U.S. Stock Market II: Income vs Revenue

In [1]:
```
import pandas as pd
import numpy as np
import requests, zipfile, io
import os
from pathlib import Path

from tiingo import TiingoClient
tiingo = TiingoClient({'api_key':'XXXX'})

# Plotting:
import matplotlib.pyplot as plt                          # Basic plot library.
plt.style.use('ggplot')                                  # Make plots look nice.
```

In [2]:
```
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



def combine_items(tags, items):
    result = items[tags[0]]
    for tag in tags[1:]:  result = result.combine_first( items[tag] )
    return result



def calculate_quarterly_annual_values(item):                      # item: tabl
    result           = pd.DataFrame()                             # Results go
    all_firms        = item.index.get_level_values('cik').unique()  # All CIKs.
    all_filing_dates = pd.read_csv('data/sec/dates/filing_dates.csv', index_col=

    for cik in all_firms:                                         # Loop over
```

```
        filing_dates = pd.Series(all_filing_dates.filed[cik])       # All filing

        # Quarterly values:
        valuesQ = item.loc[cik].value_shortest.reindex(filing_dates) # Values wi
        qtrsQ   = item.loc[cik].qtrs_shortest.astype(int)           # Number of
        for date,q in qtrsQ[qtrsQ>1].iteritems():                   # Loop over
            previous_values = valuesQ[:date][-q:-1]                 # Example: f
            if len(previous_values) == q-1:                         # If all pre
                valuesQ[date] -= previous_values.sum(skipna=False)  # Subtract p
            else:
                valuesQ[date]  = np.nan

        # Annual values:
        valuesA = item.loc[cik].value_longest.reindex(filing_dates) # Values wit
        qtrsA   = item.loc[cik].qtrs_longest.astype(int)            # Number of
        for date,q in qtrsA[qtrsA<4].iteritems():                   # Loop over
            previous_values = valuesQ[:date][-4:-q]                 # Example: f
            if len(previous_values) == 4-q:                         # If all pre
                valuesA[date] += previous_values.sum(skipna=False)  # Add previo
            else:
                valuesA[date]  = np.nan

        result = result.append( pd.DataFrame({'cik':cik, 'filed':filing_dates, '

    return result.set_index(['cik','filed'])                        # Return a t



def ffill_values(item, dates):
    data = item.unstack('cik')
    data = data.reindex(dates.union(data.index)).sort_index()           # Add sp
    filing_dates = pd.read_csv('data/sec/dates/filing_dates.csv', index_col='cik
    last_filing_date_all_firms = filing_dates.max()                     # Most r

    for cik in data.columns:                                            # Loop o
        last_filing_date      = pd.Series(filing_dates[cik]).iloc[-1]   # Last d
        days_since_last_filed = (last_filing_date_all_firms - last_filing_date).
        last_date_this_firm   = dates[-1] if days_since_last_filed < 120 else la
        data.loc[:last_date_this_firm, cik].ffill(inplace=True)         # Forwar

    return data.loc[dates]                                              # Return
```

In [4]:

```
tags_sales           = ['RevenueFromContractWithCustomerExcludingAssessedTax','S
                        'SalesAndOtherOperatingRevenueIncludingSalesBasedTaxes',

tags_earnings        = ['NetIncomeLossAvailableToCommonStockholdersBasic','NetIn

tags_operatingIncome = ['OperatingIncomeLoss']
```

In [5]:

```
items = get_items_from_SEC_files( tags_sales + tags_earnings + tags_operatingInc
```

```
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
```

Fix data errors:

