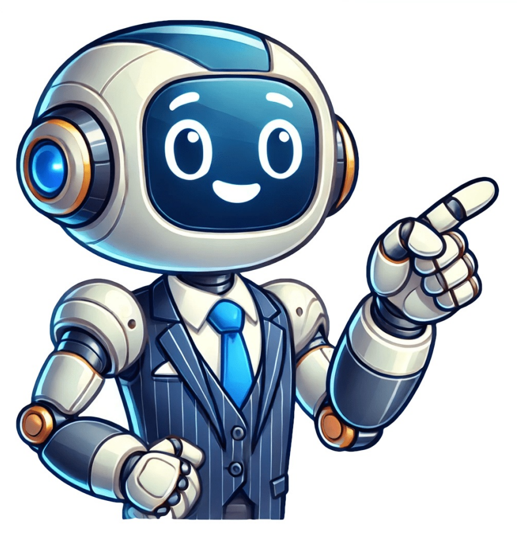
Continue



With the help of the VBA Index Match, you can look for a particular value in a table or range and get the matching value from a different column or range. The Index and Match functions in VBA can be used to accomplish the same goal programmatically. It enables you to use custom VBA code to automate tasks and manipulate data in Excel. The VBA code in this example is called "IndexMatchExample." This code looks for Lee's location in Excel data using VBA Index Match and shows the result in a message box. The first step is to look for the name "Lee" in Column A (A2:A5) within a particular range. If a match is found, the VBA Index function is used to obtain the matching value from Column B (B2:B5). The "result" variable contains the outcome. After checking for errors, the code shows a message box with the found value, such as "Value found: [result]." if there are no errors. But if there isn't a match, it displays the message "Value not found." You can programmatically search and retrieve data from Excel worksheets using VBA Index Match. Basic lookups and more complicated tasks involving numerous criteria and external workbooks can both be accomplished with VBA Index Match. When utilizing VBA Index Match, proper error handling and data validation are crucial. To perform Index Match across different worksheets within the same workbook, we can use VBA Index Match from Another Worksheet. Open Excel and press ALT + F11 to open the VBA editor. Insert a new module by clicking on "Insert" in the menu and selecting "Module." First, declare the variables you will need for your VBA Index Match operation. For example:Dim searchValue As String ' Variable to store the value you want to search forDim result As Variant ' Variable to store the result of the Index Match Next, assign the value you want to search for to the "searchValue" variable: searchValue = "YourSearchValue"Replace "YourSearchValue" with the actual value you want to find in the Excel worksheet. Now, use the "Application.Index" and "Application.Match" functions to perform the Index Match operation. For example:result = Application.Index(Range("ColumnToRetrieveDataFrom"), Application.Match(searchValue, Range("ColumnToSearchIn"), 0))• Range("ColumnToRetrieveDataFrom"): Specify the range from which you want to retrieve data based on the match.• Range("ColumnToSearchIn"): Specify the range in which you want to search for the searchValue.0, as the last argument indicates an exact match. You can use 1 for the approximate match as well. If a match is found, you can display or use the retrieved result. For example:MsgBox "Value: " & result Save your VBA code and run the macro by pressing F5 or selecting it from the Macro dialog box in Excel. Excel VBA – All in One Courses Bundle (35+ Hours of Video Tutorials) If you want to learn Excel and VBA professionally, then Excel VBA All in One Courses Bundle (35+ hours) is the perfect solution. Whether you're a beginner or an experienced user, this bundle covers it all – from Basic Excel to Advanced Excel, Macros, Power Query, and VBA. In this example, we will locate and show 'John's' salary in Excel within the same salary cell using the VBA Index Match. Step 1: In the new module, we begin with the creation of a subroutine, after which two variables are declared: "rowIndex" to hold the match result and "employeeName" to store the name we wish to look up. Step 2: We then assign the name "John" to the employeeName variable. You can replace this with the name you want to search for. Step 3: In this line, we use "Application.Match" to search for "employeeName" in the range Sheets("Example_1").Range("A2:A5"), and the 0 indicates an exact match. The result is stored in the "rowIndex" variable. Step 4: Here, we check if "rowIndex" contains an error using the IsError function. Step 5: If "rowIndex" does not contain an error, we set the value of a cell to display the salary. The "Application.Index" function retrieves the salary from column B (Range("B2:B5")) of the row where the match was found (rowIndex + 1). Step 6: If there was an error (employee not found), we display a message box indicating that the employee was not found. Step 7: Save the VBA macro and click on run. If the employee "John" is found in "Example_1," the code will search for him and display "Salary: [John's salary]" in Cell B2, which is the adjacent cell in Column B. Otherwise, it shows a message box in VBA saying "Employee not found." Here is the full code: Sub BasicIndexMatch()Dim employeeName As StringDim rowIndex As VariantemployeeName = "John"rowIndex = Application.Match(employeeName, Sheets("Example_1").Range("A2:A5"), 0)If Not IsError(rowIndex) ThenCells(rowIndex + 1, 2).Value = "Salary: " & Application.Index(Sheets("Example_1").Range("B2:B5"), rowIndex)ElseMsgBox "Employee not found."End IfEnd Sub In this example, we will use the VBA Match Index with a criterion to search for 'Monitor' in the 'SalesData' sheet and display the corresponding value. Step 1: We start by creating a new subroutine and declaring the following two variables: searchValue: This variable will store the value we want to search for in the Excel data. result: This variable will store the result of the Index Match operation. Step 2: Here, we assign the value "Monitor" to the searchValue variable. It is the value we want to find in the Excel data. Step 3: This is where the main Index Match operation occurs: "Application.Match(searchValue, Sheets("SalesData").Range("A2:A5"), 0)": This section of the code carries out the Match operation. It looks for the value in searchValue ("Monitor") in the "SalesData" worksheet's range A2:A5. The final argument, which is 0, denotes a perfect match. "Application.Index(Sheets("SalesData").Range("B2:B5"), ...)": This section of the code carries out the Match operation. Based on the row where the match was found in column A, it searches the "SalesData" worksheet for the corresponding value in the range B2:B5. Step 4: The "If" statement checks whether the result contains an error. If not, it means that the Index Match operation found a matching value. Step 5: This line shows a message box with the value "Value: " followed by the outcome of the VBA Index Match operation if the result is error-free, meaning a match was found. For example, the message box will say "Value: $200" if "Monitor" is located and the matching value in column B is $200. Step 6: If the result contains an error (i.e., no match was found), this part of the code displays a message box with the text "Value not found." Step 7: Save the macro and click on run. 'Value: [Monitor's corresponding value]' will appear in a message box when you run the code if "Monitor" is found. Else, it will show 'Value not found. Here is the full code: Sub IndexMatchWithValue()Dim searchValue As StringDim result As VariantsearchValue = "Monitor"result = Application.Index(Sheets("SalesData").Range("B2:B5"), Application.Match(searchValue, Sheets("SalesData").Range("A2:A5"), 0))If Not IsError(result) ThenMsgBox "Value: " & resultElseMsgBox "Value not found."End IfEnd Sub In this example, to search for particular criteria in Sheet 3 of the current workbook, i.e., "VBA Index Match.xlsm," we use VBA Index Match from another workbook to retrieve pertinent data from "Workbook2," which is a different Excel workbook. Workbook1 Workbook2 Step 1: In the new module, we start by creating a subroutine and assigning two following variables: criteria: This variable will store the criteria (value to search for) obtained from the current workbook. rowIndex: This variable will store the result of the Match operation. Step 2: Here, we retrieve the criteria (the value to search for) from cell A2 of "Index_Match_Example" in the current workbook (ThisWorkbook). Step 3: This is where the main Index Match operation occurs. The Match operation is carried out by this section of the code. Within the given range (A2:A100), it looks for the criteria value that was previously obtained in "Sheet1" of the external workbook "Workbook2.xlsx." The final argument, which is 0, denotes a perfect match. Step 4: This If statement checks whether "rowIndex" contains an error. If rowIndex does not contain an error, it means that the Index Match operation found a matching row. Step 5: When a match is found and "rowIndex" is not containing an error, this line shows a message box that says "Data from Workbook2: " followed by the value that is in column B of the corresponding row in "Sheet1" of "Workbook2.xlsx." To account for the header row in excel, it retrieves the data using the rowIndex plus 1. Step 6: If "rowIndex" contains an error (i.e., no match was found), this part of the code displays a message box with the text "Data not found in Workbook2." Step 7: Save the macro and click on run. After you run the code and if a match is found, the code displays 'Data from Workbook2: [Retrieved data]. If there is no match found, it will show as "Data not found in Workbook2." Here is the full code: Sub IndexMatchFromAnotherWorkbook()Dim criteria As StringDim rowIndex As Variantcriteria = ThisWorkbook.Sheets("Index_Match_Example").Cells(2, 1).ValuerowIndex = Application.Match(criteria, Workbooks("Workbook2.xlsx").Sheets("Sheet1").Range("A2:A100"), 0)If Not IsError(rowIndex) ThenMsgBox "Data from Workbook2: " & Workbooks("Workbook2.xlsx").Sheets("Sheet1").Cells(rowIndex + 1, 2).ValueElseMsgBox "Data not found in Workbook2."End IfEnd Sub Excel's VBA Index Match offers a strong and adaptable way to perform data lookups and retrievals. To further narrow down your search, you can use VBA Index Match combine multiple criteria conditions using logical operators like "AND" and "OR" within the VBA code. You can automate time-consuming Index Match operations over sizable datasets or carry out several lookups in succession by using the VBA Index Match loop. 1. Can VBA Index Match be used with non-contiguous ranges? It is possible to use VBA Index Match with non-contiguous ranges. To get the desired data, you can define multiple ranges and use the Match and Index functions for each range independently. 