Chapter 5

How to get the data

Objectives

Applied

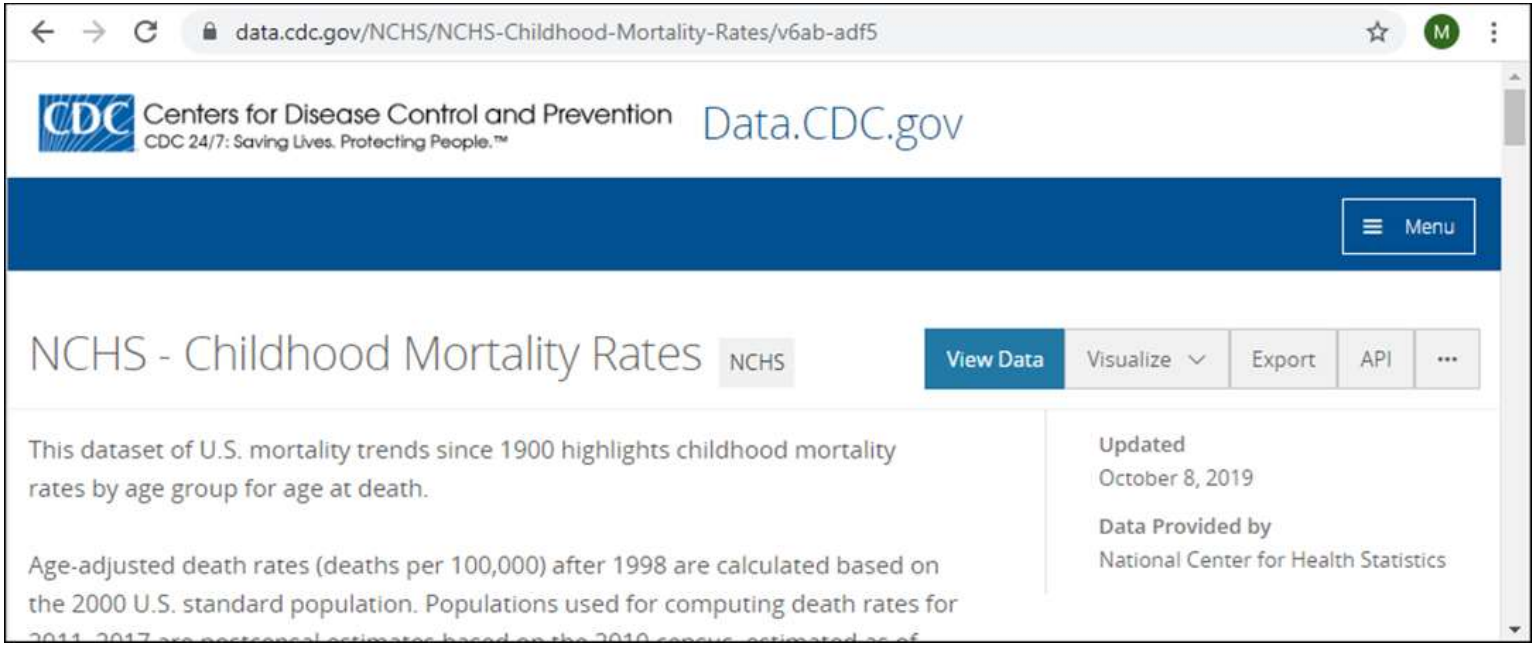
1. Use the Pandas read methods to import data into a DataFrame.
2. Download a file to disk before importing it into a DataFrame.
3. Unzip a zip file to access the files that it contains.
4. Use a SQL query to import database data into a DataFrame.
5. Use the metadata of a Stata file to analyze the data, and then read selected columns of the Stata data into a DataFrame.
6. Use JupyterLab to drill down into the data in a JSON file that has more than two levels of data, convert the JSON file to a dictionary, and then build a DataFrame from portions of the data in the dictionary.

Objectives (continued)

Knowledge

1. In general terms, describe the way that you get data in these formats into a DataFrame: CSV, Excel, database, Stata, and JSON.

The download page for the Childhood Mortality data


data.cdc.gov/NCHS/NCHS-Childhood-Mortality-Rates/v6ab-adf5
CDC
Centers for Disease Control and Prevention
CDC 24/7: Saving Lives. Protecting People.™
Data.CDC.gov
Menu
View Data
Visualize
Export
API
...

NCHS - Childhood Mortality Rates NCHS

This dataset of U.S. mortality trends since 1900 highlights childhood mortality rates by age group for age at death.

