

# 파이썬

## 40강. File to DB

## 1. File to DB

- 파이썬을 이용하여 MariaDB(MySQL)에 연결하기 위해서는 환경변수를 이용한다. 환경변수는 데이터베이스가 설치된 호스트 주소, 데이터베이스 이름, 포트 번호, 그리고 접근권한을 갖는 사용자 계정 과 비밀번호 등을 말한다.

## 2. csv to table

- CSV 파일은 콤마를 기준으로 열 단위 자료를 작성한 파일이므로 데이터베이스의 테이블 구조와 호환 성이 높다. 따라서 CSV 파일의 자료를 CRUD 작업이 가능한 테이블로 저장할 수 있다.
- 다음 예문은 1차 실행 시 테이블을 만들고, CSV 파일 자료에서 100개의 행을 테이블의 레코드에 저장한다. 2차 실행을 하면 테이블에 저장된 레코드 100개가 검색된다.

## 2. csv to table

chapter10.lecture.step05\_csv\_to\_table.py ↵

Python Console ↵

```
'''  
csv -> db table  
  1차 실행 : table 생성 -> 레코드 추가  
  2차 실행 : 레코드 검색  
'''  
  
import pandas as pd
```

## 2. csv to table

```
import pymysql

# (1) csv 파일 로드
bmi = pd.read_csv("chapter10/data/bmi.csv")
print(bmi.info())
'''
height    20000 non-null int64 -> int64
weight    20000 non-null int64 -> int64
label     20000 non-null object -> varchar(20)
'''

# (2) 각 칼럼 추출
height = bmi['height']
weight = bmi['weight']
label = bmi['label']

config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset': 'utf8',
    'use_unicode' : True}

try :
    conn = pymysql.connect(**config)
    cursor = conn.cursor()
```

<class  
'pandas.core.frame.DataFrame'  
>  
RangeIndex: 20000 entries, 0 to 19999  
Data columns (total 3 columns):  
height 20000 non-null int64  
weight 20000 non-null int64  
label 20000 non-null object  
dtypes: int64(2), object(1)  
memory usage: 390.7+ KB

## 2. csv to table

```
# (3) table 조회
cursor.execute("show tables")
tables = cursor.fetchall() ↵
```

```
# (4) 스위칭기법 ↵
sw = False ↵
for table in tables : ↵
    if table[0] == 'bmi tab' : ↵
        sw = True # table 있는 경우 swapp ↵
ing
```

```
(5) table 생성 ↵
if not sw : ↵
    print('테이블 없음') # table 없으면 생성
    sql="""create table
    bmi_tab( height int not null, ↵
    weight int not null, ↵
```

## 2. csv to table

```

        label varchar(15) not null
    ) ""
    cursor.execute(sql)

    # (6) 레코드 조회
    cursor.execute("select * from bmi tab
")
    rows = cursor.fetchall()

    if rows : # (7) 레코드 있는 경우 : 레코드 조회
        for row in rows :
            print(f"{row[0]} {row[1]} {row[2]}")
        print('전체 레코드 수 : ', len(rows))

    else: # (8) 레코드 없는 경우 : 레코드 추가
        print("100 레코드 추가")
        for i in range(100) :
            h = height[i]
            w = weight[i]
            lab = label[i]

            cursor.execute(f"insert into bmi values({h}, {w}, '{lab}')"
conn.commit()

except Exception as e:
    print('db error :', e)
finally:
    cursor.close()
    conn.close()

```

## 2. csv to table

- csv 파일을 테이블에 저장하기 예
- # (1) csv 파일 로드
- 테이블에 저장할 csv 파일을 불러온 후 info()함수를 이용하여 각 칼럼의 자료형을 확인한다. 테이블 설계 시 int64는 int형으로, object는 varchar형으로 지정한다.
- # (2) 각 칼럼 추출
- 테이블의 각 칼럼에 자료를 삽입하기 위해서 칼럼 단위로 자료를 준비한다.
- # (3) table 조회
- work 데이터베이스에서 전체 테이블을 조회하는 SQL문을 실행한다.
- # (4) 스위칭기법
- work 데이터베이스에서 'bmi\_tab' 테이블의 유무를 비교 판단하여 sw 변수에 True 또는 False값을 할당한다.(더 알아보기 참고)



## 2. csv to table

- csv 파일을 테이블에 저장하기 예
- # (5) table 생성
- sw 변수가 False 이면 테이블이 없는 것으로 가정하고, 'bmi\_tab' 테이블을 생성한다.
- # (6) 레코드 조회
- 테이블을 생성 후 해당 테이블의 전체 레코드를 조회하는 SQL문을 실행한다.
- # (7) 레코드 있는 경우 : 레코드 조회
- 만약 레코드가 1개 이상 있으면 전체 레코드를 조회하고 출력한다.
- # (8) 레코드 없는 경우 : 레코드 추가
- 만약 검색된 레코드가 없으면 방금 테이블이 생성된 것으로 가정하고, csv 파일에서 앞부분 100개의 행만 테이블의 각 칼럼에 삽입한다.

