Sum by Color in Excel (Step-by-Step Example)

Authored by

Mohammed looti

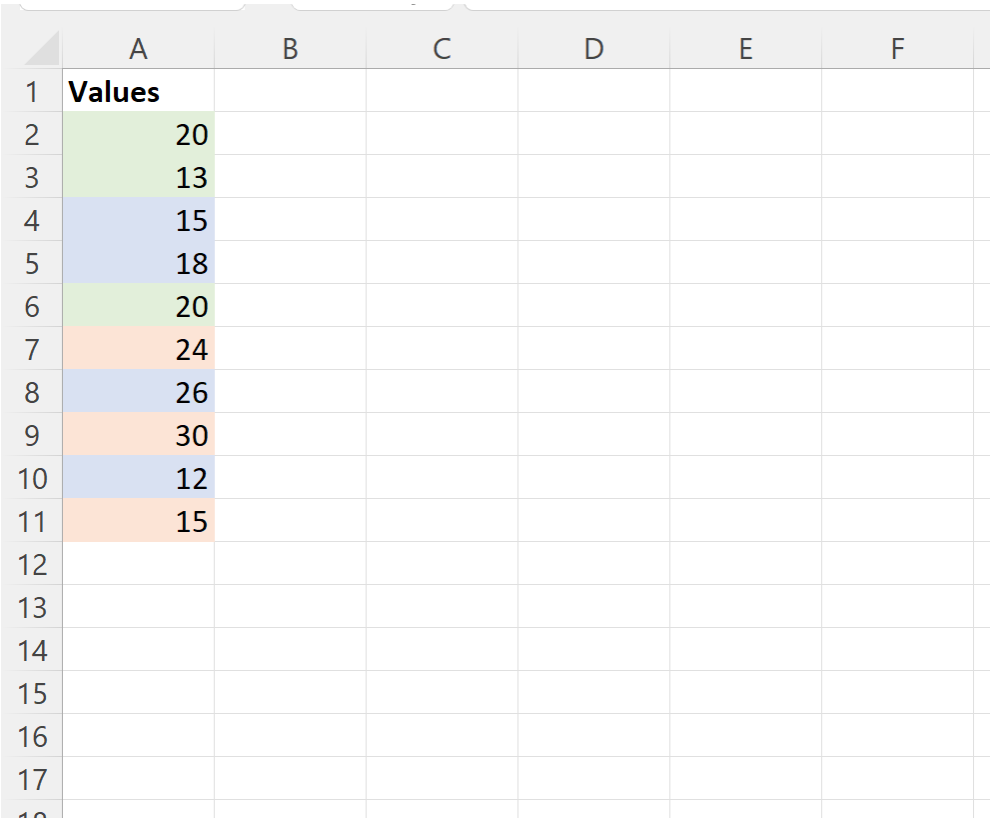
November 15, 2025

The Challenge of Summing by Cell Color in Excel

It is a common requirement in data analysis to aggregate values based on specific visual attributes, such as the background color of a cell. Standard Excel functions, including the versatile **SUMIF** or **SUMIFS**, are designed to evaluate criteria based on cell content (numbers, text, or dates), not formatting properties like fill color. This limitation necessitates a specialized approach when attempting to sum values conditionally based purely on cell color.

Consider a scenario where we are tracking sales data, and specific entries have been manually highlighted (e.g., green for high priority, yellow for moderate). To calculate the total value represented by these colored cells, we must introduce a custom solution. The most efficient and robust method within the Microsoft ecosystem is through writing a custom function using VBA (Visual Basic for Applications).

