;

CREATE TABLE sqlitestudio_temp_table AS SELECT *
FROM
cam_table;

DROP TABLE cam_table;

CREATE TABLE cam_table (
  ID          INTEGER PRIMARY KEY AUTOINCREMENT,
  Path        NOT NULL,
  Post_id     NOT NULL,
  Weight_cam
);
```

The dialog also has a 'Don't show again' checkbox and 'OK' and 'Cancel' buttons, with the 'OK' button highlighted by a red box.

cam_table (cam_1) Structure:

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate
1	ID	INTEGER	🔑					
2	Path						🚫	
3	Post_id						🚫	
4	Weight_cam							

users (cam_1) Structure:

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Gen
1	id	INTEGER	🔑						
2	name	TEXT					🚫		
3	age	INTEGER					🚫		

SQLITE STUDIO

The screenshot shows the SQLite Studio interface. On the left, the 'Databases' pane shows a project named 'cam_1' containing two tables: 'cam_table' and 'users'. The 'cam_table' is selected, and its 'Data' tab is active. The table has columns: ID, Path, Post_Id, and Weight_cam. The data is displayed in a grid view. A modal dialog titled 'Uncommitted changes' is open in the foreground, asking if the user wants to commit the structure or go back to the structure tab. The 'Commit modifications and browse data' button is highlighted with a red box.

cam_table (cam_1) Data

ID	Path	Post_Id	Weight_cam
TD_1	3	5825	300
TD_2	2	5825	620
TD_3	1	286	700
TD_4	2	5825	800
TD_5	1	282	580
TD_6	2	286	340
TD_7	1	286	390
TD_8	2	282	440
TD_9	2	282	660
TD_10	1	5825	330
TD_11	3	286	
TD_12	1	286	
TD_13	1	286	
TD_14	2	282	
TD_15	3	5825	
TD_16	3	5825	
TD_17	1	5825	
TD_18	1	282	
TD_19	1	5825	400
TD_20	1	286	680

Uncommitted changes

There are uncommitted structure modifications. You cannot browse or edit data until you have table structure settled.
Do you want to commit the structure, or do you want to go back to the structure tab?

Go back to structure tab | Commit modifications and browse data

users (cam_1) Structure

ID	Name	Data type
1	id	INTEGER
2	name	TEXT
3	age	INTEGER

DATABASE 테이블과 입력값 규정

SQLiteStudio (3.4.4) - [users (test1)]

Database Structure View Tools Help

Databases

- test (SQLite 3)
- hawxl (SQLite 3)
- cam_1 (SQLite 3)
- airpurifier (SQLite 3)
- test1 (SQLite 3)
 - Tables (2)
 - info
 - users
 - Views

test1 Table name: users

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	id	INTEGER								NULL
2	name	TEXT								NULL
3	age	INTEGER								NULL

Annotations:

- id가 식별값임 (id is the identifier)
- 연관된 DB 테이블의 PK (Primary Key of the related DB table)
- 빈칸 허용 안 함 (No space allowed)

```
cur.execute('''
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL
)
''')
```

