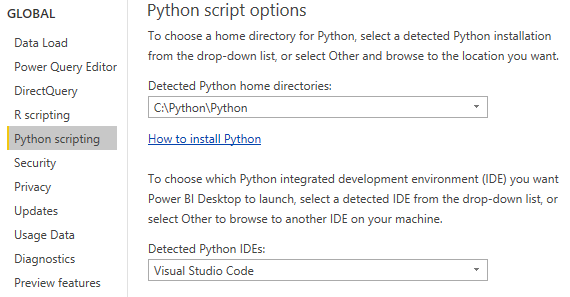
Python Stockinfo from API

# The demo report’s goal to show examples of using Python to connect to a datasource and to use the Python visuals feature of the Microsoft Power BI reporting tool.

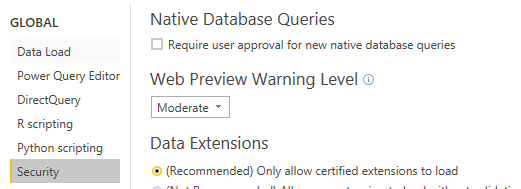
We are going to download historical data of different stocks from an API and visualize the data using built-in Power BI (PBI) and Python visuals.

# It is a step-by-step tutorial to create the report and also to explain my methods.

## First you need to enable Python scripts in the Preview features in the options and set your Python home directories.



Normally I would recommend requiring user approval for new native database queries but this time we should disable it.



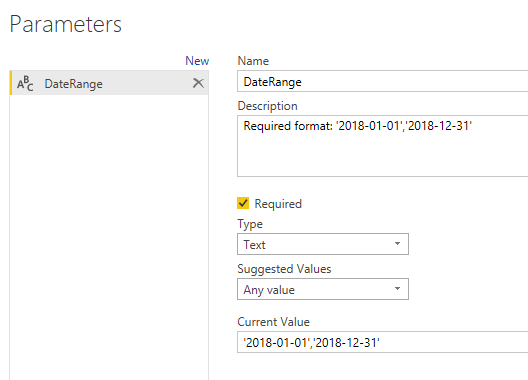
## Date Table

First we create the Date table in the query editor. Creating a Date table this way has advantages over doing it in DAX because DAX calculated columns are using your CPU resources.

I created the Date table from a script my colleague found on the internet. It’s a great script and I would give credit to the author if I knew the source. If you can identify the blog post by the code let me know and I will reference that here.

The Data script can be copied into and edited in the advanced editor after creating a blank query. The script generates a date table with the usually used calculated columns from a starting date and ending date argument.  
By editing the script you can rename the columns, change the starting day of the week from Monday to Sunday, change the culture from „en-us” to „hu” or other cultures.

## Create a Parameter for the Date Range of the historical data about different stocks.



At other cases it is useful to create a query for the suggested values, that way the user does not have to type in the argument. Our Parameter is about Dates so it is pretty straightforward.

## Source Python query

You can read here about the iexfinance package <https://pypi.org/project/iexfinance/>   
The example code below is from the documentation. By using the parameter created in step 3 we avoid hardcode-ing it.

from iexfinance import get\_historical\_data  
from datetime import datetime

start = datetime(2017, 2, 9)  
end = datetime(2017, 5, 24)

df = get\_historical\_data("AAPL", start=start, end=end, output\_format='pandas')

From the PBI Power Query editor window’s New source menu create a Python script from the other group.

from iexfinance import get\_historical\_data # Install the necessary packages on your machine I recommend the Anaconda virtual environment manager

from datetime import datetime

import pandas as pd

data = get\_historical\_data('AAPL', " & DateRange & ", output\_format='pandas') # we are using the parameter to get historical data of Apple using the ticker as argument.

data['Date']=data.index # the query editor does not transform the index of the Dataframe into a column so I added it as a column

#Normally we do not need to download the whole date range at every refresh, I could save the past data as a csv file and only get the dates #that are not in the csv from the API.

data['Returns']=data['close'].pct\_change() # calculated column that computes the daily change in the close column’s values. Difference between the current day’s closing price the closing price the day before divided by the closing price of the day before. For January 2 the value is the result of (170.2745-170.3042)/170.3042

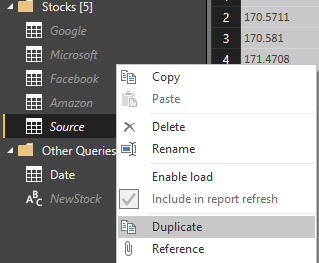
After this point, run the Python query!

