

# How to Easily Retrieve Data with Excel's DGET Function

Authored by  
**stats writer**

February 23, 2026

## RECOMMENDED CITATION

stats writer (2026). *How to Easily Retrieve Data with Excel's DGET Function*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=132368>

The **DGET** function in **Microsoft Excel** is a sophisticated and specialized tool designed to extract a specific piece of information from a **database** or a structured table of data. Unlike standard lookup functions that may return multiple results or the first match they encounter, this function is engineered to retrieve exactly one value that satisfies a unique set of predefined conditions. It is particularly effective when working with large-scale **spreadsheet** environments where precision is paramount, as it allows users to pinpoint data without the manual labor of filtering, sorting, or scrolling through thousands of rows. By leveraging a dedicated criteria range, users can perform complex queries that would otherwise require nested formulas or advanced filtering techniques.

To implement the **DGET** function successfully, a user must organize their **data set** with a clear, tabular structure that includes descriptive column headings at the very top. This structure is vital because the function relies on these headers to identify where to search and what to retrieve. Once the data is prepared, the user defines a separate criteria range--essentially a mini-table--that specifies the conditions the desired record must meet. This range can include various **logical expressions** and operators, such as "greater than" (>), "less than" (<), or "not equal to" (≠), enabling highly granular searches. This level of control makes it a preferred choice for power users who need to automate data extraction based on dynamic, multi-layered requirements.

The versatility of the **DGET** function extends across numerous professional domains, providing significant value in fields like **financial analysis**, **inventory management**, and **data reporting**. For instance, a financial analyst might use it to extract the specific revenue of a particular branch for a single fiscal quarter, while an inventory manager might use it to find the stock level of a unique SKU across multiple warehouses. By ensuring that only a single, accurate result is returned, the function helps maintain **data integrity** and reduces the risk of errors associated with manual data entry or incorrect lookup matches. Ultimately, it serves as an essential component of any robust data management strategy within the Excel ecosystem.

## Mastering the DGET Function in Excel: A Comprehensive Guide with Practical Examples

### Understanding the Fundamental Mechanics of the DGET Syntax

The **DGET** function is a member of the "Database" family of functions in **Microsoft Excel**, which are specifically designed to treat ranges of cells as if they were a relational **database**. This function is used to isolate a single value from a specific column when the row meets one or more distinct conditions defined in a criteria range. Because it is built to find unique records, it serves as a powerful alternative to **VLOOKUP** or **XLOOKUP** when your search parameters are more complex than a single key-value pair.

The **syntax** for the function follows a clear, three-argument structure:

## **DGET(database, field, criteria)**

To use this function effectively, it is essential to understand exactly what each argument requires for the formula to calculate correctly:

**database:** This represents the entire range of cells that contains your data, including the top row of column headers. It is the "source of truth" where Excel will look for both the criteria matches and the value to return.

**field:** This argument indicates which column contains the value you wish to extract. You can provide this as the text name of the column header enclosed in double quotes (e.g., "Points") or as a number representing the column's position within the database range.

**criteria:** This is a separate range of cells that specifies the conditions. This range must include at least one column header and at least one cell below it specifying the value or condition to search for.

A critical aspect of the **DGET** function is its strict requirement for uniqueness. If the **search criteria** result in more than one row being identified, the function will not return a value; instead, it will trigger a **#NUM** error. Conversely, if no records match the criteria at all, the function returns a **#VALUE!** error. This behavior ensures that the user is always aware of whether their query is specific enough to identify a single, unique data point.

## **Visualizing the Dataset: Basketball Performance Metrics**

To illustrate the practical utility of the **DGET** function, we will utilize a sample dataset containing performance statistics for several basketball players. This dataset includes columns for "Team," "Position," "Points," and "Rebounds." Such data is common in **statistical analysis** and serves as an excellent foundation for learning how to query specific records from a larger pool of information.

	A	B	C	D	E	F
1						
2						
3						
4						
5	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>		
6	Mavs	22	4	8		
7	Mavs	20	6	10		
8	Spurs	39	5	12		
9	Kings	19	3	5		
10	Rockets	15	8	8		
11	Spurs	14	12	12		
12	Lakers	22	5	5		
13	Mavs	25	7	2		
14	Rockets	28	6	4		
15	Rockets	30	2	9		
16	Nets	32	8	13		
17						
18						
19						

As seen in the image above, the data is organized into a clean table with headers in row 5. The information spans from cells A5 to D16. This structure allows Excel to easily map the "Team" or "Points" labels to their respective columns. When setting up your own **database** in Excel, maintaining this level of organization is the most important step in preventing **logic errors** within your database functions.

### Example 1: Executing a Basic Search with a Single Condition

In our first scenario, let's assume we need to find the number of points scored by a player on a specific team. Specifically, we want to know the "Points" value for the record where the "Team" column is exactly "Lakers." This is a straightforward lookup that demonstrates how the criteria range interacts with the main database.

