2.3.3 Reuse Code

Steve may want to look at a different set of stocks in the future. With this in mind, we should create a flexible macro for running multiple stocks. By carefully reusing the code we've already written for DQ, we can write a macro with this flexibility.

To write this macro, we can reuse a lot of the code we wrote for DQ. However, we'll need to determine which lines of code should go inside the loop, and which lines of code should go outside the loop. For example, we don't need to get the number of rows in the 2018 sheet for every ticker; it will be the same, so we'll put that outside both loops.

REWIND

When you reuse code, you are essentially recognizing abstract patterns: you are using code already written to solve one problem and applying it to a different problem.

When we're done with the analysis for a ticker, we'll need to output the results, which means we'll be activating the output worksheet. When we start on a new ticker, we'll need to reactivate the data worksheet, so that code goes inside the innermost loop.

Using comments to show where we're going to put our code is a good idea. So far it would look something like the following. (Hold off on making any changes to your code just yet!)

```
Sub AllStocksAnalysis()
    'Find number of rows (before both loops)

    For i = 0 to 11

        ticker = tickers(i)
        For j = 2 to RowCount
            'Activate data worksheet

        Next j
    'Output results
    Next i
End Sub
```

NOTE

> We can't initialize a variable more than once because VBA will assume that we're trying to create a new variable and accidentally gave it the name of an existing variable. Therefore, we don't want to put our `Dim` statements inside loops.

Before we start putting code into our new loop structure, we should formulate a plan. We'll use this plan to keep our code blocks organized, using comments as our structure.

We can reuse a lot of the code we've already written in the `DQAnalysis` subroutine, but we'll need to rearrange it to fit our new ticker loop. Remember, we want to perform the same kind of analysis we did for DQ, but for every stock in our ticker list. We also don't want to waste time rewriting code that we've already written.

Let's write our plan.

Map Out a Plan

Since this code might get a little complicated, we should start by writing a basic outline of the program flow. Then we'll use comments to organize it all before we write the actual code.

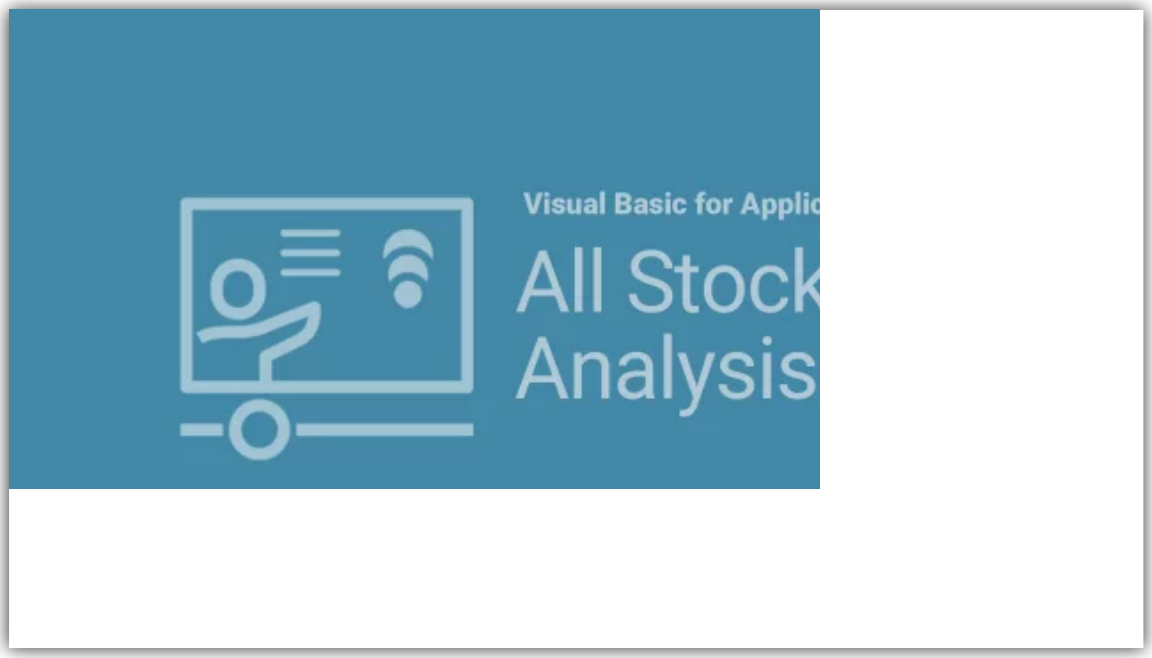
Our new macro should do the following:

