Working with Tables I: Table Access

Pandas library

SEC file:

In [1]:
```
import requests  # For downloading data
```

In [2]:
```
s = requests.get('https://www.sec.gov/files/company_tickers.json').text
# download as string
s[:200]  # Show first 200 characters
```

Out[2]:
```
'{"0":{"cik_str":320193,"ticker":"AAPL","title":"Apple Inc."},"1":{"cik
_str":789019,"ticker":"MSFT","title":"MICROSOFT CORP"},"2":{"cik_str":1
018724,"ticker":"AMZN","title":"AMAZON COM INC"},"3":{"cik_'
```

In [3]:
```
d = requests.get('https://www.sec.gov/files/company_tickers.json').json
()  # download as dictionary
d['0']
```

Out[3]:
```
{'cik_str': 320193, 'ticker': 'AAPL', 'title': 'Apple Inc.'}
```

We can also read the SEC file with pandas.
Import the library:

In [4]:
```
import pandas as pd    # Import pandas library and abbreviate as pd
```

Use pandas to read the file:

In [5]:
```
pd.read_json('https://www.sec.gov/files/company_tickers.json')
```

Out[5]:

	0	1	2	3	4	5	6	7	
cik_str	320193	789019	1018724	1652044	1318605	1326801	1293451	1577552	
ticker	AAPL	MSFT	AMZN	GOOG	TSLA	FB	TCEHY	BABA	
title	Apple Inc.	MICROSOFT CORP	AMAZON COM INC	Alphabet Inc.	Tesla, Inc.	Facebook Inc	Tencent Holdings Ltd	Alibaba Group Holding Ltd	SEMICC MANUF

3 rows × 11228 columns

Transpose this table (switch rows and columns):

In [6]:
```
symbols = pd.read_json('https://www.sec.gov/files/company_tickers.json')
.transpose()
symbols
```

Out[6]:

	cik_str	ticker	title
0	320193	AAPL	Apple Inc.
1	789019	MSFT	MICROSOFT CORP
2	1018724	AMZN	AMAZON COM INC
3	1652044	GOOG	Alphabet Inc.
4	1318605	TSLA	Tesla, Inc.
...	...	...	...
11223	1326160	DUKB	Duke Energy CORP
11224	1326205	IGCIW	India Globalization Capital, Inc.
11225	1333986	EQH-PA	Equitable Holdings, Inc.
11226	1335105	LIXTW	LIXTE BIOTECHNOLOGY HOLDINGS, INC.
11227	1338065	DCP-PB	DCP Midstream, LP

11228 rows × 3 columns

Find all rowss where ticker equals BAC:

In [7]:
```
symbols.ticker   # Get column "ticker"
```

Out[7]:
```
0          AAPL
1          MSFT
2          AMZN
3          GOOG
4          TSLA
          ...
11223      DUKB
11224     IGCIW
11225    EQH-PA
11226     LIXTW
11227    DCP-PB
Name: ticker, Length: 11228, dtype: object
```

In [8]:
```
symbols.ticker == 'BAC'    # Test if ticker equals 'BAC'
```

Out[8]:
```
0         False
1         False
2         False
3         False
4         False
          ...
11223     False
11224     False
11225     False
11226     False
11227     False
Name: ticker, Length: 11228, dtype: bool
```

In [9]:
```
symbols[ symbols.ticker == 'BAC' ]    # Select rows where ticker equals
 'BAC'
```

Out[9]:

	cik_str	ticker	title
22	70858	BAC	BANK OF AMERICA CORP /DE/

All other securities issued by this firm:

In [10]:
```
symbols[symbols.cik_str==70858]     # Select rows where cik_str equals 7
0858
```

