SP_Dataset_Framework

April 3, 2019

1 SP_Dataset_Framework

1.0.1 Data

1. Company Financials
2. Company Stock Prices

In [2]:
```
# Mount Google Drive
"""
No need to execute this block when working on local system.
"""
from google.colab import drive
drive.mount("/content/vdrive", force_remount = True)
```

```
Mounted at /content/vdrive
```

In [0]:
```
# Files to process
"""
Modify the locations below as per your directory struture.
"""
root_dir = "/content/vdrive/My Drive/Colab Notebooks/Projects/Bondai/SP 500/data/"
do_file = root_dir + "do_file.csv"
done_file = root_dir + "done_file.csv"
not_done_file = root_dir + "not_done_file.csv"

# Reading the files
import pandas as pd
do_df = pd.read_csv(do_file, header=None, names=["Tickers"])
done_df = pd.read_csv(done_file, header=None, names=["Tickers"])
not_done_df = pd.read_csv(not_done_file, header=None, names=["Tickers"])

do_set = set(do_df["Tickers"].tolist())
done_set = set(done_df["Tickers"].tolist())
not_done_set = set(not_done_df["Tickers"].tolist())
```

In [0]:
```
# URL Paths for Stockrow Website
stockrow_url_paths = {
```

```
        'company': 'https://stockrow.com/api/companies/',
        'annual': {
            'income-statement': '/financials.xlsx?dimension=MRY&section=Income%20Statement&
            'balance-sheet': '/financials.xlsx?dimension=MRY&section=Balance%20Sheet&sort=d
            'cashflow-statement': '/financials.xlsx?dimension=MRY&section=Cash%20Flow&sort=
            'metrics': '/financials.xlsx?dimension=MRY&section=Metrics&sort=desc',
            'growth': '/financials.xlsx?dimension=MRY&section=Growth&sort=desc'
        }
    }

    # Stockrow Downloader
    import requests
    def stockrow_download(ticker):
        income_statement = pd.read_excel(stockrow_url_paths['company'] + ticker + stockrow_
        balance_sheet = pd.read_excel(stockrow_url_paths['company'] + ticker + stockrow_url
        cashflow_statement = pd.read_excel(stockrow_url_paths['company'] + ticker + stockro
        metrics = pd.read_excel(stockrow_url_paths['company'] + ticker + stockrow_url_paths
        growth = pd.read_excel(stockrow_url_paths['company'] + ticker + stockrow_url_paths
        return income_statement, balance_sheet, cashflow_statement, metrics, growth
```

In [0]:
```
    # Modified Get Yahoo Quotes Script by Brad Luicas

    __author__ = "Brad Luicas"
    __copyright__ = "Copyright 2017, Brad Lucas"
    __license__ = "MIT"
    __version__ = "1.0.0"
    __maintainer__ = "Brad Lucas"
    __email__ = "brad@beaconhill.com"
    __status__ = "Production"

    import re
    import sys
    import time
    import datetime
    # import requests


    def split_crumb_store(v):
        return v.split(':')[2].strip('"')


    def find_crumb_store(lines):
        # Looking for
        # ,"CrumbStore":{"crumb":"9q.A4D1c.b9
        for l in lines:
            if re.findall(r'CrumbStore', l):
                return l
        print("Did not find CrumbStore")
```

```
        def get_cookie_value(r):
            return {'B': r.cookies['B']}


        def get_page_data(symbol):
            url = "https://finance.yahoo.com/quote/%s/?p=%s" % (symbol, symbol)
            r = requests.get(url)
            cookie = get_cookie_value(r)

            # Code to replace possible \u002F value
            # ,"CrumbStore":{"crumb":"FWP\u002F5EFll3U"
            # FWP\u002F5EFll3U
            lines = r.content.decode('unicode-escape').strip(). replace('}', '\n')
            return cookie, lines.split('\n')


        def get_cookie_crumb(symbol):
            cookie, lines = get_page_data(symbol)
            crumb = split_crumb_store(find_crumb_store(lines))
            return cookie, crumb


        def get_data(symbol, start_date, end_date, cookie, crumb):
            # filename = '%s.csv' % (symbol)
            url = "https://query1.finance.yahoo.com/v7/finance/download/%s?period1=%s&period2=
            response = requests.get(url, cookies=cookie)
            # with open (filename, 'wb') as handle:
            #     for block in response.iter_content(1024):
            #         handle.write(block)
            return response


        def get_now_epoch():
            # @see https://www.linuxquestions.org/questions/programming-9/python-datetime-to-e
            return int(time.time())


        def download_quotes(symbol):
            start_date = 0
            end_date = get_now_epoch()
            cookie, crumb = get_cookie_crumb(symbol)
            historical_prices = get_data(symbol, start_date, end_date, cookie, crumb)
            return pd.read_csv(io.StringIO(historical_prices.content.decode('utf-8')))
```

