```
import sqlite3
import pandas as pd
db_name = './database/my_books.db'
```

```
def getBooksDF(books) :
    ret_df = pd.DataFrame()

    title = list()
    published_date = list()
    publisher = list()
    pages = list()
    recommendation = list()

    column_name = ['title','published_date','publisher', 'pages', 'recommendation']
    for book in books :
        title.append(book[0])
        published_date.append(book[1])
        publisher.append(book[2])
        pages.append(book[3])
        recommendation.append(book[4])

    data = {
        'title' : title,
        'published_date' : published_date,
        'publisher'      : publisher     ,
        'pages'          : pages         ,
        'recommendation' : recommendation
    }

    ret_df = pd.DataFrame(data, columns = column_name)
    return ret_df
```

```
def select_all_books(db_name):
    """
    전체 데이터를 조회하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try :
        # 데이터베이스 커넥션 생성
        conn = sqlite3.connect(db_name)
        # 커서 확보
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books"
        cur.execute(db_sql)

        # 조회한 데이터 불러오기
        print('[1] 전체 데이터 출력하기')
        books = cur.fetchall()

        ret_df = getBooksDF(books)

    except :
        is_success = False
        print('Database Error')
    finally :
        conn.close()

    return is_success, ret_df
```

```
is_success, books_df = select_all_books(db_name)
if is_success:
    print('조회된 데이터는 총 %d 건 입니다.'%len(books_df))
else :
    print('데이터를 조회하지 못했습니다')

books_df
```

[1] 전체 데이터 출력하기
조회된 데이터는 총 5 건 입니다.

	title	published_date	publisher	pages	recommendation
0	메가트랜드	2002.03.02	A	200	0
1	인더스트리 4.0	2016.07.09	B	584	1
2	유니콘 스타트업	2011.07.15	A	248	1
3	빅데이터 마케팅	2012.08.25	A	296	1
4	사물인터넷 전망	2013.08.22	B	526	0

```
# 일부 조회용 함수
def select_some_books(db_name, number):
    """
    일부 데이터를 조회하는 함수
    Args:
        db_name : Database Name
        number  : Count of data to query
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = sqlite3.connect(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books"
        cur.execute(db_sql)

        # 조회한 데이터 일부 불러오기
        print('[2] 데이터 일부 출력하기')
        books = cur.fetchmany(number)

        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally :
        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success, ret_df
```

```
is_success, books_df = select_some_books(db_name, number=3)
if is_success:
    print('조회된 데이터는 총 %d 건 입니다.'%len(books_df))
else :
```

```
    print('데이터를 조회하지 못했습니다')

books_df
```

[2] 데이터 일부 출력하기
조회된 데이터는 총 3 건 입니다.

	title	published_date	publisher	pages	recommendation
0	메가트랜드	2002.03.02	A	200	0
1	인더스트리 4.0	2016.07.09	B	584	1
2	유니콘 스타트업	2011.07.15	A	248	1

```
# 1개 조회용 함수
def select_one_book(db_name):
    """
    최상단 하나의 데이터를 조회하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = sqlite3.connect(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books "
        cur.execute(db_sql)

        # 데이터 한개 출력하기
        print('[3] 1개 데이터 출력하기')
        # print(cur.fetchone())
        book = cur.fetchone()
        books = [book]
        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally :
        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success, ret_df
```

```
is_success, books_df = select_one_book(db_name)
if is_success:
    print('하나의 데이터를 성공적으로 조회하였습니다.')
else :
    print('데이터를 조회하지 못했습니다')

books_df
```

[3] 1개 데이터 출력하기
하나의 데이터를 성공적으로 조회하였습니다.

	title	published_date	publisher	pages	recommendation
0	메가트랜드	2002.03.02	A	200	0

```
# 쪽수 많은 책 조회용 함수
```

```
def getBooksDF(books) :
    ret_df = pd.DataFrame()

    title = list()
    published_date = list()
    publisher = list()
    pages = list()
    recommendation = list()

    column_name = ['title','published_date','publisher', 'pages', 'recommendation']
    for book in books :
        title.append(book[0])
        published_date.append(book[1])
        publisher.append(book[2])
        pages.append(book[3])
        recommendation.append(book[4])

    data = {
        'title' : title,
        'published_date' : published_date,
        'publisher'      : publisher     ,
        'pages'          : pages         ,
        'recommendation' : recommendation
    }

    ret_df = pd.DataFrame(data, columns = column_name)
    return ret_df
```

```
# try문 밖에 있는 것들을 try 문 안의 것들이 참조할 수 있음
```

```
# 해당되는 책 제목으로 책정보를 조회하는 함수

def find_books_by_title(db_name, title):
    ret_df = pd.DataFrame()
    is_success = True

    try :
        conn = sqlite3.connect(db_name)

        cur = conn.cursor()

        db_sql = "SELECT * FROM my_books "    # WHERE과 붙지 않도록 띄어쓰기
        db_sql += "WHERE title LIKE '{}%'"
        cur.execute(db_sql.format(title))

        #SQL 실행하기: cursor.execute(SQL)


        print('[3] 해당되는 책 제목으로 정보 조회하기')
        # fetchall :: fetchALL :: 한 번에 모든 rows 가져오기
        books = cur.fetchall()
        ret_df = getBooksDF(books)

    except :
        is_success = False
        print('Database Error')
```

```
    finally :
        conn.close()

    return is_success, ret_df
```

```
#title = '빅데이터'
is_success, ret_df = find_books_by_title(db_name,title ='빅데이터')
if is_success:
    print('조건에 맞는 데이터는 총 %d 건 입니다.'%len(ret_df))
else :
    print('데이터를 조회하지 못했습니다')

ret_df
```

