Imports

In [1]:
```
import requests
from bs4 import BeautifulSoup
import json
from collections import OrderedDict
from datetime import datetime
from dateutil.relativedelta import relativedelta
from __future__ import print_function
import mysql.connector
from mysql.connector import errorcode
from timeit import default_timer as timer
import time
import copy
import numpy_financial
```

In [2]:
```
# Update_or_Delete = input('Do you wish to Update or Delete/Restart the Datebase?')
# Confirmation = input(f'Are you sure you wish to {Update_or_Delete}')
# Computer_Shutdown = input(Do you wish to shut off the computer after running this fil

# if Confirmation != 'Yes':
#     x=1/0
```

Setting the conditons of what you want this file to preform.

Update: Will update only update the ticker symbols when they are behind on an 10-Q report being realeased

Delete: Will Delete the current Database and recreate it from nothing. This is recommend only if you you change the format of the Database or Starting new

Computer_Shutdown: Leaving this with Yes well shut off your computer one the file is done running

In [3]:
```
# Update or Delete
Update_or_Delete = 'Update'
# Yes or No
Computer_Shutdown = 'No'
```

Importing Functions from another file

In [6]:
```
%run ./Functions_Folder/Functions.ipynb
%run ./Functions_Folder/Updating_the_Database.ipynb
%run ./Functions_Folder/Personal_Folder/Personal_Information.ipynb
```

```
Creating table Company_Info: already exists.
Creating table CalcTable: already exists.
Creating table Yahoo_Forecast: already exists.
Creating table futureEarningsDate: already exists.
```

Connecting to the MySQL Database

In [4]:
```
cnx = mysql.connector.connect(user=MySQL_username, password = MySQL_password)
cursor = cnx.cursor()
```

If Delete was selected above this will delete the database

```
In [5]: if Update_or_Delete == 'Delete':
            # Drop Database
            try:
                query = ("drop database sec")
                cursor.execute(query)
                print('Successfully deleted previous SEC Database')
            except Exception as err:
                print(err)
```

If Delete was selected above this will recreate the format of the database

```
In [7]: if Update_or_Delete == 'Delete':
            cnx = mysql.connector.connect(user=MySQL_username, password = MySQL_password, datab
            cursor = cnx.cursor()
            # #Creating Tables
            Table_List = ['Revenue', 'Earnings', 'Earnings_Per_Share', 'Levered_Free_Cash_Flow'
                          'Long_term_Liabilites', 'Total_LTL_per_LFCF', 'Market_Cap', 'Div_per_
                          'Total_Debt_to_Equity']
            for Table in Table_List:
                Creating_a_table(Table)
            # All Stocks
            Master_List = [ 'PORT' , 'MULN' , 'NECA' , 'TSLA' , 'BBRW' , 'HCMC' , 'KEGS' , 'F'
```

If Update was selected this will find what companies are in need of an update

```
In [8]: if Update_or_Delete == 'Update':
            Master_List = Companies_that_need_updating()
            print(len(Master_List))
```

```
232
```