Age-adjusted death rates (deaths per 100,000) after 1998 are calculated based on the 2000 U.S. standard population. Populations used for computing death rates for 2011–2017 are postcensal estimates based on the 2010 census, estimated as of

Updated
October 8, 2019

Data Provided by
National Center for Health Statistics

Common sources of data

Internal datasets and databases

This can include everything from departmental spreadsheets to any of the databases used by a corporation.

Third-party websites

This includes the hundreds of websites that let you download data for your own analysis.

Good places to look for external datasets

Google Dataset Search: https://toolbox.google.com/datasearch

Kaggle: https://www.kaggle.com/datasets

Registry of Open Data on AWS: https://registry.opendata.aws

The websites for the case studies in this book

- **Child Mortality**: Centers for Disease Control (data.cdc.gov)
- **2016 Polling**: FiveThirtyEight (fivethirtyeight.com)
- **Forest Fires**: U.S. Forest Service (fs.fed.us)
- **Social Survey**: General Social Survey (gss.norc.org)
- **Sports Analytics**: NBA stats (stats.nba.com)

Some file formats for data found on websites

Type	Extension	Description	Contents
CSV	`.csv`	Comma-separated values	One table
TSV	`.tsv`	Tab-separated values	One table
Excel	`.xlsx`, `.xls`	Excel spreadsheet	One or more sheets
Stata	`.dta`	Stata statistical package	Complex data
JSON	`.json`	JavaScript Object Notation	Nested data
XML	`.xml`	Extensible Markup Language	Nested data
SAS	`.sd7`, `.sd6`	SAS statistical package	Complex data
SPSS	`.sav`	SPSS statistical package	Complex data
HDF5	`.h5`	Structured format for large datasets	Complex data
Zip	`.zip`	Archive format	One or more files

Some of the Pandas methods for importing data into a DataFrame

Method	Data format
`read_csv(file)`	CSV
`read_excel(file)`	Excel
`read_stata(file,columns)`	Stata
`read_json(file)`	JSON
`read_hdf(file,columns)`	HDF5
`read_sas(file)`	SAS
`read_sql_query(query,con)`	Database

How to import a CSV file from a website

```
mortality_url = \
    "https://data.cdc.gov/.../rows.csv?accessType=DOWNLOAD"
mortality_data = pd.read_csv(mortality_url)
```

How to import the first sheet of a downloaded Excel file

```
jobs = pd.read_excel("oesm18all/all_data_M_2018.xlsx")
```

The urlretrieve() method of the urllib.request module

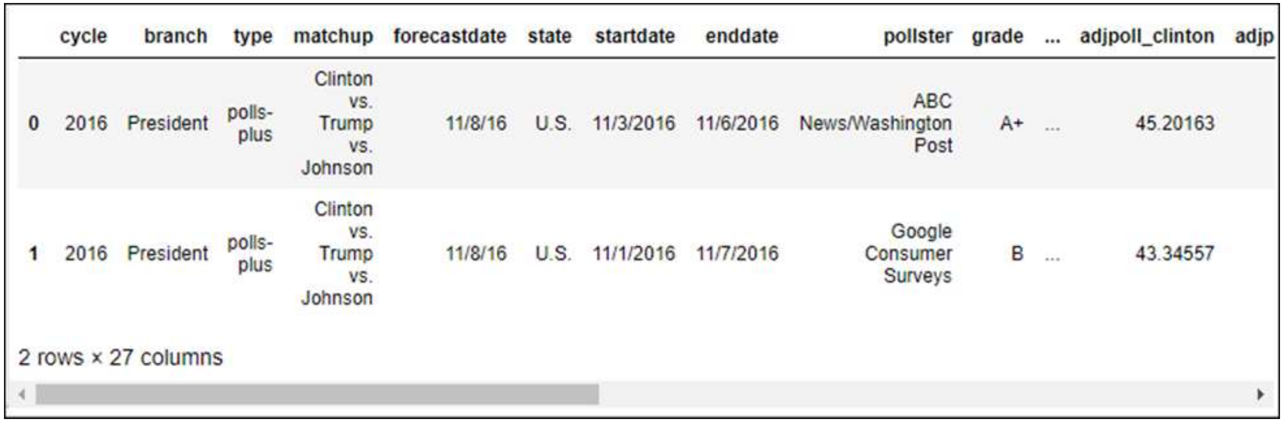
```
urlretrieve(url,filename)
```

How to download a file to disk

```
from urllib import request
polls_url = \
    'http://projects.fivethirtyeight.com/.../president_general_polls_2016.csv'
request.urlretrieve(polls_url, filename='president_polls_2016.csv')
```