2. Is it possible to nest VBA Index Match within other VBA functions? Yes, you can perform more complex operations by nesting VBA Index Match within other VBA functions and procedures. For instance, you can use Index Match in conjunction with conditional statements or inside of a loop. 3. Can VBA Index Match be used with dynamic ranges? It is possible to use VBA Index Match with dynamic ranges. Using VBA, you can dynamically calculate the range based on predefined criteria, or you can define dynamic named ranges. 4. What are the advantages of using VBA for Index Match over regular Excel formulas? Using VBA for Index Match offers several advantages:• You can save time and effort by automating tasks and performing VBA Index Match operations programmatically.• Complex logic and conditional operations can be implemented with flexibility using VBA, which can be difficult to accomplish with Excel formulas alone.• VBA is an effective tool for data manipulation and analysis because it can be integrated with other Excel features and external data sources.• Using VBA, you can design unique user-defined functions (UDFs) to fulfill particular needs that conventional Excel formulas might not be able to. This article must be helpful to understand the VBA Index Match, with formulas and examples. You can download the templates here to use it instantly. Recommended Articles This has been a guide to VBA Index Match. Here we learn to use combination of Index & Match functions in Excel VBA code, with examples & points to remember. You can learn more from the following articles – VBA Match VBA Application.Match VBA Type Mismatch Error Reader Interactions You can use the following basic syntax to perform an INDEX MATCH with multiple criteria in VBA: Sub IndexMatchMultiple() Range("F3").Value = WorksheetFunction.Index(Range("C2:C10"), _ WorksheetFunction.Match(Range("F1"), Range("A2:A10"), 0) + _ WorksheetFunction.Match(Range("F2"), Range("B2:B10"), 0) - 1) End Sub This particular example looks up the value in cell F1 within the range A2:A10 and the value in cell F2 within the range B2:B10 and returns the corresponding value in the range C2:C10 to cell F3. The following example shows how to use this syntax in practice. Example: Perform INDEX MATCH with Multiple Criteria Using VBA Suppose we have the following dataset in Excel that contains information about basketball players: Suppose we would like to look up the player that matches the team name in cell F1 and the position in cell F2 and return the name in cell F3. We can create the following macro to do so: Sub IndexMatchMultiple() Range("F3").Value = WorksheetFunction.Index(Range("C2:C10"), _ WorksheetFunction.Match(Range("F1"), Range("A2:A10"), 0) + _ WorksheetFunction.Match(Range("F2"), Range("B2:B10"), 0) - 1) End Sub When we run this macro, we receive the following output: The macro looks up "Spurs" in the Team column and "Forward" in the Position column and correctly returns the name "Eric" in cell F3. If we change the values in cells F1 and F2 and run the macro again, it will be able to find the player name based on the new values. For example, suppose we change the team name to "Mavs" and position to "Center" and run the macro again: The macro looks up "Mavs" in the Team column and "Center" in the Position column and correctly returns the name "Chad" in cell F3. Additional Resources The following tutorials explain how to perform other common tasks in VBA: VBA: How to Use INDEX MATCH VBA: How to Check if String Contains Another String VBA: How to Count Number of Rows in Range For example, look at the below data.In the above data, the lookup value is the "Department" name. Based on this department name, we need to extract the salary amount.But the problem here is that the result column is there in the first, and the lookup value column is the result column. So, in this case, VLOOKUP cannot fetch the salary amount because VLOOKUP works only from right to left, not from left to right.In these cases, we need to use the combination formula of the VBA INDEX and MATCH function. But, first, let us perform the task of finding the salary amount of each department in the VBA code. Learn to master formulas, automate tasks with VBA, build interactive dashboards, and analyze data like a pro—all in one comprehensive bundle. Develop hands-on Excel skills across all levels, earn a recognized certification, and gain practical knowledge that's essential for roles in finance, analytics, and business operations. Learn More → Step 1: Start the sun routine.Step 2: Declare the VBA Integer variable.Code:Sub INDEX_MATCH_Example1() Dim k As Integer End SubStep 3: Now, open For Next Loop in VBA.