For instance, suppose we have the following dataset and our objective is to isolate and sum the values in the cells based on their unique background colors, treating each color as its own criterion:



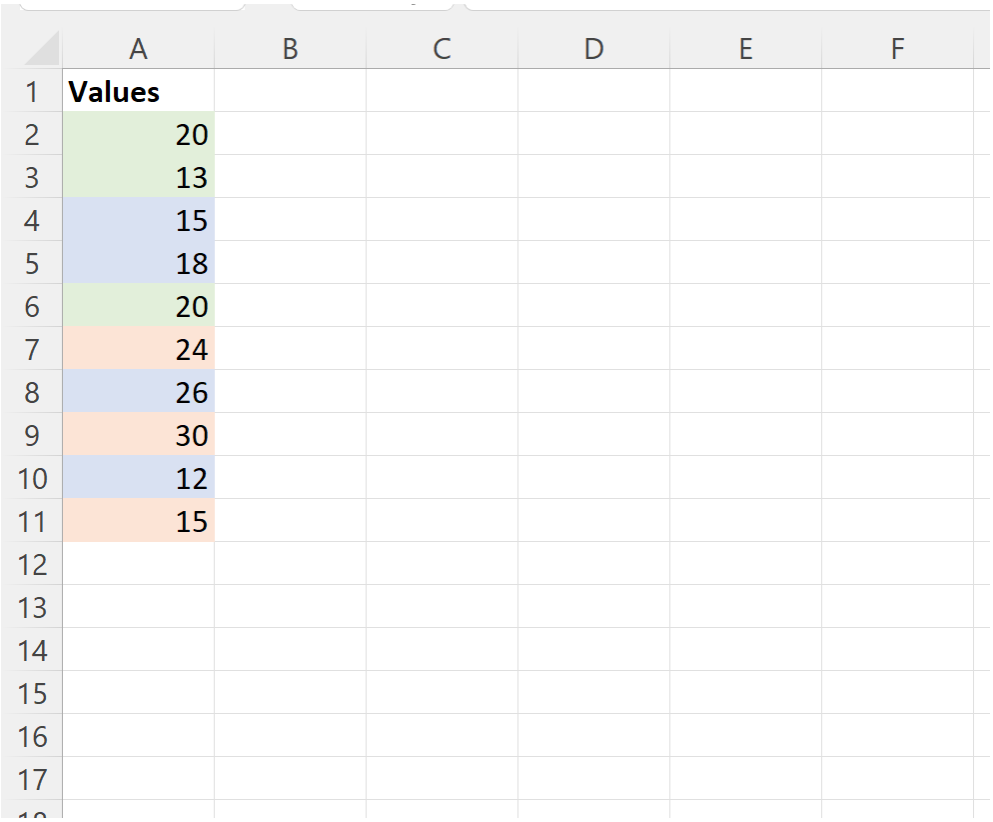
	A	B	C	D	E	F
1	**Values**					
2	20					
3	13					
4	15					
5	18					
6	20					
7	24					
8	26					
9	30					
10	12					
11	15					
12						
13						
14						
15						
16						
17						

While the prospect of writing custom code in VBA might initially seem complex, especially for users unfamiliar with programming, the necessary steps are highly structured and straightforward. The following comprehensive, step-by-step tutorial will guide you through creating and deploying a User

Defined Function (UDF) that successfully overcomes this formatting constraint, allowing for precise summation based on cell color.

Preparation: Enabling the Developer Environment

Before we can begin writing the custom code, we must ensure that the necessary tools for development are accessible within the Excel interface. The first step involves entering the raw data into your worksheet, ensuring that the necessary cells have already been formatted with the specific background colors you intend to sum.



