Macros 2

This chapter will work more closely with Visual Basic (sometimes called Visual Basic for Applications or VB). First, we'll record another macro similar to the previous chapter that creates a basic table utilizing various formatting features.

Practice Spreadsheet

Use the workbook you created in the Macros 1 chapter.

1. Press the **Record Macro** button from the **Developer** toolbar.
    a. For the purpose of this demonstration, the macro settings can remain as their default.
2. Type the letters **A–F** into separate cells down the left side of the table's column.
3. Type the numbers **1–5** into separate cells across the table's top row.
4. Format the table's cells with borders.
5. Group columns **I** and **J**.
    a. Select the I and J columns.
    b. Go to the **Data** toolbar and press the **Group** button.
6. Type a title for the table in the top left cell.
    a. For the purpose of this demonstration, the title can simply be "Table title."
    b. The title needs to be in the row above the table's row of numbers.
7. Format the table title's row with **Heading 1**.
8. Format the number and letter labels in **bold**. (See **Figure 29.1**)
9. Press the **Stop Recording** button from the **Developer** toolbar.

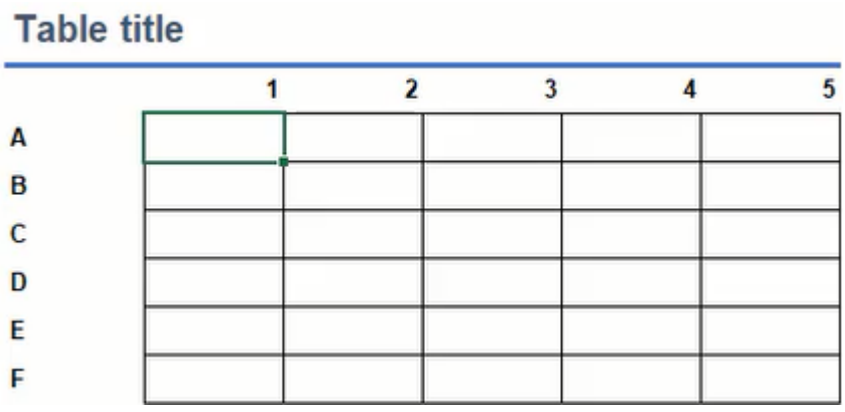

Figure 29.1

Additionally, we will record a macro to remove the table.

1. Press the **Record Macro** button from the **Developer** toolbar.
2. Ungroup columns **I** and **J**.
    a. Select columns **I** and **J**.
    b. Go to the **Data** toolbar and press the **Ungroup** button.
3. Select all of the table's cells and press the **Delete** key.
   Note: You will probably notice some of the formatting remains. The delete key removes cell values, but it does not delete formatting. Continue with the next steps to clear formats. (See **Figure 29.2**)

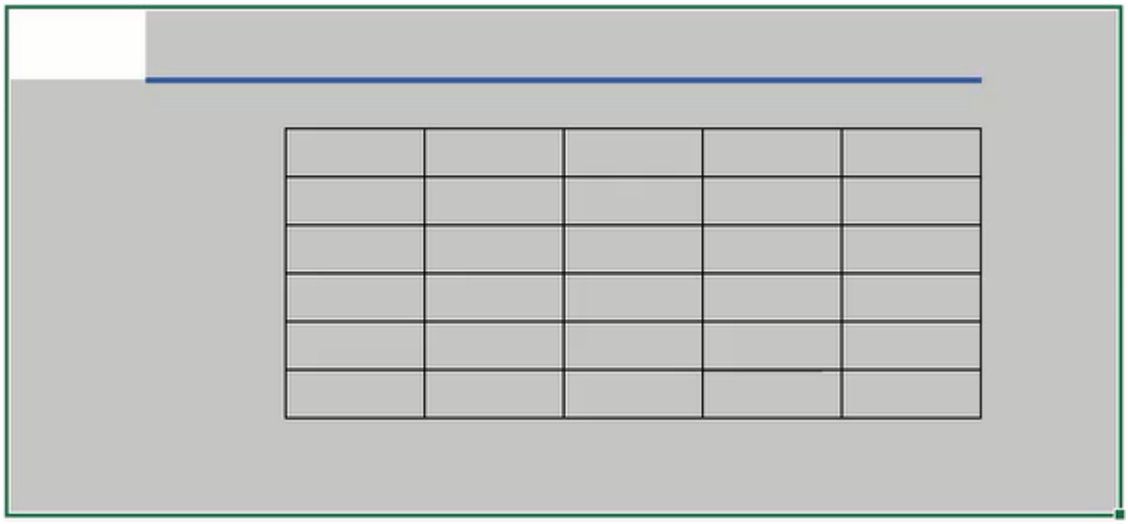

Figure 29.2

4. Go to the **Home** toolbar.
5. Press the **Clear** button and select **Clear Formats**. (See **Figure 29.3**)
6. Press the **Stop Recording** button from the **Developer** toolbar.

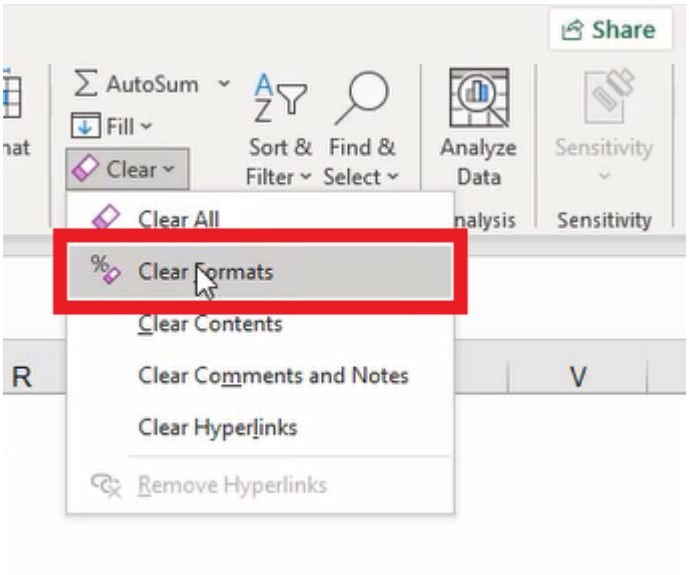

Figure 29.3

Visual Basic

Next, we'll look at the code composing the macros. Press the **Visual Basic** button from the Developer toolbar to open the code in Visual Basic. Here, we can review, edit, and create the code used in Excel macros. (See **Figure 29.4**)

