SEC Data IV: Combine Items

In [1]:
```
import pandas as pd
import numpy  as np
import os
```

Example: get data from 2020q3 file:

In [2]:
```
directory = 'data/sec/merged/'
filename  = '2020q3.csv'

data = pd.read_csv(directory+filename, parse_dates=['filed','ddate'])
data[:3]
```

Out[2]:

	cik	sic	countryinc	tag	filed	ddate	qtrs	val
0	1661920	1311.0	US	EntityCommonStockSharesOutstanding	2020-07-02	2020-06-30	0	2494489
1	1661920	1311.0	US	AccountsReceivableNetCurrent	2020-07-02	2019-12-31	0	1599100
2	1661920	1311.0	US	AccountsReceivableNetCurrent	2020-07-02	2019-12-31	0	1600000

Get revenue:

In [3]:
```
tag = 'RevenueFromContractWithCustomerExcludingAssessedTax'

item = data[data.tag==tag]

item.sort_values('cik')[:10]  # Sort by firm and display first 10 rows
```

Out[3]:

	cik	sic	countryinc	tag	filed	dda
282471	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	202 06-
282472	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	20 06-
282473	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	202 06-
282474	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	20 06-
1017497	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	202 06-
1017498	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	20 06-
1017499	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	202 06-

	cik	sic	countryinc	tag	filed	dda
1017500	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	201 06-
592339	2488	3674.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	202 06-
592340	2488	3674.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	201 06-

Find most recent values with shortest duration (smallest qtrs):

In [4]:
```
short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True,True,Tr
short[:10]
```

Out[4]:

	cik	sic	countryinc	tag	filed	dda
282474	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	201 06-
282472	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	201 06-
282473	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	202 06-
282471	1800	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	202 06-
1017500	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	201 06-
1017498	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	201 06-
1017499	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	202 06-
1017497	2178	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-08-06	202 06-
592340	2488	3674.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	201 06-
592338	2488	3674.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-07-29	201 06-

In [5]:
```
short = short.groupby(['cik','filed']).last()
short[:5]
```

Out[5]:

		sic	countryinc	tag	ddate	qtrs
cik	**filed**					
1800	**2020-07-29**	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-06-30	1

cik	filed	sic	countryinc	tag	ddate	qtrs
2178	**2020-08-06**	5172.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-06-30	1
2488	**2020-07-29**	3674.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-06-30	1
3116	**2020-08-07**	2834.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-06-30	1
3197	**2020-08-05**	3564.0	US	RevenueFromContractWithCustomerExcludingAssess...	2020-06-30	1

After we selected our rows, we only need the value and qtrs columns:

In [6]:
```
short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True,True,Tr

short = short.groupby(['cik','filed']).last() [['value','qtrs']]
short[:5]
```

Out[6]:

cik	filed	value	qtrs
1800	**2020-07-29**	7.328000e+09	1
2178	**2020-08-06**	1.436740e+08	1
2488	**2020-07-29**	1.932000e+09	1
3116	**2020-08-07**	1.203100e+08	1
3197	**2020-08-05**	7.517000e+07	1

Same for longest duration (largest qtrs):

In [7]:
```
long = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True,True,Tru
long = long.groupby(['cik','filed']).last()[['value','qtrs']]
long[:5]
```

Out[7]:

cik	filed	value	qtrs
1800	**2020-07-29**	1.505400e+10	2
2178	**2020-08-06**	4.796470e+08	2
2488	**2020-07-29**	3.718000e+09	2
3116	**2020-08-07**	3.250030e+08	2
3197	**2020-08-05**	1.556560e+08	2

Put short and long next to each other:

In [8]:
```
short_long = short.join(long, lsuffix='_shortest', rsuffix='_longest')
short_long[:10]
```

Out[8]:

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
1800	**2020-07-29**	7.328000e+09	1	1.505400e+10	2
2178	**2020-08-06**	1.436740e+08	1	4.796470e+08	2
2488	**2020-07-29**	1.932000e+09	1	3.718000e+09	2
3116	**2020-08-07**	1.203100e+08	1	3.250030e+08	2
3197	**2020-08-05**	7.517000e+07	1	1.556560e+08	2
3453	**2020-08-05**	5.241000e+08	1	1.038000e+09	2
3545	**2020-08-06**	2.612200e+07	1	8.764200e+07	3
3570	**2020-08-06**	2.424000e+09	1	4.917000e+09	2
4127	**2020-07-24**	7.368000e+08	1	2.398900e+09	3
4281	**2020-08-10**	1.253000e+09	1	2.887000e+09	2

