Chapter6
Data Loading, Storage, and File Formats

2021010702 진서영

목차

6.1
Reading and Writing Data in Text Format

XML and HTML: Web Scraping(p180~)

xml : eXtensible Markup Language
> 단순한 문자열을 넘어서서, 내부적으로 트리 구조를 가지고 있는 파일을 표현하기 위해 사용하는 마크업 언어
HTML : Hyper Text Markup Language
> 웹 페이지를 만드는 데 사용되는 언어

In [7]:
```
import pandas as pd
tables = pd.read_html('C:/Users/run07/Downloads/fdic_failed_bank_list.html')
len(tables)
```

Out[7]:
```
1
```

In [8]:
```
failures = tables[0]
failures.head()
```

Out[8]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
0	Allied Bank	Mulberry	AR	91	Today's Bank	September 23, 2016	November 17, 2016
1	The Woodbury Banking Company	Woodbury	GA	11297	United Bank	August 19, 2016	November 17, 2016
2	First CornerStone Bank	King of Prussia	PA	35312	First-Citizens Bank & Trust Company	May 6, 2016	September 6, 2016
3	Trust Company Bank	Memphis	TN	9956	The Bank of Fayette County	April 29, 2016	September 6, 2016
4	North Milwaukee State Bank	Milwaukee	WI	20364	First-Citizens Bank & Trust Company	March 11, 2016	June 16, 2016

In [9]:
```
close_timestamps = pd.to_datetime(failures['Closing Date'])
close_timestamps.dt.year.value_counts()
```

Out[9]:
```
2010    157
2009    140
2011     92
2012     51
2008     25
2013     24
2014     18
2002     11
2015      8
2016      5
2004      4
2001      4
2007      3
2003      3
2000      2
Name: Closing Date, dtype: int64
```

In [4]:
```
failures=tables[0]
close_timestamps=pd.to_datetime(failures['Closing Date'])
close_timestamps
```

Out[4]:
```
0     2016-09-23
1     2016-08-19
2     2016-05-06
3     2016-04-29
4     2016-03-11
         ...
542   2001-07-27
543   2001-05-03
544   2001-02-02
545   2000-12-14
546   2000-10-13
Name: Closing Date, Length: 547, dtype: datetime64[ns]
```

xml : eXtensible Markup Language
> 단순한 문자열을 넘어서서, 내부적으로 트리 구조를 가지고 있는 파일을 표현하기 위해 사용하는 마크업 언어
HTML : Hyper Text Markup Language
> 웹 페이지를 만드는 데 사용되는 언어

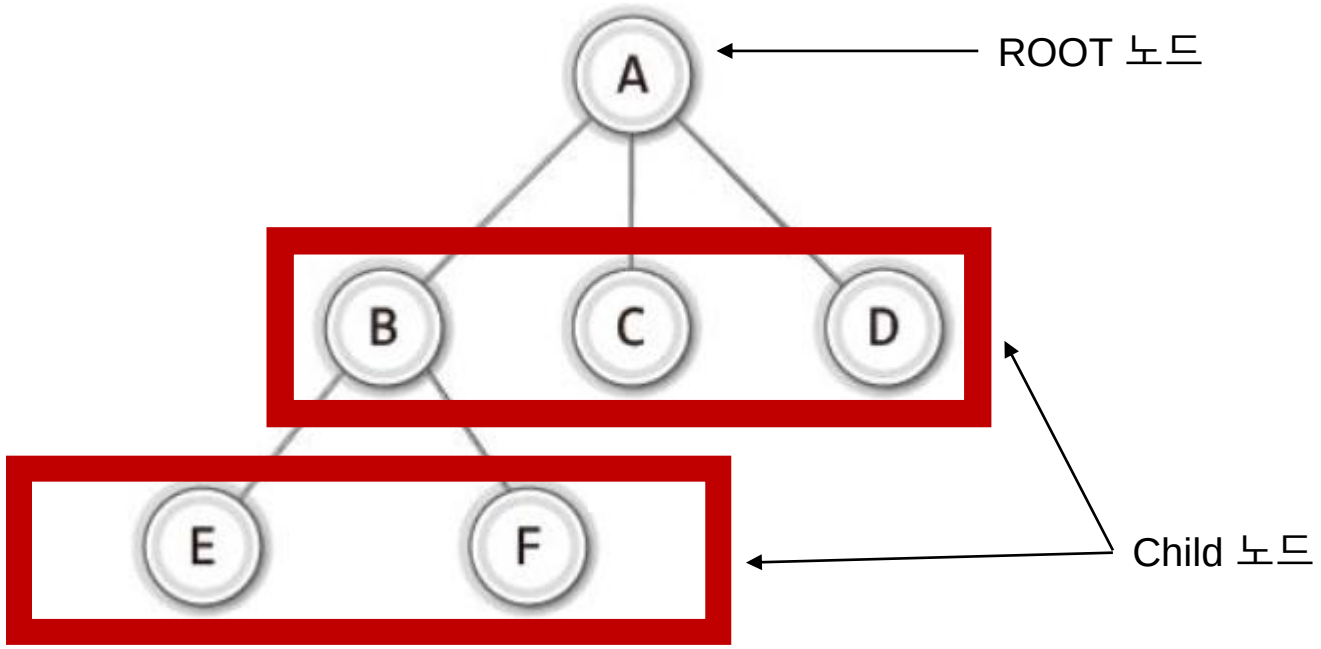
In [2]:
```
import pandas as pd
tables=pd.read_html('C:/Users/run07/Downloads/fdic_failed_bank_list.html')
tables
```

Out[2]:
```
[                          Bank Name             City  ST   CERT  \
0                     Allied Bank          Mulberry  AR     91
1         The Woodbury Banking Company        Woodbury  GA  11297
2              First CornerStone Bank  King of Prussia  PA  35312
3                  Trust Company Bank          Memphis  TN   9956
4           North Milwaukee State Bank        Milwaukee  WI  20364
..                                 ...              ...  ..    ...
542                Superior Bank, FSB         Hinsdale  IL  32646
543               Malta National Bank            Malta  OH   6629
544   First Alliance Bank & Trust Co.       Manchester  NH  34264
545  National State Bank of Metropolis       Metropolis  IL   3815
546                  Bank of Honolulu         Honolulu  HI  21029

                    Acquiring Institution       Closing Date  \
0                           Today's Bank  September 23, 2016
1                            United Bank     August 19, 2016
2    First-Citizens Bank & Trust Company        May 6, 2016
3            The Bank of Fayette County      April 29, 2016
4    First-Citizens Bank & Trust Company      March 11, 2016
..                                   ...                 ...
542                Superior Federal, FSB       July 27, 2001
543                    North Valley Bank         May 3, 2001
544  Southern New Hampshire Bank & Trust    February 2, 2001
545              Banterra Bank of Marion   December 14, 2000
546                  Bank of the Orient    October 13, 2000

          Updated Date
0    November 17, 2016
1    November 17, 2016
2    September 6, 2016
3    September 6, 2016
4        June 16, 2016
..                 ...
542    August 19, 2014
543  November 18, 2002
544  February 18, 2003
545     March 17, 2005
546     March 17, 2005

[547 rows x 7 columns]]
```

xml : eXtensible Markup Language
> 단순한 문자열을 넘어서서, 내부적으로 트리 구조를 가지고 있는 파일을 표현하기 위해 사용하는 마크업 언어

트리 구조: 비선형 자료구조의 대표적인 자료구조

* **비선형 자료구조**: 하나의 자료 뒤에 여러 개의 자료가 존재할 수 있는 것



lxml : XML을 빠르고 유연하게 처리하는 파이썬 라이브러리
objectify.parse : 파일을 구문 분석함
getroot() : root 노드 가져오기

In [4]:
```
from lxml import objectify
import pandas as pd
path = 'C:/Users/run07/Downloads/Performance_MNR.xml'
parsed = objectify.parse(open(path))
root = parsed.getroot()
```

In [5]:
```
data = []

skip_fields = ['PARENT_SEQ', 'INDICATOR_SEQ',
               'DESIRED_CHANGE', 'DECIMAL_PLACES']

for elt in root.INDICATOR:
    el_data = {}
    for child in elt.getchildren():
        if child.tag in skip_fields:
            continue
        el_data[child.tag] = child.pyval
    data.append(el_data)
```

In [6]:
```
perf = pd.DataFrame(data)
perf.head()
```

Out[6]:

	AGENCY_NAME	INDICATOR_NAME	DESCRIPTION	PERIOD_YEAR	PERIOD_MONTH	CATEGORY	FREQUENCY	INDICATOR_UNIT	YTD_TARGET	YTD_ACT
0	Metro-North Railroad	On-Time Performance (West of Hudson)	Percent of commuter trains that arrive at thei...	2008	1	Service Indicators	M	%	95	
1	Metro-North Railroad	On-Time Performance (West of Hudson)	Percent of commuter trains that arrive at thei...	2008	2	Service Indicators	M	%	95	
2	Metro-North Railroad	On-Time Performance (West of Hudson)	Percent of commuter trains that arrive at thei...	2008	3	Service Indicators	M	%	95	
3	Metro-North Railroad	On-Time Performance (West of Hudson)	Percent of commuter trains that arrive at thei...	2008	4	Service Indicators	M	%	95	
4	Metro-North Railroad	On-Time Performance (West of Hudson)	Percent of commuter trains that arrive at thei...	2008	5	Service Indicators	M	%	95	

In [7]:
```
from io import StringIO
tag = '<a href="http://www.google.com">Google</a>'
root = objectify.parse(StringIO(tag)).getroot()
```

In [8]:
```
root
root.get('href')
root.text
```

Out[8]:
```
'Google'
```

6.2
Binary Data Formats

Pickle : 파이썬의 모든 객체를 파일로 저장할 수 있는 방법
to_pickle : pandas 데이터프레임을 저장할 때 사용
read_pickle : pandas 데이터프레임을 불러올 때 사용

In [9]:
```
import pandas as pd
frame = pd.read_csv('C:/Users/run07/Downloads/ex1.csv')
frame
```

Out[9]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

In [10]:
```
frame.to_pickle('C:/Users/run07/Downloads/ex1.csv')
pd.read_pickle('C:/Users/run07/Downloads/ex1.csv')
```

Out[10]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

Using HDF5 Format

HDF5 : 많은 양의 과학적 배열 데이터를 저장하기 위한 잘 알려진 파일 형식
> 보다 단순한 포맷에 비해 다양한 압축 모드를 통해 즉각적인 압축을 지원해서 패턴이 반복되는 데이터를 효율적으로 저장할 수 있음
HDFStore : 받아쓰기처럼 작동하며 낮은 수준의 세부 정보를 처리함

In [9]:
```
import pandas as pd
import numpy as np
frame = pd.DataFrame({'a': np.random.randn(100)})
store = pd.HDFStore('mydata.h5')
store['obj1'] = frame
store['obj1_col'] = frame['a']
store
```

Out[9]:
```
<class 'pandas.io.pytables.HDFStore'>
File path: mydata.h5
```

In [10]:
```
store['obj1']
```

Out[10]:

	a
0	-0.294043
1	-1.536217
2	-0.260536
3	-0.756391
4	0.903389
...	...
95	0.327945
96	0.692768
97	1.082559
98	1.348893
99	1.044277

100 rows × 1 columns

In [11]:
```
store.put('obj2', frame, format='table')
store.select('obj2', where=['index >= 10 and index <= 15'])
store.close()
```

In [12]:
```
frame.to_hdf('mydata.h5', 'obj3', format='table')
pd.read_hdf('mydata.h5', 'obj3', where=['index < 5'])
```

Out[12]:

	a
0	-0.294043
1	-1.536217
2	-0.260536
3	-0.756391
4	0.903389

pd.read_excel(xlsx, ‘Sheet1’) : xlsx 파일에서 ‘Sheet1’을 출력

In [2]:
```
import pandas as pd
xlsx = pd.ExcelFile('C:/Users/run07/Downloads/ex1.xlsx')
```

In [3]:
```
pd.read_excel(xlsx, 'Sheet1')
```

Out[3]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

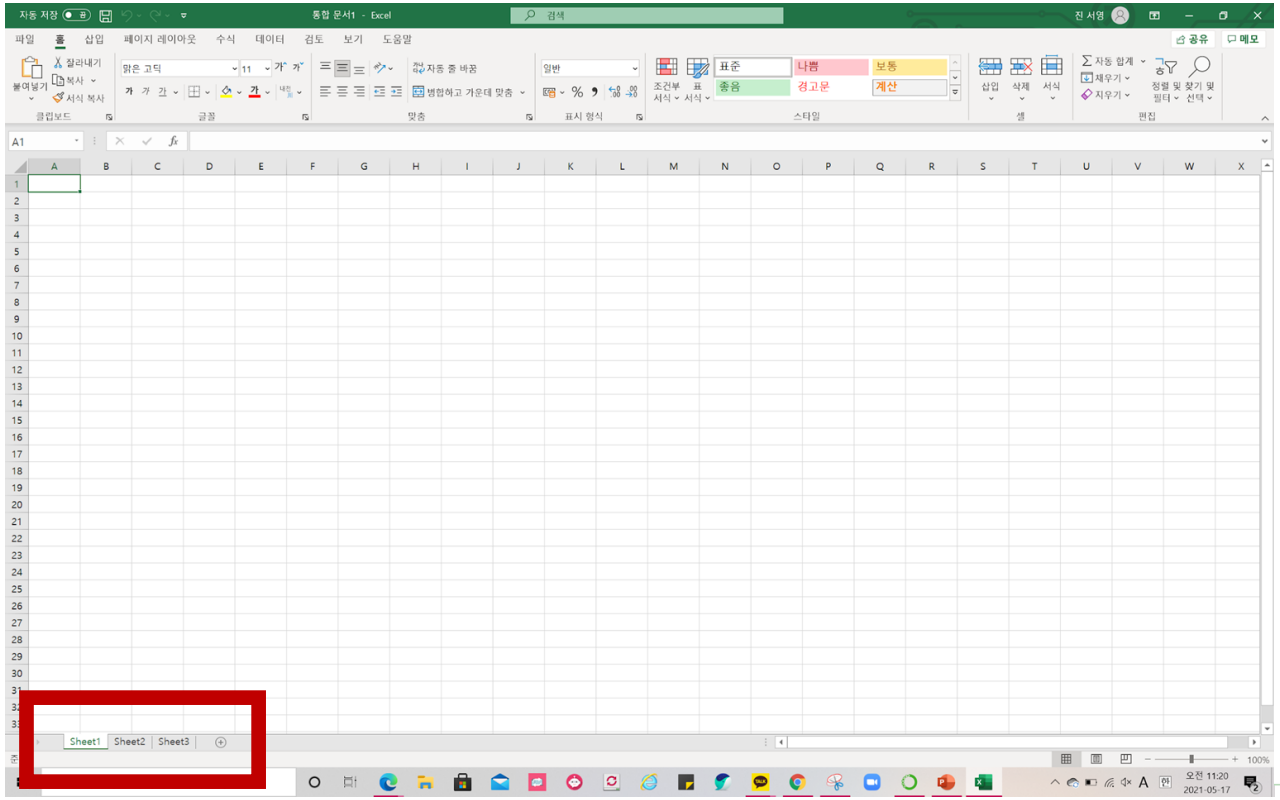
In [4]:
```
frame = pd.read_excel('C:/Users/run07/Downloads/ex1.xlsx', 'Sheet1')
frame
```

Out[4]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

Reading Microsoft Excel Files

ExcelFiles : 사용하려면 xls 또는 xlsx에 파일 경로를 전달해서 객체 만듦
to_excel : 판다스 데이터프레임을 엑셀 형태로 저장
ExcelWriter : 엑셀 파일에 데이터를 저장할 때 여러 개의 시트를 만들 수 있는데, DataFrame이 여러 개 이거나 또는 같은 내용을 여러 시트에 작성하고 싶은 경우 사용



