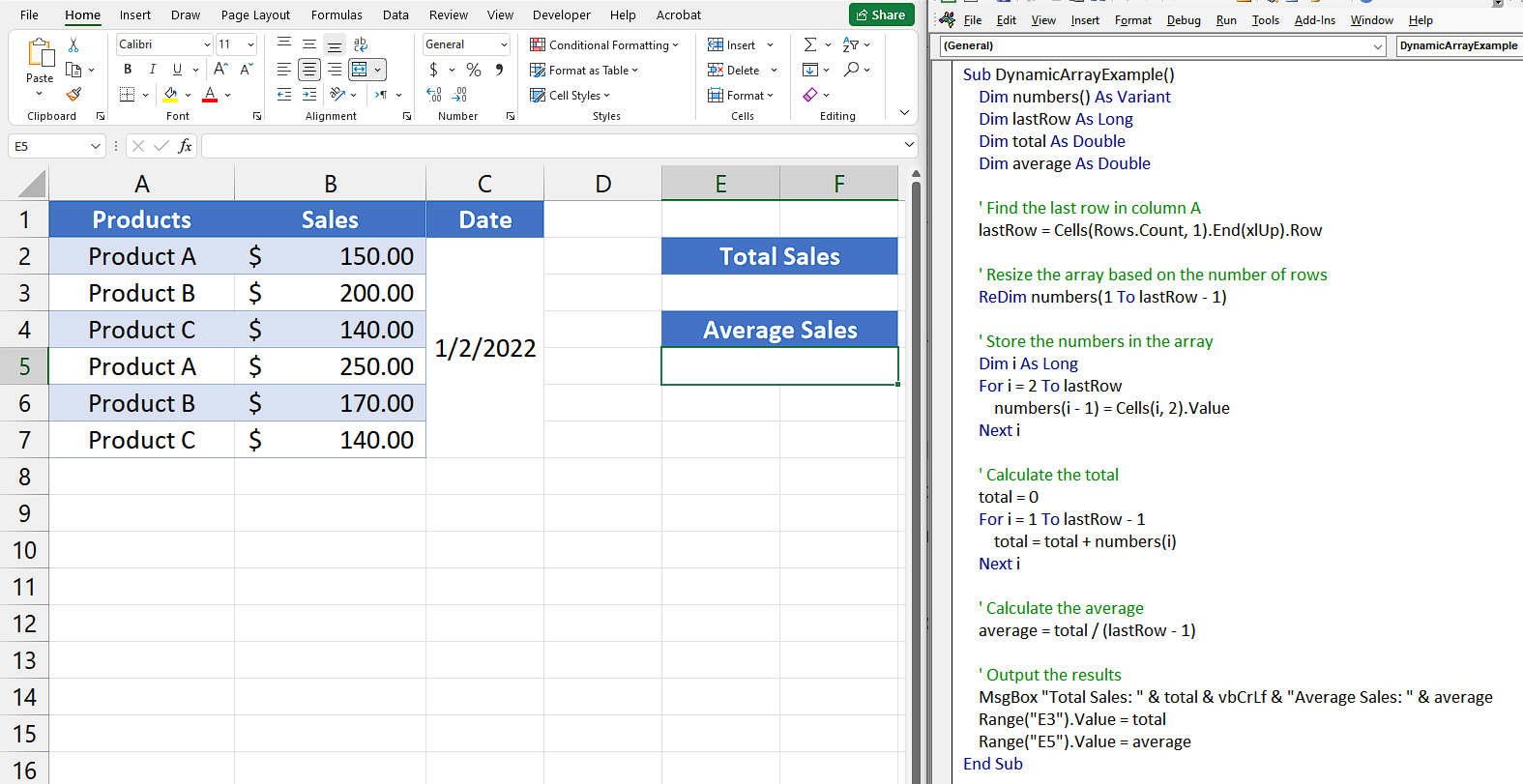
## How to use dynamic array in VBA

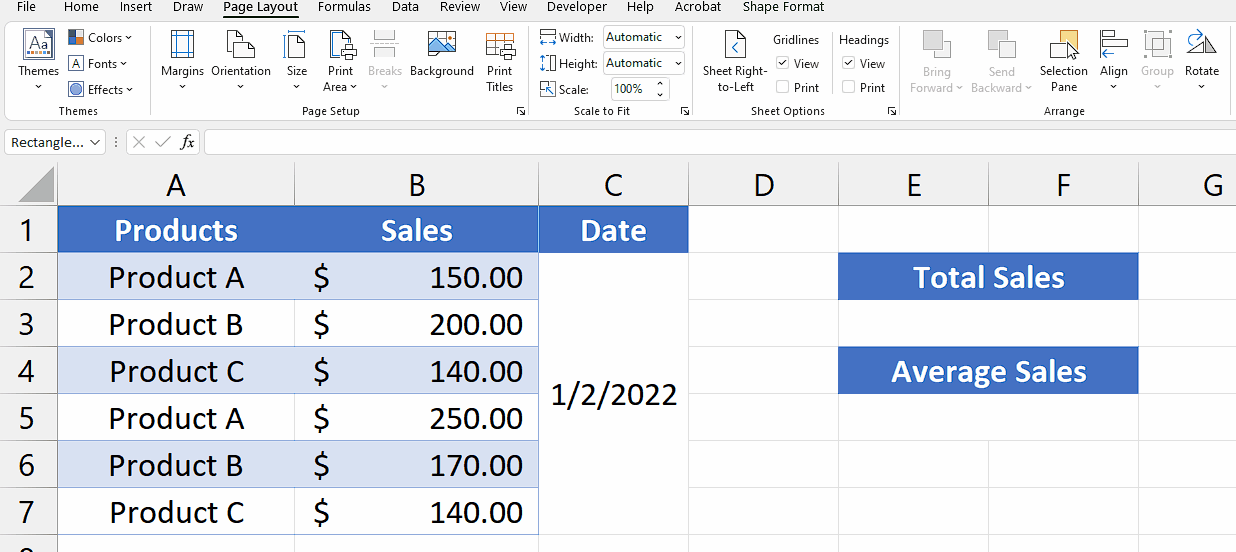
## In VBA (Visual Basic for Applications), a dynamic array is an array whose size can be changed or resized during runtime. Unlike a static array, which has a fixed size specified during declaration, a dynamic array allows for flexibility in handling data that may vary in size.

## The provided data contains sales information for three products: Product A, Product B, and Product C. The sales figures are recorded in dollars ($) and are continuously updated as new rows are added to the column. By utilizing VBA code with dynamic arrays, we can calculate the average sales and total sales for any number of rows present in the column. This allows for flexibility in handling varying amounts of sales data and enables us to obtain accurate and up-to-date insights into the overall performance of the products.



## Step 1 – Insert a module

* For adding a module, navigate to **Developer tab.**
* After that, click on the **first option** named as **Visual basic.**
* You’ll see a **new window on your screen** and you can **also** **open this window** by using **shortcut key** (**Alt+F11**) as well.
* Then, click on the **Insert tab** in this window and click on the **Module option.**
* Now, a **new module would open.**



## Step 2 – Write the code

* Write the code which contains Dynamic array the code area of module.
* For example, we are writing the following code,

Sub DynamicArrayExample()

Dim numbers() As Variant

Dim lastRow As Long

Dim total As Double

Dim average As Double

' Find the last row in column A

lastRow = Cells(Rows.Count, 1).End(xlUp).Row

' Resize the array based on the number of rows

ReDim numbers(1 To lastRow - 1)

' Store the numbers in the array

Dim i As Long

For i = 2 To lastRow

numbers(i - 1) = Cells(i, 2).Value

Next i

' Calculate the total

total = 0

For i = 1 To lastRow - 1

total = total + numbers(i)

Next i

' Calculate the average

average = total / (lastRow - 1)