Get multiple tags:

In [9]:
```
tags = ['RevenueFromContractWithCustomerExcludingAssessedTax','NetIncomeLoss']

results = {t:pd.DataFrame() for t in tags}

for t in tags:                                          # Loop over all tags.

    item  = data[data.tag==t]                           # Select all data for

    short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True,Tru
    long  = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True,Tru

    short = short.groupby(['cik','filed']).last()[['value','qtrs']]     # One va
    long  = long .groupby(['cik','filed']).last()[['value','qtrs']]

    short_long = short.join(long, lsuffix='_shortest', rsuffix='_longest') # Put

    results[t] = results[t].append( short_long )
```

All revenues:

In [10]:
```
results['RevenueFromContractWithCustomerExcludingAssessedTax'][:5]
```

Out[10]:

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
1800	**2020-07-29**	7.328000e+09	1	1.505400e+10	2
2178	**2020-08-06**	1.436740e+08	1	4.796470e+08	2
2488	**2020-07-29**	1.932000e+09	1	3.718000e+09	2
3116	**2020-08-07**	1.203100e+08	1	3.250030e+08	2

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
3197	2020-08-05	7.517000e+07	1	1.556560e+08	2

All earnings:

In [11]:
```
results['NetIncomeLoss'][:5]
```

Out[11]:

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
1750	2020-07-21	-16500000.0	1	4.400000e+06	4
	2020-09-24	-14500000.0	1	-1.450000e+07	1
1800	2020-07-29	537000000.0	1	1.101000e+09	2
1961	2020-08-13	-419314.0	1	-7.403690e+05	2
2098	2020-08-10	3199000.0	1	4.476000e+06	2

Now loop over multiple files and multiple tags:

In [13]:
```
filenames = os.listdir(directory)  [:5]
filenames = [f for f in filenames if not f.startswith(".")]   # Exclude hidden f

results   = {t:pd.DataFrame() for t in tags}                  # Dictionary of ta

for filename in filenames:                                    # Loop over all fi
    print(filename)
    data = pd.read_csv(directory+filename, parse_dates=['filed','ddate'])  # Rea

    for t in tags:                                            # Loop over all ta
        item  = data[data.tag==t]                             # Select all data
        short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True
        long  = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[True
        short = short.groupby(['cik','filed']).last()[['value','qtrs']]     # On
        long  = long .groupby(['cik','filed']).last()[['value','qtrs']]
        short_long = short.join(long, lsuffix='_shortest', rsuffix='_longest') #
        results[t] = results[t].append( short_long )
```

```
2018q4.csv
2018q3.csv
2018q2.csv
2021_01.csv
2018q1.csv
```

Check results:

In [14]:
```
results['NetIncomeLoss']
```

Out[14]:

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
1750	2018-12-19	7000000.0	1	2.210000e+07	2

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
1800	**2018-10-31**	563000000.0	1	1.714000e+09	3
1961	**2018-11-14**	40749.0	1	-5.701750e+05	3
2034	**2018-11-09**	-21092000.0	1	-2.109200e+07	1
2098	**2018-11-09**	807000.0	1	4.007000e+06	3
...	**...**	...	...	...	...
1719406	**2018-03-23**	-105033.0	1	-1.050330e+05	1
1719489	**2018-02-09**	-145928.0	1	-1.459280e+05	1
1721478	**2018-02-14**	-1250.0	1	-1.250000e+03	1
1723128	**2018-03-09**	0.0	1	0.000000e+00	1
1723596	**2018-03-23**	3678000.0	1	3.678000e+06	1

22001 rows × 4 columns

Sort the result by filedate:

In [15]:
```
results['NetIncomeLoss'] = results['NetIncomeLoss'].sort_index(level='filed')
results['NetIncomeLoss']
```

