

## Executive Summary

I spend a significant amount of time analyzing data for equity securities in the US market. I will also be doing equity research full-time for an investment bank after graduation. As such, I am constantly going online to look up current and historical data for those securities. This has become a time-consuming task for me, so I wanted to use VBA to program something that would allow me to simply enter the ticker symbol, tell it what analysis I wanted to perform, and have the program run the analysis for me.

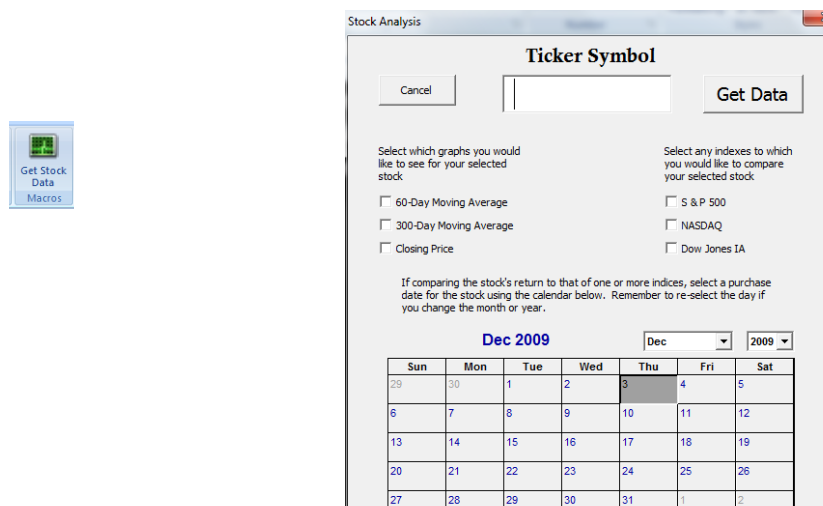
In order to be useful, the program would have to allow the user to specify a past date, and have the program look at historical prices for the security itself, and for the benchmark indexes against which I wanted to compare the security. The program, therefore, retrieves historical data for the Dow Jones Industrial Average, the S&P 500, and the Nasdaq for the indicated date. It also captures current data for those indexes as well as the selected security and calculates the annual return on each. It then compares the annual return of the security to that of each of the benchmark indexes, and calculates the *alpha* (the difference between the annual gain of the security and that of each benchmark index) of the selected security, and then displays all of the results to the user.

In addition, the program calculates the stock price's 300-day moving average, the 60-day moving average, and the actual stock price of the selected security (for the last 100 days) and graphs those three trend lines on a single graph that is then displayed to the user.

In an effort to facilitate stock comparisons, the program creates a new sheet for each security allowing the user to simply move between sheets (if the program is run multiple times) to look at the results for different securities.

## Implementation

When the user first opens the workbook containing the program, instructions are shown at the top of the sheet instructing the user to click on a button whose caption reads "Get Stock Data", that was added to the home ribbon. This button initializes the user form that will collect all necessary data from the user in order to perform the analysis. Both the button and the user form are shown below:



The image shows two parts: a button on the Excel ribbon and a user form titled 'Stock Analysis'.

**Get Stock Data Button:** A small blue button with a green icon and the text 'Get Stock Data' and 'Macros' below it.

**Stock Analysis Form:** A dialog box with the following elements:

- Ticker Symbol:** A text input field with 'Cancel' and 'Get Data' buttons.
- Select which graphs you would like to see for your selected stock:**
  - ☐ 60-Day Moving Average
  - ☐ 300-Day Moving Average
  - ☐ Closing Price
- Select any indexes to which you would like to compare your selected stock:**
  - ☐ S & P 500
  - ☐ NASDAQ
  - ☐ Dow Jones IA
- Calendar:** A calendar for December 2009. The date '3' (Thursday) is selected.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