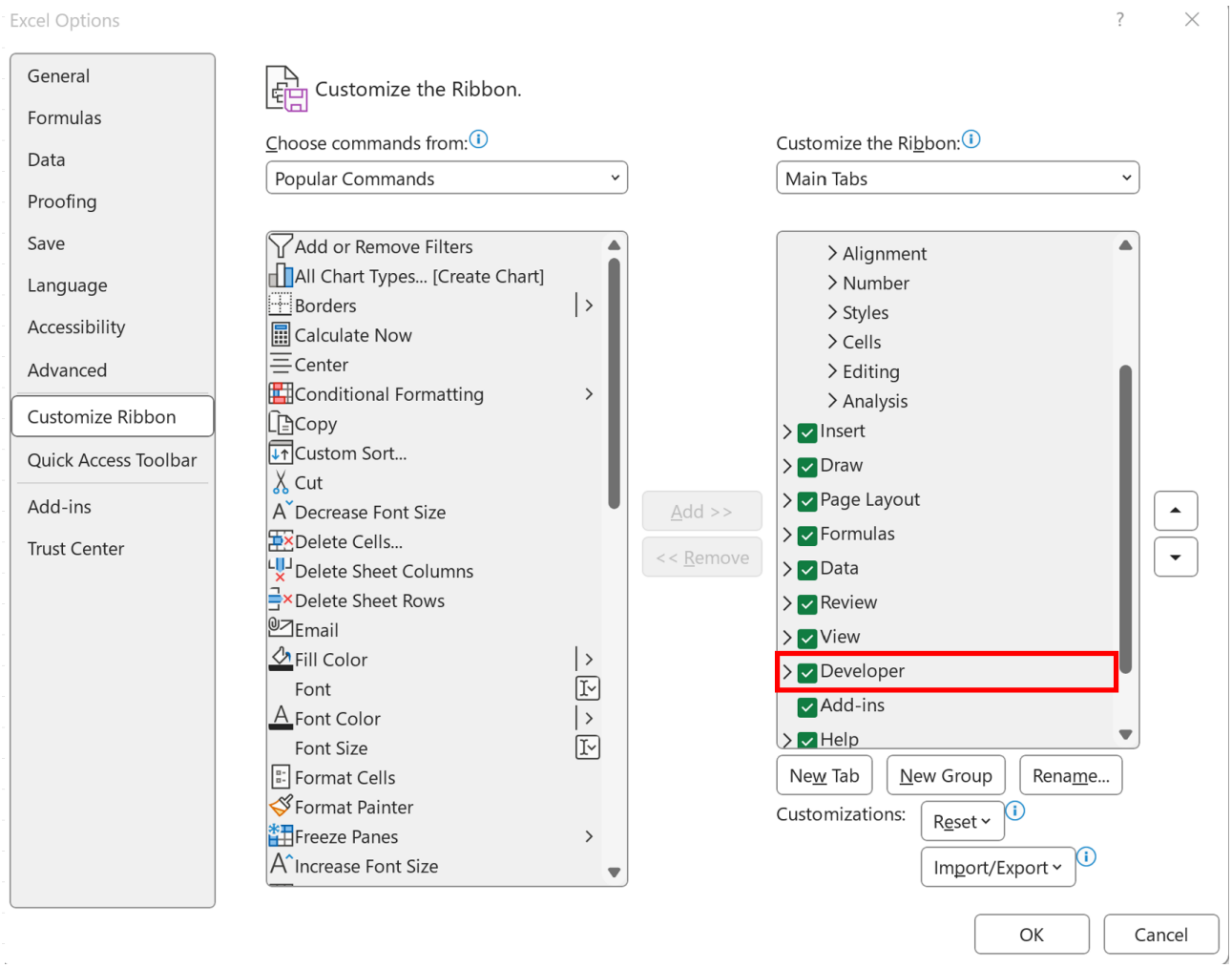
	A	B	C	D	E	F
1	**Values**					
2	20					
3	13					
4	15					
5	18					
6	20					
7	24					
8	26					
9	30					
10	12					
11	15					
12						
13						
14						
15						
16						
17						

The next critical preparatory step is to make the **Developer** tab visible on the top ribbon. This tab houses the essential commands for accessing the Visual Basic Editor (VBE) and managing Macros. If this tab is not currently visible, follow these instructions precisely to enable it:

Click the **File** tab located in the upper-left corner of the Excel window.
Select **Options** from the menu to open the Excel Options dialog box.
In the left pane of the dialog box, click **Customize Ribbon**.
Under the section labeled **Main Tabs**, locate the **Developer** option.
Check the box next to **Developer**, and then click **OK** to save the changes and close the dialog box.