In [6]:

```
# RUN THIS CELL ONLY ONCE!

# MKSI cik: 1049502
tag = 'RevenueFromContractWithCustomerExcludingAssessedTax'
t = items[tag].reset_index()
t.loc[(t.cik==1049502) & (t.filed=='2018-05-08'), 'value_longest']  /= 1000
t.loc[(t.cik==1049502) & (t.filed=='2018-05-08'), 'value_shortest'] /= 1000
items[tag] = t.set_index(['cik','filed'])

# CRAWA cik: 1049502: Income, 6 Months Ended 2013-3-31 = $263,235
# https://www.sec.gov/cgi-bin/viewer?action=view&cik=47307&accession_number=0000
tag = 'NetIncomeLossAvailableToCommonStockholdersBasic'
t = items[tag].reset_index()
t.loc[(t.cik==47307) & (t.filed=='2013-5-15'), 'value_longest'] /= 10**6
items[tag] = t.set_index(['cik','filed'])
```

```
# Bonanza Creek Energy, Inc; CIK=1509589 -> OperatingIncomeLoss 2013-03-15 = 3.6
# WMT CIK=104169, 2017-08-31 values missing, source: https://www.sec.gov/cgi-bin
tag = 'OperatingIncomeLoss'
t = items[tag].reset_index()
t.loc[(t.cik==1509589) & (t.filed=='2013-03-15'), 'value_shortest'] /= 1000
t.loc[(t.cik==104169)  & (t.filed=='2017-08-31'), ['value_shortest','qtrs_shorte
items[tag] = t.set_index(['cik','filed'])
```

Combine items:

In [7]:
```
items['Sales']    = combine_items(tags_sales,    items)
items['Earnings'] = combine_items(tags_earnings, items)
```

Calculate quarterly and annual values:

In [8]:
```
sales           = calculate_quarterly_annual_values(items['Sales'])
earnings        = calculate_quarterly_annual_values(items['Earnings'])
operatingIncome = calculate_quarterly_annual_values(items['OperatingIncomeLoss'

sales[:5]
```

Out[8]:

		valueQ	valueA
cik	**filed**		
1750	**2010-09-23**	412197000.0	NaN
	2010-12-21	447054000.0	NaN
	2011-03-22	451031000.0	NaN
	2011-07-13	465500000.0	1.775782e+09
	2011-09-23	479290000.0	1.842875e+09

Save the results:

In [9]:
```
sales          .to_csv('data/sec/items/Sales.csv')
earnings       .to_csv('data/sec/items/Earnings.csv')
operatingIncome.to_csv('data/sec/items/OperatingIncome.csv')
```

Calculate the quarterly and annual values:

In [10]:
```
trading_days = tiingo.get_dataframe('SPY','2009-04-15').index.tz_convert(None)

salesQ = ffill_values(sales.valueQ, trading_days)
salesA = ffill_values(sales.valueA, trading_days)

earningsQ = ffill_values( earnings.valueQ, trading_days )
earningsA = ffill_values( earnings.valueA, trading_days )

operatingIncomeQ = ffill_values( operatingIncome.valueQ, trading_days )
operatingIncomeA = ffill_values( operatingIncome.valueA, trading_days )

salesA[:3]
```

Out[10]:

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1818152	1818346
date													
2009-04-15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2009-04-16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2009-04-17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN

3 rows × 9826 columns

Which 10 firms have the most recent highest annual sales?

In [11]:
```
symbols = pd.read_csv('data/ticker_symbols/symbols.csv',index_col=0)
symbols[:2]
```

Out[11]:

cik	ticker	title	exchange	assetType	priceCurrency	startDate	endDate
1750	AIR	AAR CORP	NYSE	Stock	USD	1984-07-19	2021-03-01
1800	ABT	ABBOTT LABORATORIES	NYSE	Stock	USD	1983-04-06	2021-03-01

In [12]:
```
salesA.iloc[-1].nlargest(10).div(10**9).to_frame('Sales').join(symbols.title)
```

Out[12]:

cik	Sales	title
104169	544.856000	Walmart Inc.
1018724	347.946000	AMAZON COM INC
320193	294.135000	Apple Inc.
64803	266.041000	CVS HEALTH Corp
731766	252.575000	UNITEDHEALTH GROUP INC
927653	234.194000	MCKESSON CORP
34088	195.860000	EXXON MOBIL CORP
1140859	189.893926	AMERISOURCEBERGEN CORP
909832	172.929000	COSTCO WHOLESALE CORP /NEW
732717	172.890000	AT&T INC.

Which 10 firms have the most recent highest annual earnings?

