U.S. Stock Market III: Industries

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'})

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

In [2]:
```
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 [3]:
```
# Read files that we previously created
sales           = pd.read_csv('data/sec/items/Sales.csv',           parse_dates=
earnings        = pd.read_csv('data/sec/items/Earnings.csv',        parse_dates=
operatingIncome = pd.read_csv('data/sec/items/OperatingIncome.csv', parse_dates=

earnings[:5]
```

Out[3]:
```
                     valueQ      valueA
 cik        filed
1750   2010-09-23  13674000.0         NaN
       2010-12-21  16814000.0         NaN
       2011-03-22  17918000.0         NaN
       2011-07-13  21420000.0  69826000.0
       2011-09-23  16053000.0  72205000.0
```

Forward fill table for annual earnings:

In [4]:
```
trading_days = tiingo.get_dataframe('SPY','2009-04-15').index.tz_convert(None)

earningsA = ffill_values( earnings.valueA, trading_days ) / 10**9     # In USD bi
```

We want to get th industry codes from the merged files.
Example:

In [5]:
```python
directory = 'data/sec/merged/'
filename  = '2021_01.csv'
data = pd.read_csv(directory+filename, parse_dates=['filed','ddate'])
data[:5]
```

Out[5]:

	cik	sic	countryinc	tag	filed	ddate	qtrs	
0	1517389	7371.0	US	AccountsPayableAndAccruedLiabilitiesCurrent	2021-01-06	2020-11-30	0	
1	1517389	7371.0	US	AccountsPayableAndAccruedLiabilitiesCurrent	2021-01-06	2020-02-29	0	
2	1517389	7371.0	US	AccountsReceivableNetCurrent	2021-01-06	2020-11-30	0	
3	1517389	7371.0	US	AccountsReceivableNetCurrent	2021-01-06	2020-02-29	0	
4	1517389	7371.0	US	AdditionalPaidInCapital	2021-01-06	2020-11-30	0	24

For every firms and every filing (groupby(['cik','filed'])) select 1 row (we pick the first but this doesn't matter):

In [6]:
```python
data.groupby(['cik','filed']).first()
```

Out[6]:

		sic	countryinc	tag	ddate	qtrs	value
cik	**filed**						
2488	**2021-01-29**	3674.0	US	NetIncomeLoss	2018-12-31	4	337000000.0
4127	**2021-01-29**	3674.0	US	AccountsPayableCurrent	2020-09-30	0	226900000.0
6845	**2021-01-07**	3231.0	US	NetIncomeLoss	2020-11-30	3	57807000.0
6955	**2021-01-05**	3590.0	US	NetIncomeLoss	2020-11-30	1	4598000.0
8858	**2021-01-29**	5065.0	US	NetIncomeLoss	2020-12-31	2	274000.0
...	**...**	...	...	...	...	...	...
1823854	**2021-01-25**	6770.0	US	NetIncomeLoss	2020-09-30	0	-19680.0
1823896	**2021-01-27**	6770.0	US	NetIncomeLoss	2020-09-30	0	-422.0
1824301	**2021-01-19**	6770.0	US	NetIncomeLoss	2020-09-30	0	-479.0
1824734	**2021-01-21**	6770.0	US	NetIncomeLoss	2020-09-30	0	-1000.0

		sic	countryinc	tag	ddate	qtrs	value
cik	**filed**						
1825079	**2021-01-15**	6770.0	KY	NetIncomeLoss	2020-09-30	0	-5000.0

356 rows × 6 columns

Now put this into a funtion so we can apply this selection for every file:

In [7]:
```
def get_attributes_from_SEC_files(attributes, filename=None):          # Function

    directory = 'data/sec/merged/'                                     # Read dat
    filenames = [filename] if filename else os.listdir(directory)      # Supplied
    filenames = [f for f in filenames if not f.startswith(".")]        # Exclude

    results = {a:pd.DataFrame() for a in attributes}                   # Dictiona

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

        for a in attributes:                                           # Loop ove
            item =  data.groupby(['cik','filed'])[[a]].first()         # Get attr
            results[a] = results[a].append( item )

    for a in attributes:
        results[a] = results[a].sort_index(level='filed')              # Sort eac

    return results
```

In [8]:
```
attributes = get_attributes_from_SEC_files(['countryinc','sic'])
```

```
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
```

Result looks like this:

In [9]:

```
attributes['sic']
```

Out[9]:

		sic
cik	**filed**	
277948	**2009-04-15**	4011.0
883984	**2009-04-23**	3841.0
1070412	**2009-04-27**	1221.0
1335793	**2009-04-27**	1311.0
884905	**2009-04-29**	2810.0
...	**...**	...
1550603	**2021-01-29**	6035.0
1551887	**2021-01-29**	7372.0
1580149	**2021-01-29**	2834.0
1593812	**2021-01-29**	6221.0
1807707	**2021-01-29**	100.0

250426 rows × 1 columns

In [10]:

```
# Save data
Path('data/sec/attributes/').mkdir(parents=True, exist_ok=True)  # Generate the

attributes['sic']       .to_csv('data/sec/attributes/sic.csv')
attributes['countryinc'].to_csv('data/sec/attributes/countryinc.csv')
```

And now we can read the sic file like this:

In [11]:
```
# Read data
sic = pd.read_csv('data/sec/attributes/sic.csv', parse_dates=['filed'], index_co
sic[:2]
```

Out[11]:

		sic
filed	**cik**	
2009-04-15	**277948**	4011.0
2009-04-23	**883984**	3841.0

In [12]:
```
# Forward fill the table:
sic = ffill_values(sic.sic, trading_days)    # sic.sic: we select column sic from
```

Let's select the most recent sic codes for each firm (the last row of the sic table):

In [13]:
```
sic_current = sic.iloc[-1].to_frame('sic')
sic_current
```

Out[13]:

	sic
cik	
1750	3720.0
1800	2834.0
1961	7372.0
2034	NaN
2098	3420.0
...	...
1824920	6770.0
1824963	6770.0
1825024	6770.0
1825042	6770.0
1825079	6770.0

12126 rows × 1 columns

In [14]:
```
# Read the ticker symbols
symbols = pd.read_csv('data/ticker_symbols/symbols.csv',index_col=0)
symbols[:3]
```

Out[14]:

	ticker	title	exchange	assetType	priceCurrency	startDate	endDate
cik							

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
1961	WDDD	WORLDS INC	OTCQB	Stock	USD	1998-10-20	2021-03-01

Top 10 Earnings with title and SIC:

In [15]:
```
earningsA.iloc[-1].nlargest(10).to_frame('Earnings').join(symbols[['ticker','tit
```

Out[15]:

cik	Earnings	ticker	title	sic
320193	63.930	AAPL	Apple Inc.	3571.0
789019	51.310	MSFT	MICROSOFT CORP	7372.0
1067983	35.845	BRK-A	BERKSHIRE HATHAWAY INC	6331.0
1652044	35.713	GOOGL	Alphabet Inc.	7370.0
1326801	29.146	FB	Facebook Inc	7370.0
19617	23.803	JPM	JPMORGAN CHASE & CO	6021.0
50863	20.899	INTC	INTEL CORP	3674.0
104169	19.742	WMT	Walmart Inc.	5331.0
732712	18.308	VZ	VERIZON COMMUNICATIONS INC	4813.0
70858	18.013	BAC	BANK OF AMERICA CORP /DE/	6021.0

All codes: https://www.osha.gov/data/sic-manual

How to select a specific SIC:

In [16]:
```
# Example table:
t = pd.DataFrame({'A':[7372, 6000, 7385],'B':[8000,2200,7372]})
t
```

Out[16]:

	A	B
0	7372	8000
1	6000	2200
2	7385	7372

Get 7372:

In [17]:
```
t[t==7372]
```

Out[17]:

	A	B
0	7372.0	NaN
1	NaN	NaN
2	NaN	7372.0

In [18]:

```
t[t==7372].notnull()
```

Out[18]:

	A	B
0	True	False
1	False	False
2	False	True

And now we can use the True/False table to select from another table with the same rows/columns.
Apply this to our data:

In [20]:

```
codes     = sic
codes[codes==7372]  # Select all values that equal 7372
```

Out[20]:

cik	**1750**	**1800**	**1961**	**2034**	**2098**	**2178**	**2186**	**2488**	**2491**	**2969**	**...**	**1824013**	**182430**
date													
2009-04-15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2009-04-16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2009-04-17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2009-04-20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2009-04-21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	.
2021-03-03	NaN	NaN	7372.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2021-03-04	NaN	NaN	7372.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2021-03-05	NaN	NaN	7372.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2021-03-08	NaN	NaN	7372.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
2021-03-09	NaN	NaN	7372.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na

2996 rows × 12126 columns

In [21]:

```
codes    = sic
industry = codes[codes==7372].notnull()  # Generate True/False table
industry
```

Out[21]:

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1824013	1824301
date													
2009-04-15	False	False	False	False	False	False	False	False	False	False	...	False	False
2009-04-16	False	False	False	False	False	False	False	False	False	False	...	False	False
2009-04-17	False	False	False	False	False	False	False	False	False	False	...	False	False
2009-04-20	False	False	False	False	False	False	False	False	False	False	...	False	False
2009-04-21	False	False	False	False	False	False	False	False	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-03-03	False	False	True	False	False	False	False	False	False	False	...	False	False
2021-03-04	False	False	True	False	False	False	False	False	False	False	...	False	False
2021-03-05	False	False	True	False	False	False	False	False	False	False	...	False	False
2021-03-08	False	False	True	False	False	False	False	False	False	False	...	False	False
2021-03-09	False	False	True	False	False	False	False	False	False	False	...	False	False

2996 rows × 12126 columns

Use this True/False table to select specific cells from Earnings table:

In [22]:

```
earningsA[industry]
```

Out[22]:

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1824013	182
date													
2009-04-15	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	
2009-04-17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1824013	182
date													
2009-04-20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2009-04-21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2021-03-03	NaN	NaN	-0.001425	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2021-03-04	NaN	NaN	-0.001425	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2021-03-05	NaN	NaN	-0.001425	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2021-03-08	NaN	NaN	-0.001425	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2021-03-09	NaN	NaN	-0.001425	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	

2996 rows × 12051 columns

Most recent top 10 in this industry:

In [23]:
```python
earningsA[industry].iloc[-1].nlargest(10).to_frame('Earnings').join(symbols.titl
```

Out[23]:

	Earnings	title	sic
cik			
789019	51.310000	MICROSOFT CORP	7372.0
1341439	10.380000	ORACLE CORP	7372.0
796343	5.260000	ADOBE INC.	7372.0
1108524	3.557000	SALESFORCE.COM, INC.	7372.0
849399	3.258000	NortonLifeLock Inc.	7372.0
718877	2.213000	Activision Blizzard, Inc.	7372.0
896878	1.967000	INTUIT INC	7372.0
1124610	1.588000	VMWARE, INC.	7372.0
712515	1.314000	ELECTRONIC ARTS INC.	7372.0
813672	1.076581	CADENCE DESIGN SYSTEMS INC	7372.0

Another example: get all 4-digit sic codes that start with 73:

In [24]:
```python
t  # Our example table
```

Out[24]:

	A	B

	A	B
0	7372	8000
1	6000	2200
2	7385	7372

In [25]:

```
codes = t.div(100).apply(np.floor)  # divide by 100 to get the first 2 digits an
codes
```

Out[25]:

	A	B
0	73.0	80.0
1	60.0	22.0
2	73.0	73.0

In [26]:

```
codes[codes==73]  # Select 73
```

Out[26]:

	A	B
0	73.0	NaN
1	NaN	NaN
2	73.0	73.0

In [27]:

```
codes[codes==73].notnull()  # True/False table
```

Out[27]:

	A	B
0	True	False
1	False	False
2	True	True

Now apply this to our earnings table:

In [28]:

```
# get all 5800 sic ("Eating And Drinking Places"):
codes    = sic.div(100).apply(np.floor)
industry = codes[codes==58].notnull()

earningsA[industry].iloc[-1].nlargest(10).to_frame('Earnings').join(symbols.titl
```

Out[28]:

	Earnings	title	sic
cik			
63908	4.925500	MCDONALDS CORP	5812.0
1704720	1.261400	Cannae Holdings, Inc.	5810.0
1041061	1.060000	YUM BRANDS INC	5812.0

	Earnings	title	sic
cik			
1618755	0.865000	NaN	5812.0
1673358	0.723000	Yum China Holdings, Inc.	5812.0
829224	0.664800	STARBUCKS CORP	5810.0
1618756	0.560000	Restaurant Brands International Inc.	5812.0
1058090	0.237223	CHIPOTLE MEXICAN GRILL INC	5812.0
1357204	0.220245	NaN	5810.0
30697	0.105631	Wendy's Co	5810.0