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Next k End SubStep 4: Inside the VBA loop, execute the formula. In the 5th column, we need to apply the formula, so the code is CELLS (k,5).Value =Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = Next k End SubStep 5: We must apply that cell's VBA INDEX and MATCH formula. As we said, we need to use these functions as a Worksheet Function in VBA class, so open the worksheet function class.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction. Next k End SubStep 6: After entering the worksheet function class, we can see all the available worksheet functions, so select the INDEX function.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Next k End SubStep 7: While using the worksheet function in VBA, you must be sure of the arguments of the formula. The first argument is an array, i.e., from which column we need the result. In this case, we need the result from A2 to A5.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Range("A2:A5"), Next k End SubStep 8: Next up is from which row number we need the result. As we have seen in the earlier example, we cannot manually supply the row number every time. So, use the MATCH function.To use the MATCH function again, we need to open the Worksheet function class.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Range("A2:A5"), WorksheetFunction.Match(Next k End SubStep 9: The MATCH function's first argument is the LOOKUP value; here, our lookup value is department names; it is there in the cells (2, 4).Since every time the row number changes, we can supply the variable "k" in place of manual row number 2. Cells (k, 4).ValueCode:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Range("A2:A5"), WorksheetFunction.Match(Cells(k,5).Value, Next k End SubStep 10: Next, we need to mention the department value range, i.e., Range ("B2:B5").Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Range("A2:A5"), WorksheetFunction.Match(Cells(k,5).Value,Range("B2:B5"),Next k End SubStep 11: Next, put the argument as 0 because we need an exact match and close the brackets.Code:Sub INDEX_MATCH_Example1() Dim k As Integer For k = 2 To 5 Cells(k, 5).Value = WorksheetFunction.Index(Range("A2:A5"), WorksheetFunction.Match(Cells(k, 4).Value, Range("B2:B5"), 0))Next kEnd SubWe have now completed the coding part. Let us run the code to have the result in column 5.So, we got the result.We can use this formula as an alternative to the VLOOKUP function. Download Practice Workbook Download the practice workbook and practice. Step1 – Apply the INDEX and the MATCH Functions There are 2 sheets in the workbook. One is empty: UserForm, the other: StudentInformation contains a range showing student names, gender, and eye color. Use the following formula to find the gender based on the student's name: =INDEX(B2:B31, MATCH("Diana Graham", A2:A31, 0)) The value of Females is returned. Step 2 – Change the Name of Column B into StudentNames Name the range A2: A31, StudentNames. Hide the StudentInformation sheet, by right-clicking and selecting Hide. Step 3 – Open the Visual Basic Window Go to Developer > Code > Visual Basic to open the Visual Basic Editor (VBE). Go to Insert and select UserForm. Step 4 – Change the Properties and Add Text Boxes In the Properties Window, rename the form to StudentLookup. Change the Caption to Lookup Student Information. Change the BackColor to light blue, and set the height to 300 px and the width to 350 px. If the Properties Window is not displayed, press F4. Insert a label using the Toolbox (go to View, Toolbox). Change the Caption to Choose a student. Change the BackColor to white, here. Set the font to Georgia, the font style to bold, the font size to 12, and choose center to align the text. Use the special effect 1– fmSpecialEffectRaised. Insert a combo box below the label. Name it cmdStudentName. In RowSource, enter StudentNames. Click Run Sub/UserForm. By clicking the drop-down arrow in UserForm, the combo box automatically displays students' names. Close the UserForm by clicking the close button. Press Alt-F11 to go back to the VBE. Add another label to the UserForm (below the combo box) and change the Caption to Gender. Change the BackColor to white, here. Set the font to Georgia, the font style to bold, the font size to 12, and choose center to align the