### 3. json to table

- JSON은 {'키': '값'} 형식으로 제공하는 파일이다. json 모듈을 이용하여 JSON 파일의 자료를 디코딩한 이후 데이터프레임 형식으로 변경하면 테이블 구조의 칼럼 단위로 저장할 수 있다.
- 다음은 예문에서 사용될 JSON 파일이다. 각 행 단위로 {'키': '값'}을 한 쌍으로 5개 원소를 가지고 있다. {}은 테이블의 레코드, "키:값"은 레코드를 구성하는 칼럼으로 삽입될 것이다..

### 3. json to table

```
[  
  {  
    "id": 76811,  
    "url": "https://api.github.com/repos/pandas-dev/pandas/labels/Bug",  
    "name": "Bug",  
    "color": "e10c02",  
    "default": false  
  },  
  {  
    "id": 76812,  
    "url": "https://api.github.com/repos/pandas-dev/pandas/labels/Enhancement",  
    "name": "Enhancement",  
    "color": "4E9A06",  
    "default": false  
  },  
  생략 \ \ \ \ \  
]
```

### 3. json to table

chapter10.lecture.step06\_json\_to\_table.py

Python Console

```
import json

# (1) json 파일 로드
file=open("chapter10_Database/data/labels.json", mode='r'
encoding="utf-8")

# (2) decoding : json 문자열 -> dict
lines = json.load(file)
print(type(lines)) # <class 'list'>
print(len(lines)) # 30
print(type(lines[0])) # <class 'dict'>

# (3) DataFrame 생성
import pandas as pd
df = pd.DataFrame(lines)
print(df.info())
print(df.head())
```

<class 'list'>  
30  
<class 'dict'>  
<class  
'pandas.core.frame.DataFrame'  
>  
RangeIndex: 30 entries, 0 to 29  
Data columns (total 5 columns):  
id            30 non-null int64  
url          30 non-null object  
name        30 non-null object

### 3. json to table

```
import pymysql
config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset': 'utf8',
    'use_unicode' : True}

try :
    conn = pymysql.connect(**config)
    cursor = conn.cursor()

    # (4) Table 생성
    sql = """create table
labels(id int not null,
url varchar(150) not null,
name varchar(50) not null,
color varchar(50) not null,
de char(5)
)"""
    cursor.execute(sql) # table 생성
```

```
color 30 non-null object
default 30 non-null bool
dtypes: bool(1), int64(1), o
bject(3)
```

### 3. json to table

```
# (5) 레코드 조회↵
cursor.execute("select * from labels")
rows = cursor.fetchall()↵
if rows : # (6) 레코드 있는 경우↵
    print("labels 레코드 조회")↵
    for row in rows :
        print(row)↵

    print("전체 레코드 수 :", len(rows))
else : # (7) 레코드 없는 경우↵
    for i in range(30) :↵
        uid = df.id[i] # df.column
        url = df.url[i]↵
        name = df.name[i]
        color = df.color[i]↵
        de = str(df.default[i]) # bool↵

-> str↵

    sql = f"insert into labels
           values({uid}, '{url}', '↵
{name}', '{color}', '{de}')"↵
    cursor.execute(sql)
    conn.commit() # db 반영↵
```

### 3. json to table

```
except Exception as e:  
    print("db error : ", e)↵  
finally:↵  
    cursor.close()  
    conn.close()↵
```

---

### 3. json to table

- json 자료를 테이블에 저장하기 예
- # (1) json 파일 로드
- 읽기 모드(mode='r')로 파일 객체를 생성하여 json 파일을 읽어온다.
- # (2) decoding : json 문자열 -> dict
- 파일 객체를 대상으로 json 모듈의 load()함수를 이용하여 디코딩(json 문자열을 파이썬 딕트 객체로 변환)한다.
- # (3) DataFrame 생성
- 디코딩된 결과를 대상으로 pandas의 데이터프레임 객체로 변환한다. info()함수를 이용하여 데이터 프레임의 정보를 확인하면, 30개의 행과 5개의 칼럼을 갖는 2차원의 데이터프레임 객체를 확인할 수 있다. 기존 JSON 파일에서 한 개의 {}가 한 개의 행으로 만들어지고, {}를 구성하는 원소들은 각 행을 구성하는 칼럼이 된다.