Getting into the Main Code File

This will collect, clean, and store all the data need for the database from ThinkOrSwim API, Yahoo, and SEC

```
In [9]: today = datetime.today()
        Current_Year = int(today.year)

        start = timer()
        Tick_Count =0
        error_grouping = {}

        x =1
        for Ticker in Master_List:

            Tick_Count = Tick_Count + 1
            Remaining_Count = round(len(Master_List)-Tick_Count,0)
            starting_kill_timer = timer()
            Company_collected = "No"
            while (timer()- starting_kill_timer)<15 and Company_collected == "No":

                #  Deleting the Ticker if requested to update
                if Update_or_Delete == 'Update':
                    Deleting_Company(Ticker)

                try:
```

```
                # Connecting to SEC
                Ticker = Ticker.replace('/','-')
                try:
                    CIK = CIK_Finder(Ticker)
                except:
                    continue

                ## Basic Company Information
                url = 'https://data.sec.gov/submissions/CIK'+ CIK +'.json'
                try:
                    soup = SEC_link(url)
                except:
                    continue
                sic = soup['sic']
                if sic == '':
                    continue
                sic_Description = soup['sicDescription']
                ein = soup['ein']

                ## Fundamental Data
                url = 'https://data.sec.gov/api/xbrl/companyfacts/CIK'+ CIK + '.json'
                try:
                    soup = SEC_link(url)
                except:
                    continue

                # Collecting Data
                ## Data Directly From the SEC
                ### Equity Info
                Split_Hist = Split(1)
                Share_Count = Shares(['WeightedAverageNumberOfDilutedSharesOutstanding'])
                Shares_Outstanding = Shares(['CommonStockSharesOutstanding', 'SharesOutstan

                ### Storing Data from Yahoo's Data Analytics
                Yahoo_data = YAHOO(Ticker)
                nextYR_growth = Yahoo_data[0]
                fiveYR_growth = Yahoo_data[1]
                Ticker = Ticker.replace('-','.')
                price = ThinkORSwim(Ticker)
                Current_Earnings = Yahoo_data[3]
                Future_Earnings = Yahoo_data[4]
                Rev_gr = Yahoo_data[5]
                Sector = Yahoo_data[6]
                Industy = Yahoo_data[7]
                try :
                    Current_PE = round(price/Current_Earnings,2)
                except:
                    Current_PE = None
                Market_Cap = MC_fucntion(Shares_Outstanding, price)

                ### Importing
                importing_Company_data('company_info')
                importing_Yahoo_Forecast(nextYR_growth, fiveYR_growth, price , Current_Earn
                Original_importing_data('Market_Cap', Market_Cap)

                ### Clearing data, this is so values dont overlap in Yahoo's table
                YahooMySQL = [nextYR_growth, fiveYR_growth, price , Current_Earnings, Futur
                for items in YahooMySQL:
                    items = None
```

```
### Income Statement
Revenue = USD_gaap(['Revenues', 'SalesRevenueNet', 'RevenueFromContractWith
RevenueTTM = TTM(Revenue[1])

Earnings = USD_gaap(['NetIncomeLossAvailableToCommonStockholdersBasic','Net
EarningsTTM = TTM(Earnings[1])

### Balance Sheet
Cash = USD_gaap(['CashAndCashEquivalentsAtCarryingValue', 'CashCashEquivale
Property_Equipment = USD_gaap(['PropertyPlantAndEquipmentNet', 'PaymentsToA
Current_Debt = USD_gaap(['LongTermDebtCurrent', 'DebtCurrent', 'NotesPayabl
Long_Term_Debt = USD_gaap(['LongTermDebtNoncurrent', 'LongTermDebt', 'LongT
Total_Debt =USD_gaap(['DebtAndCapitalLeaseObligations', 'DebtandFinanceLeas
Total_current_Liabilites = USD_gaap(['LiabilitiesCurrent'])[1]
Total_Long_Term_Liabilities = USD_gaap(['LiabilitiesNoncurrent'])[1]
Total_Liabilities = USD_gaap(['Liabilities'])[1]
Equity = USD_gaap(['StockholdersEquityIncludingPortionAttributableToNoncont
Liabilities_And_StockholdersEquity = USD_gaap(['LiabilitiesAndStockholdersE

### Cashflow Statement
Operating_Cash_Flow = USD_gaap(['NetCashProvidedByUsedInOperatingActivities
Div_Cash = USD_gaap(['Dividends', 'PaymentsOfDividendsCommonStock', 'Divide
Div_per_Share = USD_gaap_per_shares(['CommonStockDividendsPerShareDeclared'

## Auto Fill Missing Numbers with Calculations
for num in range(3):
    ### Balance Sheet
    Calc1_Current_Debt = Subtraction(Total_Debt, Long_Term_Debt)
    Current_Debt = Append(Current_Debt, Calc1_Current_Debt)

    Calc1_Long_Term_Debt = Subtraction(Total_Debt, Current_Debt)
    Long_Term_Debt = Append(Long_Term_Debt, Calc1_Long_Term_Debt)

    Calc1_Total_Debt = Addition(Current_Debt, Long_Term_Debt)
    Total_Debt = Append(Total_Debt, Calc1_Total_Debt)

    Calc1_Total_current_Liabilites = Subtraction(Total_Liabilities, Total_L
    Total_current_Liabilites = Append(Total_current_Liabilites, Calc1_Total

    Calc1_Total_Long_Term_Liabilities = Subtraction(Total_Liabilities, Tota
    Calc2_Total_Long_Term_Liabilities = Addition(Equity, Total_current_Liab
    Calc2_Total_Long_Term_Liabilities = Subtraction(Liabilities_And_Stockho
    Appended1 = Append(Calc1_Total_Long_Term_Liabilities,Calc2_Total_Long_T
    Total_Long_Term_Liabilities = Append(Total_Long_Term_Liabilities,Append

    Calc1_Total_Liabilities = Subtraction(Liabilities_And_StockholdersEquit
    Total_Liabilities = Append(Total_Liabilities, Calc1_Total_Liabilities)

    Calc1_Equity = Subtraction(Liabilities_And_StockholdersEquity, Total_Li
    Equity = Append(Equity, Calc1_Equity)