text Use the special 1– fmSpecialEffectRaised. Create a textbox below the Gender label. Name it txtGender. Add another label called Eye Colour and a textbox named txtEyeColour. Use the same properties for the labels. Select all the controls added to the UserForm, using the control key. Similar Readings Step 5 – Add a Button from the Toolbox Add a button to the form using the Toolbox. Change the Name of the button to cmdLookUp, the BackColor to light orange, keep the Tahoma font and change the style to bold. Change the Caption of the button to Look up Student Details. Step 6 – Insert a VBA Code Right-click the button and select View Code. Enter the following code for the button click event: Dim a As Variant Dim b As Variant Dim c As Variant a = cmdStudentName.Value Sheets("StudentInformation").Activate If a = "" Then b = "" Let txtGender.Text = b c = "" Let txtEyeColour.Text = c Else b = Application.WorksheetFunction.Index(Sheets("StudentInformation").Range("B2:B31"), Application.WorksheetFunction.Match(a, Sheets("StudentInformation").Range("A2:A31"), 0)) c = Application.WorksheetFunction.Index(Sheets("StudentInformation").Range("C2:C31"), Application.WorksheetFunction.Match(a, Sheets("StudentInformation").Range("A2:A31"), 0)) Let txtEyeColour.Text = c End If Three variables are declared and the variant data type is assigned. Draw a Variable value from the option selected in the drop-down combo box inn the UserForm. If there is no selection, all the other textboxes are empty. If you select a student name from the combo box in the UserForm, variable b draws value by using the INDEX Worksheet Function in combination with the MATCH Function in the VBA code. It looks up the value using the same syntax as the worksheet function. Variable c draws value by using the INDEX Worksheet Function in combination with the MATCH Function in the VBA code when an option in the combo box is selected. Variable b attains value from the gender column, whereas variable c gets value from the Eye colour column. The gender textbox is populated with b's value and the eye color textbox is populated with c's value. Read More: Excel VBA Events (A Complete Guideline) Step 7 – Insert a Command Button Go to the UserForm worksheeti. Format it and insert the image provided by ExcelDemy. Go to Developer > Controls > Insert > ActiveX Controls. Select the button and go to Developer > Controls > Properties. Change the Name of the button to cmdShowForm and the Caption to Lookup Student Information. Step 8 – View the Lookup Code Right-click the button and select View Code. Enter the following code: Private Sub cmdShowForm_Click() StudentLookup.Show End Sub Step 9 – Display the Final Result Go back to the worksheet. Make sure to uncheck Design Mode. Click the button in order to show the form. Select a student's name using the combo box. The code will return the student's gender and eye color automatically. Save your workbook as a macro-enabled workbook. Read More: Excel INDEX-MATCH Formula to Return Multiple Values Horizontally Practice Section: Examples 1) Set up a list of three items in column A: tangerines, carrots, and oranges. In the cell next to each item in column B list whether the items are fruits or vegetables. Use the INDEX & MATCH combination function to see whether carrots are fruits or vegetables. 2) Set up a list of ESPN in the NFL head coaches and the respective team they are coaching. Create a user form that allows the user to input a coach's and see the team he is coaching in another textbox, by clicking submit. Use the INDEX & MATCH functions within the VBA code. Further Readings Get FREE Advanced Excel Exercises with Solutions! Have you ever struggled with performing precise lookups in your Excel spreadsheets? The index match function, when combined with Excel VBA code, can enhance your spreadsheet's functionality and efficiency. In this section, we will provide a step by step guide on how to use Excel VBA code for the index match function, allowing you to perform targeted lookups with ease. Key Takeaways The index match function is a powerful tool for performing precise lookups in Excel spreadsheets Excel VBA code can enhance the functionality and efficiency of the index match function By following a step by step guide, you can easily use Excel VBA code for the index match function Understanding the purpose and benefits of index match is essential before using VBA code Proper spreadsheet setup and troubleshooting techniques are crucial for successful implementation of VBA code for index match Understanding the Index Match Function Excel's Index Match function is a powerful tool that allows you to search for and return values in your spreadsheet with more precision than other lookup functions, like VLOOKUP. Understanding how to use Index