Out[10]:

	cik_str	ticker	title
22	70858	BAC	BANK OF AMERICA CORP /DE/
8370	70858	BML-PL	BANK OF AMERICA CORP /DE/
8371	70858	BAC-PE	BANK OF AMERICA CORP /DE/
8372	70858	BML-PG	BANK OF AMERICA CORP /DE/
8373	70858	BML-PH	BANK OF AMERICA CORP /DE/
8374	70858	BML-PJ	BANK OF AMERICA CORP /DE/
8375	70858	BAC-PL	BANK OF AMERICA CORP /DE/
8376	70858	BAC-PB	BANK OF AMERICA CORP /DE/
8378	70858	BAC-PK	BANK OF AMERICA CORP /DE/
8379	70858	BAC-PC	BANK OF AMERICA CORP /DE/
10739	70858	BAC-PM	BANK OF AMERICA CORP /DE/
10740	70858	BAC-PN	BANK OF AMERICA CORP /DE/
10741	70858	MER-PK	BANK OF AMERICA CORP /DE/
10742	70858	BAC-PO	BANK OF AMERICA CORP /DE/
10743	70858	BAC-PA	BANK OF AMERICA CORP /DE/

Creating a table from scratch

In [11]:
```
pd.DataFrame()  # Create empty table
```

Out[11]:

In [12]:
```
pd.DataFrame(index = ['a','b','c'])   # Create empty table and specify r
ow labels
```

Out[12]:

a	
b	
c	

In [13]:
```
pd.DataFrame(index = ['a','b','c'], columns=['A','B','C'])  # Create emp
ty table with row and column labels
```

Out[13]:

	A	B	C
a	NaN	NaN	NaN
b	NaN	NaN	NaN
c	NaN	NaN	NaN

In [14]:
```
t = pd.DataFrame()     # Create empty table and name it "t"
t['A'] = [1,2,3]       # Add column 'A'
t['B'] = [4,5,6]       # Add column 'B'
t                      # Show result
```

Out[14]:

	A	B
0	1	4
1	2	5
2	3	6

If we don't specify the index, we get the default index 0,1,2, ...
We can also change the index after we created the table:

In [15]:
```
t.index = ['a','b','c']   # Change index of table "t"
t
```

Out[15]:

	A	B
a	1	4
b	2	5
c	3	6

CSV files

In [16]:
```
t.to_csv('test.csv')          # Save table "t" to current working direct
ory (folder) as "test.csv"
```

In [17]:
```
pd.read_csv('test.csv')       # Read the file (default index: 0,1,2)
```

Out[17]:

	Unnamed: 0	A	B
0	a	1	4
1	b	2	5
2	c	3	6

In [18]:
```
pd.read_csv('test.csv',index_col=0)  # Read and set column 0 to index
```

Out[18]:

	A	B
a	1	4
b	2	5
c	3	6

Accessing parts of tables

In [19]:
```
# read file from URL:
data = pd.read_csv('http://www.janschneider.website/data/AAPL.csv')
data
```

Out[19]:

	name	value	segment	startdate	endd
0	CashAndCashEquivalentsAtCarryingValue	9.352000e+09	NaN	NaN	N
1	Assets	3.957200e+10	NaN	NaN	N
2	AvailableForSaleSecuritiesCurrent	1.023600e+10	NaN	NaN	N
3	CommitmentsAndContingencies	0.000000e+00	NaN	NaN	N
4	CommonStockSharesAuthorized	1.800000e+09	NaN	NaN	N
...	...	...	...	...	
32314	EarningsPerShareDiluted	6.100000e-01	NaN	2018-12-30	20 03
32315	RevenueFromContractWithCustomerExcludingAssess...	8.431000e+10	NaN	2018-09-30	20 12
32316	GrossProfit	3.203100e+10	NaN	2018-09-30	20 12
32317	NetIncomeLoss	1.996500e+10	NaN	2018-09-30	20 12
32318	EarningsPerShareBasic	1.050000e+00	NaN	2018-09-30	20 12

32319 rows × 7 columns

In [20]:
```
data = data.set_index('name')    # Use set_index() method to set column
 "name" to index
data
```

