Learn How to Return All Matching Values with XLOOKUP and FILTER in Excel

Authored by

Mohammed looti

June 3, 2026

The Default Behavior and Limitations of XLOOKUP

The introduction of the **XLOOKUP** function revolutionized how users perform lookups within **Excel**, offering a powerful, flexible alternative to older functions like VLOOKUP and HLOOKUP. By design, **XLOOKUP** is optimized to locate a specific value within a defined lookup range and return the corresponding data point from an adjacent return range. However, it is crucial to understand that its standard configuration dictates that it will return a corresponding value *only for the first match* encountered. This inherent limitation becomes apparent when working with datasets containing non-unique identifiers where a single search criterion might relate to multiple distinct results. When data integrity relies on capturing all associated records--such as finding all transaction dates for a specific customer ID or all scores associated with a particular team name, as in our upcoming example--relying solely on **XLOOKUP** will lead to incomplete data retrieval, making the function unsuitable for scenarios requiring a complete list of related entries.

While **XLOOKUP** excels in simple, singular lookups, data analysts frequently encounter situations requiring the extraction of an entire array of results. Prior to the widespread implementation of dynamic array capabilities, achieving this required complex workarounds involving INDEX/MATCH combinations, traditional array formulas (requiring Ctrl+Shift+Enter), or even pivot tables. These legacy methods often proved cumbersome, difficult to maintain, and prone to error. Understanding that **XLOOKUP** stops processing the moment it identifies the initial match is key to appreciating why a different approach is necessary for comprehensive data extraction. This constraint forces the user to seek out functions specifically designed to handle and spill multiple results simultaneously, moving beyond the traditional single-cell output paradigm that dominated older versions of spreadsheet software.

Introducing the Dynamic Array Solution: The FILTER Function

To effectively address the challenge of returning multiple corresponding values in modern **Excel** environments, we turn to the **FILTER** function. This function is a cornerstone of Excel's dynamic array capabilities, allowing users to effortlessly filter a range of data based on criteria provided by a boolean array. Unlike the traditional lookup functions, which are constrained to a single output cell, **FILTER** is designed to 'spill' its results across multiple cells, automatically adjusting the output range as the underlying data changes. This makes it the definitive tool for performing a lookup that returns *all matches*, effectively overcoming the primary limitation of **XLOOKUP** in this context.

The basic syntax of the **FILTER** function requires three arguments: the array to return (what data you want to see), the include criterion (the logical test defining which rows should be included), and an optional 'if_empty' argument. When used to mimic a multi-match lookup, the structure is elegantly simple, focusing on matching a specified lookup value against an entire column. Consider the standard formula structure used to achieve this crucial multi-match extraction:

=FILTER(C2:C11, E2=A2:A11)

In this specific example, the formula instructs **Excel** to look at the desired return range, specified here as **C2:C11** (which typically holds the output values, like 'Points'). The crucial second argument, **E2=A2:A11**, generates a logical array of TRUEs and FALSEs. A TRUE is generated for every row where the value in cell **E2** (our lookup criteria) matches the corresponding value in the lookup column **A2:A11**. The **FILTER** function then intelligently uses this logical array to extract and return all corresponding values from **C2:C11** where the condition evaluated to TRUE. This streamlined approach replaces several lines of code from legacy systems and provides a robust, real-time solution for dynamic data extraction.

Practical Demonstration: Setting Up the Dataset

To illustrate the fundamental difference between the single-match nature of **XLOOKUP** and the multi-match capability of **FILTER**, we will work with a specific sample dataset. Suppose we have compiled performance data for various basketball teams, where teams might appear multiple times across different records or games, indicating non-unique entries. Our dataset, organized into columns for Team, Date, and Points, is structured clearly, allowing us to focus on data retrieval based on the team name:

	A	B	C	D	E	F
1	**Team**	**Rebounds**	**Points**			
2	Mavericks	12	22			
3	Pacers	14	25			
4	Pacers	8	24			
5	Hornets	7	24			
6	Rockets	11	25			
7	Pacers	19	19			
8	Rockets	15	15			
9	Nets	14	24			
10	Rockets	10	30			
11	Hornets	12	34			
12						
13						
14						
15						
16						
17						
18						