Out[15]:

cik	filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
863894	**2018-01-02**	-2679335.0	1	-6612047.0	4
1392694	**2018-01-02**	-967577.0	1	-967577.0	1
1606364	**2018-01-02**	-103693.0	1	-183631.0	2
1619227	**2018-01-02**	-7928.0	1	-53505.0	3
1634293	**2018-01-02**	-61443.0	4	-61443.0	4
...	**...**	...	...	...	...
1527102	**2021-01-29**	-183760.0	1	-1195973.0	3
1550603	**2021-01-29**	-546000.0	1	3601000.0	4
1551887	**2021-01-29**	-485916.0	4	-485916.0	4
1580149	**2021-01-29**	-3057807.0	1	4276109.0	2
1807707	**2021-01-29**	-3458412.0	3	-3458412.0	3

22001 rows × 4 columns

Put all of this into a function:

In [16]:
```
def get_items_from_SEC_files(tags, filename=None):                    # Function inp
    directory = 'data/sec/merged/'                                    # Read data fr
```

```
    filenames = [filename] if filename else os.listdir(directory) # Supplied fil
    filenames = [f for f in filenames if not f.startswith(".")]   # Exclude hidd

    results   = {t:pd.DataFrame() for t in tags}                  # Dictionary o

    for filename in filenames:                                    # Loop over al
        print(filename)
        data = pd.read_csv(directory+filename, parse_dates=['filed','ddate'])  #

        for t in tags:                                            # Loop over al
            item  = data[data.tag==t]                             # Select all d
            short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[
            long  = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[
            short = short.groupby(['cik','filed']).last()[['value','qtrs']]
            long  = long .groupby(['cik','filed']).last()[['value','qtrs']]
            short_long = short.join(long, lsuffix='_shortest', rsuffix='_longest
            results[t] = results[t].append( short_long )

    for t in tags:                                                # Now sort all
        if not results[t].empty: results[t] = results[t].sort_index(level='filed

    return results
```

Optional arguments for functions:

In [17]:

```
def do_this(a, b=1):  # set b to 1 if b not specified
    print(a,b)

do_this(5)            # b not specified
do_this(5, b=6)
```

```
5 1
5 6
```

Use the function to get items from specific file:

In [18]:

```
tags = ['RevenueFromContractWithCustomerExcludingAssessedTax','NetIncomeLoss']

items = get_items_from_SEC_files(tags, filename='2020q2.csv')
```

```
2020q2.csv
```

Check result:

In [19]:

```
items['NetIncomeLoss'][:5]
```

Out[19]:

		value_shortest	qtrs_shortest	value_longest	qtrs_longest
cik	**filed**				
18498	**2020-04-01**	35562000.0	1	6.138400e+07	4
56873	**2020-04-01**	327000000.0	1	1.659000e+09	4
78239	**2020-04-01**	417300000.0	4	4.173000e+08	4
744187	**2020-04-01**	4402000.0	1	7.427000e+06	4
795266	**2020-04-01**	59748000.0	1	5.974800e+07	1

Use the function to get items from all files:

In [20]:
```
tags = ['RevenueFromContractWithCustomerExcludingAssessedTax','SalesRevenueNet',

items = get_items_from_SEC_files(tags)
```

```
2018q4.csv
2018q3.csv
2018q2.csv
2021_01.csv
2018q1.csv
2020q2.csv
2020q3.csv
2020q1.csv
2019q4.csv
2019q1.csv
2019q3.csv
2019q2.csv
2013q4.csv
2015q2.csv
2015q3.csv
2017q1.csv
2020_12.csv
2020_10.csv
2017q3.csv
2015q1.csv
2017q2.csv
2011q4.csv
2020_11.csv
2013q2.csv
2015q4.csv
2011q1.csv
2013q3.csv
2013q1.csv
2011q3.csv
2017q4.csv
2011q2.csv
2009q4.csv
2014q1.csv
2016q3.csv
2010q4.csv
2016q2.csv
2014q2.csv
2012q4.csv
2016q1.csv
2014q3.csv
2009q2.csv
2010q3.csv
2012q1.csv
2010q2.csv
2016q4.csv
2009q3.csv
2009q1.csv
2014q4.csv
2012q2.csv
2012q3.csv
2010q1.csv
```