' Output the results

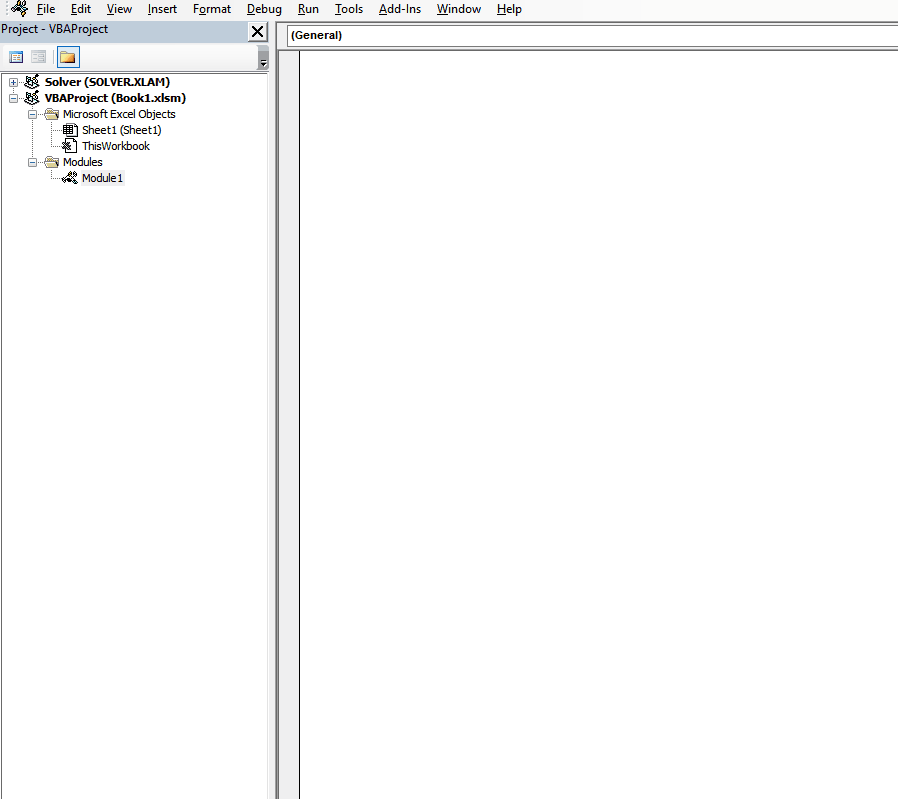
MsgBox "Total Sales: " & total & vbCrLf & "Average Sales: " & average

Range("E3").Value = total

Range("E5").Value = average

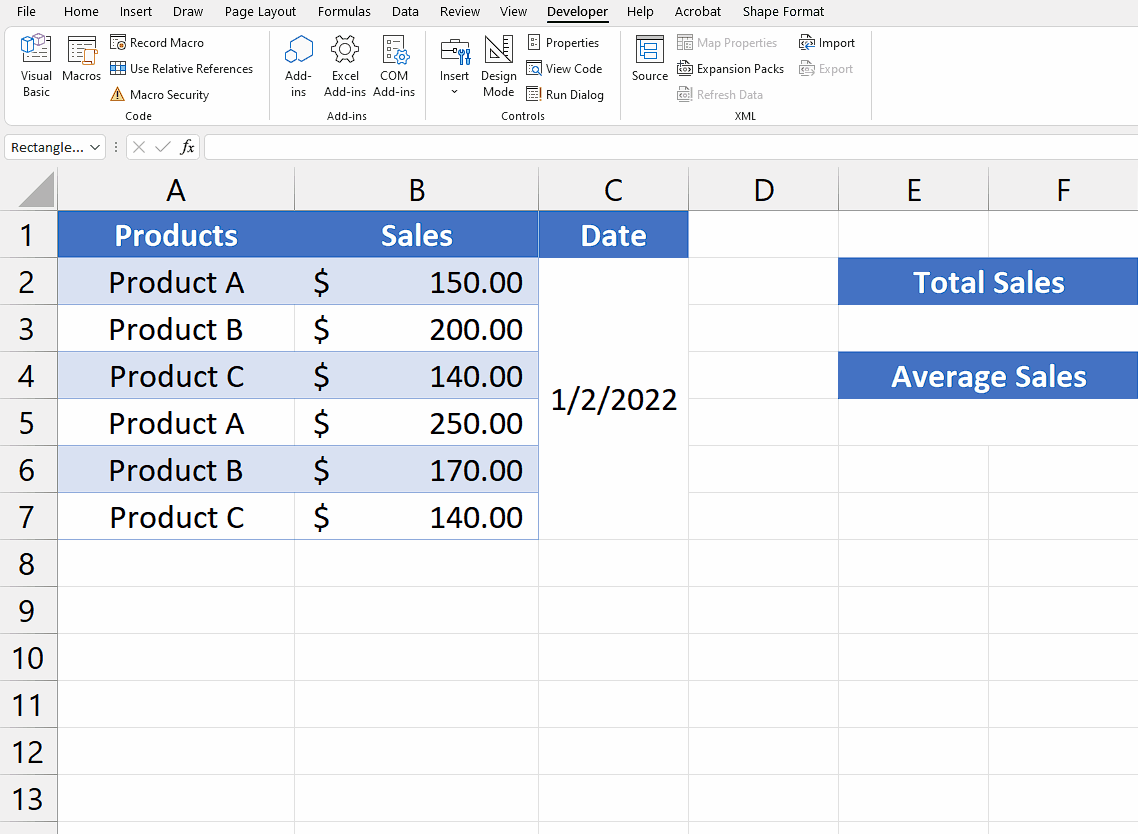
End Sub

* Then, close the VBA Editor by pressing “Alt+Q”.