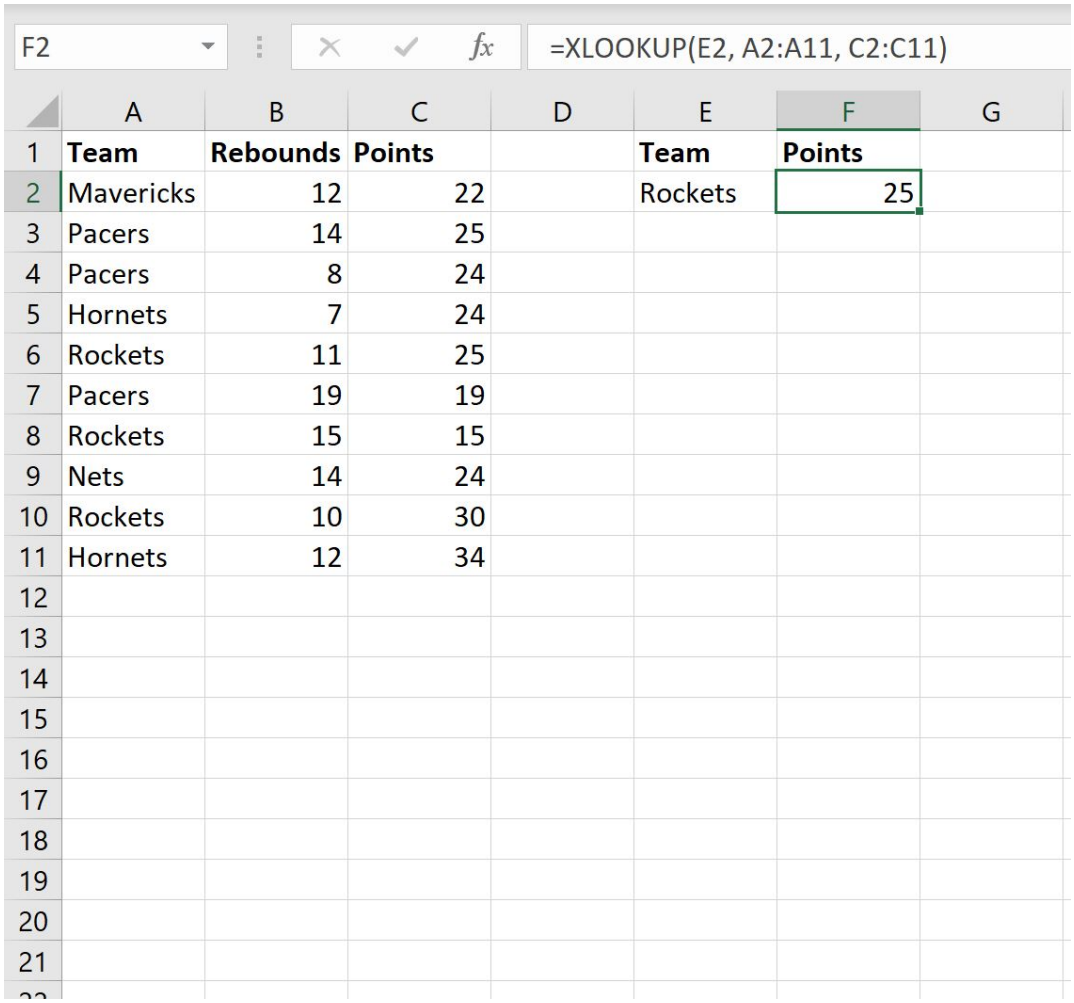
Our specific analytical objective is to retrieve all 'Points' values associated with a single team name, designated as "Rockets." A quick inspection of the data reveals that "Rockets" appears three times within the 'Team' column (Column A). This scenario perfectly highlights why a traditional lookup function will inevitably fall short and why a modern dynamic array solution is mandated. The need for comprehensive data retrieval, rather than merely the first successful record, drives the choice of function in modern spreadsheet analytics, emphasizing the importance of utilizing tools capable of handling redundant keys effectively.