Match is essential to making the most of Excel's capabilities. Index Match works by searching for a value within a table and then returning a related value in a chosen column. The function requires two arguments: the lookup value and the range where the lookup value can be found. Additionally, Index Match is a dynamic formula and can search in any direction, left to right or right to left. This makes it especially useful for data sets with changing column orders. The Benefits of Index Match One of the biggest benefits of Index Match is its ability to perform exact and approximate matches in the same formula. This can save you time and effort by avoiding the need to use multiple lookup functions on a single data set. Index Match also allows you to look up values in a table with increased accuracy, as it uses two individual searches. Once it locates the value, it can return any related data you need. This versatility makes it a valuable tool, particularly in complex data sets. How Index Match Differs from Other Lookup Functions Compared to other lookup functions like VLOOKUP and HLOOKUP, Index Match has several advantages. Firstly, Index Match can search through columns both left and right of a lookup value. This is advantageous since VLOOKUP and HLOOKUP can only search to the right and below, respectively. Additionally, the Index Match function runs faster, improving the efficiency of your lookup operations, particularly with larger data sets. Using Index Match in Excel can be a game-changer, and understanding its functionality is essential to capitalizing on its benefits. With this knowledge, you can perform more powerful and efficient lookups in your spreadsheets. Setting Up Your Spreadsheet To use the power of index match with VBA, correctly setting up your spreadsheet is crucial. Follow these step-by-step instructions to ensure your data is organized and ready to perform the index match lookup. Step 1: Open a new Excel workbook and enter data into two separate tables. The first table should hold the data you want to look up, and the second table should contain the data you want to retrieve. Step 2: Name each table with a relevant and descriptive name by selecting the range of cells, right-click, and choose "Define Name". Step 3: Ensure each table has a unique header name and that there are no blank rows or columns in either table. Step 4: Arrange the data in a way that will make it easy for you to identify and retrieve the information you need. For instance, if you need to find a product price, make sure the product names and their prices are in the same row or column in both tables. Product Name Price Product 1 $10.00 Product 2 $15.00 Step 5: Create a new module in the VBA editor by clicking on "Developer" -> "Visual Basic". By following these steps, you'll have successfully set up your spreadsheet for using the index match function with VBA. Writing the VBA Code With your spreadsheet organized and prepared for the index match function, it's time to start writing the VBA code. Follow these simple steps: Open the Visual Basic Editor – click on the Developer tab and select "Visual Basic". Insert a new module – right-click on the VBAProject in the Project window and click "Insert Module". Begin the VBA code – use the Sub and End Sub commands to define the code. Declare your variables – use the Dim command to declare your variables and their data type. Write the code for index match function – use the Range and Cells commands in combination with the index match function to perform the lookup. Test the code – execute the code and verify that the index match lookup is functioning accurately. Note: Remember to use proper syntax and adhere to the coding conventions of VBA. Indentation and comments will aid both you and others who review the code. Writing the VBA code for the index match function may seem intimidating, but with these simple steps, you can perform precise and efficient lookups in your Excel spreadsheets. As you become more proficient in VBA programming, you can add additional functionality to your code to further enhance your spreadsheets' performance. Understanding Variables and Arrays in VBA Variables and arrays are crucial concepts in VBA programming, and understanding them is essential to effectively using VBA code for the index match function. Variables represent a value that can change during the execution of the code, while arrays are a collection of related values that share a common name. In VBA, you can declare variables and assign values to them using the Dim keyword. Arrays can be one-dimensional, two-dimensional, or multidimensional, and can be used to store data in a structured format. To declare an array, use the Dim keyword, followed by the name of the array and its dimensions. For