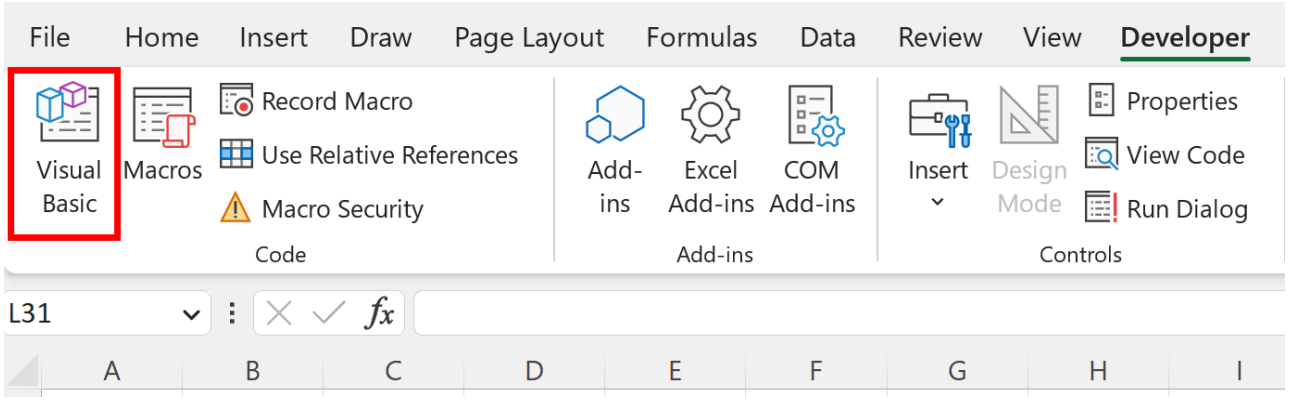
Once the **Developer** tab is enabled, you will have access to the full suite of Developer tab tools

required to create and manage the custom function we need for summing by color.

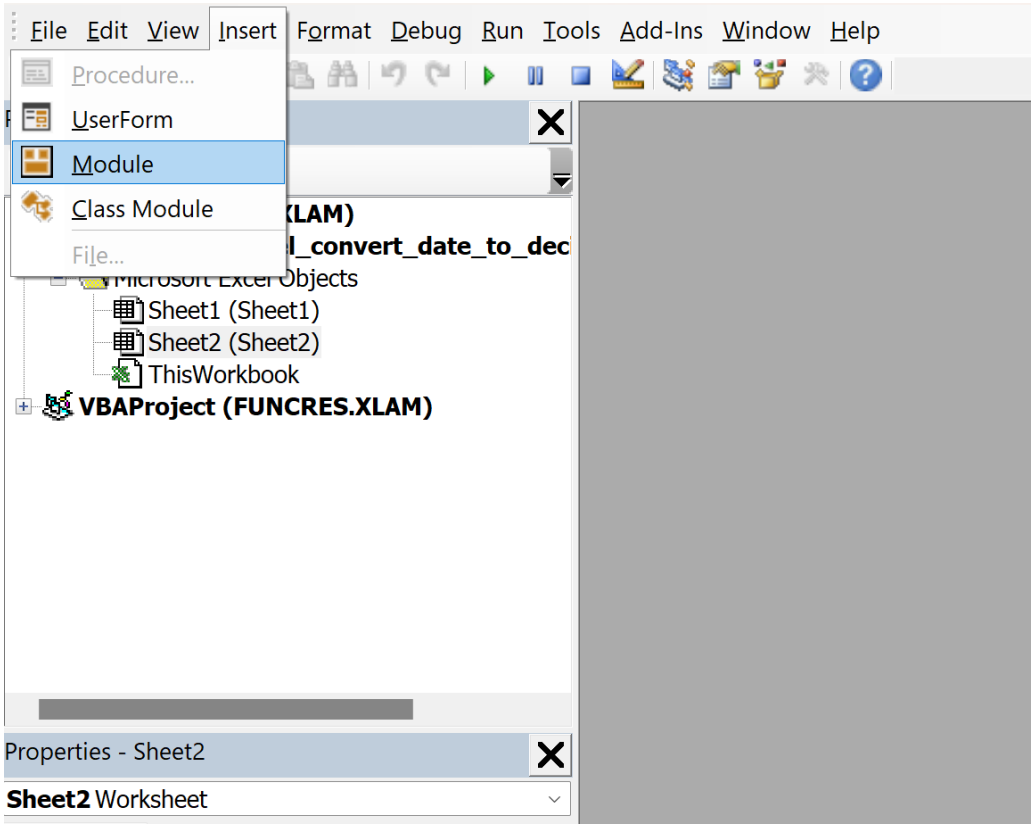


Developing the Custom Function: Writing the VBA Macro

With the Developer environment active, we can proceed to create the custom function, often referred to as a Macro, that will perform the summation logic. Begin by navigating to the newly visible **Developer** tab along the top ribbon. Within the Code group, click the **Visual Basic** icon to launch the Visual Basic Editor (VBE).