## Step 3 – Add the button

* For inserting a shape that will act as a button, navigate to “Insert” tab.
* Click on the “Shapes” option under “Illustrations” group.
* After that, select the any shape you like.
* To insert a shape at a specific location, hold down the left-click button on your mouse and drag the cursor.
* Then, you can change “Shape Fill”, “Shape Effects” and etc. in the “Shape Format” Tab.
* Now, you can add text such as “RUN CODE” in the newly inserted shape.

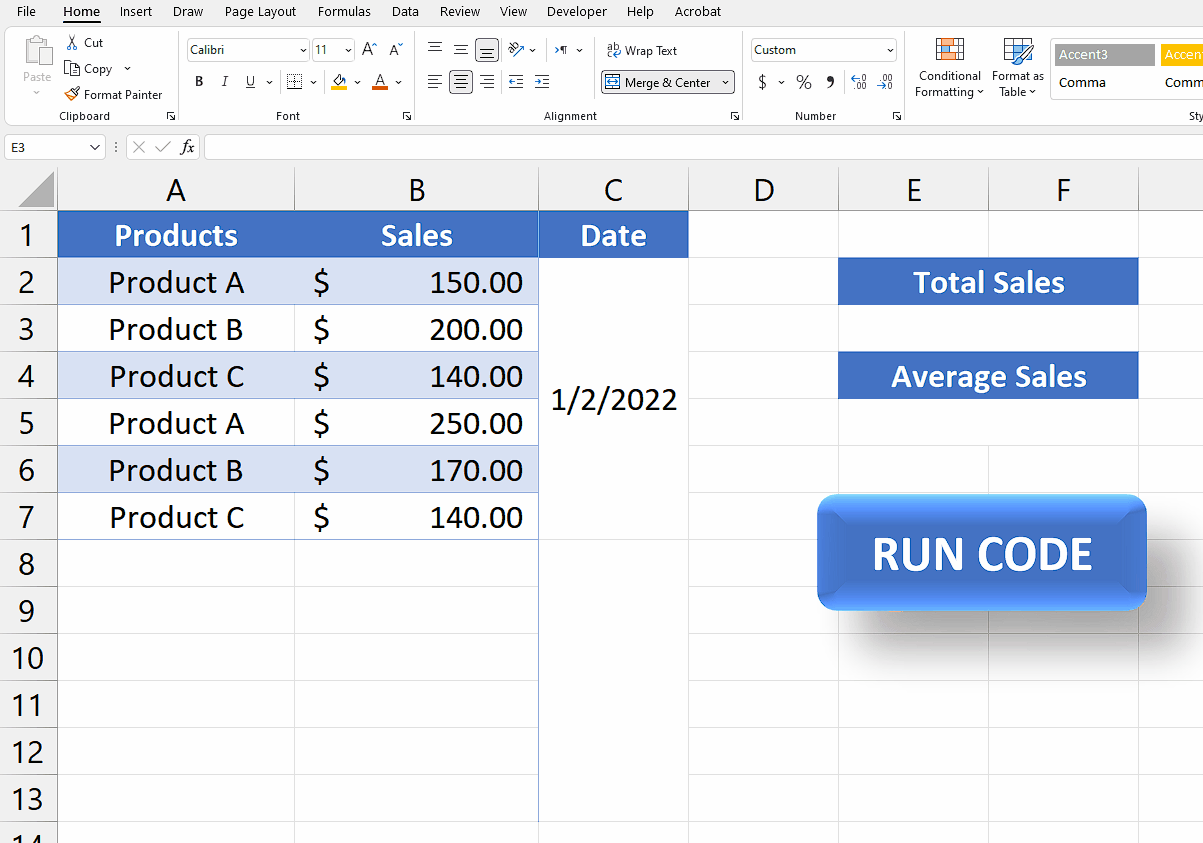


## Step 4 – Assign Macro

* For assigning the macro, right-click on the shape and click on the “Assign Macro” option.
* Click on the name of your macro such as “DynamicArrayExample”.
* Then, press “OK” button.