In [0]:
```
        import os
        import io
```

```
        def main():
            counter = 0
            total = len(do_set)

            print(do_set)
            for ticker in do_set.copy():
                counter = counter + 1
                try:
                    print("Downloading data for: " + ticker + "(" + str(counter) + "/" + str(to
                    income_statement, balance_sheet, cashflow_statement, metrics, growth = stoc
                    historical_prices = download_quotes(ticker)
                    with pd.ExcelWriter(root_dir + "raw/" + ticker + '.xlsx') as writer:
                        historical_prices.to_excel(writer, sheet_name="historical_prices")
                        balance_sheet.to_excel(writer, sheet_name="balance_sheet")
                        income_statement.to_excel(writer, sheet_name="income_statement")
                        cashflow_statement.to_excel(writer, sheet_name="cashflow_statement")
                        metrics.to_excel(writer, sheet_name="metrics")
                        growth.to_excel(writer, sheet_name="growth")
                except:
                    not_done_set.add(ticker)
                    do_set.discard(ticker)
                    pd.DataFrame(list(not_done_set)).to_csv(not_done_file, header=None, index=F
                    pd.DataFrame(list(do_set)).to_csv(do_file, header=None, index=False)
                    continue


                done_set.add(ticker)
                do_set.discard(ticker)
                pd.DataFrame(list(done_set)).to_csv(done_file, header=None, index=False)
                pd.DataFrame(list(not_done_set)).to_csv(not_done_file, header=None, index=Falsc
                pd.DataFrame(list(do_set)).to_csv(do_file, header=None, index=False)
```

In [0]:
```
main()
```

```
{'ESRX', 'LUK', 'BRK.B', 'NFX', 'ANDV', 'AET', 'WYN', 'MON', 'CSRA', 'SCG', 'XL', 'PX', 'COL',
Downloading data for: ESRX(1/19); Failed(0)
Downloading data for: LUK(2/19); Failed(1)
Downloading data for: BRK.B(3/19); Failed(2)
Downloading data for: NFX(4/19); Failed(3)
Downloading data for: ANDV(5/19); Failed(4)
Downloading data for: GGP(6/19); Failed(5)
Downloading data for: UNP(7/19); Failed(6)


/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:42: DeprecationWarning: invalid es


Downloading data for: AET(8/19); Failed(6)
Downloading data for: WYN(9/19); Failed(7)
```

```
Downloading data for: MON(10/19); Failed(8)
Downloading data for: CSRA(11/19); Failed(9)
Downloading data for: SCG(12/19); Failed(10)
Downloading data for: XL(13/19); Failed(11)
Downloading data for: PX(14/19); Failed(12)
Downloading data for: COL(15/19); Failed(13)
Downloading data for: SYK(16/19); Failed(14)
Downloading data for: HRL(17/19); Failed(14)
Downloading data for: BF.B(18/19); Failed(14)
Downloading data for: KORS(19/19); Failed(15)
```