In [13]:
```
earningsA.iloc[-1].nlargest(10).div(10**9).to_frame('Earnings').join(symbols.tit
```

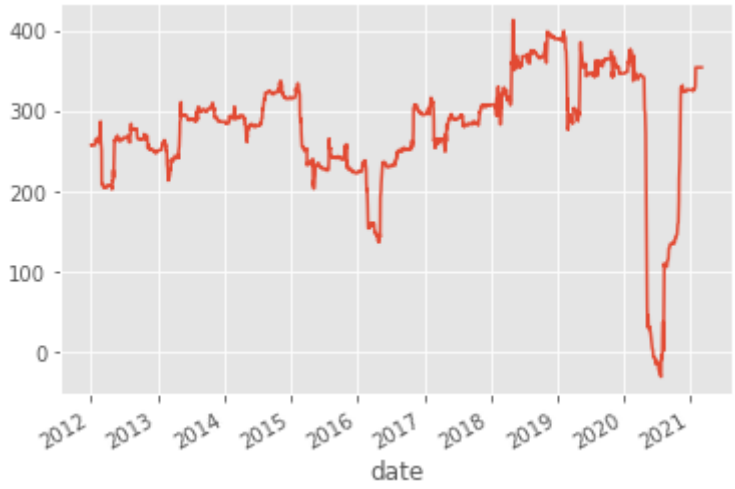
Out[13]:

cik	Earnings	title
320193	63.930	Apple Inc.
789019	51.310	MICROSOFT CORP
1067983	35.845	BERKSHIRE HATHAWAY INC
1652044	35.713	Alphabet Inc.
1326801	29.146	Facebook Inc
19617	23.803	JPMORGAN CHASE & CO
50863	20.899	INTEL CORP
104169	19.742	Walmart Inc.
732712	18.308	VERIZON COMMUNICATIONS INC
70858	18.013	BANK OF AMERICA CORP /DE/

Aggregate quarterly earnings:

In [14]:
```
earningsQ.sum('columns')['2012':].div(10**9).plot()
```

Out[14]: <AxesSubplot:xlabel='date'>

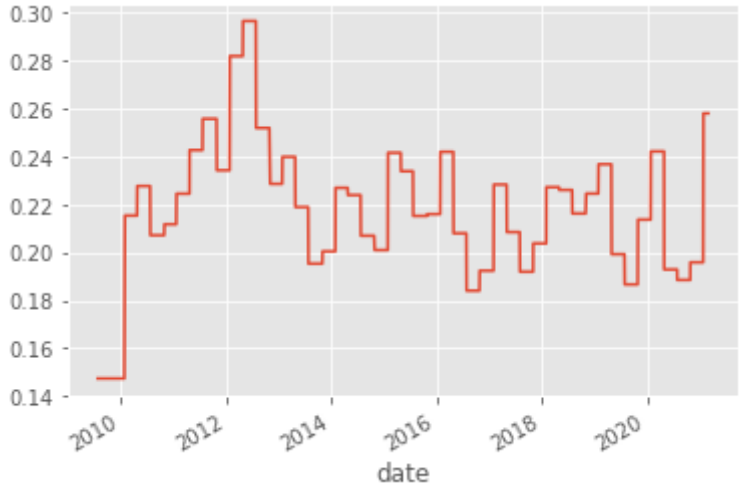


Earnings relative to sales for a specific firm:

In [15]:
```
cik = symbols[symbols.ticker=='AAPL'].index[0]

(earningsQ[cik] / salesQ[cik]).plot()
```

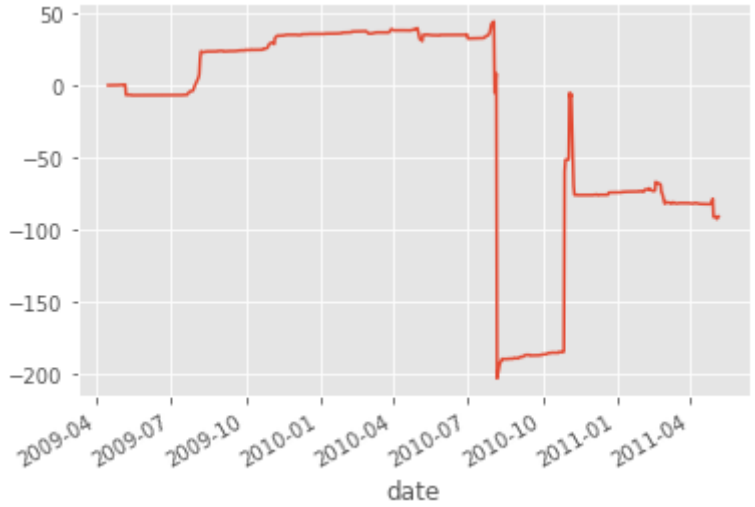
Out[15]: <AxesSubplot:xlabel='date'>



Earnings relative to sales income for the entire market:

In [17]:
```
(earningsQ / salesQ).sum('columns').plot()
```

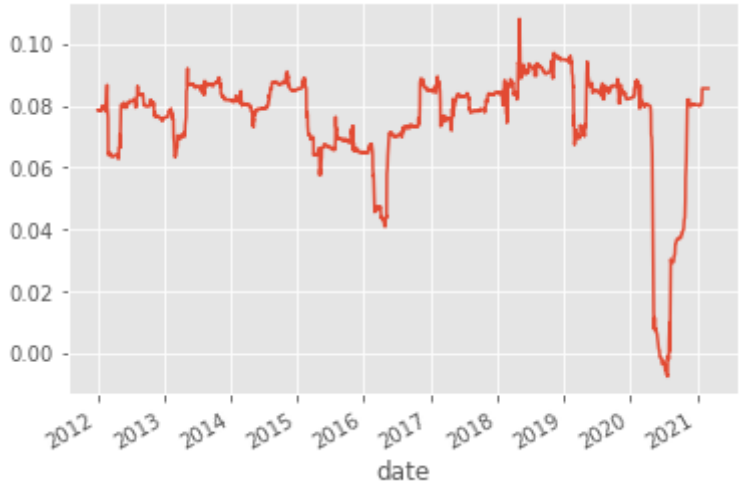
Out[17]: <AxesSubplot:xlabel='date'>



This result is dominated by outliers (firms with very small values in the denominator of the ratio). A better way to calculate the ratio for the entire market is divided the total earnings by the total sales (instead of first dividing each firm and then aggreating):

In [18]:
```
(earningsQ.sum('columns') / salesQ.sum('columns')) ['2012':].plot()
```

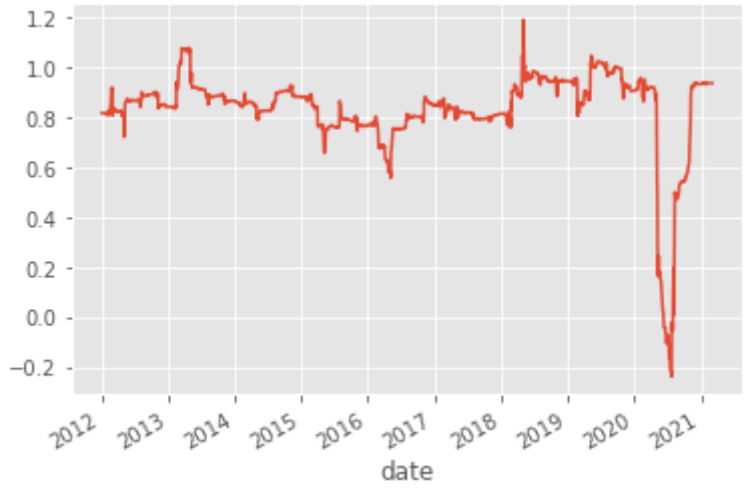
Out[18]: <AxesSubplot:xlabel='date'>



Earnings relative to operating income:

In [19]:
```
(earningsQ.sum('columns') / operatingIncomeQ.sum('columns')) ['2012':].plot()
```

Out[19]:
```
<AxesSubplot:xlabel='date'>
```



Note how this ratio is sometimes above 1. Problem: some firms report earings but nor operating income (for example Goldman Sachs) and then we count these firms in the nominator but not in the denominator.
→ we need to calculate this ratio only for firms that report both values.