First, we define our criteria in a separate area of the **spreadsheet**. In this example, we have placed the headers in row 2 and the search values in row 3 (Range A2:D3). By typing "Lakers" under the "Team" header in cell A3, we instruct Excel to filter the database for that specific string. We then enter the following **code** into cell G2:

**=DGET(A5:D16, "Points", A2:D3)**

The screenshot below captures the implementation of this formula and the resulting output:

	A	B	C	D	E	F	G
1							
2	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>		<b>Points Value</b>	22
3	Lakers						
4							
5	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>			
6	Mavs	22	4	8			
7	Mavs	20	6	10			
8	Spurs	39	5	12			
9	Kings	19	3	5			
10	Rockets	15	8	8			
11	Spurs	14	12	12			
12	Lakers	22	5	5			
13	Mavs	25	7	2			
14	Rockets	28	6	4			
15	Rockets	30	2	9			
16	Nets	32	8	13			
17							
18							

The function successfully scans the "Team" column within the range A5:D16, finds the single row where the team is "Lakers," and returns the value **22** from the "Points" column. This demonstrates the efficiency of **DGET** when dealing with distinct, identifiable records. It is a clean way to pull data into a summary **dashboard** or report without cluttering the sheet with complicated filter settings.

### Troubleshooting the #NUM! Error: Handling Non-Unique Results

While **DGET** is powerful, it is also highly sensitive to the uniqueness of your data. If you attempt to use the function to retrieve a value where the criteria are not specific enough--meaning multiple rows match the condition--Excel will be unable to choose which value to return. This leads to the **#NUM** error code, which acts as a safeguard against returning ambiguous or incorrect data.

	A	B	C	D	E	F	G
1							
2	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>		<b>Points Value</b>	<b>#NUM!</b>
3	Rockets						
4							
5	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>			
6	Mavs	22	4	8			
7	Mavs	20	6	10			
8	Spurs	39	5	12			
9	Kings	19	3	5			
10	Rockets	15	8	8			
11	Spurs	14	12	12			
12	Lakers	22	5	5			
13	Mavs	25	7	2			
14	Rockets	28	6	4			
15	Rockets	30	2	9			
16	Nets	32	8	13			
17							
18							

In the instance shown above, we changed the criteria to search for the "Rockets." Looking at the database, we can see that there are two separate entries for the Rockets team. Because the **DGET** function cannot determine which of these two records you are interested in, it generates the **#NUM** error. This highlights the primary difference between **DGET** and a function like **VLOOKUP**, which would simply return the first match it found. To resolve this, the user must add more criteria to narrow the search down to one specific row.

## Example 2: Refining Queries with Multiple Logical Conditions

One of the most compelling reasons to use **DGET** is its ability to handle multiple conditions simultaneously using **Boolean logic**. By adding values under multiple headers in the criteria range, you create an "AND" condition. This means the function will only return a result if a row matches every single condition specified in the criteria row. This is perfect for differentiating between players on the same team based on their individual stats.

Suppose we want to find the number of "Rebounds" for a player on the "Rockets" who also scored fewer than 20 points. We set up our criteria range (A2:D3) by entering "Rockets" under the "Team" header and the expression "<20" under the "Points" header. We then apply the following formula:

**=DGET(A5:D16, "Rebounds", A2:D3)**

The resulting application is displayed in the following screenshot:

	A	B	C	D	E	F	G
1							
2	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>		<b>Points Value</b>	8
3	Rockets	<20					
4							
5	<b>Team</b>	<b>Points</b>	<b>Assists</b>	<b>Rebounds</b>			
6	Mavs	22	4	8			
7	Mavs	20	6	10			
8	Spurs	39	5	12			
9	Kings	19	3	5			
10	Rockets	15	8	8			
11	Spurs	14	12	12			
12	Lakers	22	5	5			
13	Mavs	25	7	2			
14	Rockets	28	6	4			
15	Rockets	30	2	9			
16	Nets	32	8	13			
17							
18							

By applying these dual constraints, the function ignores the Rockets player who scored 28 points and focuses exclusively on the player who scored 15 points. Consequently, it returns the value **8**, which is the correct number of rebounds for that specific individual. This capability makes **DGET** an invaluable tool for complex **data analysis** where multiple variables must be considered to identify a specific record.

## Comparing DGET with Modern Lookup Alternatives

In modern versions of **Microsoft Excel**, users often have a choice between several different functions for data retrieval. While **VLOOKUP** is the most famous, it is limited by its inability to look to the left and its tendency to return the first match without warning if duplicates exist. **INDEX-MATCH** is more flexible but requires a more complex formula structure. **XLOOKUP** has recently become the standard for many due to its simplicity and power.

However, **DGET** maintains a unique niche because of the criteria range. The ability to see your

search conditions clearly printed in cells (the criteria range) rather than hidden inside a long string of formula **syntax** makes your **spreadsheet** much easier to audit and troubleshoot. Furthermore, the built-in error handling for non-unique values is a feature that other functions lack, providing an extra layer of **quality assurance** for your data reports. When you must guarantee that you are looking at a unique entry in a database, **DGET** is often the superior choice.

## Best Practices for Building Robust Database Formulas

To ensure your **DGET** formulas remain accurate and your workbooks remain performant, there are several best practices to follow. First, always ensure that your column headers in the criteria range are an exact match--including spelling and spacing--to the headers in your main database. While Excel is generally not case-sensitive in this context, extra spaces can cause the function to fail. Using **data validation** drop-down lists in your criteria range can help maintain consistency and prevent typing errors.

Second, consider using **structured references** by converting your database into a formal Excel Table (Ctrl+T). This allows your **DGET** formula to automatically expand if you add more rows to your data, ensuring that your queries always cover the entire **database** without needing to manually update range coordinates. By combining these professional techniques with the inherent power of the **DGET** function, you can build dynamic, reliable, and highly efficient data management systems in Excel.