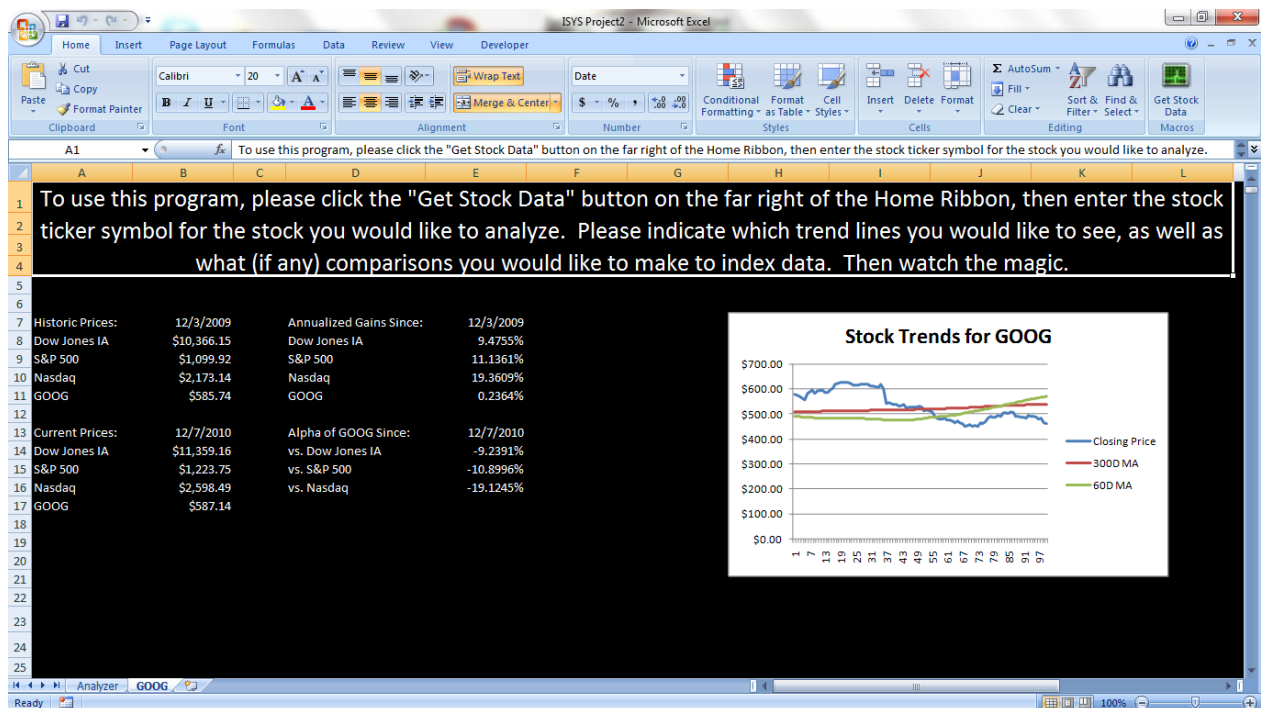
When the userform is shown, the user is instructed how to use it. Namely, it instructs the user to enter a ticker symbol, and then select which analyses the user would like performed by checking the corresponding checkboxes for those analyses.

When the user clicks “Get Data”, several actions are performed by the various macros programmed in the workbook. The program:

1. Checks to see if there is room for another sheet in the workbook. If the maximum number of worksheets (256) already exists in the workbook, the user is instructed to delete at least one so that the program can create the sheet necessary to display the analysis.
2. Checks to see if a ticker symbol was actually entered. If not, the user is kindly informed of their epic failure and asked to enter a ticker symbol and try again.
3. Checks to see if index comparisons were wanted, and if so runs macros that do the following:
  - a. Takes the selected purchase date from the user form
  - b. Checks to make sure the purchase date is in the past
    - i. If not, the user is kindly reminded that the program cannot predict stock prices for a future date, and is asked to choose a different date and try again. Code is then terminated.
  - c. Uses the date and ticker symbol to create the proper URL at which to find the necessary data at Yahoo Finance.
  - d. Collects data from Yahoo and copies it to a sheet that is “very hidden”, meaning that the user cannot unhide the sheet without doing so through the VBA code.
  - e. Checks to make sure that historical data exists for the selected security at the selected date by searching for the purchase date in the data. If it isn’t found, then it returns a message box informing the user that historical data is unavailable, but that any desired charts would still be created using more current data.
  - f. Searches the data returned from Yahoo and finds current price data for the security and indexes on the desired date and copies it to the worksheet
  - g. Finds the necessary historical price data and copies it to another “veryhidden” worksheet.
  - h. Calculates gains by comparing price differences between historic and current data, and divides that difference by the number of years between the selected purchase date, and the current date. This returns annual gains (or losses) and places that data on the worksheet.
  - i. Calculates the difference in gains (or losses) of the security and those of the benchmark indexes and displays that difference under the *alpha* heading on the worksheet.
4. Checks to see if the user opted for chart data, and if so, what chart data was requested. If chart data was requested, the following actions are performed:
  - a. The appropriate URL is created the same way as for the index comparisons, but instead of retrieving data for only one historic date, the program captures all data for the last 400 days, copies it to a hidden sheet, and manipulates that data through a sort so that it is properly ordered for a chart.