Check revenue for Apple:

In [21]:
```
items['RevenueFromContractWithCustomerExcludingAssessedTax'].loc[320193]  # 3201
```

Out[21]:

filed	value_shortest	qtrs_shortest	value_longest	qtrs_longest
2019-01-30	8.431000e+10	1	8.431000e+10	1
2019-05-01	5.801500e+10	1	1.423250e+11	2
2019-07-31	5.380900e+10	1	1.961340e+11	3
2019-10-31	6.404000e+10	1	2.601740e+11	4
2020-01-29	9.181900e+10	1	9.181900e+10	1
2020-05-01	5.831300e+10	1	1.501320e+11	2
2020-07-31	5.968500e+10	1	2.098170e+11	3
2020-10-30	6.469800e+10	1	2.745150e+11	4
2021-01-28	1.114390e+11	1	1.114390e+11	1

Compare all 3 tags for Apple:

In [22]:

```
cik = 320193

t = pd.DataFrame(index = pd.date_range('2009','2021'))    # Create table with dat

t[tags[0]] = items[tags[0]].loc[cik].value_shortest       # Add column tag[0] (=
t[tags[1]] = items[tags[1]].loc[cik].value_shortest
t[tags[2]] = items[tags[2]].loc[cik].value_shortest

t.dropna(how='all')
```

Out[22]:

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2009-07-22	NaN	8.337000e+09	NaN
2009-10-27	NaN	3.653700e+10	NaN
2010-01-25	NaN	1.568300e+10	NaN
2010-04-21	NaN	1.349900e+10	NaN
2010-07-21	NaN	1.570000e+10	NaN
2010-10-27	NaN	2.034300e+10	NaN
2011-01-19	NaN	2.674100e+10	NaN
2011-04-21	NaN	2.466700e+10	NaN
2011-07-20	NaN	2.857100e+10	NaN
2011-10-26	NaN	2.827000e+10	NaN

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2012-01-25	NaN	4.633300e+10	NaN
2012-04-25	NaN	3.918600e+10	NaN
2012-07-25	NaN	3.502300e+10	NaN
2012-10-31	NaN	3.596600e+10	NaN
2013-01-24	NaN	5.451200e+10	NaN
2013-04-24	NaN	4.360300e+10	NaN
2013-07-24	NaN	3.532300e+10	NaN
2013-10-30	NaN	3.747200e+10	NaN
2014-01-28	NaN	5.759400e+10	NaN
2014-04-24	NaN	4.564600e+10	NaN
2014-07-23	NaN	3.743200e+10	NaN
2014-10-27	NaN	4.212300e+10	NaN
2015-01-28	NaN	7.459900e+10	NaN
2015-04-28	NaN	5.801000e+10	NaN
2015-07-22	NaN	4.960500e+10	NaN
2015-10-28	NaN	5.150100e+10	NaN
2016-01-27	NaN	7.587200e+10	NaN
2016-04-27	NaN	5.055700e+10	NaN
2016-07-27	NaN	4.235800e+10	NaN
2016-10-26	NaN	4.685200e+10	NaN
2017-02-01	NaN	7.835100e+10	NaN
2017-05-03	NaN	5.289600e+10	NaN

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2017-08-02	NaN	4.540800e+10	NaN
2017-11-03	NaN	5.257900e+10	NaN
2018-02-02	NaN	8.829300e+10	NaN
2018-05-02	NaN	6.113700e+10	NaN
2018-08-01	NaN	5.326500e+10	NaN
2018-11-05	NaN	NaN	6.290000e+10
2019-01-30	8.431000e+10	NaN	NaN
2019-05-01	5.801500e+10	NaN	NaN
2019-07-31	5.380900e+10	NaN	NaN
2019-10-31	6.404000e+10	NaN	NaN
2020-01-29	9.181900e+10	NaN	NaN
2020-05-01	5.831300e+10	NaN	NaN
2020-07-31	5.968500e+10	NaN	NaN
2020-10-30	6.469800e+10	NaN	NaN

We can combine the items like this:

In [23]:
```
# Example:
t1 = pd.DataFrame({'A':[1,2,np.nan],               },  index=['a','b','c'])
t2 = pd.DataFrame({'A':[10,20,30], 'B':[40,50,60]  },  index=['a','b','c'])

t1.combine_first(t2)
```