The VBE is where we will write the code. Since we are creating a function that will be callable directly from the worksheet (a User Defined Function), the code must reside within a standard Module. To insert a new module, click the **Insert** tab within the VBE menu bar, and then select **Module** from the dropdown menu. This action opens a blank code editor pane where you can input the necessary script.

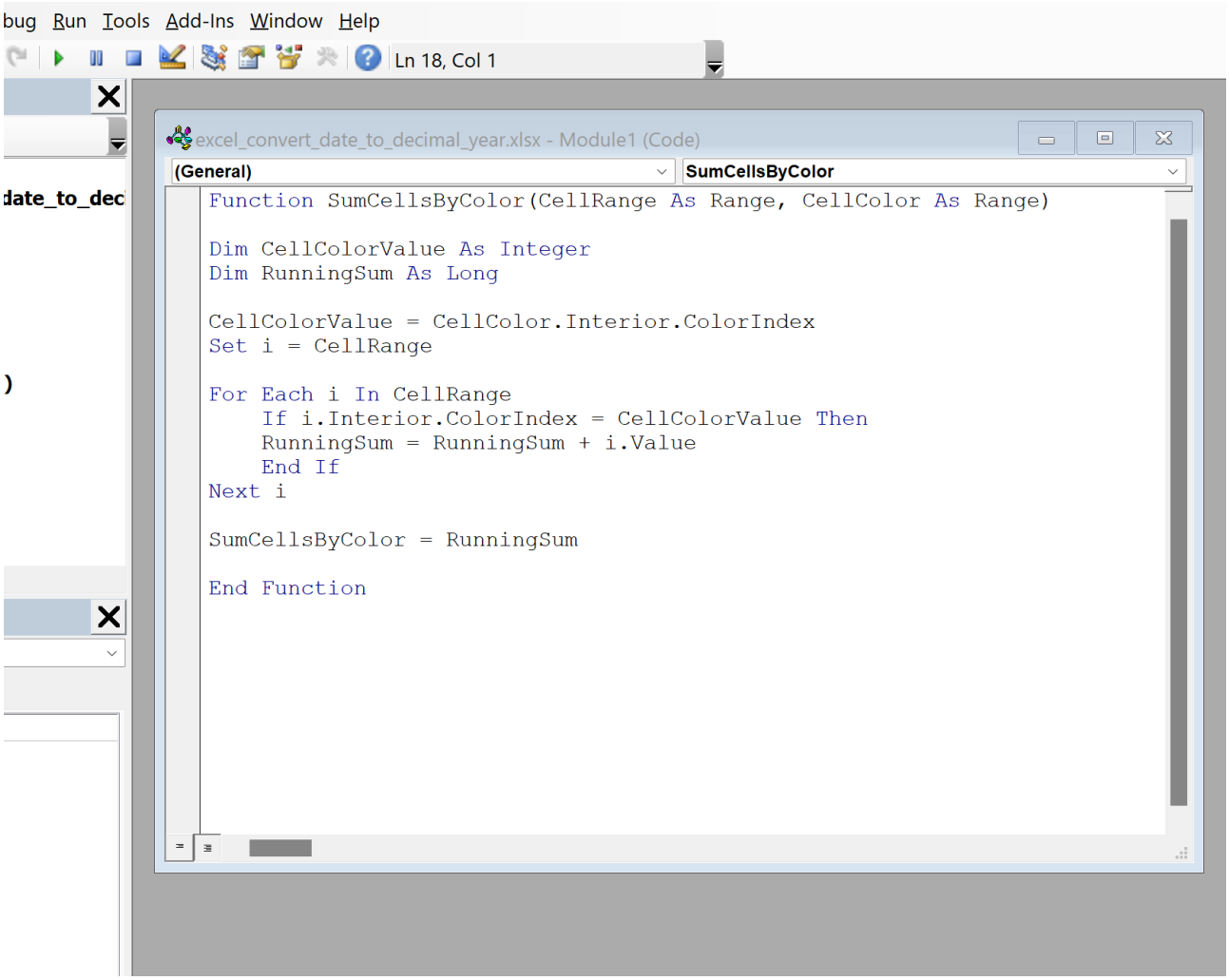


Carefully paste the following VBA code into the newly created module. This script defines a function named `SumCellsByColor` which accepts two arguments: the range of cells to be summed, and a single cell that serves as the color reference:

Function SumCellsByColor(CellRange As Range, CellColor As Range)

```
Dim CellColorValue As Integer
Dim RunningSum As Long

CellColorValue = CellColor.Interior.ColorIndex
Set i = CellRange

For Each i In CellRange
If i.Interior.ColorIndex = CellColorValue Then
RunningSum = RunningSum + i.Value
End If
Next i

SumCellsByColor = RunningSum

End Function
```

The correct placement of the code within the module editor is illustrated in the following screenshot, confirming that the function is properly encapsulated within the standard Module structure for immediate use within the workbook. Once the code is accurately pasted, you can close the Visual Basic Editor. The function is now saved within your workbook and ready to be called like any native Excel formula.



Deconstructing the VBA Code Logic

Understanding the logic behind the `SumCellsByColor` function is essential for troubleshooting and future customization. The function is specifically engineered to iterate through a defined set of cells and compare the formatting of each cell against a single reference color.