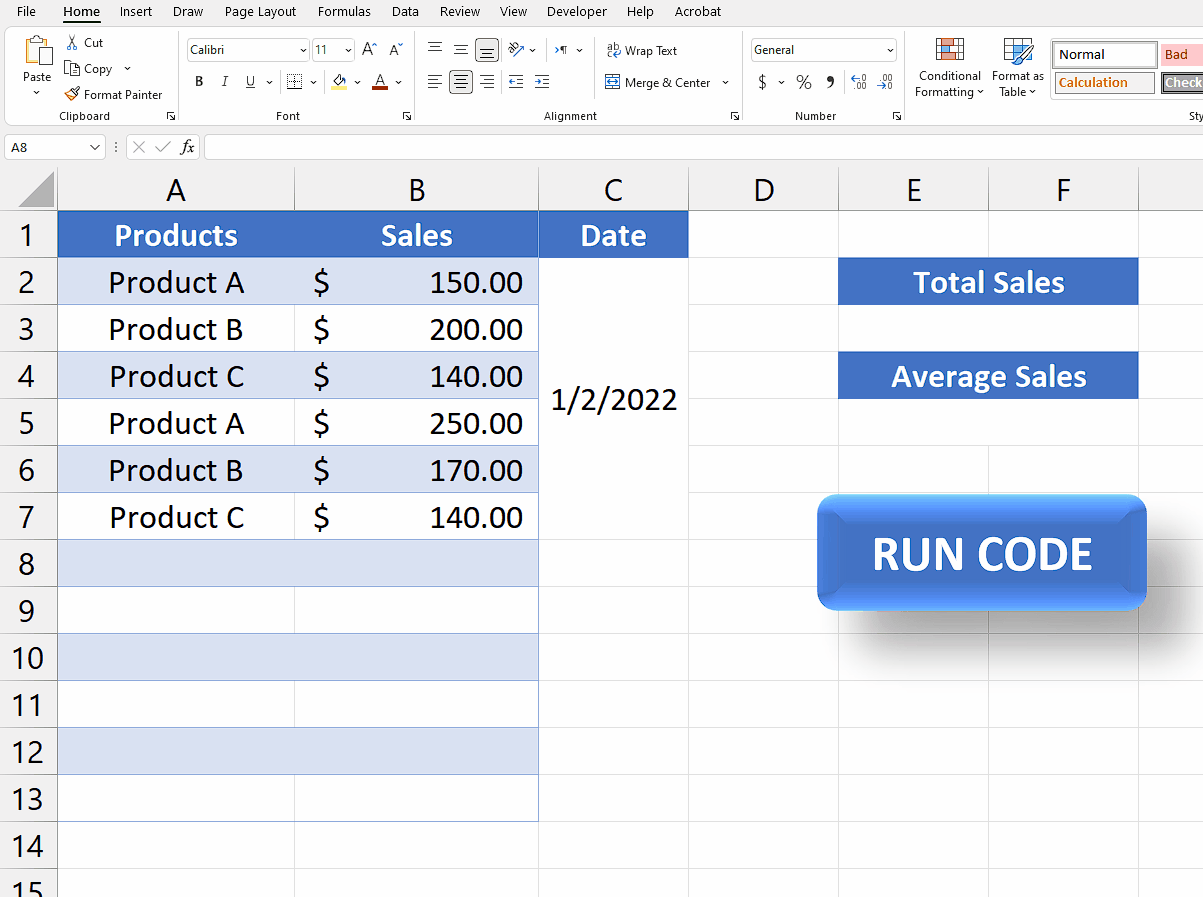
## Step 5 – Run the code

* Click on the button to run the code.
* The use of a dynamic array allows the code to run smoothly and accurately calculate the total sales and average sales, regardless of the number of cells in the "Sales" column.



## Step 6 – Modify the Dataset

* Since the sales data accumulates in the columns daily, the recently added data for the next day has now modified the existing data.
* To evaluate the functionality of the dynamic array, you have to input or alter the data.



## Step 7 – Run the code Again

* Click on the button to execute the code.
* Even though the dataset has been modified and new cells have been added, the code continues to function correctly. This is due to the utilization of a dynamic array in our code, which operates seamlessly regardless of the number of cells in the "Sales" column.
* We can observe that the accurate values for "Total sales" and "Average Sales" are calculated for the two-day period.