```
cur.execute('''
CREATE TABLE employee_info (
    no INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT(20) NOT NULL,
    age INTEGER CHECK (1 < age AND age < 120),
    level TEXT(2),
    mobile TEXT,
    email TEXT NOT NULL
)
''')
```

SQLITE STUDIO MISSION

미션 : 간단한 products 테이블 생성 및 데이터 삽입

목표 : SQLiteStudio에서 데이터베이스 생성, products 테이블을 추가하고 제품 정보를 저장.
테이블에 데이터를 삽입하고 조회.

세부 요구사항

1. 데이터베이스 생성

- 데이터베이스 파일 이름: store.db

2. 테이블 생성

- 테이블 이름: products
- 테이블의 스키마(컬럼):
 - id: INTEGER, PRIMARY KEY, AUTOINCREMENT
 - name: TEXT, NOT NULL (제품 이름)
 - price: REAL, NOT NULL (제품 가격)
 - stock: INTEGER, NOT NULL (재고 수량)

3. 데이터 삽입 : 테이블에 다음 데이터를 삽입:

- (1) name: "Laptop", price: 1500.50, stock: 10
- (2) name: "Mouse", price: 25.99, stock: 100
- (3) name: "Keyboard", price: 45.00, stock: 50

4. 데이터 확인

- SQLiteStudio의 "Browse Data" 기능을 사용하여 데이터를 조회하고 확인.

SQLITE3 기본 문법

Python에서 Sqlite3 모듈을 사용하여 SQLite 데이터베이스를 처리하는 방법

- 접속관련 함수

connect()	SQLite 데이터베이스에 연결 데이터베이스 파일이 존재하지 않으면 새로운 데이터베이스 파일을 생성 연결된 데이터베이스 객체를 반환
예제	<code>conn = sqlite3.connect('example.db')</code>
cursor()	데이터베이스와 상호작용하기 위한 cursor 객체를 생성 cursor 객체는 SQL 쿼리를 실행하고, 결과를 가져오는 등의 작업을 수행
예제	<code>cursor = conn.cursor()</code>
close()	데이터베이스 연결을 닫음. cursor 객체와 데이터베이스 연결 객체가 모두 호출 필요
예제	<code>cursor.close()</code> <code>conn.close()</code>

SQLITE3 기본 문법

▪ 쿼리 전송 관련 함수