How to import the file into a DataFrame

```
polls = pd.read_csv('president_polls_2016.csv')
polls.head(2)
```



	cycle	branch	type	matchup	forecastdate	state	startdate	enddate	pollster	grade	...	adjpoll_clinton	adjp
0	2016	President	polls-plus	Clinton vs. Trump vs. Johnson	11/8/16	U.S.	11/3/2016	11/6/2016	ABC News/Washington Post	A+	...	45.20163	
1	2016	President	polls-plus	Clinton vs. Trump vs. Johnson	11/8/16	U.S.	11/1/2016	11/7/2016	Google Consumer Surveys	B	...	43.34557	

2 rows × 27 columns

Two methods of the ZipFile class

Method	Description
`extractall()`	Extracts all of the files from the zip file and saves them in the default directory.
`infolist()`	Reads the zip file and returns the file information as a list.

How to download a zip file to disk

```
from urllib import request
zip_url = 'https://www.bls.gov/oes/special.requests/oesm18all.zip'
request.urlretrieve(zip_url, filename='oesm18all.zip')
```

How to extract the files and list their names

```
from zipfile import ZipFile
file_names = list()
with ZipFile('oesm18all.zip', mode='r') as zip:
    zip.extractall()
    for file in zip.infolist():
        file_names.append(file.filename)
        print(file.filename, file.compress_size, file.file_size)
=================================================================
oesm18all/all_data_M_2018.xlsx 70296790 71834374
```

Two ways to read an extracted file into a DataFrame

By specifying the filename

```
jobs = pd.read_excel("oesm18all/all_data_M_2018.xlsx")
```

By specifying the position of the file in the file_names list

```
jobs = pd.read_excel(file_names[0])
```

Packages for connecting to databases in Python

Package name	Database	Availability
`sqlite3`	SQLite	Built-in
`pymysql`	MySQL	Needs to be installed
`psycopg2`	PostgreSQL	Needs to be installed
`cx_oracle`	Oracle	Needs to be installed
`pymssql`	SQL Server	Needs to be installed

The connect() method of the SQLite module

```
connect(path)
```

The cursor() method of a connection object

```
cursor()
```

Two methods of a cursor object

```
execute(sql)
fetchall()
```

How to run queries on a SQLite database

Create a connection object and a cursor object

```
import sqlite3
fires_con = sqlite3.connect('Data/FPA_FOD_20170508.sqlite')
fires_cur = fires_con.cursor()
```

Run a query that lists the tables in the database

```
fires_cur.execute(
    'SELECT name FROM sqlite_master WHERE type="table"').fetchall()
===================================================================
[('spatial_ref_sys',),
 ('spatialite_history',),
 ...
 ('Fires',),
 ...
 ('NWCG_UnitIDActive_20170109',)]
```

How to get information about a table

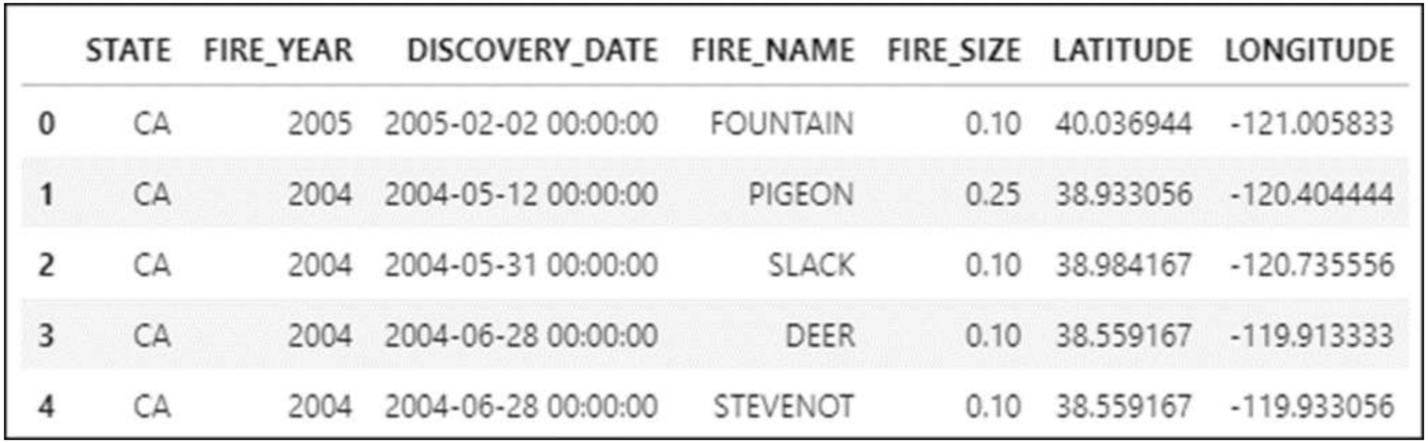
```
fires_cur.execute('PRAGMA table_info(Fires)').fetchall()
========================================================
[(0, 'OBJECTID', 'integer', 1, None, 1),
 (1, 'FOD_ID', 'int32', 0, None, 0),
 (2, 'FPA_ID', 'text(100)', 0, None, 0),
 ...
 (12, 'FIRE_CODE', 'text(10)', 0, None, 0),
 (13, 'FIRE_NAME', 'text(255)', 0, None, 0),
 ...
 (19, 'FIRE_YEAR', 'int16', 0, None, 0),
 (20, 'DISCOVERY_DATE', 'realdate', 0, None, 0),
 ...
 (28, 'FIRE_SIZE', 'float64', 0, None, 0),
 (29, 'FIRE_SIZE_CLASS', 'text(1)', 0, None, 0),
 (30, 'LATITUDE', 'float64', 0, None, 0),
 (31, 'LONGITUDE', 'float64', 0, None, 0),
 ...
 (34, 'STATE', 'text(255)', 0, None, 0),
 ...
 (38, 'Shape', 'POINT', 1, None, 0)]
```