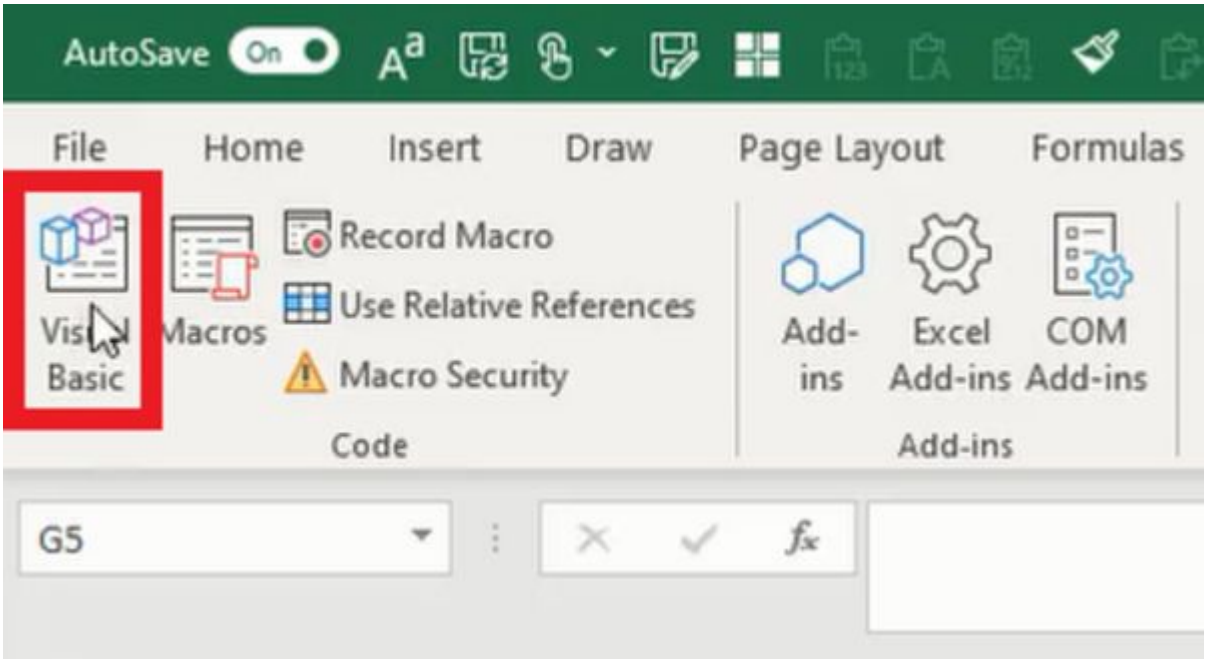

Figure 29.4

Organizing Code

Organizing code by inserting notes is an important practice for you and others to make sense of the code. Notes are written beginning with a single apostrophe (') which the program recognizes as separate from the macro code. If text is written without an apostrophe, the text will be colored red to indicate an error because it is not actual code.
(See **Figure 29.5**)

Figure 29.5

Editing Code

Recording macros can be helpful for completing repetitive tasks. However, the recorded macro will likely contain a lot of extra code as a result of selecting cells prior to performing a specific action. In **Figure 29.6** below, we can see each action begins with identifying rows, columns, or a range of cells and performing the action on the selected cell(s) (for example, **Range("E3:L12").Select**). As in **Figure 29.7**, the code can be simplified by removing the selection and performing the next action on the designated cell(s) directly (for example, **Range("E3:L12").ClearContents**).

We can take it a step further by simplifying the clear contents and clear formats lines. Instead of performing the actions to clear content and format separately, the **ClearContents** action can be changed to **Clear**, and the code will clear everything and the clear formats line can be deleted.

```
Sub Remove_Table()
' removes the table and clears formats

    ' ungroup columns
    Columns("I:J").Select
    Selection.Columns.Ungroup

    'clear contents
    Range("E3:L12").Select
    Selection.ClearContents

    'clear formats
    Selection.ClearFormats

    'correct row height
    Rows("3:3").Select
    Selection.RowHeight = 20
    Range("A1").Select
End Sub
```

Figure 29.6

```
Sub new_macro()
' removes the table and clears formats

    ' ungroup columns
    Columns("I:J").Columns.Ungroup

    'clear contents
    Range("E3:L12").Clear

    'correct row height
    Rows("3:3").RowHeight = 20
    Range("A1").Select

End Sub
```

Figure 29.7

Debugging Code

There will be situations wherein misspellings, changes, or other continuity problems between the macro code and the spreadsheet cause a macro to fail. In the event of an error, Visual Basic will produce an error message and prompt the user to **End** or **Debug** the macro. (See **Figure 29.8**)

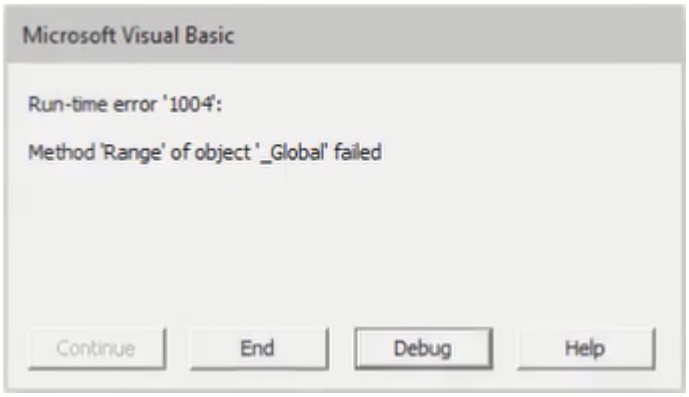

Figure 29.8

Pressing **Debug** will direct the user to the line of code that failed. The error will be highlighted yellow with an arrow indicating the line of code. In the following example, the macro attempted to select a range labeled as "abc." However, the range didn't exist. It may have been a typing error, or the user forgot to update the desired cell(s) with the indicated label. The code or the spreadsheet can be modified to correct the error. (See **Figure 29.9**)

```
Sub Create_A_Table()
' This macro will make a table


    'row headings
    Range("abc").Select
    ActiveCell.FormulaR1C1 = "A"
    Range("F6").Select
    ActiveCell.FormulaR1C1 = "B"
    Range("F7").Select
    ActiveCell.FormulaR1C1 = "C"
    Range("F8").Select
    ActiveCell.FormulaR1C1 = "D"
```

Figure 29.9

Supplemental Resource