A Comprehensive Guide to INDEX MATCH with Partial Text Matching in Excel

Authored by

Mohammed looti

November 13, 2025

In the expansive and data-rich environment of Microsoft Excel, mastering efficient data retrieval techniques is crucial for sophisticated analysis and reporting. While traditional functions like VLOOKUP are commonplace, the dynamic pairing of the **INDEX function** and the **MATCH function** offers superior flexibility and power. This powerful combination is particularly valuable when your lookup criteria involves finding a target cell that contains only a partial text string or a specific **substring**, moving beyond the limitation of requiring an exact, character-for-character match.

This comprehensive tutorial is dedicated to exploring the methodology of employing the **INDEX MATCH** combination to accurately locate and return a corresponding value from a data range based on a partial text match. We will thoroughly examine the necessary syntax, highlight the indispensable role of **wildcard characters** in enabling this flexibility, and walk through a detailed practical example. By the end of this guide, you will possess the specialized knowledge required to perform dynamic, highly adaptable lookups that dramatically enhance your data manipulation capabilities in Excel, regardless of how messy or incomplete your source data might be.

Deconstructing the Power Duo: INDEX and MATCH Functions

Before diving into the integration required for partial text searching, it is fundamental to grasp the independent capabilities of the **INDEX** and **MATCH** functions. Their joint power stems from the fact that they address different parts of the lookup problem: **MATCH** locates the position, and **INDEX** retrieves the data. This combined synergy creates a robust and versatile solution that consistently outperforms single-function alternatives, especially when dealing with complex datasets or non-standard search criteria.

The **INDEX function** serves as the final retriever of the data. Its primary role is to fetch a value or a reference to a value from a specified table or array. It requires defining the boundaries of the search area (the **array**), and crucially, specifying the numerical position of the row (the **row_num**) and, optionally, the column (the **column_num**) from which the value should be returned. For instance, if you write `INDEX(D1:F10, 5, 3)`, Excel will return the value found at the intersection of the 5th row and the 3rd column within that D1:F10 search array. It is essential to recognize that **INDEX** relies entirely on receiving a numerical position, which is precisely the information the **MATCH function** provides.

Conversely, the **MATCH function** is the dynamic locator; it determines the relative position of a specific item within a one-dimensional range of cells. It takes three arguments: the **lookup_value** (what you are searching for), the **lookup_array** (where you are searching), and the **match_type**. For almost all exact or partial text lookups, the **match_type** must be set to 0, which signals **MATCH** to look for an **exact match** to the specified criteria or pattern. When combined, **MATCH** dynamically generates the **row_num** argument needed by **INDEX**, transforming a static data

retrieval method into a dynamic lookup system capable of handling complex search patterns.

Achieving Flexibility with Combined INDEX and MATCH

The true architectural advantage of the **INDEX MATCH** structure over legacy functions is its unrestricted nature and ability to look up data in any direction. Unlike **VLOOKUP**, which is fundamentally limited to searching the leftmost column and can only return values from columns to the right, **INDEX MATCH** can search any column in your dataset and retrieve data from any other column, regardless of their relative positioning. This crucial flexibility is what has established it as the preferred method for advanced data manipulation tasks in Excel, especially when dealing with data structures that cannot be rearranged.

In a combined formula, the **MATCH function** executes its role first, scanning a designated column to find the exact relative row position of the item being sought. This numerical row position is then seamlessly passed as the primary argument (the **row_num**) to the **INDEX function**. The **INDEX function** uses this position, alongside a specified column number, to precisely extract the desired value from the larger data range. This clear division of labor--locating by **MATCH** and retrieving by **INDEX**--is the source of its superior adaptability and efficiency.

The requirement for partial matching arises frequently, especially when dealing with inconsistent data, descriptive fields, or when only a keyword or a **substring** is known, not the entire cell contents. To bridge the gap between a standard exact lookup and a flexible partial lookup, we must integrate **wildcard characters** directly into the **MATCH** component's **lookup_value**. This transformation enables the lookup to successfully identify cells that contain the target text anywhere within their content, dramatically improving the utility of the lookup.

Implementing Wildcard Characters for Partial Lookups

The crucial technique for instructing Excel to search for text that "contains" a specific string--rather than strictly equaling it--lies in utilizing Excel's powerful **wildcard characters**. These special symbols act as flexible placeholders, allowing you to define broad search patterns. Within the context of text lookups using the **MATCH function**, the most essential wildcard for partial matching is the **asterisk (*)**.