The function declaration `Function SumCellsByColor(CellRange As Range, CellColor As Range)` defines the inputs: `CellRange` is the entire area where values reside, and `CellColor` is a single cell whose background color acts as the criterion. Internally, the script declares two variables: `CellColorValue` (an Integer to hold the color code) and `RunningSum` (a Long integer to accumulate the total value).

The core of the selection process lies in the line `CellColorValue = CellColor.Interior.ColorIndex`. Every background color in Excel is assigned a unique numerical identifier, known as the ColorIndex. This line captures the specific index of the reference cell provided by the user. This numerical index is far more reliable for programmatic comparison

than trying to capture complex RGB values.

A loop is then initiated using `For Each i In CellRange`, which sequentially examines every cell within the specified data range. Inside this loop, an `If` condition checks whether the current cell's ColorIndex matches the stored `CellColorValue`. If the condition is met, the cell's numeric content (`i.Value`) is added to the `RunningSum` variable. After iterating through all cells, the final accumulated value is assigned as the output of the function using `SumCellsByColor = RunningSum`, making the result available on the worksheet.

Execution: Applying the Custom SumCellsByColor Formula

The final stage involves deploying the newly created custom function within your Excel worksheet. To effectively use the function, it is highly recommended to set up a dedicated area where you can list the unique colors you wish to sum, thereby creating the criteria necessary for the formula.