Out[20]:

name	value	segment	startdate	endd
CashAndCashEquivalentsAtCarryingValue	9.352000e+09	NaN	NaN	N
Assets	3.957200e+10	NaN	NaN	N
AvailableForSaleSecuritiesCurrent	1.023600e+10	NaN	NaN	N
CommitmentsAndContingencies	0.000000e+00	NaN	NaN	N
CommonStockSharesAuthorized	1.800000e+09	NaN	NaN	N
...	...	...	...	
EarningsPerShareDiluted	6.100000e-01	NaN	2018-12-30	20 03
RevenueFromContractWithCustomerExcludingAssessedTax	8.431000e+10	NaN	2018-09-30	20 12
GrossProfit	3.203100e+10	NaN	2018-09-30	20 12
NetIncomeLoss	1.996500e+10	NaN	2018-09-30	20 12
EarningsPerShareBasic	1.050000e+00	NaN	2018-09-30	20 12

32319 rows × 6 columns

In [34]:
```
data.to_csv('data/apple.csv')
```

Select rows and columns like this:

> table.loc[rows, columns]
>
> table.iloc[row numbers, column numbers]

In [21]:
```
data.loc[ 'Assets' ]  # no columns specified -> get all columns
```

Out[21]:

name	value	segment	startdate	enddate	instant	filedate
Assets	3.957200e+10	NaN	NaN	NaN	2008-09-27	2009-07-22
Assets	4.814000e+10	NaN	NaN	NaN	2009-06-27	2009-07-22
Assets	3.957200e+10	NaN	NaN	NaN	2008-09-27	2009-10-27
Assets	5.385100e+10	NaN	NaN	NaN	2009-09-26	2009-10-27
Assets	5.392600e+10	NaN	NaN	NaN	2009-12-26	2010-01-25
...	...	...	...	...	...	...
Assets	3.204000e+11	NaN	NaN	NaN	2020-03-28	2020-05-01
Assets	3.173440e+11	NaN	NaN	NaN	2020-06-27	2020-07-31
Assets	3.385160e+11	NaN	NaN	NaN	2019-09-28	2020-07-31
Assets	3.238880e+11	NaN	NaN	NaN	2020-09-26	2020-10-30
Assets	3.385160e+11	NaN	NaN	NaN	2019-09-28	2020-10-30

137 rows × 6 columns

In [22]:
```
data.loc[ 'Assets', 'value' ]  # Rows with "Assets", column "value"
```

Out[22]:
```
name
Assets    3.957200e+10
Assets    4.814000e+10
Assets    3.957200e+10
Assets    5.385100e+10
Assets    5.392600e+10
              ...
Assets    3.204000e+11
Assets    3.173440e+11
Assets    3.385160e+11
Assets    3.238880e+11
Assets    3.385160e+11
Name: value, Length: 137, dtype: float64
```

For the rows and columns inside loc[] or iloc[] we can use:

- single value
- list
- slice (from:to)

In [23]:
```
data.loc[['NetIncomeLoss','GrossProfit'], 'startdate':'instant']   # List of rows, slice of columns
```

Out[23]:

name	startdate	enddate	instant
NetIncomeLoss	2008-03-30	2008-06-28	NaN
NetIncomeLoss	2009-03-29	2009-06-27	NaN
NetIncomeLoss	2007-09-30	2008-06-28	NaN
NetIncomeLoss	2008-09-28	2009-06-27	NaN
NetIncomeLoss	2006-10-01	2007-09-29	NaN
...	...	...	...
GrossProfit	2019-09-29	2019-12-28	NaN
GrossProfit	2019-06-30	2019-09-28	NaN
GrossProfit	2019-03-31	2019-06-29	NaN
GrossProfit	2018-12-30	2019-03-30	NaN
GrossProfit	2018-09-30	2018-12-29	NaN

550 rows × 3 columns

In [24]:
```
data.iloc[3]  # Row 3, all columns
```

Out[24]:
```
value                     0
segment                 NaN
startdate               NaN
enddate                 NaN
instant          2008-09-27
filedate         2009-07-22
Name: CommitmentsAndContingencies, dtype: object
```

