Use Print Preview in VBA (With Examples)

Authored by

Mohammed looti

November 15, 2025

Optimizing Document Output with Print Preview in VBA for Excel

In professional environments utilizing Excel for critical reporting and data management, the integrity and presentation of printed documents are paramount. When dealing with complex spreadsheets, dynamic data tables, or multi-page financial reports, the process of manually adjusting print settings and repeatedly sending test prints to a physical device is both inefficient and costly. This is precisely where the automation capabilities offered by Visual Basic for Applications (**VBA**) become indispensable, specifically through the use of the **`PrintPreview`** method.

The primary function of the **`PrintPreview`** method is to provide a digital visualization of the document's final printed appearance before any physical resources are consumed. This essential functionality allows users to meticulously inspect critical formatting elements, such as pagination, scaling, header and footer placement, and overall layout fidelity. By integrating this method into your existing macros, you gain granular control over output quality, dramatically reducing errors and ensuring that every print job meets professional standards.

This comprehensive tutorial is designed to demystify the application of the **`PrintPreview`** method within the VBA environment. We will explore the different scopes under which this method operates--specifically, how to apply it to an entire worksheet versus a designated range of cells. By understanding these distinctions and implementing the practical examples provided, you will be equipped to streamline your printing workflow and ensure that your Excel documents are always perfectly prepared for distribution or archival.

Mastering the PrintPreview Method: Syntax and Scope

The **`PrintPreview`** method is a versatile command within the Excel VBA Object Model, primarily belonging to objects that represent printable content, such as `Worksheet`, `Chart`, `Range`, or `Selection` objects. Its core purpose remains consistent: to interrupt the direct printing process and instead display the output in the standard Excel Print Preview window. Understanding the object to which you apply this method dictates the scope of the preview.

There are two fundamental applications of the **`PrintPreview`** method that address the majority of user requirements. The first application involves the **`ActiveSheet`** object, which is utilized when the objective is to review the entire contents of the currently visible worksheet. This is typically used for general document reviews where the full context of the data, spanning potentially multiple pages, must be verified before printing. The second application, involving the **`Selection`** object, offers a targeted approach, focusing solely on user-defined data boundaries.

Choosing between these two methods depends entirely on the specific printing requirements of the task at hand. If the goal is to produce a complete, multi-page report, applying **`PrintPreview`** to the

`ActiveSheet` is the correct approach. Conversely, if you are generating a quick, focused summary or extracting only a specific table from a much larger sheet, using the **`Selection`** object ensures efficiency and prevents the inclusion of extraneous data, which is often crucial for clean report generation.

Practical Implementation: Previewing the Entire Active Sheet

When preparing a document where all data on the active sheet must be printed, the most efficient VBA technique involves leveraging the **`ActiveSheet`** object. This approach guarantees that the print preview includes every populated cell, along with all associated formatting, headers, footers, and page break configurations that have been established for that specific worksheet. It is the definitive method for performing a final quality check on a comprehensive document before production.

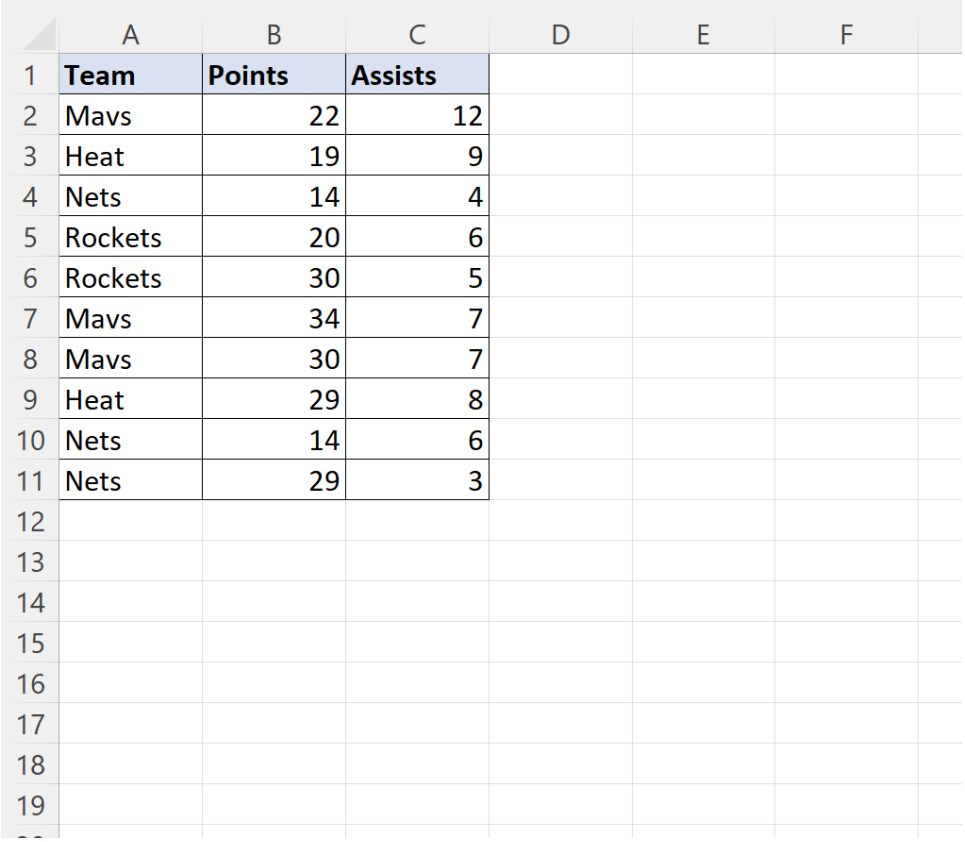
To execute this process, a straightforward macro is utilized, calling the **`PrintPreview`** method directly on the `ActiveSheet` object. This simple line of code eliminates the need for manual navigation to the print menu, integrating the preview functionality directly into an automated workflow. The following code snippet demonstrates the required structure for this operation:

```
Sub UsePrintPreview_EntireSheet()
ActiveSheet.PrintPreview
End Sub
```

Once this routine is executed, the standard Excel Print Preview interface will launch, displaying a precise replica of how the entire sheet will appear when printed. This functionality is crucial for verifying aspects such as proper page orientation (portrait or landscape), ensuring column widths are appropriate for the paper size, and confirming that automatic page breaks fall in logical locations, thereby maintaining the document's readability and professional structure.

Setting the Stage for Practical Demonstrations

To visualize the impact of the different **`PrintPreview`** applications, we will use a common scenario: a simple dataset contained within an active Excel worksheet. This visual example allows us to clearly contrast the output generated when previewing the entire sheet versus previewing only a selected subset of the data. This foundational setup is critical for understanding the practical utility of scoping the print preview action.



	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	12			
3	Heat	19	9			
4	Nets	14	4			
5	Rockets	20	6			
6	Rockets	30	5			
7	Mavs	34	7			
8	Mavs	30	7			
9	Heat	29	8			
10	Nets	14	6			
11	Nets	29	3			
12						
13						
14						
15						
16						
17						
18						
19						

As illustrated in the preceding image, our active sheet contains a matrix of data. We will use this visual context to walk through two distinct examples. The first will show the comprehensive preview generated by the `ActiveSheet` command, and the second will demonstrate the focused output achieved by applying the **`PrintPreview`** method to a specific range, utilizing the `Selection` object, thus providing concrete evidence of the method's behavior under various conditions.

When the `ActiveSheet.PrintPreview` command is executed on the sample data, the resulting window confirms that all elements, including the surrounding white space and any data extending beyond the visible viewport, are included in the print job calculation.

Team	Points	Assists
Mavs	22	12
Heat	19	9
Nets	14	4
Rockets	20	6
Rockets	30	5
Mavs	34	7
Mavs	30	7
Heat	29	8
Nets	14	6
Nets	29	3

This comprehensive view, as shown above, offers invaluable confirmation regarding the document's integrity. Users can confirm that the data is not truncated and that all established page setup parameters, such as margins and scaling factors, are being correctly applied before

proceeding with the costly physical printing step.

Targeted Printing: Utilizing Print Preview for Specific Selections

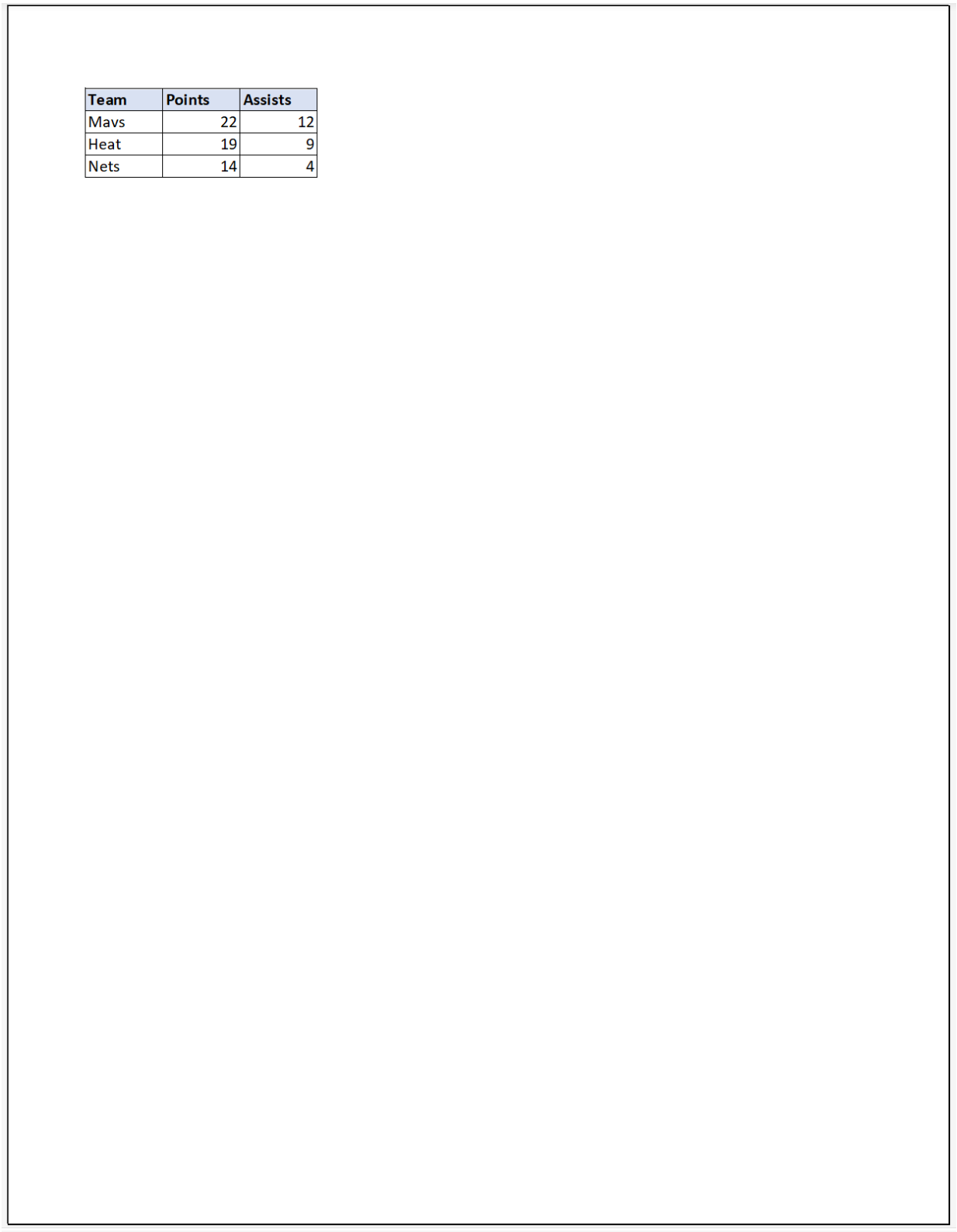
A frequent requirement in data analysis and reporting is the need to print only a highly specific section of a worksheet, perhaps a single table or a summary range, while deliberately excluding surrounding data or calculations. The `Selection.PrintPreview` method, when utilized within VBA, offers the ideal solution for this targeted printing need, ensuring that only the relevant information is included in the final output. This capability significantly enhances document production efficiency.

The prerequisite for using this method is that a specific range of cells must be actively selected by the user before the macro is executed. The `Selection` object dynamically refers to whatever range the user has highlighted. By combining the `Selection` object with the `PrintPreview` method, the VBA code instructs Excel to calculate the print job exclusively based on that highlighted area, ignoring all other data on the sheet.

The code required for this operation is structurally identical to the `ActiveSheet` example, differing only in the object reference. This consistency simplifies implementation and maintenance for developers utilizing the VBA Object Model:

```
Sub UsePrintPreview_Selection()
Selection.PrintPreview
End Sub
```

Upon executing this macro after selecting, for instance, only cells A1:C10 in the sample data, the resulting print preview window will display content limited strictly to that range. This focused output is invaluable for creating concise reports or extracting specific data sections without the clutter of extraneous worksheet information, thereby enhancing clarity and professionalism.



Team	Points	Assists
Mavs	22	12
Heat	19	9
Nets	14	4

As evidenced in the image above, the print preview confirms that only the selected data range is considered for printing. This targeted capability is a cornerstone of efficient document management, allowing users to precisely define the scope of their output and minimize wasted

effort and materials.

Advanced Workflow Enhancements and Best Practices

While the basic **`PrintPreview`** implementations covered are highly effective, the true power of this VBA method emerges when it is combined with programmatic control over page setup parameters. For sophisticated workflows, developers often precede the **`PrintPreview`** command with code that dynamically adjusts margins, sets the paper orientation, enforces specific scaling percentages, or defines custom headers and footers based on dynamic data.

For instance, a macro could first check the dimensions of the data range and then programmatically set the `ActiveSheet.PageSetup.Orientation` to landscape if the data is wide, ensuring optimal layout before calling **`PrintPreview`**. This automated configuration eliminates manual setup steps and guarantees consistent formatting across different worksheets or reports generated by the same system. Integrating such programmatic adjustments transforms the **`PrintPreview`** method from a simple visualization tool into a core component of a fully automated printing solution.

For developers seeking to explore the full depth of this functionality, consultation of the official Microsoft documentation for the **`PrintPreview`** method is essential. This resource provides detailed information on optional arguments and related properties, such as `PrintOut` and the various members of the `PageSetup` object, enabling the creation of highly customized and resilient printing routines. Mastering these advanced techniques ensures that your VBA solutions are robust and highly efficient, especially in environments requiring frequent, high-volume document generation from Excel.

To further enhance your VBA skills and explore other common tasks related to printing and document management in Excel, consider reviewing the following tutorials:

VBA: How to Print to PDF