The next steps of the query are done in the PBI query editor.

Expanded the values of the data DataFrame, added a custom column to contain the StockName,

and removed a column [’Name’] that refers to the name of the DataFrame.

Great! We have our base query. We can copy and alter it to get data about other stocks as well.



Right click on Duplicate query!

From the new query you can create the sources Amazon, Facebook, Google and Microsoft.

You only have to change 3 things: rename the query from Source(2) to Microsoft for example, change the custom column to be equal to the stock’s name and change the ticker in the Python query.

Click on the Enable load so these queries won’t show up on the report page.

## Delay the queries

This step might be unnecessary this time but useful in other cases. We are accessing this free API without an API key so if we are asking for too much data, it could fail to execute.

By adding the time.sleep line I can manage the execution order and the execution time of the different queries.

from iexfinance import get\_historical\_data

from datetime import datetime

import pandas as pd

*import os*

*import time*

*import sys*

*time.sleep(20)*

data = get\_historical\_data('AMZN', " & DateRange & ", output\_format='pandas')

data['Date']=data.index

data['Returns']=data['close'].pct\_change()

## Create the staging query named StockInfo

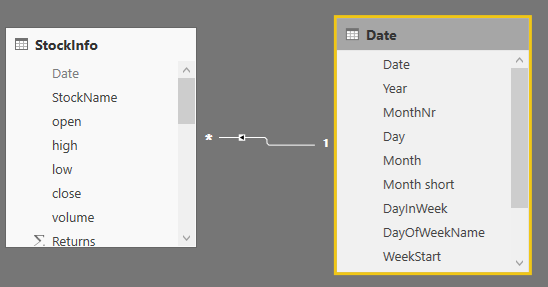
* In the first step the query is referring to the Source query that contains the Apple data.

*let*

*Source = Source,*

* The second step is appending the Microsoft query.
* The 3rd is appending the other 3 queries in one step. Yes, we could have appended all of them at once.
* The changed type with locale steps are changing the numerical values from using point as a decimal separator to use the comma as we do it in Europe. Here the locale value „en-us” is the value we change from to the regional locale of the computer running the Power BI Desktop.
* I changed the type of Date to Date and StockName to text but it is probably not necessary. Generally you should only choose column types if it is necessary and preferably as the last step.

## Create the relationship



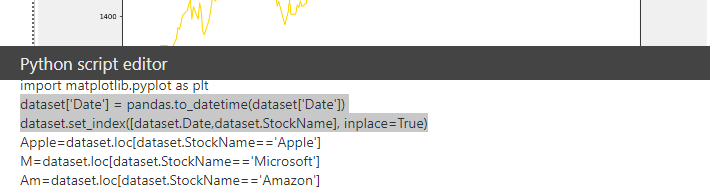
This way we can analyze StockInfor by all the useful columns of the Date table.

## Python visuals

You can find the source code of the Python visuals by clicking on them and opening the Python script editor window.

The DataFrame used by the Python visuals consists of column you drag to the values field list. You can refer to a column that is not in the field list.

You have to manually add an index to the dataframe when you create a time series. Let’s see the code managing that in the visual on the Time Series Linear Python page:



dataset['Date'] = pandas.to\_datetime(dataset['Date']) # First convert the column to Datetime

dataset.set\_index([dataset.Date,dataset.StockName], inplace=True) # then set it as the index. Only the Date does not identify a row, therefore I added the StockName column to have unique indexing.

## DAX examples

The DAX pages shows some basic DAX examples. You can see the DAX codes behind the measures by clicking on them.