1. Format the output sheet on the "All Stocks Analysis" worksheet.
2. Initialize an array of all tickers.
3. Prepare for the analysis of tickers.
    - Initialize variables for the starting price and ending price.
    - Activate the data worksheet.
    - Find the number of rows to loop over.
4. Loop through the tickers.
5. Loop through rows in the data.
    - Find the total volume for the current ticker.
    - Find the starting price for the current ticker.
    - Find the ending price for the current ticker.

6. Output the data for the current ticker.

Write the Macro

Let's put the plan into action. The following video provides an overview of this process from start to finish so you know what your code should look like.



Now you are going to put fingers to keys. Every coder goes through the experience of planning out what they're going to code, writing every line, and then running their program—only to find out it doesn't even run. Or it runs, but gives the wrong answer. This is where debugging comes into play.

Debugging is the process of going through your code, figuring out why it's not doing what you expected, and fixing the error. Debugging is something you've probably done already, but maybe you didn't know it had a name. If this happens to you, just remember that debugging is a big part of being a coder.

With both loops written in code, our game plan looks like this:

```
Sub AllStocksAnalysis()
    '1) Format the output sheet on All Stocks Analysis workshe

    '2) Initialize array of all tickers
    '3a) Initialize variables for starting price and ending pr

    '3b) Activate data worksheet
    '3c) Get the number of rows to loop over

    '4) Loop through tickers
    For i = 0 to 11
        ticker = tickers(i)
        '5) loop through rows in the data
        For j = 2 to RowCount
            '5a) Get total volume for current ticker
            '5b) get starting price for current ticker
            '5c) get ending price for current ticker

        Next j
        '6) Output data for current ticker

    Next i

End Sub
```

Step 1: Format the Output Sheet on the "All Stocks Analysis" Worksheet

Copy the code from `DQAnalysis` and make the following changes:

- Activate "All Stocks Analysis" instead of "DQ Analysis."
- Change the A1 value to "All Stocks (2018)."
- Change the first column header to "Ticker."

```
'1) Format the output sheet on All Stocks Analysis worksheet
Worksheets("All Stocks Analysis").Activate
Range("A1").Value = "All Stocks (2018)"
'Create a header row
Cells(3, 1).Value = "Ticker"
Cells(3, 2).Value = "Total Daily Volume"
Cells(3, 3).Value = "Return"
```

Step 2: Initialize an Array of All Tickers

There's no shortcut for this step. We have to type every assignment of a ticker to the array:

```
'2) Initialize array of all tickers
Dim tickers(11) As String

tickers(0) = "AY"
tickers(1) = "CSIQ"
tickers(2) = "DQ"
tickers(3) = "ENPH"
tickers(4) = "FSLR"
tickers(5) = "HASI"
tickers(6) = "JKS"
tickers(7) = "RUN"
tickers(8) = "SEDG"
tickers(9) = "SPWR"
tickers(10) = "TERP"
tickers(11) = "VSLR"
```

Step 3: Prepare for the Analysis of Tickers

In this step, we'll do the following:

- Initialize variables for the starting price and ending price.

- Activate the data worksheet.
- Get the number of rows to loop over.