Analyzing the XLOOKUP Default Behavior

When attempting to retrieve all associated points using the **XLOOKUP** function, we define the search criteria (Team Name in E2), the lookup array (Column A), and the return array (Column C). We are instructing the function to search for the value in E2 within the range A2:A11 and return the corresponding value from C2:C11. The formula used would be constructed as follows, respecting the standard syntax requirements for this powerful function:

=XLOOKUP(E2, A2:A11, C2:C11)

Executing this formula yields a result, but critically, it only returns the point value corresponding to the very first instance of "Rockets" found within the dataset. The screenshot below clearly demonstrates this outcome, where only the score 110 is returned in the output cell, neglecting the subsequent scores (101 and 105) that are also associated with the same team name. This behavior, while intended for efficiency in single-match scenarios, proves inadequate for comprehensive data reporting when duplicates exist.



	A	B	C	D	E	F	G
1	**Team**	**Rebounds**	**Points**		**Team**	**Points**	
2	Mavericks	12	22		Rockets	25	
3	Pacers	14	25				
4	Pacers	8	24				
5	Hornets	7	24				
6	Rockets	11	25				
7	Pacers	19	19				
8	Rockets	15	15				
9	Nets	14	24				
10	Rockets	10	30				
11	Hornets	12	34				
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							

This failure to return comprehensive results underscores the inherent design limitation of **XLOOKUP** when confronted with duplicate keys. It is crucial for users to recognize that **XLOOKUP** is optimized for unique identifiers or scenarios where only the initial record holds relevance. For data aggregation and complete record extraction, relying on **XLOOKUP** is counterproductive, necessitating the adoption of functions capable of handling and spilling an array of results. The subsequent records containing "Rockets" are simply ignored once the first successful match is identified, confirming the need for a dynamic array function to achieve the goal of returning all corresponding data points.

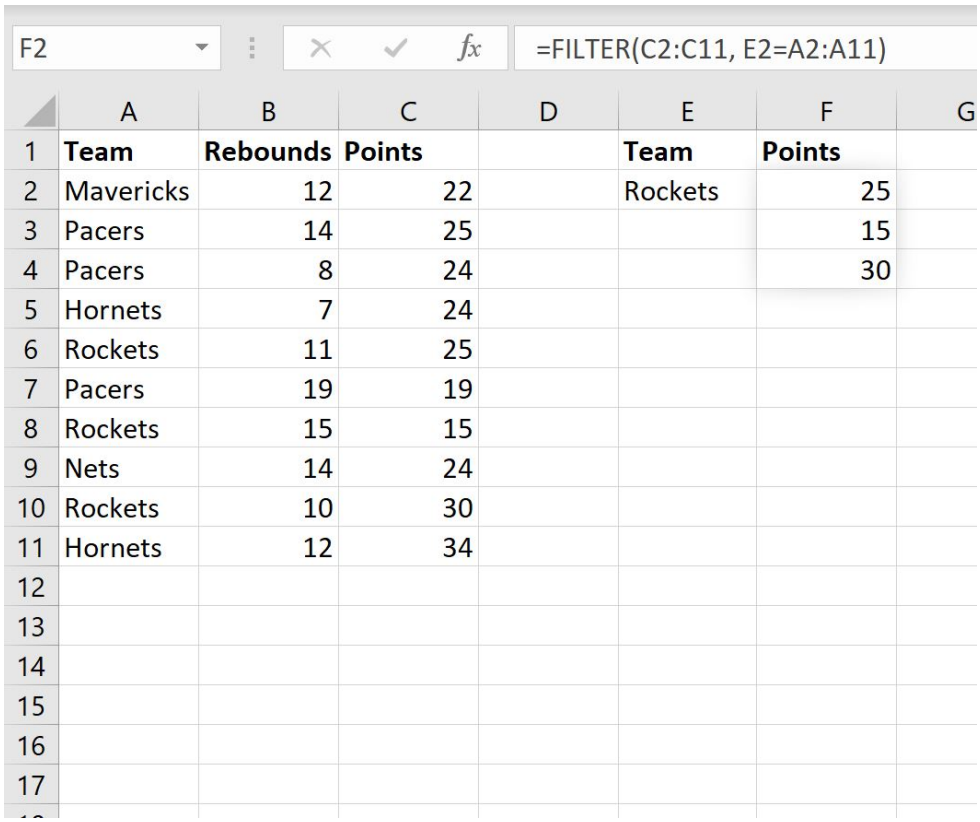
Achieving Multiple Matches Using the FILTER Function