### 3. json to table

- # (4) Table 생성
- 데이터프레임을 구성하는 5개의 칼럼을 테이블로 저장하기 위해서 'labels' 테이블을 생성한다. 테이블을 구성하는 칼럼의 자료형은 데이터프레임을 구성하는 칼럼의 자료형을 참조하여 지정한다. 예를 들면 id는 정수형(int)으로 지정하고, 나머지 4개 칼럼은 문자형(varchar or char)으로 지정한다.
- ※ 테이블을 생성은 SQL문은 한 번만 실행해야 되기 때문에 1회 실행 후 주석을 처리한다.
- # (5) 레코드 조회
- 테이블을 생성 후 해당 테이블의 전체 레코드를 조회하는 SQL문을 실행한다.

### 3. json to table

- # (6) 레코드 있는 경우
- 만약 레코드가 1개 이상 있으면 전체 레코드를 조회하고 출력한다.
- # (7) 레코드 없는 경우
- 만약 검색된 레코드가 없으면 방금 테이블이 생성된 것으로 가정하고, 데이터프레임의 각 칼럼을 테이블의 각 칼럼에 1:1로 삽입한다.
- 데이터프레임을 구성하는 마지막 default 칼럼은 자료형이 bool이기 때문에 str()함수를 이용하여 문자열 객체로 변환한 후 테이블에 저장해야 한다.

## 4. Table join

- 데이터베이스는 여러 개의 관련 있는 테이블로 구성된다. 이러한 테이블들은 특정 칼럼을 이용하여 두 개 이상의 테이블들을 하나로 연결할 수 있는데, 이러한 기법을 테이블 조인(Table join)이라고 한다.
- 다음은 사원 테이블(EMP)과 부서 테이블(DEPT)이 부서명에 의해서 테이블이 연결되고 있다.

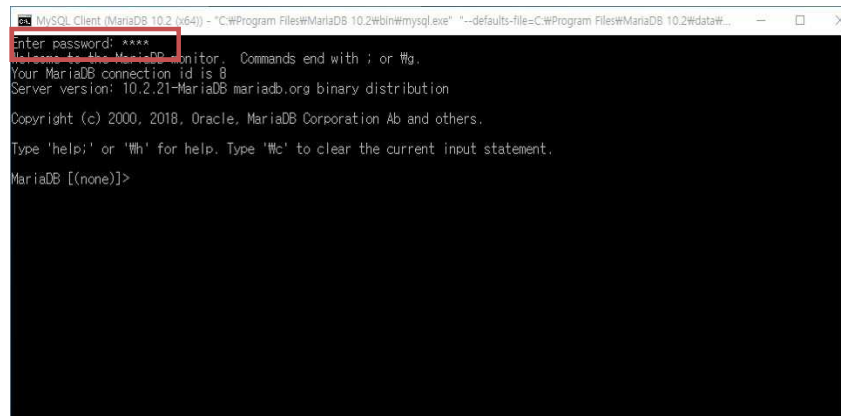
사원 테이블(EMP)						부서 테이블(DEPT)	
사번	이름	입사일	급여	보너스	부서명	부서명	부서주소
int	char	date	int	int	char	char	date

## 5. 테이블 만들기

- 테이블을 조인하기 위해서 다음과 같은 단계를 거쳐서 사원 테이블과 부서 테이블을 작성한다.
- (1) 사원 테이블

## 5. 테이블 만들기

- [단계 1] MariaDB 실행과 접속
- ① Window의 시작 메뉴에서 [모든 프로그램] → [MariaDB 10.1 (x64)] → [MySQL Client (MariaDB 10.1 (x64))] 메뉴를 선택한다.
- ② MySQL Client 창에서 Enter password : 관리자 비밀번호를 입력하고, 뿔키를 누르면 MariaDB 프롬프트(MariaDB [(none)]>)가 나타난다.



```
MySQL Client (MariaDB 10.2 (x64)) - "C:\Program Files\MariaDB 10.2\bin\mysql.exe" "--defaults-file=C:\Program Files\MariaDB 10.2\data#...
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.2.21-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\w' to clear the current input statement.
MariaDB [(none)]>
```

## 5. 테이블 만들기

- [단계 2] 사용자 데이터베이스(work) 선택
- MariaDB [<none>] use work; // db 선택
- Databases changed