We can use the some of the code from `DQAnalysis` as-is.

```
'3a) Initialize variables for starting price and ending price
Dim startingPrice As Double
Dim endingPrice As Double
'3b) Activate data worksheet
Worksheets("2018").Activate
'3c) Get the number of rows to loop over
RowCount = Cells(Rows.Count, "A").End(xlUp).Row
```

Step 4: Loop Through the Tickers

Before we get to the inner loop, we need to consider any values that need to be initialized before the inner loop starts. Every time we finish analysis on one ticker, we need to reset the total volume to zero. This means the line `totalVolume = 0` is inside the ticker loop, but outside of the row loop. Add it now.

```
'4) Loop through tickers
For i = 0 to 11
   ticker = tickers(i)
   totalVolume = 0
   '5) loop through rows in the data
   For j = 2 to RowCount
       '5a) Get total volume for current ticker
       '5b) get starting price for current ticker
       '5c) get ending price for current ticker

   Next j
```

```
    '6) Output data for current ticker
Next i
```

Step 5: Loop Through Rows in the Data

Now we can focus on the inner loop. Before starting the loop, make sure that the right worksheet is active by using the `Worksheets().Activate` method.

```
    '5) loop through rows in the data
    Worksheets("2018").Activate
    For j = 2 to RowCount
        '5a) Find total volume for current ticker
        '5b) Find starting price for current ticker
        '5c) Find ending price for current ticker

    Next j
```

Step 5 consists of three parts:

- Find the total volume for the current ticker.
- Find the starting price for the current ticker.
- Find the ending price for the current ticker.

For these steps, we can copy the code from `DQAnalysis,` but be careful! Now `j` is the variable iterating over the tickers, so we'll have to change every `i` reference to `j` after we copy and paste.

Find the total volume for the current ticker:

```
'5a) Find total volume for current ticker
If Cells(j, 1).Value = ticker Then
```

```
    totalVolume = totalVolume + Cells(j, 8).Value

End If
```

Find the starting price for the current ticker:

```
'5b) Find starting price for current ticker
If Cells(j - 1, 1).Value <> ticker And Cells(j, 1).Value = ti

    startingPrice = Cells(j, 6).Value

End If
```

Find the ending price for the current ticker:

```
'5c) Find ending price for current ticker
If Cells(j + 1, 1).Value <> ticker And Cells(j, 1).Value = ti

    endingPrice = Cells(j, 6).Value
End If
```

Step 6: Output the Data for the Current Ticker

Finally, we need to slightly alter the code so that the output for each ticker prints on a new row. This is a case where using `Cells()` is much easier than using `Range().` Instead of printing on the 4th row only, we print on the 4th row *plus* `i`.

```
'6) Output data for current ticker
Worksheets("All Stocks Analysis").Activate
```

```
Cells(4 + i, 1).Value = ticker
Cells(4 + i, 2).Value = totalVolume
Cells(4 + i, 3).Value = endingPrice / startingPrice - 1
```