Out[23]:

	A	B
a	1.0	40
b	2.0	50
c	30.0	60

In [24]:
```
# Replace missing values from first tag (RevenueFromContractWithCustomerExcludin
# Replace values that are still missing with 3rd tag (Revenues)
```

```
# Call the resulting table "sales".
sales = items[tags[0]].combine_first( items[tags[1]] ).combine_first( items[tags
```

Now compare all 3 items and "sales":

In [25]:

```
cik = 320193

t = pd.DataFrame(index = pd.date_range('2009','2021')) # Create table with dates

t[tags[0]] = items[tags[0]].loc[cik].value_shortest
t[tags[1]] = items[tags[1]].loc[cik].value_shortest
t[tags[2]] = items[tags[2]].loc[cik].value_shortest

t['Sales'] = sales.loc[cik].value_shortest

t.dropna(how='all')
```

Out[25]:

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2009-07-22	NaN	8.337000e+09	NaN
2009-10-27	NaN	3.653700e+10	NaN
2010-01-25	NaN	1.568300e+10	NaN
2010-04-21	NaN	1.349900e+10	NaN
2010-07-21	NaN	1.570000e+10	NaN
2010-10-27	NaN	2.034300e+10	NaN
2011-01-19	NaN	2.674100e+10	NaN
2011-04-21	NaN	2.466700e+10	NaN
2011-07-20	NaN	2.857100e+10	NaN
2011-10-26	NaN	2.827000e+10	NaN
2012-01-25	NaN	4.633300e+10	NaN
2012-04-25	NaN	3.918600e+10	NaN
2012-07-25	NaN	3.502300e+10	NaN
2012-10-31	NaN	3.596600e+10	NaN
2013-01-24	NaN	5.451200e+10	NaN

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2013-04-24	NaN	4.360300e+10	NaN
2013-07-24	NaN	3.532300e+10	NaN
2013-10-30	NaN	3.747200e+10	NaN
2014-01-28	NaN	5.759400e+10	NaN
2014-04-24	NaN	4.564600e+10	NaN
2014-07-23	NaN	3.743200e+10	NaN
2014-10-27	NaN	4.212300e+10	NaN
2015-01-28	NaN	7.459900e+10	NaN
2015-04-28	NaN	5.801000e+10	NaN
2015-07-22	NaN	4.960500e+10	NaN
2015-10-28	NaN	5.150100e+10	NaN
2016-01-27	NaN	7.587200e+10	NaN
2016-04-27	NaN	5.055700e+10	NaN
2016-07-27	NaN	4.235800e+10	NaN
2016-10-26	NaN	4.685200e+10	NaN
2017-02-01	NaN	7.835100e+10	NaN
2017-05-03	NaN	5.289600e+10	NaN
2017-08-02	NaN	4.540800e+10	NaN
2017-11-03	NaN	5.257900e+10	NaN
2018-02-02	NaN	8.829300e+10	NaN
2018-05-02	NaN	6.113700e+10	NaN
2018-08-01	NaN	5.326500e+10	NaN

	RevenueFromContractWithCustomerExcludingAssessedTax	SalesRevenueNet	Revenues
2018-11-05	NaN	NaN	6.290000e+10
2019-01-30	8.431000e+10	NaN	NaN
2019-05-01	5.801500e+10	NaN	NaN
2019-07-31	5.380900e+10	NaN	NaN
2019-10-31	6.404000e+10	NaN	NaN
2020-01-29	9.181900e+10	NaN	NaN
2020-05-01	5.831300e+10	NaN	NaN
2020-07-31	5.968500e+10	NaN	NaN
2020-10-30	6.469800e+10	NaN	NaN

And now we can get the entire sales history for any firm like this:

In [26]:
```
# Get ticker symbol file from SEC
symbols = pd.read_json('https://www.sec.gov/files/company_tickers.json').transpo
```

In [27]:
```
cik = symbols[symbols.ticker=='MSFT'].index[0]

sales.loc[cik].value_shortest.div(10**9).plot()           # Plot quarterly sales
```

Out[27]: <AxesSubplot:xlabel='filed'>