The **asterisk (*)** is defined as a placeholder that represents any sequence of characters, including zero characters. This feature makes it perfect for enabling a partial match. By placing an asterisk before and after your desired **substring**, you instruct Excel to locate any text string that includes that substring at any point (beginning, middle, or end). For example, if your **lookup_value** is defined as `"*Proj*"`, the **MATCH function** will successfully match cells containing "Project Alpha," "Initial Project Scope," or even just "Proj."

It is important to note the distinction with the **question mark (?)**, which is also a useful wildcard but represents only a single character. For instance, `"r?n"` would match "run" or "ran" but would fail to match "rain" because "rain" requires two characters between 'r' and 'n'. When incorporating these wildcards into the **MATCH function**, it remains mandatory to set the **match_type** argument to 0 (exact match). This configuration ensures that the function correctly processes the wildcard pattern as the precise criterion for the lookup, thus treating the pattern itself as the 'exact' thing it must find.

Dissecting the Core Formula Structure

The fundamental syntax for deploying **INDEX** and **MATCH** to perform a lookup where the target cell contains specific text is a structured and highly effective formula. This structure enables the identification of a relevant row based on a flexible text criterion and the subsequent, precise retrieval of data from a corresponding column.

We will analyze the following formula structure, which serves as the blueprint for partial text lookups. It aims to find the result in the second column based on a partial match in the first column:

=INDEX(A2:B11,MATCH("*Warr*",A2:A11,0),2)

Let's systematically break down the roles of each component within this powerful formula:

The INDEX Component: `INDEX(A2:B11, ... , 2)`:

`A2:B11` defines the **array**, which is the complete data range containing all potential results. The **INDEX function** will retrieve the final output from within this specific range.
The final argument, `2`, is the **column_num**. This specifies that the desired value should be returned from the second column of the specified **array** (A2:B11). It is vital to remember this number is relative to the array's starting column (Column A is 1, Column B is 2), not the entire worksheet.

The MATCH Component: `MATCH("*Warr*", A2:A11, 0)`:

`"*Warr*"` is the **lookup_value**. The surrounding **wildcard characters** (asterisks) instruct the **MATCH function** to find any cell containing the **substring** "Warr," regardless of surrounding text.
`A2:A11` is the **lookup_array**. This is the single column where the **MATCH function** performs the linear search for the specified lookup value pattern.
`0` is the **match_type**, which mandates an **exact match** to the pattern defined by the text and wildcards.

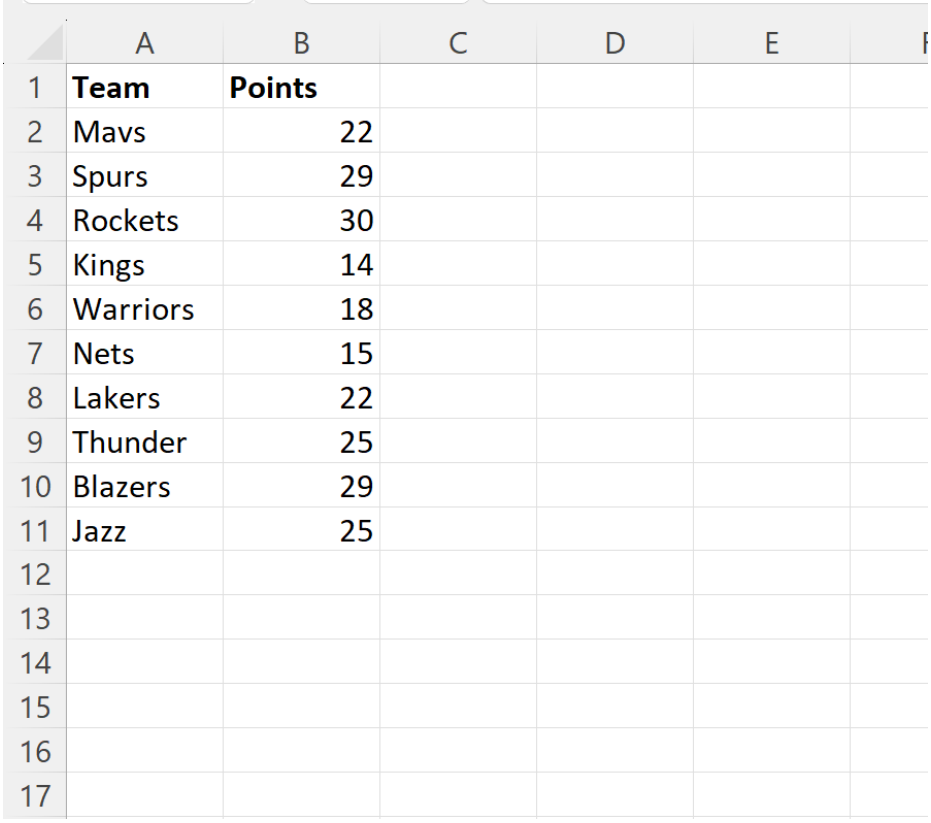
In practice, the **MATCH function** executes first, returning the relative row number where the partial match is found. This number is then fed into **INDEX**, which uses it to locate and retrieve the