### Cash Flow Statement
Calc1Div_per_Share = Division(Div_Cash,Shares_Outstanding)
Div_per_Share = Append(Div_per_Share,Calc1Div_per_Share)


## Calculated Values numbers the SEC does not Offer
### Gaap
Earnings_per_share = Division(Earnings[0],Share_Count)
```

```
        ### Non-gaap
        Free_Cash_Flow = Subtraction(Operating_Cash_Flow[0],Property_Equipment)
        Levered_Free_Cash_Flow = Subtraction(Free_Cash_Flow,Current_Debt)
        Total_LTL_per_LFCF = Division_Flipped(Total_Long_Term_Liabilities, Levered_

        CashperTotalDebt = Division(Cash,Total_Debt)
        Total_Debt_to_Equity = Division(Total_Debt, Equity)


        #Storing Data to from SEC
        ## Income Statement
        importing_data('Revenue', Revenue[0])
        importing_data('Revenue', RevenueTTM)
        Combination_of_Calc_Func('Revenue', Revenue[0], ['YoY', 'CAGR', 'YoYAvg', '
        Combination_of_Calc_Func('Revenue', RevenueTTM, ['YoY', 'CAGR', 'YoYAvg', '

        importing_data('Earnings', Earnings[0])
        importing_data('Earnings', EarningsTTM)
        Combination_of_Calc_Func('Earnings', Earnings[0], ['YoY', 'CAGR', 'YoYAvg',
        Combination_of_Calc_Func('Earnings', EarningsTTM, ['YoY', 'CAGR', 'YoYAvg',

        ## Balance Sheet
        importing_data('Cash', Cash)
        Combination_of_Calc_Func('Cash', Cash, ['YoY', 'CAGR', 'YoYAvg', 'Avg'])

        importing_data('Total_Debt', Total_Debt)
        Combination_of_Calc_Func('Total_Debt', Total_Debt, ['YoY', 'CAGR', 'YoYAvg'

        importing_data('Long_term_Liabilites', Total_Long_Term_Liabilities)
        Combination_of_Calc_Func('Long_term_Liabilites', Total_Long_Term_Liabilitie

        ## Calculated Functions
        importing_data('Earnings_Per_Share', Earnings_per_share)
        Combination_of_Calc_Func('Earnings_Per_Share', Earnings_per_share, ['YoY',

        importing_data('Div_per_Share', Div_per_Share)
        Combination_of_Calc_Func('Div_per_Share', Div_per_Share, ['YoY', 'CAGR', 'Y

        importing_data('Levered_Free_Cash_Flow', Levered_Free_Cash_Flow)
        Combination_of_Calc_Func('Levered_Free_Cash_Flow', Levered_Free_Cash_Flow,

        importing_data('Total_LTL_per_LFCF', Total_LTL_per_LFCF)
        Combination_of_Calc_Func('Total_LTL_per_LFCF', Total_LTL_per_LFCF, ['YoY',

        importing_data('CashperTotalDebt', CashperTotalDebt)
        Combination_of_Calc_Func('CashperTotalDebt', CashperTotalDebt, ['YoY', 'CAG

        importing_data('Total_Debt_to_Equity', Total_Debt_to_Equity)
        Combination_of_Calc_Func('Total_Debt_to_Equity', Total_Debt_to_Equity, ['Yo

        end = timer()
        Time_remaining = round(((end-start)/Tick_Count)*Remaining_Count,0)
        Time_remaining_str = '{} seconds'.format(Time_remaining)
        if Time_remaining>60:
            Time_remaining = round(Time_remaining/60,1)
            Time_remaining_str = '{} minutes'.format(Time_remaining)
            if Time_remaining>60:
                Time_remaining = round(Time_remaining/60,1)
                Time_remaining_str = '{} hours'.format(Time_remaining)
                #Error Code String
                count = 0
                List_str = ['Revenue[0]', 'Revenue[1]', 'Earnings[0]', 'Earnings[1]', 'Oper
                            'Operating_Cash_Flow[1]', 'Property_Equipment', 'Current_Debt',
                            'Total_current_Liabilites', 'Total_Long_Term_Liabilities', 'Tot
                            'Liabilities_And_StockholdersEquity', 'Cash', 'Shares_Outstandi
                List = [Revenue[0], Revenue[1], Earnings[0], Earnings[1], Operating_Cash_Fl
                        Operating_Cash_Flow[1], Property_Equipment, Current_Debt, Long_Term
                        Total_current_Liabilites, Total_Long_Term_Liabilities, Total_Liabil
                        Liabilities_And_StockholdersEquity, Cash, Shares_Outstanding, Marke

                # Code for calculating the future date for earnings
                Calander_Data = dataForFurtureEarnings(List)
                importing_data_futureEarningsDate(Calander_Data)

                string = 'We have collected {} companies, with {} remaining. Estimated rema
                print(string, end='\r')

                for table in List:
                    count= count +1
                    x = 1
                    for items in table:
                        if table[items] != {}:
                            x = 2
                    if x == 1:
                        x=1/0
                Company_collected = "Yes"
            except ZeroDivisionError:
                try:
                    error_grouping[List_str[count-1]][sic_Description].append(Ticker)
                    Company_collected = "Yes"
                except:
                    try:
                        error_grouping[List_str[count-1]][sic_Description] = []
                        error_grouping[List_str[count-1]][sic_Description].append(Ticker)
                        Company_collected = "Yes"
                    except:
                        error_grouping[List_str[count-1]] = {}
                        error_grouping[List_str[count-1]][sic_Description] = []
                        error_grouping[List_str[count-1]][sic_Description].append(Ticker)
                        Company_collected = "Yes"
            except mysql.connector.Error:
                # Filters out duplicate errors
                x=1
                Company_collected = "Yes"
            except Exception as err:
                print("An error of {} has occured in the company {}
                Company_collected = "Yes"
error_grouping
```

```
We have collected 232 companies, with 0 remaining. Estimated remaining time 0.0 seconds
MXSG
```

Out[9]:

```
{'Long_Term_Debt': {'Services-Miscellaneous Amusement & Recreation': ['PRCX'],
  'Beverages': ['TGGI'],
  'Services-Computer Programming Services': ['EHVVF'],
  'Services-Prepackaged Software': ['DUSYF'],
  'Services-Advertising': ['BLIS'],
  'Services-Computer Integrated Systems Design': ['KBNT'],
  'Services-Business Services, NEC': ['LGIQ'],
  'Pharmaceutical Preparations': ['UPC'],
  'Finance Services': ['RKFL'],
  'Medicinal Chemicals & Botanical Products': ['BGXX'],
  'Real Estate': ['GMPW', 'NIHK']},
 'Current_Debt': {'Fire, Marine & Casualty Insurance': ['AFHIF'],
  'Electric Services': ['APSI'],
  'Blank Checks': ['GTVI'],
  'Services-Prepackaged Software': ['CYCA'],
  'Computer Peripheral Equipment, NEC': ['WETH'],
  'Security & Commodity Brokers, Dealers, Exchanges & Services': ['TOP'],
  'Metal Mining': ['LTUM']},
 'Revenue[0]': {'Metal Mining': ['MLYF'],
  'Pharmaceutical Preparations': ['SIOX', 'ONCR', 'BWV', 'APGN'],
  'Industrial Organic Chemicals': ['WNDW'],
  'Blank Checks': ['GPAC', 'SGII', 'USCT', 'NSTB', 'TBCP'],
  'Biological Products, (No Diagnostic Substances)': ['MYMX', 'ALNAQ'],
  'Services-Computer Integrated Systems Design': ['EMBK'],
  'Savings Institution, Federally Chartered': ['CNNB'],
  'Motor Vehicles & Passenger Car Bodies': ['FFIE'],
  'Gold and Silver Ores': ['GKIN']},
 'Property_Equipment': {'Crude Petroleum & Natural Gas': ['PARG'],
  'Pharmaceutical Preparations': ['QBIO'],
  'Services-Amusement & Recreation Services': ['AFOM'],
  'Finance Services': ['WDLF'],
  'Services-Prepackaged Software': ['ITOX'],
  'Services-Computer Processing & Data Preparation': ['SDIG']},
 'Market_Cap': {'Agricultural Production-Crops': ['APPH'],
  'Pharmaceutical Preparations': ['NBRV'],
  'Communications Equipment, NEC': ['WTT'],
  'Crude Petroleum & Natural Gas': ['VKIN']},
 'Total_current_Liabilites': {'Fire, Marine & Casualty Insurance': ['UIHC']},
 'Shares_Outstanding': {'Services-Prepackaged Software': ['MSGM']},
 'Cash': {'Radiotelephone Communications': ['GZIC'],
  'Services-Business Services, NEC': ['EDXC'],
  'Fabricated Structural Metal Products': ['FATH'],
  'Retail-Catalog & Mail-Order Houses': ['PIK'],
  'Wholesale-Groceries, General Line': ['PFGC'],
  'Cutlery, Handtools & General Hardware': ['TBLT']}}
```