- b. The program calculates the moving average of the stock price for a 300-day moving average, a 60-day moving average, and the daily closing price. These three values are listed in separate columns so as to facilitate the creation of a line chart.
  - c. A line chart is then created plotting all of the requested averages. A copy of that chart is generated as a .jpeg image (so that the image created isn't subject to change if the cells containing the data were inadvertently altered) and displays that picture on the sheet.
  - d. Deletes the original chart and associated data.
5. Copies the newly altered sheet to a new sheet and renames that sheet to match the stock ticker whose data is shown on it, then deletes all of that data from the original "Analyzer" sheet.

The final outcome of an analysis of Google stock, if purchased on 12/3/2009, would look like this:



## Learning and Conceptual Difficulties

This project presented an opportunity for significant learning. There were some difficulties along the way, and working through (or sometimes around) those difficulties gave me an opportunity to better explore the capabilities of VBA.

Creating the userform was easily the most difficult part. In order to create a valid URL, I had to dissect both the purchase date and the current date into separate day, month, and year segments so that those numbers could be inserted into the appropriate places in the URL. I originally intended to simply use list boxes that would auto-populate with the names of the months, the number of days, and the desired year. After a couple of hours struggling to find a useable way to do that, I discovered the calendar object that can be added to a userform. This certainly made capturing the dates significantly easier.

In addition to the calendar difficulties, I also had problems finding a way to capture 400 days of historical data in order to create a 300-day moving average and graph that average over 100 days. Yahoo Finance allows the user to download a table of values from its site, but I couldn't figure out how to automate that download through sendkeys or any other tools that I was aware of. I finally discovered that I could access the file by attempting to open a workbook with the URL as the name of that workbook (Workbooks.Open Filename:= "url"). This command automatically downloaded and opened the table of values that I needed. I was then able to simply activate the newly opened worksheet, copy the needed data into my original workbook, and manipulate the data as needed.

There were certainly several moments of frustration when I would complete a section of code feeling confident that it was written perfectly, only to get an error message that I didn't understand when I tried to execute the code. The most frustrating moments came after segments of code would run perfectly one minute, only to break when I tried to run them in conjunction with other code. Working through these problems provided the greatest opportunities to drill down and better understand the properties and methods of certain objects within VBA.

In the creation of this program I was able to draw on almost all of what we learned over the course of the semester, but I was also forced to learn several things on my own that were not covered. The application of our classroom learning really helped me to see the real-world use of those concepts, and helped me to better understand how to research and learn new things on my own. Both of those facts will undoubtedly contribute greatly to my future endeavors in programming.

After working through all of the problems, I was able to effectively implement all of the things I wanted to do with the program. I was even able to add certain features that I wasn't intending to add when I began the project. My original plan was to simply create a button that would initiate the code, but I was able to change the actual ribbon in a way that allows the user to execute from that location instead. I was also able to change the code so that the user could perform multiple analyses on various securities and compare them rather easily. My original intent was to allow for only a single analysis that would simply overwrite each time the program was run. Certainly, the end result is much better than I had originally hoped for, and has executed flawlessly on over 100 trial runs.