## 5. 테이블 만들기

- [단계 3] 테이블 만들기(emp)

```
MariaDB [(none)]> use work;  
Database changed  
MariaDB [work]> create or replace table emp(  
    -> eno int auto_increment primary key,  
    -> ename varchar(25) not null,  
    -> hiredate date not null,  
    -> pay int default 0,  
    -> bonus int default 0,  
    -> dname varchar(50)  
    -> );  
Query OK, 0 rows affected (0.99 sec)  
  
MariaDB [work]> alter table emp auto_increment = 1001;  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

## 5. 테이블 만들기

- [단계 4] 사번 1001부터 1씩 자동 증가(1001 ~ n)
- MariaDB [<work>] alter table auto\_increment = 1001;
- [단계 5] 테이블에 레코드 추가(4개)
- MariaDB [<work>] insert into emp(ename,hiredate,pay,bonus,dname) values('홍길동','2008-03-10', 300, 15, '영업부');
- MariaDB [<work>] insert into emp(ename,hiredate,pay,dname) values('강호동','2010-03-10', 250, '판매부');
- MariaDB [<work>] insert into emp(ename,hiredate,pay,bonus,dname) values('유관순','2008-03-10', 200, 10, '회계부');
- MariaDB [<work>] insert into emp(ename,hiredate,pay,bonus,dname) values('강감찬','2007-01-10', 400, 25, '영업부');



## 5. 테이블 만들기

- [단계 6] 레코드 조회
- MariaDB [<work>] select \* from emp;

```
MariaDB [work]> select * from emp;
```

eno	ename	hiredate	pay	bonus	dname
1001	홍길동	2008-03-10	300	15	영업부
1002	강호동	2010-03-10	250	0	판매부
1003	유관순	2008-03-10	200	10	회계부

```
3 rows in set (0.04 sec)

MariaDB [work]> commit;
Query OK, 0 rows affected (0.04 sec)

MariaDB [work]>
```

## 5. 테이블 만들기

- (2) 부서 테이블
- [단계 1] 테이블 만들기(dept)
- MariaDB [<work>] create or replace table dept(
  - -> dname varchar(50) not null,
  - -> daddr varchar(100)
  - -> );
- [단계 2] 테이블에 레코드 추가(3개)
- MariaDB [<work>] insert into dept values('영업부', '뉴욕시'); MariaDB [<work>] insert into dept values('판매부', '서울시'); MariaDB [<work>] insert into dept values('회계부', '대전시');

## 5. 테이블 만들기

- [단계 3] 레코드 조회
- MariaDB [<work>] select \* from dept;

```
MariaDB [work]> select * from dept;
+-----+-----+
| dname | daddr |
+-----+-----+
| 영업부 | 뉴욕시 |
| 판매부 | 서울시 |
| 회계부 | 대전시 |
+-----+-----+
3 rows in set (0.00 sec)

MariaDB [work]> commit;
Query OK, 0 rows affected (0.00 sec)
```

## 6. 테이블 조인

- 두 개의 테이블을 대상으로 부서명(dname) 칼을 기준으로 조인하는 과정을 예문으로 알아본다.

## 6. 테이블 조인

```
chapter10.lecture.step07_table_join.py Python Console

import pymysql

config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset':'utf8',
    'use_unicode' : True}

try :
    conn = pymysql.connect(**config)
    cursor = conn.cursor()

    # (1) Table join
    pay = int(input('join 급여 입력 : '))
    sql = f"""select e.eno, e.ename, e.pay, d.
    dname, daddr
    from emp e inner join dept d
    on e.dname = d.dname and e.pay >= {pay}"""

    # (2) 레코드 조회
    cursor.execute(sql)
    data = cursor.fetchall()
    for row in data :
        print(row[0], row[1], row[2], row[3],
row[4])

    print('검색된 레코드 수 : ', len(data))

except Exception as e :
    print('db error : ', e)
finally:
    cursor.close()
    conn.close()
```

join 급여 입력 : 250

1001 홍길동 300 영남부 뉴욕시  
1002 강호동 250 판매부 서울시  
검색된 레코드 수 : 2

## 6. 테이블 조인

- 테이블 조인 예
- # (1) Table join
- from절에서 emp 테이블은 별칭 e, dept 테이블은 별칭 d로 지정하고, on절에서는 별칭을 이용하여 각 테이블의 dname 칼럼으로 조인한다. 또한 키보드로 입력받은 pay를 조건식으로 추가한다. 끝으로 select절에서는 각 테이블의 별칭을 이용하여 조회할 컬럼명을 지정한다.
- # (2) 레코드 조회
- 조인 조건식에 만족하는 레코드를 조회하여 결과를 출력한다. 출력 결과에서 사번, 이름, 급여는 emp 테이블에서 가져온 컬럼이고, 부서명과 부서주소는 dept 테이블에서 가져온 컬럼이다.

THANK YOU