SQL 활용 데이터 프레임 생성

Python에서 Pandas와의 연계를 위한 MySQL과의 연동은 다양한 방법이 있지만 그 중에서 `pymysql`과 `SQLAlchemy`가 가장 널리 사용된다.

#01. 패키지 참조

`pymysql`과 `sqlalchemy` 패키지가 미리 설치되어 있어야 한다.

```
import pymysql
from sqlalchemy import create_engine
from pandas import DataFrame
from pandas import read_sql, read_sql_table
```

#02. `pymysql` 사용

`pymysql`은 가장 기본적인 python MySQL 관련 패키지

SQL문을 직접적으로 사용할 수 있다.

1) 데이터베이스 접속

```
dbcon = pymysql.connect(host="127.0.0.1",    # 서버주소
                        port=3306,           # 포트번호
                        user="root",         # 계정이름
                        password="root",     # 비밀번호
                        db="board",          # 데이터베이스이름
                        charset="utf8"       # 인코딩
                    )
```

2) 데이터 조회

기본 사용 방법

테이블의 각 record를 튜플로 표현하는 리스트 객체를 얻을 수 있다.

데이터 조회를 위한 커서 객체 생성

```
cursor = dbcon.cursor()
```

데이터 조회를 위한 SQL문 처리

```
sql = "SELECT * FROM board_main_post"
cursor.execute(sql)
result = cursor.fetchall()
result
```

딕셔너리 형태로 데이터 조회

데이터 조회를 위한 커서 객체 생성

객체 생성시 파라미터를 전달해야 한다.

```
cursor = dbcon.cursor(pymysql.cursors.DictCursor)
```

데이터 조회하기

```
sql = "SELECT * FROM board_main_post"
cursor.execute(sql)
result = cursor.fetchall()
result
```

조회결과를 데이터프레임으로 변환

```
df = DataFrame(result)
df
```

데이터 프레임에 대한 인덱스 설정

```
df.set_index('id', inplace=True)
df
```

3) 입력, 수정, 삭제

`INSERT`, `UPDATE`, `DELETE` 문의 수행 방식은 동일하다.

> 여기서는 데이터 조회 과정에서 생성한 `cursor` 객체를 재사용 한다.

데이터 입력

```
sql = """INSERT INTO board_main_post (title, contents, created_at, updated_at)
        VALUES ('pandas test', '이것은 테스트 입니다.', now(), now())"""
print(sql)

rows = cursor.execute(sql)
print("%d개의 행이 저장됨" % rows)
print("생성된 Primary Key: %d" % cursor.lastrowid)

# 처리 결과를 실제로 반영함
dbcon.commit()

# 되돌리기
# --> 이미 commit()한 내역은 적용안됨
#dbcon.rollback()
```

데이터 수정

```
sql = """UPDATE board_main_post
        SET title='수정된 제목',
            contents='수정된 내용',
            updated_at=now()
        WHERE id=14"""
print(sql)

rows = cursor.execute(sql)
print("%d개의 행이 갱신됨" % rows)

dbcon.commit()
```

데이터 삭제

```
sql = "DELETE FROM board_main_post WHERE id > 10"
print(sql)

rows = cursor.execute(sql)
print("%d개의 행이 삭제됨" % rows)

dbcon.commit()
```

데이터베이스 접속 해제

```
cursor.close()
dbcon.close()
```

#02. SQLAlchemy 사용

1) 데이터베이스 접속

데이터베이스 접속 패키지 설치

```
pymysql.install_as_MySQLdb()
import MySQLdb
```

접속 문자열 생성

`mysql+mysqldb://계정이름:비밀번호@:포트번호/데이터베이스이름?charset=인코딩`

```
conStr = "mysql+mysqldb://root:root@127.0.0.1:3306/board?charset=utf8"
conStr
```

데이터베이스 접속하기

```
engine = create_engine(conStr)
conn = engine.connect()
```

2) 데이터 조회하기

SQL문 사용하기

```
df = read_sql("SELECT * FROM board_main_post", index_col="id", con=conn)
df
```

테이블의 데이터를 직접 가져오기

모든 데이터 조회

```
df = read_sql_table('board_main_post', con=conn)
df
```

인덱스를 지정한 조회

`read_sql_table` 함수를 사용할 경우 WHERE절은 사용할 수 없다.

```
df = read_sql_table('board_main_post', index_col='id', con=conn)
df
```

특정 컬럼만 가져오기

```
df = read_sql_table('board_main_post', index_col='id', columns=['title', 'contents'],
df
```

3) 데이터 내보내기

- name='테이블명' 이름으로 기존 테이블이 있으면 해당 테이블의 컬럼명에 맞게 데이터를 넣을 수 있음
- if_exists='append' 옵션이 있으면, 기존 테이블에 데이터를 추가로 넣음
- if_exists='fail' 옵션이 있으면, 기존 테이블이 있을 경우, 아무일도 하지 않음
- if_exists='replace' 옵션이 있으면, 기존 테이블이 있을 경우, 기존 테이블을 삭제하고, 다시 테이블을 만들어서, 새로 데이터를 넣음

이미 만들어진 테이블이 없으면, name='테이블명' 이름으로 테이블을 자동으로 만들고, 데이터를 넣을 수 있음

테이블이 자동으로 만들어지므로, 테이블 구조가 최적화되지 않아 자동으로 테이블 만드는 것은 추천하지 않음

```
df.to_sql(name='new_table', con=conn, if_exists='append', index=False)
conn.commit()
```

4) 데이터베이스 접속 해제

```
conn.close()
```