In [4]:
```
import pandas as pd
xlsx = pd.ExcelFile('C:/Users/run07/Downloads/ex1.xlsx')
```

In [5]:
```
pd.read_excel(xlsx, 'Sheet1')
```

Out[5]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

In [6]:
```
frame = pd.read_excel('C:/Users/run07/Downloads/ex1.xlsx', 'Sheet1')
frame
```

Out[6]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

In [11]:
```
writer = pd.ExcelWriter('C:/Users/run07/Downloads/ex2.xlsx')
frame.to_excel(writer, 'Sheet1')
writer.save()
```

In [12]:
```
frame.to_excel('C:/Users/run07/Downloads/ex2.xlsx')
```

6.3
Interacting With Web APIs

requests : 파이썬 에서 HTTP와 관련된 작업, 즉 웹페이지에 원하는 작업 요청을 할 수 있는 라이브러리
> 웹페이지에서 우리가 원하는 데이터를 얻을 수 있도록, 해당 데이터의 주소인 HTML 코드를 불러오는 역할
get() : () 안에 들어갈 웹페이지의 HTML 구성요소들을 얻어주는 함수

```
In [6]: import requests
        url = 'https://api.github.com/repos/pandas-dev/pandas/issues'
        resp = requests.get(url)
        resp

Out[6]: <Response [200]>

In [7]: data = resp.json()
        data[0]['title']

Out[7]: 'Pandas Timedelta gives different result than numpy timedelta64'
```

requests : 파이썬 에서 HTTP와 관련된 작업, 즉 웹페이지에 원하는 작업 요청을 할 수 있는 라이브러리
> 웹페이지에서 우리가 원하는 데이터를 얻을 수 있도록, 해당 데이터의 주소인 HTML 코드를 불러오는 역할

get() : () 안에 들어갈 웹페이지의 HTML 구성요소들을 얻어주는 함수

```
In [9]: issues = pd.DataFrame(data, columns=['number', 'title',
                                             'labels', 'state'])
        issues
```

Out[9]:

	number	title	labels	state
0	41515	Pandas Timedelta gives different result than n...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
1	41514	DOC: Add clearer info when copy is False but m...	[{'id': 1465286368, 'node_id': 'MDU6TGFiZWwxND...	open
2	41513	TYP: ExtensionArray delete() and searchsorted()	[]	open
3	41512	ENH: add `styler` option context for sparsific...	[]	open
4	41511	deprecate non-keywordargs in clip	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
5	41510	deprecate nonkeywordargs in interpolate	[]	open
6	41509	deprecate nonkeyword args for bfill	[]	open
7	41508	deprecate nonkeywordargs in ffill	[]	open
8	41506	deprecate passing args as positional in sort_i...	[]	open
9	41505	deprecate passing args as positional in sort_v...	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
10	41504	deprecate passing args as positional in dropna	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
11	41503	PERF: load plotting entrypoint only when neces...	[]	open
12	41502	[ArrowStringArray] REF: dedup df creation in s...	[{'id': 127681, 'node_id': 'MDU6TGFiZWwxMjc2OD...	open
13	41501	TST/CLN: parameterize/dedup replace test2	[{'id': 211029535, 'node_id': 'MDU6TGFiZWwyMTE...	open
14	41500	Deprecate non-keyword arguments for drop_dupli...	[]	open
15	41499	TST/CLN: parameterize/dedup replace test	[{'id': 211029535, 'node_id': 'MDU6TGFiZWwyMTE...	open
16	41498	ENH: groupby rank supports object dtype	[{'id': 134699, 'node_id': 'MDU6TGFiZWwxMzQ2OT...	open
17	41497	BUG: columns name retention in groupby methods	[]	open
18	41496	deprecate default arguments as positional in r...	[]	open
19	41495	deprecate default args as positional in set_index	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
20	41494	TYP simplify overloads for fillna	[{'id': 1280988427, 'node_id': 'MDU6TGFiZWwxMj...	open
21	41493	TST: Add tests for old issues 2	[{'id': 127685, 'node_id': 'MDU6TGFiZWwxMjc2OD...	open
22	41492	ENH: load plotting entry point only when neces...	[{'id': 76865106, 'node_id': 'MDU6TGFiZWw3Njg2...	open
23	41491	Deprecate nonkeyword args set axis	[]	open
24	41490	[ArrowStringArray] PERF: str.extract object fa...	[{'id': 8935311, 'node_id': 'MDU6TGFiZWw4OTM1M...	open
25	41488	BUG: CustomBusinessMonthBegin(End) sometimes i...	[{'id': 53181044, 'node_id': 'MDU6TGFiZWw1MzE4...	open
26	41487	[ArrowStringArray] REF: _str_startswith/_str_e...	[{'id': 127681, 'node_id': 'MDU6TGFiZWwxMjc2OD...	open
27	41486	deprecate non-keyword arguments in drop	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
28	41485	Deprecate non-keyword arguments for methods wi...	[{'id': 87485152, 'node_id': 'MDU6TGFiZWw4NzQ4...	open
29	41484	TST/CLN: tests/strings/test_strings.py	[{'id': 211029535, 'node_id': 'MDU6TGFiZWwyMTE...	open