corresponding data from the desired result column. This elegant synergy facilitates highly specific data retrieval based on flexible, non-exact criteria.

Step-by-Step Practical Application in Excel

To fully grasp the application of the **INDEX MATCH** formula for partial text lookups, let us work through a practical scenario using a common Excel dataset. This example will clearly demonstrate how to extract specific information when only a portion of the identifying text is available, mirroring real-world data cleanup challenges.

Consider a simple dataset tracking basketball team names and their corresponding scores. Our objective is to efficiently locate the score associated with a team name that includes a particular **substring**. The sample data is structured as follows:



	A	B	C	D	E
1	**Team**	**Points**			
2	Mavs	22			
3	Spurs	29			
4	Rockets	30			
5	Kings	14			
6	Warriors	18			
7	Nets	15			
8	Lakers	22			
9	Thunder	25			
10	Blazers	29			
11	Jazz	25			
12					
13					
14					
15					
16					
17					

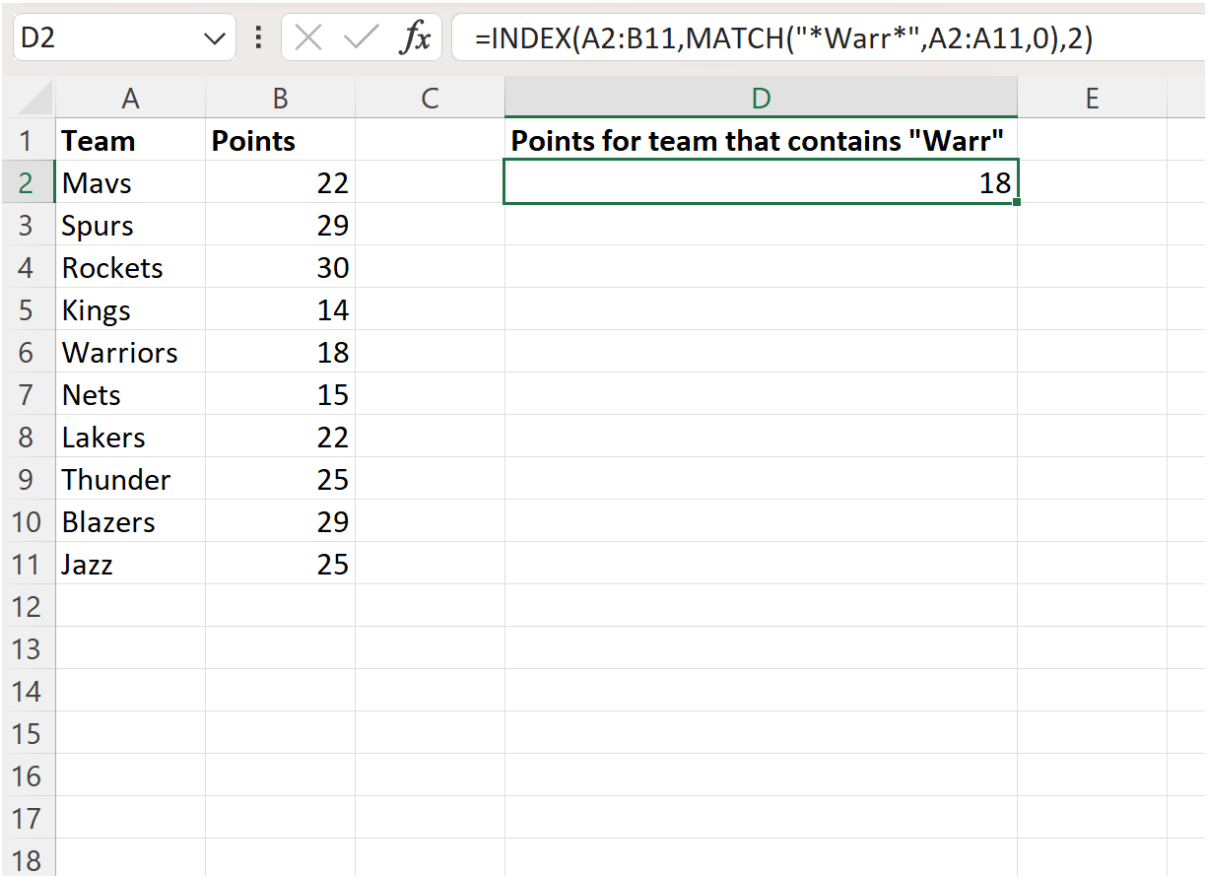
In this task, we aim to find the points scored by the first team whose name contains the substring "Warr" within the **Team** column (Column A), and then retrieve the corresponding points from the **Points** column (Column B). This is the perfect use case for our **INDEX MATCH** formula utilizing **wildcard characters**, specifically the asterisk, to handle the ambiguity.