execute()	지정된 SQL 쿼리를 실행 쿼리에는 "?"와 같은 placeholder를 사용하여 파라미터를 전달할 수 있음
예제	<code>cursor.execute("SELECT * FROM users WHERE name=?", ('John',))</code>
executemany()	같은 SQL 쿼리를 반복적으로 실행 여러 개의 파라미터 세트를 처리할 때 사용
예제	<code>data = [('John', 30), ('Jane', 25)]</code> <code>cursor.executemany("INSERT INTO users (name, age) VALUES (?, ?)", data)</code>

물음표 순서대로 튜플의 값들이
들어감

SQLITE3 기본 문법

- 쿼리 결과 관련 함수
- select 쿼리 결과를 이용할 때 사용

fetchone()	결과 세트에서 다음 행(row)을 반환
예제	<code>row = cursor.fetchone()</code>

fetchmany()	결과 세트에서 다음 여러 개의 행을 가져옴 size 파라미터를 지정하지 않으면 기본값으로 cursor.arraysize를 사용(기본값:1) cursor.arraysize = 100과 같이 변경 가능
예제	<code>rows = cursor.fetchmany(5)</code>

fetchall()	결과 세트에서 모든 행을 가져옴
예제	<code>rows = cursor.fetchall()</code>

SQLITE3 기본 문법

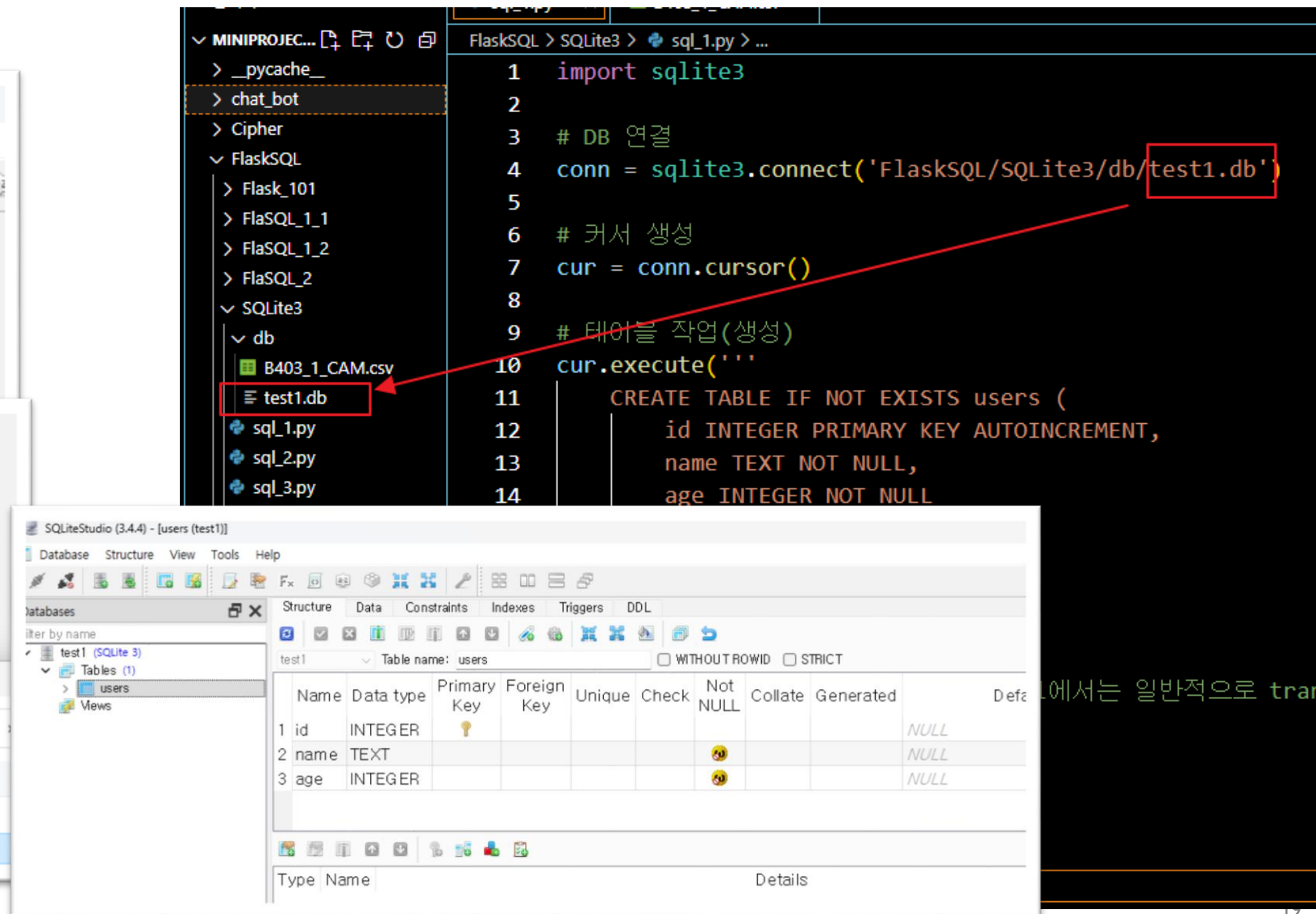
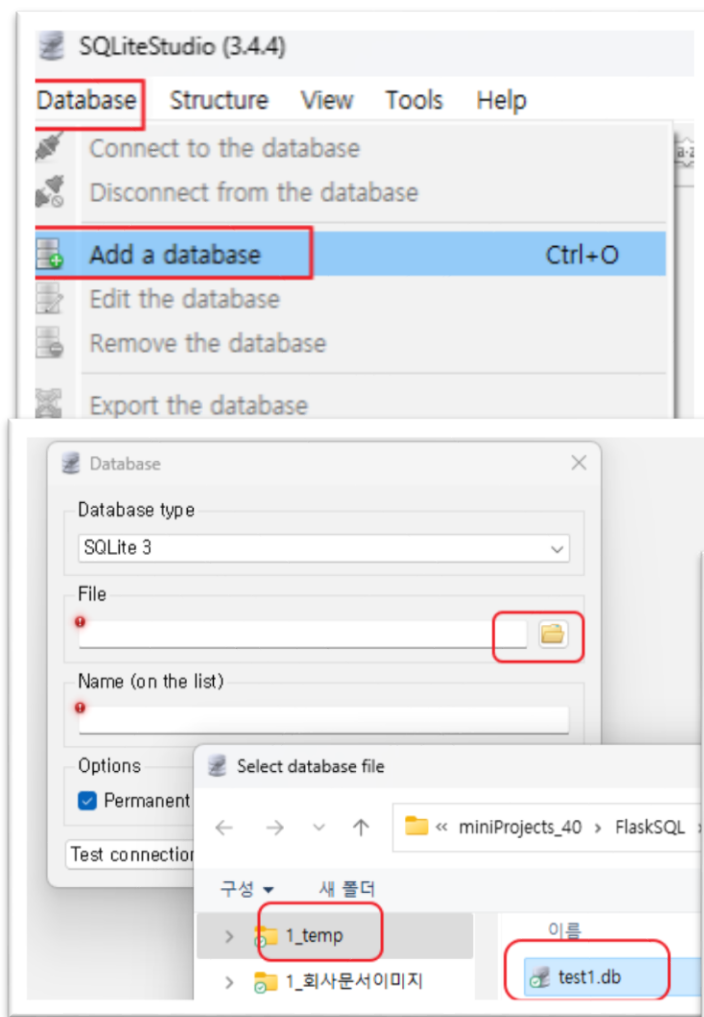
- 트랜잭션 관련 함수
- sqlite3모듈은 자동 커밋 모드로 동작
수동으로 begin을 전송하여 직접 트랜잭션 제어 가능

commit()	데이터베이스에 트랜잭션을 커밋 데이터베이스에 변경 사항을 영구적으로 적용하기 위해서는 반드시 이 함수를 호출
예제	conn.commit()
rollback()	이전 커밋 시점으로 데이터베이스를 되돌림 이 함수를 호출하면, 이전 커밋 이후에 변경된 데이터는 모두 삭제됨
예제	conn.rollback()

- commit()은 save 역할을 함. (SQLite는 DB를 파일로 저장하므로)
- DDL(create, alter, drop)은 정규 SQL에서는 커밋 없이 수행 가능. 하지만 SQLite에서는 save 역할이므로 항상 커밋해줘야 함.

SQLITE3

- 기본 구조 (DB 생성 및 연결)



SQLITE3 (SQL_2.PY)

- CRUD: Create `cur.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Alice', 20))`