To successfully extract all associated point values for the team "Rockets," we must deploy the **FILTER** function. As discussed, **FILTER** is specifically engineered to handle multiple matches by generating a dynamic array output. Instead of searching for the first match and stopping, **FILTER** evaluates the entire range, compiling a list of all successful matches based on the specified criteria. By inputting the Points column (C2:C11) as the array to return and the logical test

matching E2 against the Team column (A2:A11) as the include argument, we create a formula that captures every instance of the criteria:

=FILTER(C2:C11, E2=A2:A11)

Upon execution, the results automatically spill down the column, starting from the cell where the formula is entered. This output mechanism is a defining characteristic of **dynamic arrays** in modern **Excel**. The resulting output accurately lists all three scores--110, 101, and 105--demonstrating that **FILTER** successfully located and returned all corresponding values for the rows where the "Team" column contained "Rockets." This result directly addresses the requirement for complete data extraction, solving the problem inherent in the single-match nature of **XLOOKUP**.



	A	B	C	D	E	F	G
1	**Team**	**Rebounds**	**Points**		**Team**	**Points**	
2	Mavericks	12	22		Rockets	25	
3	Pacers	14	25			15	
4	Pacers	8	24			30	
5	Hornets	7	24				
6	Rockets	11	25				
7	Pacers	19	19				
8	Rockets	15	15				
9	Nets	14	24				
10	Rockets	10	30				
11	Hornets	12	34				
12							
13							
14							
15							
16							
17							

The visual representation confirms the effectiveness of the **FILTER** function. Notice that the formula returns all three points values for the three rows where the "Team" column matched the lookup criterion. Furthermore, the **FILTER** function is highly adaptable; if the underlying data changes--for example, if a fourth record for "Rockets" is added or an existing record is deleted--the spilled array will automatically update and expand or contract, ensuring data integrity without manual intervention. This adaptability is the core reason why **FILTER**, alongside other dynamic array functions, has become the preferred method for complex data manipulation and retrieval in contemporary spreadsheet analysis, offering both power and resilience against data volatility.

	A	B	C	D	E	F
1	**Team**	**Rebounds**	**Points**		**Team**	**Points**
2	Mavericks	12	22		Rockets	25
3	Pacers	14	25			15
4	Pacers	8	24			30
5	Hornets	7	24			
6	Rockets	11	25			
7	Pacers	19	19			
8	Rockets	15	15			
9	Nets	14	24			
10	Rockets	10	30			
11	Hornets	12	34			
12						
13						
14						
15						
16						
17						
18						

Comparison and Conclusion: The Power of Dynamic Arrays

The comparison between **XLOOKUP** and **FILTER** is not about determining which function is inherently superior, but rather understanding which function is appropriate for the specific data retrieval task at hand. **XLOOKUP** remains the most efficient choice for single-match lookups, effectively replacing complicated nested formulas. However, when the requirement shifts to extracting a subset of data based on a criterion that yields multiple results, the **FILTER** function provides a clear, concise, and dynamic solution. This capability is deeply integrated with the concept of **dynamic arrays**, which fundamentally changed how formulas handle and return multiple values to adjacent cells automatically.

The primary advantage of using **FILTER** for multi-match lookups is the elimination of legacy array entry requirements and complex indexing logic previously needed for this task. The formula is intuitive, closely mirroring a simple logical filtering operation: "Show me this range, where this condition is met." This simplicity greatly improves the readability and maintainability of complex spreadsheets, reducing the cognitive load on the analyst. Furthermore, the automatic spilling behavior means that the output range is always perfectly sized to fit the results, preventing errors related to insufficient output space or the inclusion of irrelevant blank cells. This shift in methodology represents a significant advancement in data management within **Excel**, ensuring data analysts can retrieve comprehensive results with minimal effort and maximum reliability.

Additional Resources for Advanced Excel Operations

To further enhance proficiency in dynamic data manipulation, we highly recommend exploring related functionalities and complex operations utilizing these modern functions. Understanding the full potential of **dynamic arrays** is essential for moving beyond basic spreadsheet tasks and mastering advanced data retrieval techniques that are scalable and robust.

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

Related: Exploring the intricacies of conditional array processing and complex filtering criteria.

Further Reading: Techniques for combining **FILTER** with functions like SORT, UNIQUE, or SORTBY for custom output ordering.

Mastering Lookups: When to choose **XLOOKUP** versus **FILTER** in diverse data environments, including fuzzy matching scenarios.