We will input the core formula into an empty cell, such as **G4**. The specific formula is:

=INDEX(A2:B11,MATCH("*Warr*",A2:A11,0),2)

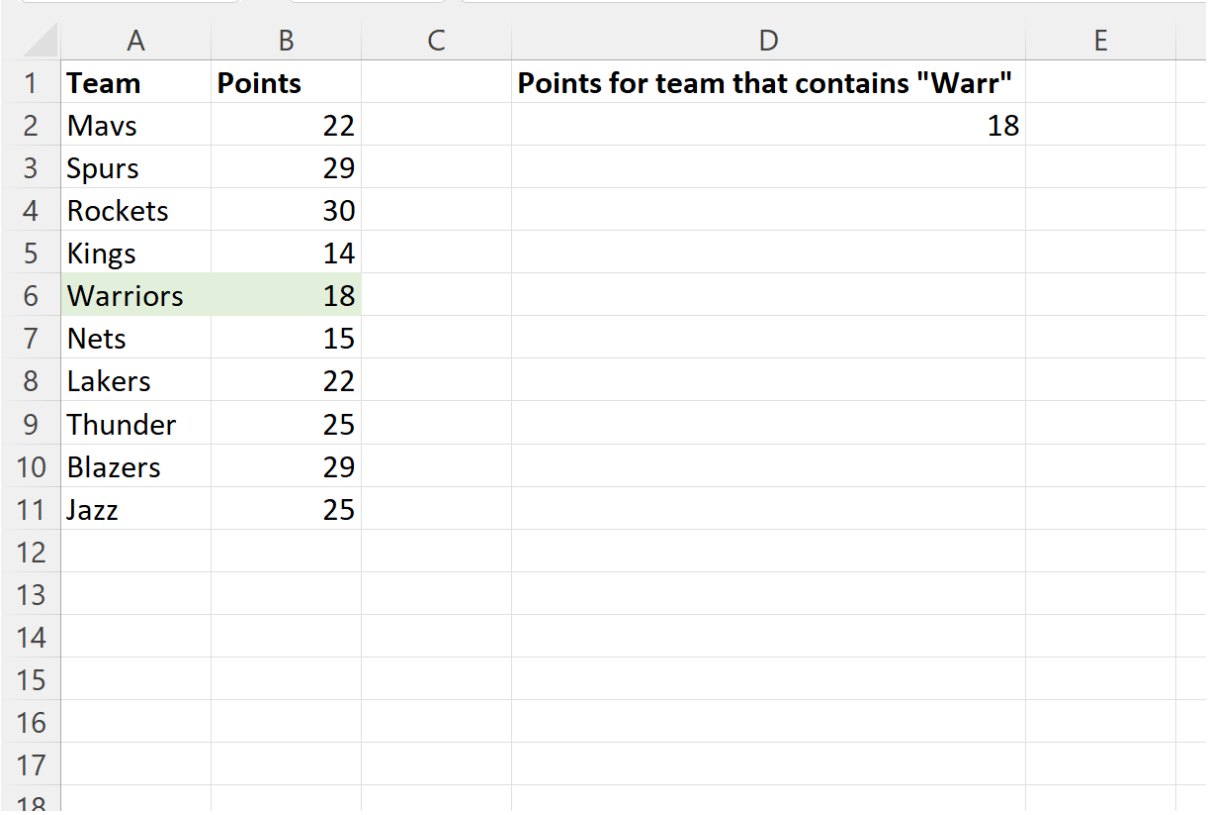
Upon execution, the **MATCH function** first scans the lookup range (A2:A11) for the first cell containing the pattern "*Warr*". It successfully identifies "Warriors" in cell A3, which corresponds to the second relative row within the lookup array (since A2 is row 1). This position (2) is then passed to **INDEX**. The **INDEX function** then navigates the full data range (A2:B11), moves to the 2nd row, and retrieves the value from the 2nd column (Column B).