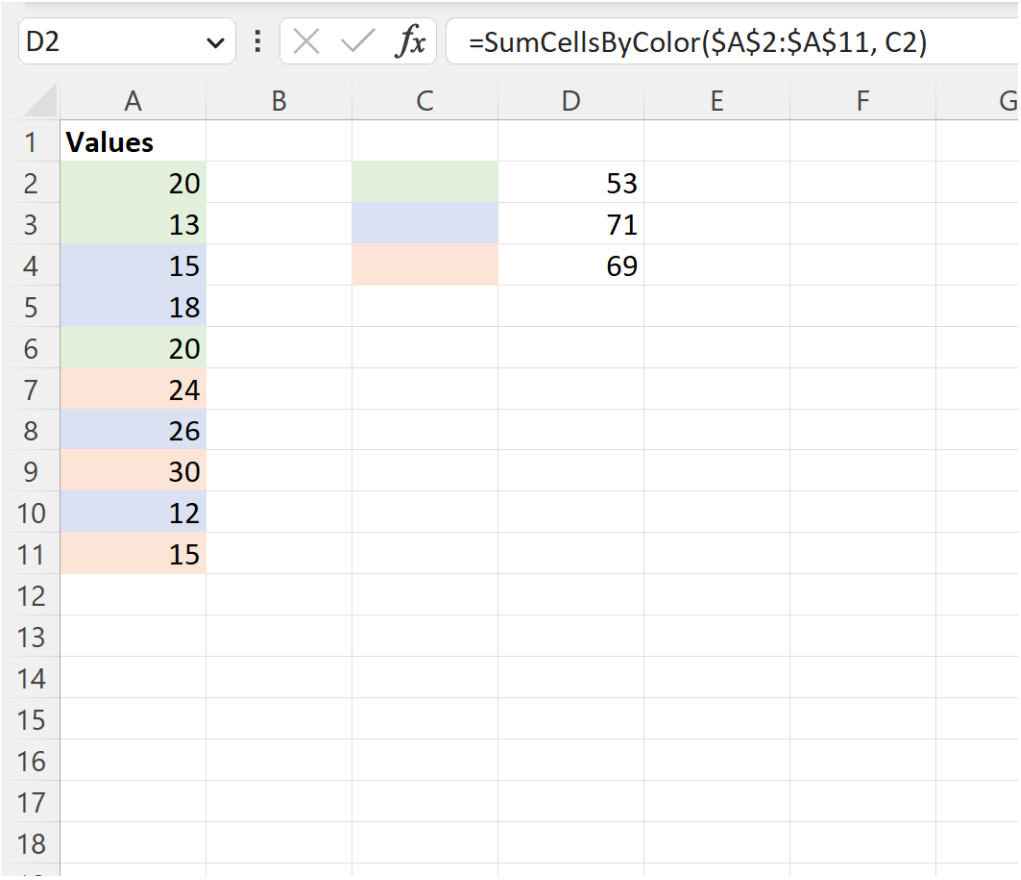
Begin by filling cells, for instance, in the range **C2:C4**, with the exact background colors that correspond to the data you are analyzing (e.g., light green, yellow, and blue). These cells will serve as the necessary color references (the `CellColor` argument) for the custom function.

Next, navigate to the cell where you want the first sum to appear (e.g., cell **D2**) and input the following formula. This specific formula targets the data range **$A$2:$A$11** and uses cell **C2** as the color reference:

=SumCellsByColor($A$2:$A$11, C2)

Note the use of absolute references (the dollar signs, e.g., **$A$2:$A$11**) for the data range. This is crucial because when you drag the formula down, the data range must remain fixed, while the color reference (**C2**) needs to change relative to the row (becoming C3, C4, etc.) to evaluate each unique color criterion.

Drag the formula from cell **D2** down to cover the remaining cells in column D (D3 and D4). The formula will automatically adjust its color reference, calculating the sum for each background color listed in column C.



As demonstrated in the final result, the function successfully calculated the aggregate value for each color group. For instance, the sum of all cells marked with a light green background is correctly identified as **53**. We can quickly verify this result by manually checking the corresponding entries in the original dataset: 20 + 13 + 20 = **53**. This confirmation ensures that the VBA function is operating accurately based on the cell formatting.

Conclusion and Verification

The creation and deployment of this custom VBA function, `SumCellsByColor`, provides a powerful solution for aggregating numerical data based on visual formatting attributes that are typically ignored by native Excel functions. By leveraging the **Developer** tools and understanding the concept of the ColorIndex, users can extend Excel's capabilities significantly, tailoring data processing to meet highly specific reporting needs.

This method is particularly useful in scenarios where conditional formatting has been applied, or where manual coloring is used for quick visual categorization. Remember that this function relies on static formatting; if the cell colors are changed, the function will not automatically recalculate unless the cell containing the function (e.g., D2) is manually refreshed or the worksheet is forced to recalculate.

For those interested in exploring further customization within Excel, numerous other tasks can be automated or enhanced using VBA.

Additional Resources

The following tutorials explain how to perform other common tasks in Excel: