# pandas 라이브러리와 pymysql

1. read_sql()
    - sql 연결객체를 활용하여 쿼리 구문으로 반환된 결과를 데이터프레임으로 바로 생성해 주는 함수

- 테이블 생성

In [1]:
```
# DB 생성 쿼리
create_db_query = """
CREATE DATABASE IF NOT EXISTS student_mgmt;
"""
# DB 사용 쿼리
use_db_query = """
USE student_mgmt;
"""
# 테이블 삭제 쿼리
drop_table_query = """
DROP TABLE IF EXISTS students;
"""
# 테이블 생성 쿼리
create_table_query = """
CREATE TABLE students (
id TINYINT NOT NULL AUTO_INCREMENT,
name VARCHAR(10) NOT NULL,
gender ENUM('man','woman') NOT NULL,
birth DATE NOT NULL,
english TINYINT NOT NULL,
math TINYINT NOT NULL,
korean TINYINT NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
"""
```

- 데이터 입력

In [2]:
```
# 데이터 입력 쿼리
inserte_data_query = """
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('dave', 'man', '1983-07-16', 90, 80, 71);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('minsun', 'woman', '1982-10-16', 30, 88, 60);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('david', 'man', '1982-12-10', 78, 77, 30);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('jade', 'man', '1979-11-1', 45, 66, 20);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('jane', 'man', '1990-11-12', 65, 32, 90);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('wage', 'woman', '1982-1-13', 76, 30, 80);
INSERT INTO students (name, gender, birth, english, math, korean) VALUES ('tina', 'woman', '1982-12-3', 87, 62, 71);
"""
```

- read_sql()
- sql 연결객체를 활용하여 쿼리 구문으로 반환된 결과를 데이터프레임으로 바로 생성해 주는 함수

In [3]:
```
import pymysql
import pandas as pd
```

In [4]:
```
# password에는 본인의 MySQL 비밀번호를 입력

host_name = 'localhost'
host_port = 3306
username = 'root'
password = '0000'
charset = 'utf8'
```

In [5]:
```
# db 연결
db = pymysql.connect(
    host=host_name,     # MySQL Server Address
    port=host_port,     # MySQL Server Port
    user=username,      # MySQL username
    password=password,  # password for MySQL username
    charset=charset
)
```

In [6]:
```
cursor = db.cursor()
```

In [7]:
```
# DB 및 TABLE 생성
cursor.execute(create_db_query)
cursor.execute(use_db_query)
cursor.execute(drop_table_query)
cursor.execute(create_table_query)
db.commit()
```

In [8]:
```
# pymysql은 기본적으로 한 번에 여러 INSERT 문을 실행하는 multi-statement를 허용하지 않음
# execute()로는 ';'로 끝나는 하나의 Query문만 처리 가능
# Query문을 ';' 기준으로 split 하여 for문으로 하나씩 실행
for query in inserte_data_query.split(";"):
    query = query.strip()
    if query:
        cursor.execute(query)
db.commit()
```

pandas.read_sql(쿼리, 연결된 db connection 객체)

In [9]:
```
sql = "SHOW TABLES"
```

In [10]:
```
df = pd.read_sql(sql, db)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21416\3349604202.py:1: UserWarning: pandas only supports SQLAlchemy connectable (eng
ine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df = pd.read_sql(sql, db)
```

In [11]:
```
df
```

Out[11]:

| | Tables_in_student_mgmt |
|---|---|
| **0** | students |

In [12]:
```
sql = "SELECT * FROM students"
df = pd.read_sql(sql,db)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21416\3088303204.py:2: UserWarning: pandas only supports SQLAlchemy connectable (eng
ine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df = pd.read_sql(sql,db)
```

In [13]:
```
df
```

Out[13]:

| | id | name | gender | birth | english | math | korean |
|---|---|---|---|---|---|---|---|
| **0** | 1 | dave | man | 1983-07-16 | 90 | 80 | 71 |
| **1** | 2 | minsun | woman | 1982-10-16 | 30 | 88 | 60 |
| **2** | 3 | david | man | 1982-12-10 | 78 | 77 | 30 |
| **3** | 4 | jade | man | 1979-11-01 | 45 | 66 | 20 |
| **4** | 5 | jane | man | 1990-11-12 | 65 | 32 | 90 |
| **5** | 6 | wage | woman | 1982-01-13 | 76 | 30 | 80 |
| **6** | 7 | tina | woman | 1982-12-03 | 87 | 62 | 71 |

In [14]:
```
type(df['math'][0]) # 테이블의 컬럼 형식을 그대로 유지
```

Out[14]:
```
numpy.int64
```

In [15]:
```
df.to_csv('students.csv', sep=',', index=False, encoding='utf-8')
df
```

Out[15]:

| | id | name | gender | birth | english | math | korean |
|---|---|---|---|---|---|---|---|
| **0** | 1 | dave | man | 1983-07-16 | 90 | 80 | 71 |
| **1** | 2 | minsun | woman | 1982-10-16 | 30 | 88 | 60 |
| **2** | 3 | david | man | 1982-12-10 | 78 | 77 | 30 |
| **3** | 4 | jade | man | 1979-11-01 | 45 | 66 | 20 |
| **4** | 5 | jane | man | 1990-11-12 | 65 | 32 | 90 |
| **5** | 6 | wage | woman | 1982-01-13 | 76 | 30 | 80 |
| **6** | 7 | tina | woman | 1982-12-03 | 87 | 62 | 71 |

In [16]:
```
db.close()
```

## 외래키(FOREIGN KEY)를 만드는 이유

- 두 테이블 사이에 관계를 선언해서, 데이터의 무결성을 보장

In [17]:
```
import pymysql
import pandas as pd
```

In [18]:
```
host_name = 'localhost'
host_port = 3306
username = 'root'
password = '0000'
```

In [19]:
```
db = pymysql.connect(
    host=host_name,
    port=host_port,
    user=username,
    password=password,
    charset='utf8'
)
```

In [20]:
```
cursor = db.cursor()
```

In [21]:
```
# 여러 쿼리문이 들어있는 sqlDB.sql 파일을 읽기
with open('sqlDB.sql', 'r', encoding='utf-8') as f:
    sql_file = f.read()

sql_querys = sql_file.split(";")
```

In [22]:
```
# 각각의 쿼리문 실행
for query in sql_querys:
    query = query.strip()
    if query:
        cursor.execute(query)
```

In [23]:
```
db.commit()
```

In [24]:
```
sql = "SELECT * FROM usertbl"
df = pd.read_sql(sql, db)
df
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21416\2085659957.py:2: UserWarning: pandas only supports SQLAlchemy connectable (eng
ine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df = pd.read_sql(sql, db)
```

Out[24]:

| | userID | name | birthYear | addr | mobile1 | mobile2 | height | mDate |
|---|---|---|---|---|---|---|---|---|
| **0** | BBK | 바비킴 | 1973 | 서울 | 010 | 0000000 | 176 | 2013-05-05 |
| **1** | EJW | 은지원 | 1972 | 경북 | 011 | 8888888 | 174 | 2014-03-03 |
| **2** | JKW | 조관우 | 1965 | 경기 | 018 | 9999999 | 172 | 2010-10-10 |
| **3** | JYP | 조용필 | 1950 | 경기 | 011 | 4444444 | 166 | 2009-04-04 |
| **4** | KBS | 김범수 | 1979 | 경남 | 011 | 2222222 | 173 | 2012-04-04 |
| **5** | KKH | 김경호 | 1971 | 전남 | 019 | 3333333 | 177 | 2007-07-07 |
| **6** | LJB | 임재범 | 1963 | 서울 | 016 | 6666666 | 182 | 2009-09-09 |
| **7** | LSG | 이승기 | 1987 | 서울 | 011 | 1111111 | 182 | 2008-08-08 |
| **8** | SSK | 성시경 | 1979 | 서울 | None | None | 186 | 2013-12-12 |
| **9** | YJS | 윤종신 | 1969 | 경남 | None | None | 170 | 2005-05-05 |

In [25]:
```
sql = "SELECT * FROM buytbl"
df = pd.read_sql(sql, db)
df
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21416\2541138876.py:2: UserWarning: pandas only supports SQLAlchemy connectable (eng
ine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df = pd.read_sql(sql, db)
```

Out[25]:

| | num | userID | prodName | groupName | price | amount |
|---|---|---|---|---|---|---|
| **0** | 1 | KBS | 운동화 | None | 30 | 2 |
| **1** | 2 | KBS | 노트북 | 전자 | 1000 | 1 |
| **2** | 3 | JYP | 모니터 | 전자 | 200 | 1 |
| **3** | 4 | BBK | 모니터 | 전자 | 200 | 5 |
| **4** | 5 | KBS | 청바지 | 의류 | 50 | 3 |
| **5** | 6 | BBK | 메모리 | 전자 | 80 | 10 |
| **6** | 7 | SSK | 책 | 서적 | 15 | 5 |
| **7** | 8 | EJW | 책 | 서적 | 15 | 2 |
| **8** | 9 | EJW | 청바지 | 의류 | 50 | 1 |
| **9** | 10 | BBK | 운동화 | None | 30 | 2 |
| **10** | 11 | EJW | 책 | 서적 | 15 | 1 |
| **11** | 12 | BBK | 운동화 | None | 30 | 2 |

## buyTbl에 데이터를 추가

- 외래키로 지정되어 있는 userID에 입력되는 새로운 값 STJ가 userTbl에 없는 값이어서 무결성 오류 발생

In [26]:
```
cursor = db.cursor()
sql = "INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('STJ', '운동화', '의류', 30, 2);"
cursor.execute(sql)
db.commit()
```

```
---------------------------------------------------------------------------
IntegrityError                            Traceback (most recent call last)
Cell In[26], line 3
      1 cursor = db.cursor()
      2 sql = "INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('STJ', '운동화', '의류', 30, 2);"
----> 3 cursor.execute(sql)
      4 db.commit()

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\cursors.py:153, in Cursor.execute(self, query, args)
    149     pass
    151 query = self.mogrify(query, args)
--> 153 result = self._query(query)
    154 self._executed = query
    155 return result

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\cursors.py:322, in Cursor._query(self, q)
    320 conn = self._get_db()
    321 self._clear_result()
--> 322 conn.query(q)
    323 self._do_get_result()
    324 return self.rowcount

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:575, in Connection.query(self, sql, unbuffered)
    573     sql = sql.encode(self.encoding, "surrogateescape")
    574 self._execute_command(COMMAND.COM_QUERY, sql)
--> 575 self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    576 return self._affected_rows

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:826, in Connection._read_query_result(self, unbuffered)
    824     result.init_unbuffered_query()
    825 else:
--> 826     result.read()
    827 self._result = result
    828 if result.server_status is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:1203, in MySQLResult.read(self)
   1201 def read(self):
   1202     try:
-> 1203         first_packet = self.connection._read_packet()
   1205         if first_packet.is_ok_packet():
```

```
   1206             self._read_ok_packet(first_packet)

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:782, in Connection._read_packet(self, packet_type)
    780     if self._result is not None and self._result.unbuffered_active is True:
    781         self._result.unbuffered_active = False
--> 782     packet.raise_for_error()
    783 return packet

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\protocol.py:219, in MysqlPacket.raise_for_error(self)
    217 if DEBUG:
    218     print("errno =", errno)
--> 219 err.raise_mysql_exception(self._data)

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\err.py:150, in raise_mysql_exception(data)
    148 if errorclass is None:
    149     errorclass = InternalError if errno < 1000 else OperationalError
--> 150 raise errorclass(errno, errval)

IntegrityError: (1452, 'Cannot add or update a child row: a foreign key constraint fails (`sqldb`.`buytbl`, CONSTRAINT `buytbl_
ibfk_1` FOREIGN KEY (`userID`) REFERENCES `usertbl` (`userID`))')
```

### 에러가 나면 정상임

- CONSTRAINT `buyTbl_ibfk_1` FOREIGN KEY ( `userID` ) REFERENCES `userTbl` ( `userID` )
- userTbl 에 userID가 STJ인 데이터가 없기 때문에,
  - FOREIGN KEY (userID) REFERENCES userTbl(userID)
  - buyTbl 테이블의 userID 커럼은 userTbl 테이블의 userID를 참조할 때, userTbl 테이블에 userID가 STJ인 데이터가 없으면, 입력이 안 됨
  - 데이터 무결성 (두 테이블간 관계에 있어서, 데이터의 정확성을 보장하는 제약 조건을 넣는 것임)
  - 현업에서는 꼭 필요한 경우만 사용하는 경우가 많음 (비즈니스 로직이 다양하기 때문에, 제약을 걸어놓을 경우, 예외적인 비즈니스 로직 처리가 어렵기 때문)

In [27]:
```
cursor = db.cursor()
sql = "INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('BBK', '운동화', '의류', 30, 2);"
cursor.execute(sql)
db.commit()
```

In [28]:
```
db.close()
```

In [29]:
```
# db 연결을 활성화 해주는 함수 구현
def conn(d_name) :
    import pymysql
    host_name = 'localhost'
    host_port = 3306
    username = 'root'
    password = '0000'
    database_name = d_name

    db = pymysql.connect(
    host=host_name, # MySQL Server Address
    port=host_port, # MySQL Server Port
    user=username, # MySQL username
    password=password, # password for MySQL username
    database=database_name, # Database name
    charset='utf8'
    )
    return db
```

In [30]:
```
db = conn('sqlDB')
```

이번에는 userTbl 에 userID가 STJ 인 데이터를 넣어준 후에, 다시 buyTbl userID에 STJ 관련 데이터를 넣어줌

In [31]:
```
cursor = db.cursor()
sql = "INSERT INTO userTbl VALUES('STJ', '서태지', 1975, '경기', '011', 'toortoor', 171, '2014-4-4');"
cursor.execute(sql)
db.commit()
```

In [32]:
```
sql = "INSERT INTO buyTbl (userID, prodName, groupName, price, amount) VALUES('STJ', '운동화', '의류', 30, 2);"
cursor.execute(sql)
db.commit()
```

이번에는 userTbl에 userID가 STJ 관련 데이터를 삭제해봄

In [33]:
```
sql = "DELETE FROM userTbl WHERE userID='STJ'"
cursor.execute(sql)
db.commit()
```

```
---------------------------------------------------------------------------
IntegrityError                            Traceback (most recent call last)
Cell In[33], line 2
      1 sql = "DELETE FROM userTbl WHERE userID='STJ'"
----> 2 cursor.execute(sql)
      3 db.commit()

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\cursors.py:153, in Cursor.execute(self, query, args)
    149     pass
    151 query = self.mogrify(query, args)
--> 153 result = self._query(query)
    154 self._executed = query
    155 return result

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\cursors.py:322, in Cursor._query(self, q)
    320 conn = self._get_db()
    321 self._clear_result()
--> 322 conn.query(q)
    323 self._do_get_result()
    324 return self.rowcount

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:575, in Connection.query(self, sql, unbuffered)
    573     sql = sql.encode(self.encoding, "surrogateescape")
    574 self._execute_command(COMMAND.COM_QUERY, sql)
--> 575 self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    576 return self._affected_rows

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:826, in Connection._read_query_result(self, unbuffered)
    824     result.init_unbuffered_query()
    825 else:
--> 826     result.read()
    827 self._result = result
    828 if result.server_status is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:1203, in MySQLResult.read(self)
   1201 def read(self):
   1202     try:
-> 1203         first_packet = self.connection._read_packet()
   1205         if first_packet.is_ok_packet():
   1206             self._read_ok_packet(first_packet)
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\connections.py:782, in Connection._read_packet(self, packet_type)
    780     if self._result is not None and self._result.unbuffered_active is True:
    781         self._result.unbuffered_active = False
--> 782     packet.raise_for_error()
    783 return packet

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\protocol.py:219, in MysqlPacket.raise_for_error(self)
    217 if DEBUG:
    218     print("errno =", errno)
--> 219 err.raise_mysql_exception(self._data)

File C:\ProgramData\anaconda3\Lib\site-packages\pymysql\err.py:150, in raise_mysql_exception(data)
    148 if errorclass is None:
    149     errorclass = InternalError if errno < 1000 else OperationalError
--> 150 raise errorclass(errno, errval)

IntegrityError: (1451, 'Cannot delete or update a parent row: a foreign key constraint fails (`sqldb`.`buytbl`, CONSTRAINT `buy
tbl_ibfk_1` FOREIGN KEY (`userID`) REFERENCES `usertbl` (`userID`))')
```

에러나면 정상입니다.

- buyTbl 에 해당 userID를 참조하는 데이터가 있기 때문

In [34]:
```
db.close()
```

In [ ]:
```
```