The resulting output clearly demonstrates the success of the partial match lookup:



	A	B	C	D	E
1	**Team**	**Points**		**Points for team that contains "Warr"**	
2	Mavs	22		18	
3	Spurs	29			
4	Rockets	30			
5	Kings	14			
6	Warriors	18			
7	Nets	15			
8	Lakers	22			
9	Thunder	25			
10	Blazers	29			
11	Jazz	25			
12					
13					
14					
15					
16					
17					
18					

The formula correctly returns the value **18**, which is the score associated with the "Warriors" team. This outcome verifies the efficiency of using **INDEX MATCH** coupled with **wildcard characters** for dynamic data retrieval in Microsoft Excel, enabling powerful searches that do not require perfect data alignment.



	A	B	C	D	E
1	**Team**	**Points**		**Points for team that contains "Warr"**	
2	Mavs	22		18	
3	Spurs	29			
4	Rockets	30			
5	Kings	14			
6	Warriors	18			
7	Nets	15			
8	Lakers	22			
9	Thunder	25			
10	Blazers	29			
11	Jazz	25			
12					
13					
14					
15					
16					
17					
18					

Contrasting Partial Match with Exact Match Lookups

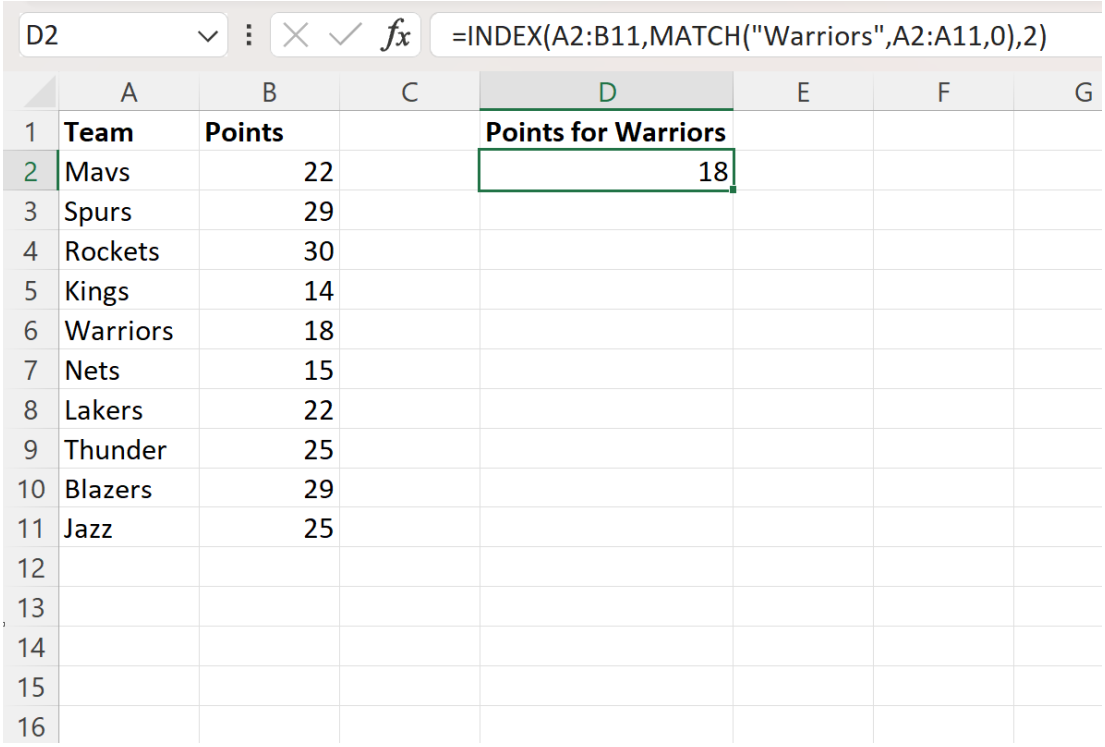
While the utility of searching for a partial **substring** using **wildcard characters** is undeniable, many data retrieval tasks necessitate a strict **exact match**. The **INDEX MATCH** framework offers the core flexibility to pivot seamlessly between these two requirements simply by adjusting the **lookup_value** argument within the **MATCH function**.