The read_sql_query() method

```
read_sql_query(SQL,connection)
```

How to import the data from a query into a DataFrame

```
fires = pd.read_sql_query(
    '''SELECT STATE, FIRE_YEAR,
           DATETIME(DISCOVERY_DATE) AS DISCOVERY_DATE,
           FIRE_NAME, FIRE_SIZE, LATITUDE, LONGITUDE
      FROM Fires''', fires_con)
```



	STATE	FIRE_YEAR	DISCOVERY_DATE	FIRE_NAME	FIRE_SIZE	LATITUDE	LONGITUDE
0	CA	2005	2005-02-02 00:00:00	FOUNTAIN	0.10	40.036944	-121.005833
1	CA	2004	2004-05-12 00:00:00	PIGEON	0.25	38.933056	-120.404444
2	CA	2004	2004-05-31 00:00:00	SLACK	0.10	38.984167	-120.735556
3	CA	2004	2004-06-28 00:00:00	DEER	0.10	38.559167	-119.913333
4	CA	2004	2004-06-28 00:00:00	STEVENOT	0.10	38.559167	-119.933056

The read_dta() method of the pyreadstat package

```
read_dta(filename,columns,metadataonly)
```

Some of the attributes for a metadata container

```
column_names
column_labels
number_columns
number_rows
```

How to install the pyreadstat module with a conda command

```
conda install --channel conda-forge pyreadstat --yes
```

How to get the metadata from a Stata file on disk

```
import pyreadstat
gss_stata_filename = 'GSS7218_R3.DTA'

gss_empty, gss_meta = pyreadstat.read_dta(
    gss_stata_filename,
    metadataonly=True)
type(gss_meta)
===============================================
<pyreadstat._readstat_parser.metadata_container
```

What the attributes of the metadata container can tell you

```
print("Number of columns:", gss_meta.number_columns)
print("Number of rows:", gss_meta.number_rows)
print("Column names:", gss_meta.column_names)
==============================================================
Number of columns: 6110
Number of rows: 64814
Column names: ['year', 'id', 'wrkstat', 'hrs1', 'hrs2',
'evwork', 'occ', 'prestige', 'wrkslf', 'wrkgovt', 'commute',
'industry', 'occ80', 'prestg80', 'indus80', 'indus07',
'occonet', 'found', 'occ10', 'occindv', 'occstatus', 'occtag',
'prestg10', 'prestg105plus', 'indus10', 'indstatus', 'indtag',
...
```

The DataFrame() constructor for a Stata file

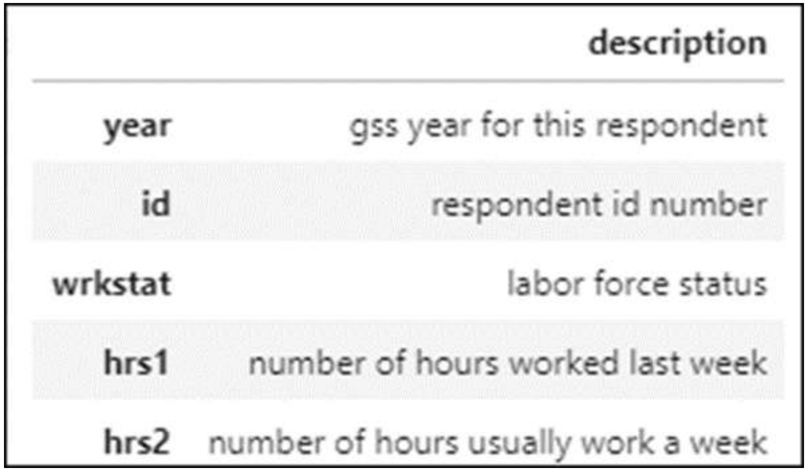
Method	Description
`DataFrame(params)`	Constructs a DataFrame.

