sqlite 응용

야후 증권에서 삼성 주식 데이터를 DataFrame형태로 DB에 저장

web 데이터 읽기

```
# web 데이터 읽기

import pandas_datareader.data as web

import pandas as pd

import datetime

# 야후 증권
import yfinance

import sqlite3
```

추출할 시작 날짜 종료 날짜 설정

```
# 추출할 시작 날짜 종료 날짜 설정
start = datetime.datetime(2018, 1, 1)
end = datetime.datetime(2019, 1, 1)
```

야후 증권으로부터 삼성전자 주식 추출

```
samsung = web.get_data_yahoo("005930.KS", start, end)
```

```
# 상위 5개만 출력
sf = samsung.tail(5)
print(sf)

                High      Low     Open    Close      Volume     Adj Close
Date
2018-12-21  38650.0  38100.0  38200.0  38650.0  14947080.0  38293.230469
2018-12-24  39050.0  38300.0  38500.0  38800.0   9729530.0  38441.843750
```

```
2018-12-26  38750.0  38300.0  38400.0  38350.0  12707675.0  37996.000000
2018-12-27  38800.0  38100.0  38700.0  38250.0  10510643.0  38250.000000
2018-12-28  38900.0  38200.0  38250.0  38700.0   9900267.0  38700.000000
```

삼성전자 1년 데이터 데이터베이스에 저장

```
# 삼성전자 1년 데이터 데이터베이스에 저장
con = sqlite3.connect("C:/Users/USER/Desktop/YKS/Python2/kospi3.db")
```

dataframe은 DB에 저장 할 수 있는 함수를 제공 : to_sql()
Auto Commit이 기본

```
samsung.to_sql('samsung', con, chunksize=1000)
```

dataframe은 DB를 통해 조회할 수 있는 쿼리도 제공

```
readed_df = pd.read_sql("SELECT * FROM samsung", con, index_col = 'Date')
print(readed_df)

                        High      Low  ...      Volume     Adj Close
Date                                   ...
2018-01-03 00:00:00  52560.0  51420.0  ...  10013500.0  32073.728516
2018-01-04 00:00:00  52180.0  50640.0  ...  11695450.0  31738.205078
2018-01-05 00:00:00  52120.0  51200.0  ...   9481150.0  32384.400391
2018-01-08 00:00:00  52520.0  51500.0  ...   8383650.0  32322.263672
2018-01-09 00:00:00  51720.0  49980.0  ...  18013600.0  31315.691406
                     ...      ...  ...         ...           ...
2018-12-21 00:00:00  38650.0  38100.0  ...  14947080.0  38293.230469
2018-12-24 00:00:00  39050.0  38300.0  ...   9729530.0  38441.843750
2018-12-26 00:00:00  38750.0  38300.0  ...  12707675.0  37996.000000
2018-12-27 00:00:00  38800.0  38100.0  ...  10510643.0  38250.000000
2018-12-28 00:00:00  38900.0  38200.0  ...   9900267.0  38700.000000

[242 rows x 6 columns]
```

기본 설정값(to_sql)

```
DataFrame.to_sql(name,
                 con,
                 flavor = 'sqlite',
                 schema= None,
                 if_exists ='fail',
                 index = True,
                 index_label = None,
                 chunksize =None,
                 dtype=None)

name : SQL 테이블 이름으로 파이썬 문자열 형태로 나타낸다.
con : Cursor 객체

flavor : 사용한 DBMS를 지정할 수 있는데 'sqlite'또는 'mysql'을 사용할 수 있다.
        기본값은 sqlite

schema : Schema를 지정할 수 있는데 기본값은 None 이다.

if_exists : 데이터베이스에 테이블이 존재할 때 수행 동작을 지정한다.
            'fail', 'replace', 'append'중 하나를 사용할 수 있는데 기본값은 'fai
            'fail'은 데이터베이스에 테이블이 있다면 아무 동작도 수행하지 않는다.
            'repalce'는 테이블이 존재하면 기존 테이블을 삭제하고
            새로 테이블을 생성한 후 데이터를 삽입한다.

            'append'는 테이블이 존재하면 데이터만을 추가한다.

index : DataFrame의 index를 데이터베이스에 칼럼으로 추가할지에 대한 여부를 지정한다
        기본값은 True이다.

index_label : 인덱스 칼럼에 대한 라벨을 지정할 수 있다. 기본값은 None 이다.

chunksize : 한 번에 저장되는 로우 크기를 정수값으로 지정할 수 있다.
            기본값은 None으로 DataFrame 내의 모든 로우가 한번에 저장된다.

dtype : 칼럼에 대한 SQL 타입을 파이썬 딕셔너리로 넘겨줄 수 있다.
```