6.4
Interacting With Databases

sqlite3 : 파이썬 표준 라이브러리(파이썬이 설치될 때 기본적으로 설치되는 모듈)로 SQLite에 대한 인터페이스를 제공함
> 표준 라이브러리인 sqlite3 모듈을 이용하면 따로 모듈을 설치할 필요 없이 데이터베이스를 쉽게 이용가능

sqlite3.connect() : 새로운 데이터베이스 파일을 만들거나, 기존의 데이터베이스와 연결할 때 사용
con.execute(): 데이터 조회
con.executemany() : 여러 개의 데이터를 저장 할 경우 사용
con.commit() : 데이터 저장
.fetchall() : 한 번에 모든 열(rows)를 읽기 위해 사용

In [1]:
```
import sqlite3
query = """
CREATE TABLE test
(a VARCHAR(20), b VARCHAR(20),
 c REAL,        d INTEGER
);"""
con = sqlite3.connect('mydata.sqlite')
con.execute(query)
con.commit()
```

In [2]:
```
data = [('Atlanta', 'Georgia', 1.25, 6),
        ('Tallahassee', 'Florida', 2.6, 3),
        ('Sacramento', 'California', 1.7, 5)]
stmt = "INSERT INTO test VALUES(?, ?, ?, ?)"
con.executemany(stmt, data)
con.commit()
```

In [3]:
```
cursor = con.execute('select * from test')
rows = cursor.fetchall()
rows
```

Out[3]:
```
[('Atlanta', 'Georgia', 1.25, 6),
 ('Tallahassee', 'Florida', 2.6, 3),
 ('Sacramento', 'California', 1.7, 5)]
```

cusor.description : cusor에 각각의 칼럼에 대한 정보를 따로 담음
sqlalchemy : 사용해서 동일한 sqlite 데이터베이스에 연결하고 이전에 생성된 테이블에서 데이터를 읽음

In [7]:
```
import pandas as pd
cursor.description
```

Out[7]:
```
(('a', None, None, None, None, None, None),
 ('b', None, None, None, None, None, None),
 ('c', None, None, None, None, None, None),
 ('d', None, None, None, None, None, None))
```

In [8]:
```
pd.DataFrame(rows, columns=[x[0] for x in cursor.description])
```

Out[8]:

	a	b	c	d
0	Atlanta	Georgia	1.25	6
1	Tallahassee	Florida	2.60	3
2	Sacramento	California	1.70	5

In [9]:
```
import sqlalchemy as sqla
db = sqla.create_engine('sqlite:///mydata.sqlite')
pd.read_sql('select * from test', db)
```

Out[9]:

	a	b	c	d
0	Atlanta	Georgia	1.25	6
1	Tallahassee	Florida	2.60	3
2	Sacramento	California	1.70	5

6.5
Conclusion

Conclusion

CH6: 데이터에 액세스 할 때 유용한 도구