Parameters of the DataFrame constructor

Parameter	Description
`data`	Can be an array, dictionary, or other object that’s shaped like a table.
`columns`	Column labels. If they aren’t specified, they will be generated.
`index`	Row labels. If they aren’t specified, they will be generated.

How to build a DataFrame for the column descriptions in the metadata

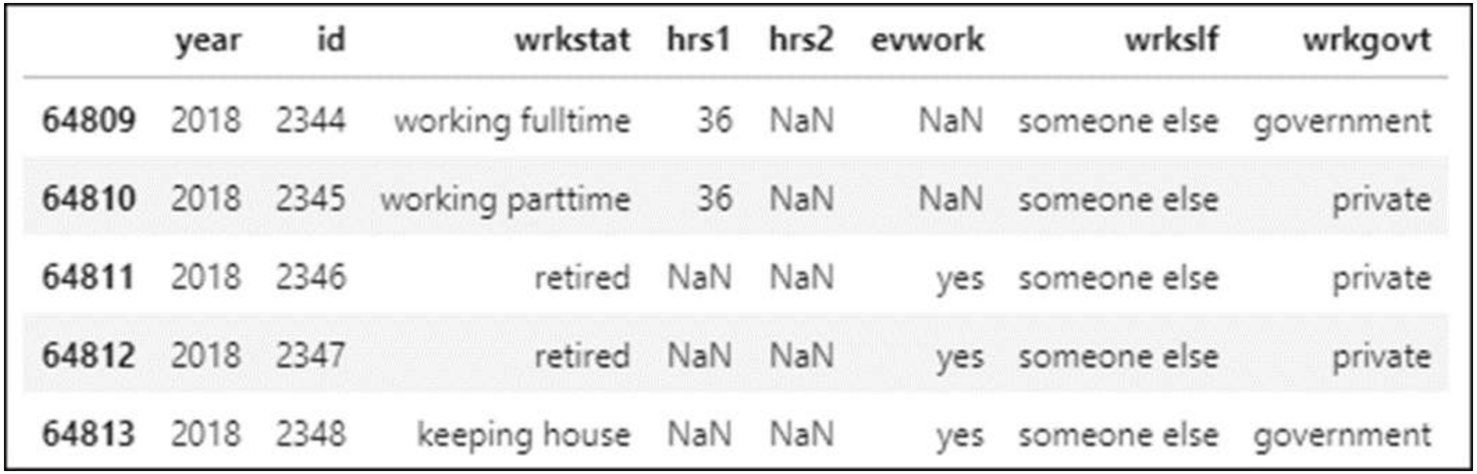
```
import pandas as pd
meta_cols=pd.DataFrame(
    data=gss_meta.column_labels,
    index=gss_meta.column_names,
    columns=['description'])
meta_cols.head(5)
```



	description
year	gss year for this respondent
id	respondent id number
wrkstat	labor force status
hrs1	number of hours worked last week
hrs2	number of hours usually work a week

How to import seven columns of the data into a DataFrame

```
gss_data = pd.read_stata('GSS7218_R3.DTA',
    columns=['year','id','wrkstat','hrs1','hrs2','evwork','wrkslf','wrkgovt'])
gss_data.tail()
```

