2.2.1 Create a Worksheet for Your Analysis

Now that we know that VBA is working correctly, let's start analyzing some stock data. Steve wants to find the total daily volume and yearly return for each stock. Daily volume is the total number of shares traded throughout the day; it measures how actively a stock is traded. The yearly return is the percentage difference in price from the beginning of the year to the end of the year. Steve's parents are starting to pester him about Daqo's stock, so we'll start with DQ.

Work with Worksheets

We've enabled and tested macros, so now we can start analyzing actual stock data. First, we'll need a worksheet to hold this data. The following video walks you through the process of setting up a worksheet for your analysis. Watch the video for that corresponds to your operating system.

macOS

Windows

Work with Cells

Since Excel holds its data in cells, we want to be able to access them in VBA. There are two ways to access cells in VBA: the `Range()` method and the `Cells()` method. For our project, we're going to use both.

IMPORTANT

> Everything we interact with in Excel—for instance, cells, ranges, charts, and worksheets—are **objects** in VBA. VBA objects have properties that we read and methods that we call. **Properties** are like predefined variables that hold values about the object. A **method** is like a subroutine: a collection of instructions that can be called. Methods often take in arguments and can return values.
>
> In this case, the `Range()` method belongs to the Worksheet object that we activated.

First, we'll use the `Range()` method, which selects cells with the same range format that Excel formulas use. The `Range()` method can also select a range of only one cell, which is what we are going to use here. We'll set the value of cell A1 to "DAQO (Ticker: DQ)" with the code `Range("A1").Value = "DAQO (Ticker: DQ)"`, as shown below:

```
Sub DQAnalysis()
    Worksheets("DQ Analysis").Activate

    Range("A1").Value = "DAQO (Ticker: DQ)"

End Sub
```

Next, we'll use the `Cells()` method. It works similarly to the `Range()` method, but it takes two arguments:

- how many rows down from the top the target cell is
- how many columns over from the left the target cell is

For example, to put "Year" in the cell A3, we would use `Cells(3, 1).Value = "Year".`

Let's use `Cells()` to create a header for cells A3 through C3 with column names Year, Total Daily Volume, and Return.

In this example, we could use `Range()` to accomplish the same goal, but `Cells()` will be more flexible as we move to automated code because individual numbers are easier to work with than strings of cell coordinates. When filling in the table below the header, use the same pattern of code but specify the row value using a variable instead.

```
Sub DQAnalysis()
    Worksheets("DQ Analysis").Activate

    Range("A1").Value = "DAQO (Ticker: DQ)"

    Cells(3, 1).Value = "Year"
    Cells(3, 2).Value = "Total Daily Volume"
    Cells(3, 3).Value = "Return"
End Sub
```

Let's get some practice using the `Range()` and `Cells()` methods.

SKILL DRILL

Using only `Range()` method, rewrite `DQAnalysis` so that it creates the same output. Then rewrite it again, using only the `Cells()` method.

ADD/COMMIT/PUSH

Save your changes to `green_stocks.xlsm` and upload it to the "stocks-analysis" repository in GitHub (you may need to temporarily close your spreadsheet before it can be pushed to GitHub). GitHub will update the file to the new version while keeping a history of all the changes we've made.