```
try:
    # 데이터 삽입
    cur.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Alice', 20))
    cur.execute('INSERT INTO users (name, age) VALUES (?, ?)', ('Bob', 26))
    cur.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Charlie", 30))
    # 커밋
    conn.commit()
```

```
# DB 내용 보기
cur.execute("SELECT * FROM users")
rows = cur.fetchall()
for row in rows:
    print(row)
```

문제 출력 디버그 콘솔 터미널 포트

- 자료\5 파이썬\1 2기교안\miniProjects_40'; & s\ms-python.debugpy-2024.6.0-win32-x64\bund
- 로봇\감의자료\5 파이썬\1 2기교안\miniProjec
- PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇
- PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇
- PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇
- 자료\5 파이썬\1 2기교안\miniProjects_40'; & s\ms-python.debugpy-2024.6.0-win32-x64\bund
- 로봇\감의자료\5 파이썬\1 2기교안\miniProjec
- (1, 'Alice', 20)
- (2, 'Bob', 26)
- (3, 'Charlie', 30)
- PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇

SQLiteStudio (3.4.4) - [users (test1)]

Database Structure View Tools Help

Databases

Filter by name

test1 (SQLite 3)

Tables (1)

users

Views

Structure Data Constraints Indexes

Grid view Form view

id	name	age
1	Alice	20
2	Bob	26
3	Charlie	30

SQLITE3

데이터 여러 개 삽입

```
cur.executemany("INSERT INTO users (name, age) VALUES (?, ?)", data)
```

```
# 데이터 파라미터 리스트
data = [('John', 30), ('Jane', 24), ('Mike', 40), ('Hawx', 17)]

try:
    # 데이터 여러 개 삽입
    cur.executemany("INSERT INTO users (name, age) VALUES (?, ?)", data)
    # 커밋
    conn.commit()
```

문제 출력 디버그 콘솔 터미널 포트

```
PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇\강의자료\
PS D:\Dropbox\1_temp\2 자료\3 교육자료\로봇\강의자료\
자료\5 파이썬\1_2기교안\miniProjects_40'; & 'c:\Users\
s\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\de
로봇\강의자료\5 파이썬\1_2기교안\miniProjects_40\Flash
(1, 'Alice', 20)
(2, 'Bob', 26)
(3, 'Charlie', 30)
(4, 'John', 30)
(5, 'Jane', 24)
(6, 'Mike', 40)
(7, 'Hawx', 17)
```

	id	name	age
1	1	Alice	20
2	2	Bob	26
3	3	Charlie	30
4	4	John	30
5	5	Jane	24
6	6	Mike	40
7	7	Hawx	17

SQLITE3 (SQL_4.PY, SQL_5.PY)

- fetchone(), fetchmany(), fetchall()