How to select values that multiple tables have in common:

In [20]:
```
t1 = pd.DataFrame({'A':[1,2,3],'B':[4,5,np.nan],'C':[7,8,9]}, index=['a','b','c'
t1
```

Out[20]:

	A	B	C
a	1	4.0	7
b	2	5.0	8
c	3	NaN	9

In [21]:
```
t2 = pd.DataFrame({'B':[1,2,3],'C':[np.nan,5,6],'D':[7,8,9]}, index=['b','c','d'
t2
```

Out[21]:

	B	C	D
b	1	NaN	7
c	2	5.0	8
d	3	6.0	9

In [22]:
```
t1 * t2  # Note that only cells [B,b] and [C,c] have values in both tables.
```

Out[22]:

	A	B	C	D
a	NaN	NaN	NaN	NaN
b	NaN	5.0	NaN	NaN
c	NaN	NaN	45.0	NaN
d	NaN	NaN	NaN	NaN

In [23]:
```
(t1 * t2).notnull()   # Turn the table above into True/False.
```

Out[23]:

	A	B	C	D
a	False	False	False	False
b	False	True	False	False
c	False	False	True	False
d	False	False	False	False

In [24]:
```
mask = (earningsQ * operatingIncomeQ).notnull()  # maks = table with True for va

total_operatingIncome = operatingIncomeQ[mask].sum('columns')
total_earnings        = earningsQ       [mask].sum('columns')

(total_earnings / total_operatingIncome)['2012':].plot()
```

Out[24]:
```
<AxesSubplot:xlabel='date'>
```

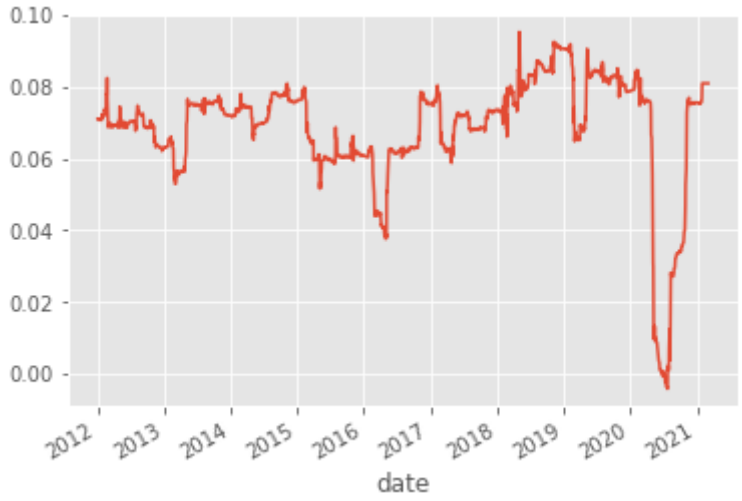
Same for sales:

In [25]:
```
mask = (earningsQ * salesQ).notnull()

total_sales = salesQ[mask].sum('columns')
total_earnings        = earningsQ[mask].sum('columns')

(total_earnings / total_sales)['2012':].plot()
```

Out[25]: <AxesSubplot:xlabel='date'>



And now if we want to use these data again we can read them like this:

In [26]:
```
sales = pd.read_csv('data/sec/items/Sales.csv',parse_dates=['filed'], index_col=
sales[:3]
```

Out[26]:

		valueQ	valueA
filed	**cik**		
2010-09-23	**1750**	412197000.0	NaN
2010-12-21	**1750**	447054000.0	NaN
2011-03-22	**1750**	451031000.0	NaN

And now forward fill the quarterly values:

In [27]:
```
salesQ = ffill_values(sales.valueQ, trading_days)
salesQ
```

Out[27]:

cik	1750	1800	1961	2034	2098	2178	2186	2
date								
2009-04-15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2009-04-16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2009-04-17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2009-04-20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2009-04-21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
2021-02-25	9800000.0	8.853000e+09	-398.0	NaN	43316000.0	263987000.0	12760000.0	3.244000
2021-02-26	9800000.0	8.853000e+09	-398.0	NaN	43316000.0	263987000.0	12760000.0	3.244000
2021-03-01	9800000.0	8.853000e+09	-398.0	NaN	43316000.0	263987000.0	12760000.0	3.244000
2021-03-02	9800000.0	8.853000e+09	-398.0	NaN	43316000.0	263987000.0	12760000.0	3.244000
2021-03-03	9800000.0	8.853000e+09	-398.0	NaN	43316000.0	263987000.0	12760000.0	3.244000

2992 rows × 9826 columns