In [10]:

```python
for labels in error_grouping:
    print(labels)
    for sic_group in error_grouping[labels]:
        print(sic_group)
        print(error_grouping[labels][sic_group])
    print('')
```

```
Long_Term_Debt
Services-Miscellaneous Amusement & Recreation
['PRCX']
Beverages
['TGGI']
Services-Computer Programming Services
['EHVVF']
Services-Prepackaged Software
['DUSYF']
Services-Advertising
['BLIS']
Services-Computer Integrated Systems Design
['KBNT']
Services-Business Services, NEC
['LGIQ']
Pharmaceutical Preparations
['UPC']
Finance Services
['RKFL']
Medicinal Chemicals & Botanical Products
['BGXX']
```

```
Real Estate
['GMPW', 'NIHK']

Current_Debt
Fire, Marine & Casualty Insurance
['AFHIF']
Electric Services
['APSI']
Blank Checks
['GTVI']
Services-Prepackaged Software
['CYCA']
Computer Peripheral Equipment, NEC
['WETH']
Security & Commodity Brokers, Dealers, Exchanges & Services
['TOP']
Metal Mining
['LTUM']

Revenue[0]
Metal Mining
['MLYF']
Pharmaceutical Preparations
['SIOX', 'ONCR', 'BWV', 'APGN']
Industrial Organic Chemicals
['WNDW']
Blank Checks
['GPAC', 'SGII', 'USCT', 'NSTB', 'TBCP']
Biological Products, (No Diagnostic Substances)
['MYMX', 'ALNAQ']
Services-Computer Integrated Systems Design
['EMBK']
Savings Institution, Federally Chartered
['CNNB']
Motor Vehicles & Passenger Car Bodies
['FFIE']
Gold and Silver Ores
['GKIN']

Property_Equipment
Crude Petroleum & Natural Gas
['PARG']
Pharmaceutical Preparations
['QBIO']
Services-Amusement & Recreation Services
['AFOM']
Finance Services
['WDLF']
Services-Prepackaged Software
['ITOX']
Services-Computer Processing & Data Preparation
['SDIG']

Market_Cap
Agricultural Production-Crops
['APPH']
Pharmaceutical Preparations
['NBRV']
Communications Equipment, NEC
['WTT']
Crude Petroleum & Natural Gas
['VKIN']

Total_current_Liabilites
Fire, Marine & Casualty Insurance
```

```
['UIHC']

Shares_Outstanding
Services-Prepackaged Software
['MSGM']

Cash
Radiotelephone Communications
['GZIC']
Services-Business Services, NEC
['EDXC']
Fabricated Structural Metal Products
['FATH']
Retail-Catalog & Mail-Order Houses
['PIK']
Wholesale-Groceries, General Line
['PFGC']
Cutlery, Handtools & General Hardware
['TBLT']
```

This will shut down your computer

In [11]:

```
if Computer_Shutdown == 'Yes':
    import os
    def shutdown():
        return os.system("shutdown /s /t 1")
    shutdown()
```

In [12]:

```
cursor.close()
cnx.close()
```

In [13]:

```
# Label = 'CommonStockDividendsPerShareDeclared'
# for items in soup['facts']['us-gaap'][Label]['units']['USD/shares']:
# for items in soup['facts']['us-gaap']:

#     print(items)
```