In [1]:
```
import yfinance as yf
import pandas as pd
import numpy as np
import datetime as dt

import time
import random

import pymysql
import mysql.connector
from sqlalchemy import create_engine

pd.options.mode.chained_assignment = None
```

In [2]:
```
print("yfinance v:",yf.__version__)
print("pandas v:",pd.__version__)
print("numpy v:",np.__version__)
```

```
yfinance v: 0.1.54
pandas v: 1.2.3
numpy v: 1.19.2
```

In [3]:
```
user = 'flatiron'
pwd = 'flatiron'
host = '127.0.0.1'
shporty = '3306'
db = 'stock'
connector =
'mysql+mysqlconnector://'+user+':'+pwd+'@'+host+':'+shporty+'/'+db
engine = create_engine(connector, echo=False)
```

In [4]:
```
# string, listOfDates, bool -> pd.DataFrame
# returns a dataframe of calls or puts for list of expiry dates
#    for a given ticker symbol string
def c_p_df(exp_dates, is_call):

    df = pd.DataFrame()

    columns_to_drop =
['contractSymbol','bid','ask','change','percentChange','openInterest
    columns_maybe_drop = ['contractSize',
'lastTradeDate','currency']
    both_dropped = columns_to_drop + columns_maybe_drop

    for exp in exp_dates:
        if is_call:
            call = stock.option_chain(exp).calls
            call['ticker'] = ticker # WILL HAVE TO CHANGE THIS
WHEN MORE TICKERS ADDED
            call['expDate'] = np.array(exp)
            call['mineDate'] = dt.datetime.today().__format__('
%Y-%m-%d')
            df = pd.concat([df, call])
            df.drop(both_dropped, axis=1, inplace=True)
        else:
            put = stock.option_chain(exp).puts
            put['ticker'] = ticker # WILL HAVE TO CHANGE THIS WHEN
MORE TICKERS ADDED
            put['expDate'] = np.array(exp)
            put['mineDate'] = dt.datetime.today().__format__('
%Y-%m-%d')
            df = pd.concat([df, put])
            df.drop(both_dropped, axis=1, inplace=True)
    return df
```

In [5]:
```
ticker_list = pd.read_csv('/home/steve/documents/flatIron
/fIProject/Stocks/master_list_no_empty.csv')['ticker']
```

In [ ]:

```
for tick in ticker_list:
    #although currently unnecessary can be moved outside
    #loop to build one complete list of all c/p
    all_calls = pd.DataFrame()
    all_puts = pd.DataFrame()
    c_and_p = pd.DataFrame()
    try:
        #generates call/put options chains for each ticker
        ticker = tick
        stock = yf.Ticker(ticker)
        expiry_dates = stock.options

        # populate calls/puts dataframes with c/p data
        all_calls = pd.concat([all_calls, c_p_df(expiry_dates,
is_call=True)])
        all_puts = pd.concat([all_puts, c_p_df(expiry_dates,
is_call=False)])

        # add col call to both dfs for join
        all_calls['call'] = True
        all_puts['call'] = False

        # join dfs - note: this could all be done more efficiently
in less steps
        c_and_p = pd.concat([all_calls, all_puts])

        # appends calls and puts to db
        c_and_p.to_sql(name='options', con=engine,
if_exists='append', index=False)

        # for separate tables - not efficient, much redundancy
#         all_calls.to_sql(name='calls', con=engine, if_exists =
'append', index=False)
#         all_puts.to_sql(name='puts', con=engine, if_exists =
'append', index=False)
        time.sleep(random.randrange(1,4))
    except:
        pass
```

```
In [ ]:
```