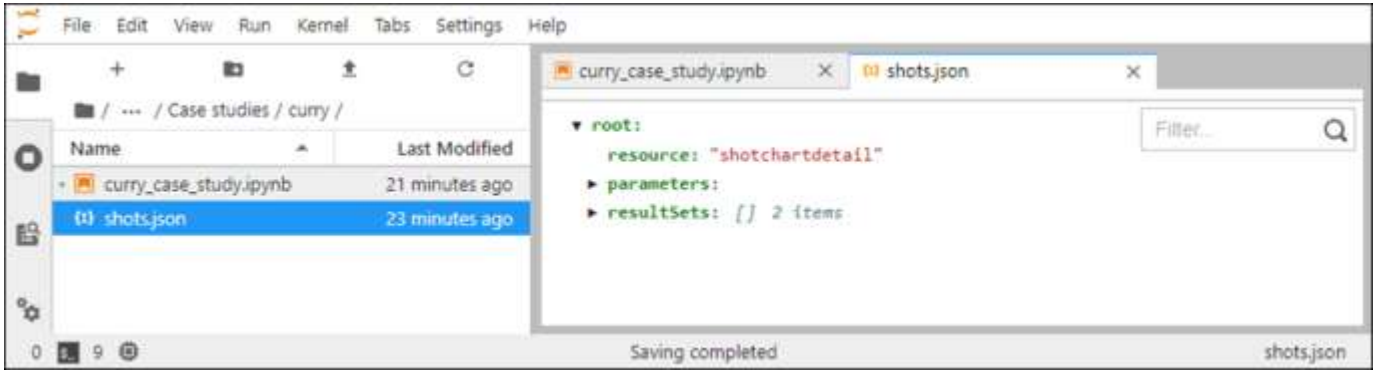


	year	id	wrkstat	hrs1	hrs2	evwork	wrkslf	wrkgovt
64809	2018	2344	working fulltime	36	NaN	NaN	someone else	government
64810	2018	2345	working parttime	36	NaN	NaN	someone else	private
64811	2018	2346	retired	NaN	NaN	yes	someone else	private
64812	2018	2347	retired	NaN	NaN	yes	someone else	private
64813	2018	2348	keeping house	NaN	NaN	yes	someone else	government

How to download a JSON file to disk

```
import json
from urllib import request
shots_url = \
    'https://www.murach.com/python_analysis/shots.json'
request.urlretrieve(shots_url, filename='shots.json')
```

Double-click on the JSON file in JupyterLab to open it



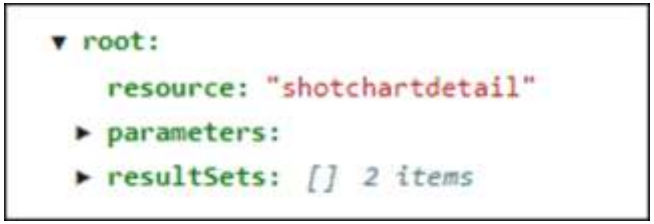
/ ··· / Case studies / curry /

Name	Last Modified
curry_case_study.ipynb	21 minutes ago
shots.json	23 minutes ago

```
▾ root:
    resource: "shotchartdetail"
  ▸ parameters:
  ▸ resultSets: [] 2 items
```

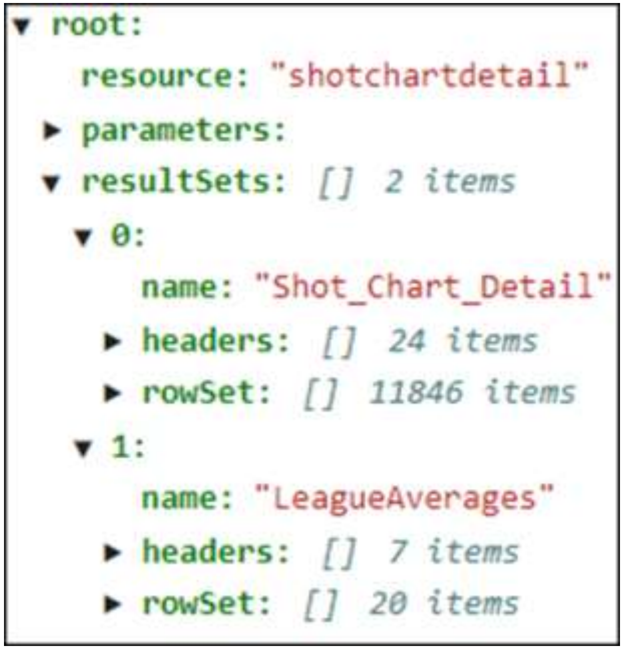
0 9

The root level of the JSON file



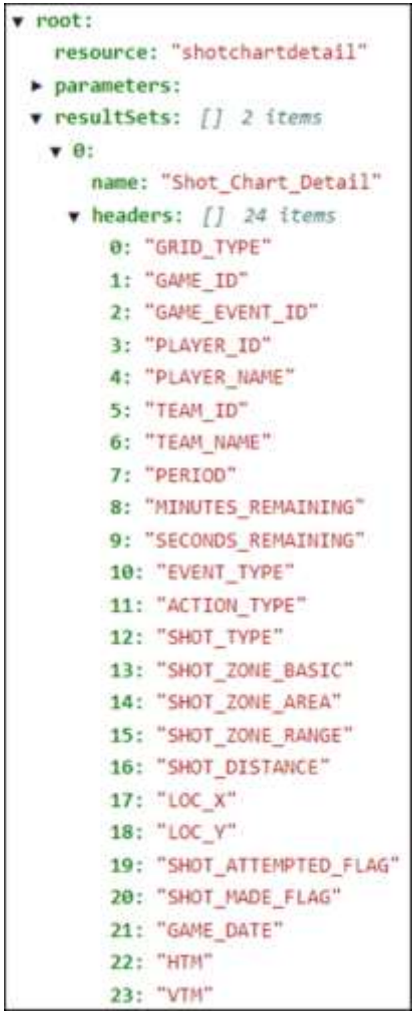
```
▾ root:
    resource: "shotchartdetail"
  ▸ parameters:
  ▸ resultSets: [] 2 items
```