[3] 해당되는 책 제목으로 정보 조회하기
조건에 맞는 데이터는 총 1 건 입니다.

	title	published_date	publisher	pages	recommendation
0	빅데이터 마케팅	2012.08.25	A	296	1

데이터 업데이트

```
# UPDATE 대상DB SET 바꿀 칼럼=바꿀 값 WHERE 대상=''
# 제목이 ? 인 책의 추천 유무를 ? 로 변경하라
## "UPDATE my_books SET recommendation=? WHERE title=?"

def update_books(db_name):
    """
    데이터를 수정하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = sqlite3.connect(db_name)

        # 커서 확보
        cur = conn.cursor()

        db_sql = "UPDATE my_books SET recommendation=? WHERE title =?"

        cur.execute(db_sql, (1,'메가트랜드'))

    except:
        is_success = False
        print("Database Error!")

    finally :
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success
```

```
is_success, books_df1 = select_one_book(db_name)

if update_books(db_name):
    print('데이터가 성공적으로 수정되었습니다.')
else :
    print('데이터가 수정되지 않았습니다')

is_success, books_df2 = select_one_book(db_name)

books_df = pd.concat([books_df1, books_df2], axis=0)
books_df['update'] = ['수정전', '수정후']
books_df.set_index('update', inplace=True)
books_df
```

[3] 1개 데이터 출력하기
데이터가 성공적으로 수정되었습니다.
[3] 1개 데이터 출력하기

update	title	published_date	publisher	pages	recommendation
수정전	메가트랜드	2002.03.02	A	200	1
수정후	메가트랜드	2002.03.02	A	200	1

데이터 삭제

```
import sqlite3

db_name = './database/my_books.db'
```

```
# 데이터 삭제용 함수

def delete_books_by_title(db_name, title):
    """
    책제목에 해당하는 데이터를 삭제하는 함수
    Args:
        db_name : Database Name
        title   : Title of the book to be removed
    Returns :
        is_success : Boolean
    """
    is_success = True

    try :
        conn = sqlite3.connect(db_name)

        cur = conn.cursor()

        db_sql = "DELETE FROM my_books "
        db_sql += "WHERE title=?"

    except :
        is_success = False
        print("Database Error")

    finally :
        if is_success :
            #데이터베이스 반영
            conn.commit()
        else :
            conn.rollback()
```

```
        conn.close()
    return is_success
```

```
title = '메가트랜드'
if delete_books_by_title(db_name, title) :
    print('데이터가 성공적으로 삭제되었습니다.')
else :
    print('데이터가 삭제되지 않았습니다')

is_success, books_df = select_all_books(db_name)
books_df
```

데이터가 성공적으로 삭제되었습니다.
[1] 전체 데이터 출력하기

	title	published_date	publisher	pages	recommendation
0	메가트랜드	2002.03.02	A	200	1
1	인더스트리 4.0	2016.07.09	B	584	1
2	유니콘 스타트업	2011.07.15	A	248	1
3	빅데이터 마케팅	2012.08.25	A	296	1
4	사물인터넷 전망	2013.08.22	B	526	0

```
# 조건에 맞는 데이터 삭제
def delete_books(db_name, col_name, col_val):
    """
    조건에 맞는 데이터를 삭제하는 함수
    Args:
        db_name  : Database Name
        col_name : Column Name
        col_val  : Column Value
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = sqlite3.connect(db_name)

        # 커서 확보
        cur = conn.cursor()

        # # 데이터 삭제 SQL
        db_sql = "DELETE FROM my_books "
        db_sql += "WHERE {}=?"
        db_sql = db_sql.format(col_name)

        # 수정 SQL 실행
        cur.execute(db_sql, (col_val,))
    except :
        is_success = False
        print("Database Error!")

    finally :
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        conn.close()
```

```
    return is_success

is_success, books_df = select_all_books(db_name)
books_df
```

[1] 전체 데이터 출력하기

	title	published_date	publisher	pages	recommendation
0	메가트랜드	2002.03.02	A	200	1
1	인더스트리 4.0	2016.07.09	B	584	1
2	유니콘 스타트업	2011.07.15	A	248	1
3	빅데이터 마케팅	2012.08.25	A	296	1
4	사물인터넷 전망	2013.08.22	B	526	0

```
col_name = 'publisher'
col_val = "A"
if delete_books(db_name, col_name, col_val):
    print('데이터가 성공적으로 삭제되었습니다.')
else :
    print('데이터가 삭제되지 않았습니다')

is_success, books_df = select_all_books(db_name)
books_df
```

데이터가 성공적으로 삭제되었습니다.
[1] 전체 데이터 출력하기

	title	published_date	publisher	pages	recommendation
0	인더스트리 4.0	2016.07.09	B	584	1
1	사물인터넷 전망	2013.08.22	B	526	0

```
col_name = 'title'
col_val = '사물인터넷 전망'
if delete_books(db_name, col_name, col_val) :
    print('데이터 성공적 삭제')
else :
    print('데이터 삭제되지 않음')

is_success, books_df = select_all_books(db_name)
books_df
```

데이터 성공적 삭제
[1] 전체 데이터 출력하기

	title	published_date	publisher	pages	recommendation
0	인더스트리 4.0	2016.07.09	B	584	1