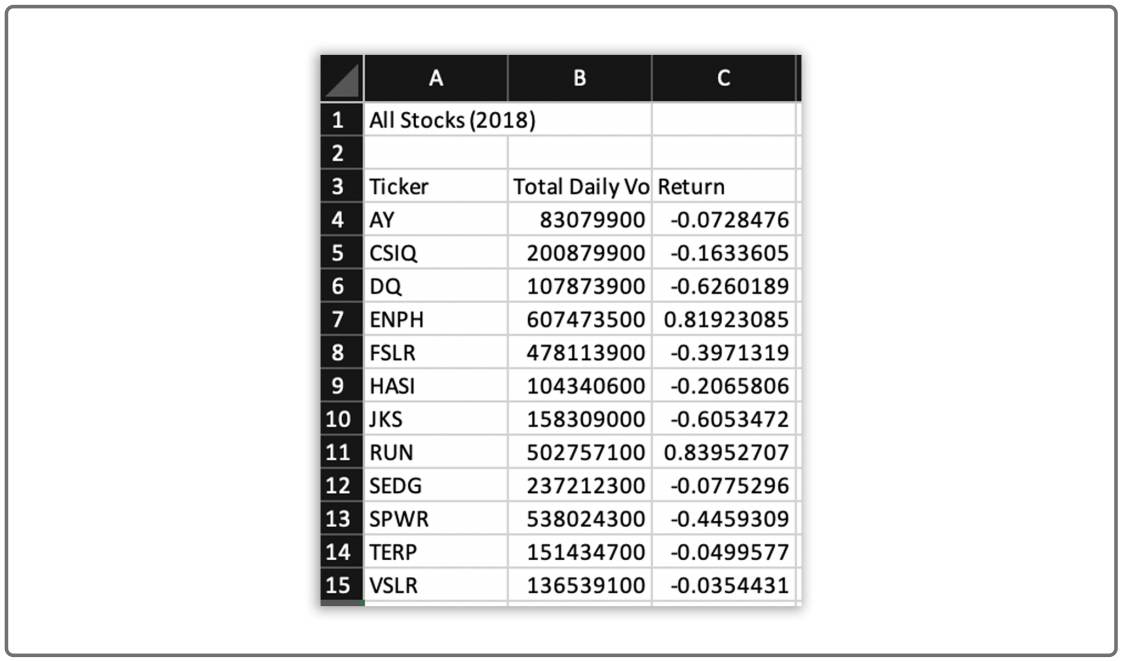
The macro should now look like the following. Compare your code to this and make sure you haven't missed anything.

```
Sub AllStocksAnalysis()
    '1) Format the output sheet on All Stocks Analysis workshe
    Worksheets("All Stocks Analysis").Activate
    Range("A1").Value = "All Stocks (2018)"
    'Create a header row
    Cells(3, 1).Value = "Ticker"
    Cells(3, 2).Value = "Total Daily Volume"
    Cells(3, 3).Value = "Return"

    '2) Initialize array of all tickers
    Dim tickers(11) As String
    tickers(0) = "AY"
    tickers(1) = "CSIQ"
    tickers(2) = "DQ"
    tickers(3) = "ENPH"
    tickers(4) = "FSLR"
    tickers(5) = "HASI"
    tickers(6) = "JKS"
    tickers(7) = "RUN"
    tickers(8) = "SEDG"
    tickers(9) = "SPWR"
    tickers(10) = "TERP"
    tickers(11) = "VSLR"
    '3a) Initialize variables for starting price and ending pr
    Dim startingPrice As Single
    Dim endingPrice As Single
    '3b) Activate data worksheet
```

```
    Worksheets("2018").Activate
    '3c) Get the number of rows to loop over
    RowCount = Cells(Rows.Count, "A").End(xlUp).Row

    '4) Loop through tickers
    For i = 0 to 11
        ticker = tickers(i)
        totalVolume = 0
        '5) loop through rows in the data
        Worksheets("2018").Activate
        For j = 2 to RowCount
            '5a) Get total volume for current ticker
            If Cells(j, 1).Value = ticker Then

                totalVolume = totalVolume + Cells(j, 8).Value

            End If
            '5b) get starting price for current ticker
            If Cells(j - 1, 1).Value <> ticker And Cells(j, 1)

                startingPrice = Cells(j, 6).Value

            End If

            '5c) get ending price for current ticker
            If Cells(j + 1, 1).Value <> ticker And Cells(j, 1)

                endingPrice = Cells(j, 6).Value

            End If
        Next j
        '6) Output data for current ticker
        Worksheets("All Stocks Analysis").Activate
        Cells(4 + i, 1).Value = ticker
        Cells(4 + i, 2).Value = totalVolume
        Cells(4 + i, 3).Value = endingPrice / startingPrice -
```

```
    Next i

End Sub
```

The below image is an example of how your sheet should look after running the macro.



	A	B	C
1	All Stocks (2018)		
2			
3	Ticker	Total Daily Vo	Return
4	AY	83079900	-0.0728476
5	CSIQ	200879900	-0.1633605
6	DQ	107873900	-0.6260189
7	ENPH	607473500	0.81923085
8	FSLR	478113900	-0.3971319
9	HASI	104340600	-0.2065806
10	JKS	158309000	-0.6053472
11	RUN	502757100	0.83952707
12	SEDG	237212300	-0.0775296
13	SPWR	538024300	-0.4459309
14	TERP	151434700	-0.0499577
15	VSLR	136539100	-0.0354431

ADD/COMMIT/PUSH

Don't forget to save your changes and push `green_stocks.xlsm` to the "stocks-analysis" repository in GitHub.