If our requirement changed from finding any team containing "Warr" to finding the team specifically named "Warriors," the structure of the formula must be updated by removing the surrounding wildcards. The revised formula for a pure exact match lookup would be:

=INDEX(A2:B11,MATCH("Warriors",A2:A11,0),2)

In this modified formula, the **lookup_value** is now the precise text string "Warriors." The **MATCH function** is strictly instructed to search for a cell that contains only "Warriors"--no preceding or succeeding characters are permitted. The **match_type** of 0 still enforces an **exact match**, but against the literal string defined. This methodology is crucial when dealing with standardized identifiers, product codes, or financial classifications where ambiguity must be avoided to ensure data integrity.

Running this exact match formula against our sample data yields the following result, which, while identical in this specific case, highlights the change in lookup mechanism:



	A	B	C	D	E	F	G
1	**Team**	**Points**		**Points for Warriors**			
2	Mavs	22		18			
3	Spurs	29					
4	Rockets	30					
5	Kings	14					
6	Warriors	18					
7	Nets	15					
8	Lakers	22					
9	Thunder	25					
10	Blazers	29					
11	Jazz	25					
12							
13							
14							
15							
16							

As expected, the formula returns **18**. Understanding when to use **wildcard characters** for flexibility versus literal strings for precision is key to maximizing the efficiency and reliability of your lookups in Microsoft Excel. Choosing the right approach ensures your data retrieval matches your intended analysis goal.

Important Considerations and Best Practices

To ensure the **INDEX MATCH** partial lookup formula is robust and handles diverse data situations effectively, several important considerations and best practices should be observed. These techniques help manage common challenges like errors and inconsistencies in your Excel worksheets, moving your formula from functional to flawless.

A primary consideration is **Case Sensitivity**. By default, the **MATCH function** operates without regard for case, even when using **wildcard characters**. This means a search for `"*proj*"` will match "Project," "PROJECT," or "project" equally. If your business logic strictly requires a case-sensitive partial lookup, you would need to integrate more complex functions like **FIND** or **EXACT** into an array formula structure. Since array formulas must be entered using the special Ctrl+Shift+Enter key combination, they significantly increase complexity and are generally avoided unless absolutely necessary for case-sensitive searching.

Another critical practice is robust **Error Handling**. If the **MATCH function** fails to find any cell containing the specified **substring** within the **lookup_array**, it will return the standard Excel #N/A error. To prevent this disruptive error from appearing on your sheet, it is highly recommended to wrap the entire **INDEX MATCH** formula within the **IFERROR function**. A clean implementation would look like `=IFERROR(INDEX(A2:B11,MATCH("*Warr*",A2:A11,0),2), "Keyword Missing")`, which returns a user-friendly message or zero rather than an error code if no match is found, ensuring a cleaner visual output.

Finally, always remember the fundamental rule of the **First Match**. The **MATCH function** is hardwired to return the row position of the *very first* entry that satisfies the lookup criteria within the **lookup_array**, searching strictly from top to bottom. If your data contains duplicate or multiple partial matches (e.g., both "Warriors" and "Warrhawks" are present), the formula will only acknowledge and return the value corresponding to the entry that appears earliest in the list. To retrieve subsequent matches or the last match, more sophisticated indexing, filtering techniques, or iterative array formulas would be required, extending beyond the scope of a standard partial lookup.

Conclusion: Mastering Flexible Data Retrieval

The strategic combination of the **INDEX function** and the **MATCH function**, particularly when enhanced by **wildcard characters**, establishes the most flexible and powerful method for performing dynamic lookups in Microsoft Excel. This technique effectively bypasses the rigidity of simpler functions, allowing analysts to search for cells that contain a specific text **substring**, rather than demanding a perfect, exact match.

We have demonstrated how **INDEX** acts as the retriever, using the position dynamically calculated by **MATCH**, which incorporates the asterisk wildcard (`*`) to enable partial text matching across a data set. This partial matching capability is essential for working efficiently with large, often messy, or descriptive datasets where exact identifiers may be inconsistent or incomplete. By successfully implementing this sophisticated lookup method, you can significantly improve the dynamism and accuracy of your data analysis workflows, making your spreadsheets highly responsive to varied data conditions and elevating your Excel proficiency.

Additional Resources

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