```
#테이블의 모든 행 선택
cur.execute("SELECT * FROM users")
#커서가 있는 한 행 가져와서 출력하고 다음 행으로 커서 이동시킴
for _ in range(3):
    row = cur.fetchone()
```

```
try:
    # 테이블의 모든 행 선택
    cur.execute("SELECT * FROM users")
    rows = cur.fetchmany(3)

    # 출력
    for row in rows:
        print(row)

    print("-----")
    # 커서가 있는 행 이후 모든 행 가져오기
    rowAll = cur.fetchall()
    for row in rowAll:
        print(row)
```

SQLITE3 - 함수를 이용한 DB 관리

- info 테이블의 모든 항목 선택

```
cur.execute("SELECT * FROM INFO")
```

- info 테이블에서 no로 조회 선택

```
cur.execute("SELECT * FROM INFO WHERE  
NO=?", (no,))
```

- info 테이블에서 no, name 조건에 부합하는 항목 선택

```
cur.execute("SELECT * FROM info WHERE no > ?  
AND name = ?", (no, name))
```

- 새 데이터로 기존 정보 갱신

```
def u_info(no, name, age, btype, birth):  
    cur.execute("UPDATE info SET NAME=?, AGE=?, BTYPE=?,  
    BIRTH=? WHERE NO=?", (name, age, btype, birth, no))
```

```
## R: READ  
# 전체 조회  
def r_info_all():  
    cur.execute("SELECT * FROM INFO")  
    return cur.fetchall()  
  
# key로 조회  
def r_info_no(no):  
    cur.execute("SELECT * FROM INFO WHERE NO=?", (no,))  
    return cur.fetchall()  
  
# 이름으로 조회  
def r_info_name(name):  
    cur.execute("SELECT * FROM INFO WHERE NAME=?", (name,))  
    return cur.fetchall()  
  
# 조건으로 조회  
def r_info_condition(name, no):  
    cur.execute("SELECT * FROM info WHERE no > ? AND name = ?", (no, name))  
    return cur.fetchall()  
  
# U: UPDATE  
def u_info(no, name, age, btype, birth):  
    cur.execute("UPDATE info SET NAME=?, AGE=?, BTYPE=?, BIRTH=? WHERE NO=?")  
    print(f"{no}번 행이 변경됨.")  
# 주: '?' 순서대로임!! 함수의 인자 순서가 아님, 함수 사용 시에는 인자 순서대로이드
```


SQLITE3

(함수 실행 결과)

```
## CRUD 함수 생성
# C: CREATE (= INSERT)
def c_info(name, age, btype, birth):
    cur.execute("INSERT INTO info (NAME, AGE, BTYPE, BIRTH) VALUES(?, ?, ?, ?)", (name, age, btype, birth))
    conn.commit()
    print("새 정보가 추가됨")
```

The screenshot shows a database management tool interface. On the left, a tree view shows a database named 'test1 (SQLite 3)' containing a table named 'info'. The 'info' table is selected, and its structure is displayed in the main area. The table has five columns: NO, NAME, AGE, BTYPE, and BIRTH. The data is shown in a grid view with 5 rows. A red box highlights the 'info' table in the tree view and the table data in the grid view.

NO	NAME	AGE	BTYPE	BIRTH
1	홍길동	18	A	1678-09-10
2	강감찬	21	B	948-09-23
3	신사임당	34	O	1504-07-26
4	박사람	19	AB	2002-01-11
5	이순신	27	A	1545-09-28

```
python.exe d:/dropbox/2_camp/2_제1과/3_과제/3_과제.py
새 정보가 추가됨
새 정보가 추가됨
새 정보가 추가됨
새 정보가 추가됨
새 정보가 추가됨
(1, '홍길동', 18, 'A', '1678-09-10')
(2, '강감찬', 21, 'B', '948-09-23')
(3, '신사임당', 34, 'O', '1504-07-26')
(4, '박사람', 19, 'AB', '2002-01-11')
(5, '이순신', 27, 'A', '1545-09-28')
```