The first two levels of the resultSets data



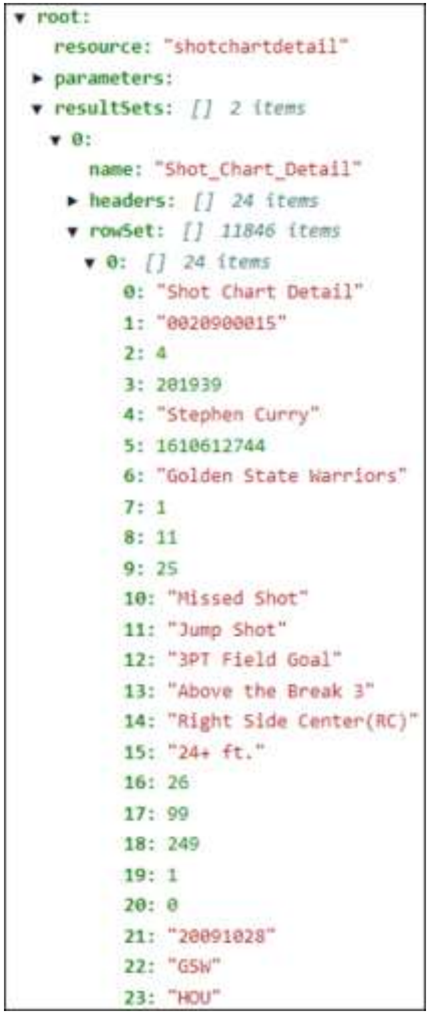
```
▼ root:
    resource: "shotchartdetail"
  ▶ parameters:
  ▼ resultSets: [] 2 items
    ▼ 0:
        name: "Shot_Chart_Detail"
      ▶ headers: [] 24 items
      ▶ rowSet: [] 11846 items
    ▼ 1:
        name: "LeagueAverages"
      ▶ headers: [] 7 items
      ▶ rowSet: [] 20 items
```

The headers level of the resultSets[0] data



```
▼ root:
    resource: "shotchartdetail"
  ▶ parameters:
  ▼ resultSets: [] 2 items
    ▼ 0:
        name: "Shot_Chart_Detail"
      ▼ headers: [] 24 items
          0: "GRID_TYPE"
          1: "GAME_ID"
          2: "GAME_EVENT_ID"
          3: "PLAYER_ID"
          4: "PLAYER_NAME"
          5: "TEAM_ID"
          6: "TEAM_NAME"
          7: "PERIOD"
          8: "MINUTES_REMAINING"
          9: "SECONDS_REMAINING"
          10: "EVENT_TYPE"
          11: "ACTION_TYPE"
          12: "SHOT_TYPE"
          13: "SHOT_ZONE_BASIC"
          14: "SHOT_ZONE_AREA"
          15: "SHOT_ZONE_RANGE"
          16: "SHOT_DISTANCE"
          17: "LOC_X"
          18: "LOC_Y"
          19: "SHOT_ATTEMPTED_FLAG"
          20: "SHOT_MADE_FLAG"
          21: "GAME_DATE"
          22: "HTM"
          23: "VTM"
```

The rowSet level of the resultSets[0] data



```
▼ root:
    resource: "shotchartdetail"
  ▶ parameters:
  ▼ resultSets: [] 2 items
    ▼ 0:
        name: "Shot_Chart_Detail"
      ▶ headers: [] 24 items
      ▼ rowSet: [] 11846 items
        ▼ 0: [] 24 items
            0: "Shot Chart Detail"
            1: "0020900015"
            2: 4
            3: 201939
            4: "Stephen Curry"
            5: 1610612744
            6: "Golden State Warriors"
            7: 1
            8: 11
            9: 25
            10: "Missed Shot"
            11: "Jump Shot"
            12: "3PT Field Goal"
            13: "Above the Break 3"
            14: "Right Side Center(RC)"
            15: "24+ ft."
            16: 26
            17: 99
            18: 249
            19: 1
            20: 0
            21: "20091028"
            22: "GSW"
            23: "HOU"
```

The load() method of a JSON object

Method	Description
`load()`	Converts the data in a JSON file to a dictionary.