In [25]:
```
data.iloc[-3:]  # Last 3 rows, all columns
```

Out[25]:

name	value	segment	startdate	enddate	instant	filedate
GrossProfit	3.203100e+10	NaN	2018-09-30	2018-12-29	NaN	2020-10-30
NetIncomeLoss	1.996500e+10	NaN	2018-09-30	2018-12-29	NaN	2020-10-30
EarningsPerShareBasic	1.050000e+00	NaN	2018-09-30	2018-12-29	NaN	2020-10-30

In [26]:
```
data.iloc[-3:, [0,2]]   # Last 3 rows, columns 0 and 2
```

Out[26]:

name	value	startdate
GrossProfit	3.203100e+10	2018-09-30
NetIncomeLoss	1.996500e+10	2018-09-30
EarningsPerShareBasic	1.050000e+00	2018-09-30

There are several ways to abbreviate the selection, without writing loc or iloc:

In [27]:
```
data.value                          # Get a single column (returns a panda
s series)
data['value']                       # Same
data[['value','startdate']]         # Get multiple columns (need double br
acket, returns dataFrame)
data[['value']]                     # Get list of columns (only 1 element
 in list), returns dataFrame
```

Out[27]:

name	value
CashAndCashEquivalentsAtCarryingValue	9.352000e+09
Assets	3.957200e+10
AvailableForSaleSecuritiesCurrent	1.023600e+10
CommitmentsAndContingencies	0.000000e+00
CommonStockSharesAuthorized	1.800000e+09
...	...
EarningsPerShareDiluted	6.100000e-01
RevenueFromContractWithCustomerExcludingAssessedTax	8.431000e+10
GrossProfit	3.203100e+10
NetIncomeLoss	1.996500e+10
EarningsPerShareBasic	1.050000e+00

32319 rows × 1 columns

In [28]:
```
type(data['value'])  # Single [] returns series
```

Out[28]:
```
pandas.core.series.Series
```

In [29]:
```
data.iloc[2:6]                      # Get rows 2,3,4,5
data[2:6]                           # Same

data[2:6][['value','filedate']]  # Get rows 2,3,4,5 and then select colu
nns "value", "filedate"
data[['value','filedate']][2:6]  # Select colunns "value", "filedate", a
nd then get rows 2,3,4,5
```

Out[29]:

name	value	filedate
AvailableForSaleSecuritiesCurrent	1.023600e+10	2009-07-22
CommitmentsAndContingencies	0.000000e+00	2009-07-22
CommonStockSharesAuthorized	1.800000e+09	2009-07-22
CommonStockSharesIssued	8.883260e+08	2009-07-22

In [32]:
```
data[1:4]                           # If we want to get rows without iloc,
we have to use a slice (from:to)
data.iloc[0]                        # For a single row number, we need to
 use iloc[]
```

Out[32]:
```
value         9.352e+09
segment             NaN
startdate           NaN
enddate             NaN
instant      2007-09-29
filedate     2009-07-22
Name: CashAndCashEquivalentsAtCarryingValue, dtype: object
```

In [33]:
```
data.loc[ data.value>10**11, 'value' ]      # All rows where value > 10B

data[ data.value>10**11 ].value             # Same without .loc[]
```

Out[33]:
```
name
Assets                                                  1.067580e+11
Assets                                                  1.163710e+11
Assets                                                  1.039220e+11
PaymentsToAcquireAvailableForSaleSecurities             1.023170e+11
SalesRevenueNet                                         1.082490e+11
                                                                ...
RevenueFromContractWithCustomerExcludingAssessedTax     1.091970e+11
RevenueFromContractWithCustomerExcludingAssessedTax     1.022660e+11
RevenueFromContractWithCustomerExcludingAssessedTax     1.250100e+11
RevenueFromContractWithCustomerExcludingAssessedTax     1.142300e+11
RevenueFromContractWithCustomerExcludingAssessedTax     1.155920e+11
Name: value, Length: 1128, dtype: float64
```