How to convert a JSON file to a dictionary

```
with open('shots.json') as json_data:
    shots = json.load(json_data)
```

The DataFrame() constructor for a JSON file

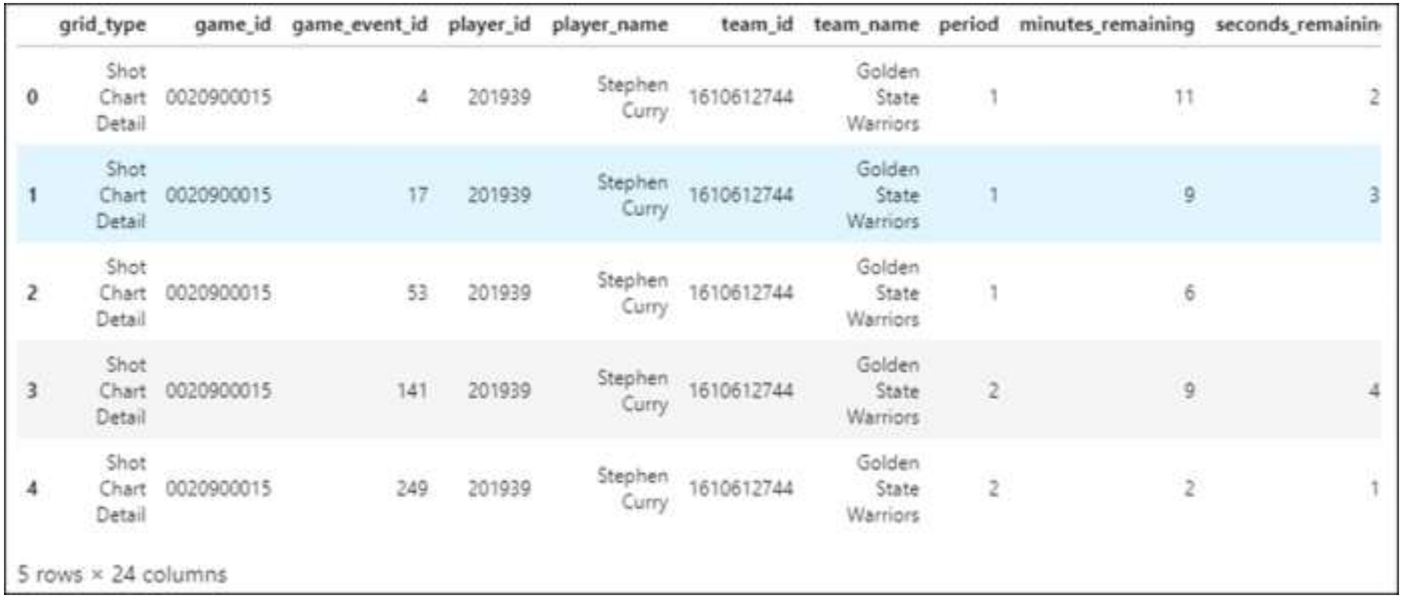
Method	Description
`DataFrame(params)`	Constructs a DataFrame.

Parameters of the DataFrame constructor

Parameter	Description
`data`	Can be an array, dictionary, or other object that's shaped like a table.
`columns`	Column labels. If they aren't specified, they will be generated.
`index`	Row labels. If they aren't specified, they will be generated.

How to to build a DataFrame for the shots

```
allRows = shots['resultSets'][0]['rowSet']
columnHeaders = \
    [x.lower() for x in shots['resultSets'][0]['headers']]
shots = pd.DataFrame(data=allRows, columns=columnHeaders)
shots.head()
```



	grid_type	game_id	game_event_id	player_id	player_name	team_id	team_name	period	minutes_remaining	seconds_remainin
0	Shot Chart Detail	0020900015	4	201939	Stephen Curry	1610612744	Golden State Warriors	1	11	2
1	Shot Chart Detail	0020900015	17	201939	Stephen Curry	1610612744	Golden State Warriors	1	9	3
2	Shot Chart Detail	0020900015	53	201939	Stephen Curry	1610612744	Golden State Warriors	1	6	
3	Shot Chart Detail	0020900015	141	201939	Stephen Curry	1610612744	Golden State Warriors	2	9	4
4	Shot Chart Detail	0020900015	249	201939	Stephen Curry	1610612744	Golden State Warriors	2	2	1

5 rows × 24 columns