example, Dim myArray(5) As Integer declares an array named "myArray" with five integer elements. When using the index match function with VBA, variables and arrays can be used to store the lookup values, search ranges, and results. For instance, you can use a variable to hold the lookup value and an array to store the search range. This can make your code more concise and manageable. Confused about how to use variables and arrays in your VBA code for the index match function? Don't worry, we've got you covered! In the next section, we will provide a step by step guide on how to implement these concepts in your code. Executing the Code and Testing the Lookup Now that you have written the VBA code for the index match function, it's time to execute it and test the lookup accuracy. Follow the steps below to execute the code: Open the Excel workbook with the index match function VBA code. Press "Alt + F8" to open the Macro dialog box. Select the macro you want to execute from the list. Click "Run" to execute the macro. Verify that the lookup returns the correct values. It's crucial to test the lookup accuracy to ensure that it's functioning correctly. To do this, you can use a small dataset and compare the results with the expected outcome. If the results match, your VBA code is functioning correctly. If, however, the lookup returns errors or unexpected results, you may need to troubleshoot using the techniques outlined in the next section. Troubleshooting Common Issues While working with Excel VBA code for the index match function, you may encounter common issues that affect the accuracy of your lookups. In this section, we will discuss these issues and provide troubleshooting techniques to ensure that your code delivers the expected results. Issue 1: Incorrect Lookup Value One of the most common issues when using the index match function is entering an incorrect lookup value. Ensure that the lookup value is correctly typed and formatted. If the lookup value is a string, enclose it in quotation marks to help Excel identify it as a text value. Alternatively, check if the source data contains leading or trailing spaces that may cause the lookup to fail. Issue 2: Duplicates in Source Data If the source data contains duplicates, the index match function may return unexpected results or errors. To resolve this issue, you can either remove the duplicates or add a helper column to the source data that assigns a unique identifier to each record. Issue 3: Incorrect Reference Range When using the VBA code for index match, ensure that the reference range is correctly specified. The reference range should include all the data needed to perform the lookup, excluding column headers and other non-data information. Check that all references in the code are correct and that the range includes the correct columns and rows. Issue 4: Syntax Errors in the Code One of the main causes of unexpected errors in VBA code is syntax errors. These errors occur when the code contains missing or incorrect syntax elements, such as commas, brackets, or quotation marks. To troubleshoot this issue, carefully review your code and check all syntax elements to ensure that they are correctly typed and formatted. Issue 5: Incompatible Data Types If the data types of the reference range and lookup values are not compatible, you may encounter errors or unexpected results. For instance, if the lookup value is a string, and the source data contains numbers, the lookup may not work correctly. To resolve this issue, ensure that the data types are consistent and compatible. Advanced Techniques and Modifications In this section, we will delve into some advanced techniques and modifications that you can make to the VBA code for the index match function. These techniques will allow you to further customize the lookup process to suit your specific needs. Let's have a closer look. 1. Using Wildcards in the Lookup Value By using wildcards in the lookup value, you can make the index match function more versatile. Wildcards can be used to replace one or more characters in the lookup value, which can be helpful when searching for partial matches. For example, using an asterisk (*) in the lookup value will match any number of characters in that position. To use wildcards in the lookup value, you need to modify the code slightly. Instead of using "=" as the criteria, use the "Like" operator to enable wildcards. Here is an example: Dim lookupvalue As String lookupvalue = "App*" Cells(2, 6).Value = Application.Index(Range("B2:B10"), Application.Match(lookupvalue & "*", Range("A2:A10"), 0)) You can use the following basic syntax to perform an INDEX MATCH in VBA: Sub IndexMatch() Dim i As Integer 'Perform index match For i = 2 To 11 Cells(i, 5).Value = WorksheetFunction.Index(Range("A2:A11"), _ WorksheetFunction.Match(Cells(i, 4).Value, Range("B2:B11"), 0)) Next i End Sub This particular example looks up the values in cells 2 through 11 of the fourth column of the worksheet within the range B2:B11 and then returns the corresponding values in the range A2:A11 to the fifth column of the worksheet. The following example shows how to use this syntax in practice. Example: Perform INDEX MATCH Using VBA Suppose we have the following dataset in Excel that contains information about basketball players: For each player in column D, suppose we would like to find their team name from column A and then write the team name in column E. We can create the following macro to do so: Sub IndexMatch() Dim i As Integer 'Perform index match For i = 2 To 11 Cells(i, 5).Value = WorksheetFunction.Index(Range("A2:A11"), _ WorksheetFunction.Match(Cells(i, 4).Value, Range("B2:B11"), 0)) Next i End Sub When we run this macro, we receive the following output: Notice that this macro was able to look up each player name and then return their corresponding team name in column E. Note that within the For loop, the syntax Cells(i,5).value specifies that we would like the team names to be returned in the fifth column of the worksheet, i.e. column E. If we change this syntax to Cells(i,6).value then the team names will be returned in the sixth column of the worksheet instead: Sub IndexMatch() Dim i As Integer 'Perform index match For i = 2 To 11 Cells(i, 6).Value = WorksheetFunction.Index(Range("A2:A11"), _ WorksheetFunction.Match(Cells(i, 4).Value, Range("B2:B11"), 0)) Next i End Sub When we run this macro, we receive the following output: Notice that the team names are now returned in the sixth column of the worksheet (column F) instead. Additional Resources The following tutorials explain how to perform other common tasks in VBA: VBA: How to Check if String Contains Another String VBA: How to Count Number of Rows in Range VBA: How to Write COUNTIF and COUNTIFS Functions This is the sample dataset for illustration. We will extract a certain result residing in one column based on conditions from the other two columns. Method 1 – Embed VBA with INDEX MATCH for Multiple (Two) Dimensional Lookup in Excel We have stored a specific student's name "Edge" in Cell G4; and the column that we will be searching the Result in, Exam Marks, is stored in Cell G5. We will search in the Exam Marks column and store the Marks that "Edge" got in Cell G6. Steps: Press Alt + F11 or go to the tab Developer -> Visual Basic to open Visual Basic Editor. In the pop-up code window, click Insert -> Module. Copy the following code and paste it into the code window. Sub IndexMatchStudent() Dim iSheet As Worksheet Set iSheet = Worksheets("Two Dimension") iSheet.Range("G6").Value = Application.WorksheetFunction.Index(iSheet.Range("C5:D14"), Application.WorksheetFunction.Match(iSheet.Range("G4"), iSheet.Range("B5:B14"), 0), Application.WorksheetFunction.Match(iSheet.Range("G5"), iSheet.Range("C4:D4"), 0)) End Sub Press F5 on your keyboard, or from the menu bar select Run -> Run Sub/UserForm. You can also just click on the small Run icon in the sub-menu bar to run the macro. The Marks that "Edge" got in the exam, 67 is retrieved in Cell G7. VBA Code Explanation Dim iSheet As Worksheet Defining the variable of Worksheet. Set iSheet = Worksheets("Two Dimension") Store the worksheet name. The name of our sheet is "Two Dimension", you should provide the name according to your spreadsheet. iSheet.Range("G6").Value = Application.WorksheetFunction.Index(iSheet.Range("C5:D14"), Application.WorksheetFunction.Match(iSheet.Range("G4"), iSheet.Range("B5:B14"), 0), Application.WorksheetFunction.Match(iSheet.Range("G5"), iSheet.Range("C4:D4"), 0)) This piece of code selects the range C5:D14 as the lookup range. Search for the match that is stored in cell G4 in range B5:B14 and search for the match that is stored in cell G5 in range C4:D4 and pass the result to cell G6. Method 2 – Apply Macro to Find MATCH Value by INDEX with User-Defined Function (UDF) Steps: Open Visual Basic Editor from the Developer tab and Insert a Module in the code window. Enter the following code: Function MatchByIndex(x As Double, y As Double)   Const StartRow = 4   Dim EndRow As Long   Dim iRow As Long   With Worksheets("UDF")   EndRow = .Range("C:D").Find(What:="*", SearchOrder:=xlByRows, SearchDirection:=xlPrevious).Row   For iRow = StartRow To EndRow   If .Range("C" & iRow).Value = x And .Range("D" & iRow).Value = y Then   MatchByIndex = .Range("B" & iRow).Value   Exit Function   End If   Next iRow   End With   MatchByIndex = "Data Not found" End Function Don't run this code, save Go back to the worksheet of interest. Pick any cell that you want to store the result. In our case, it is Cell F5. In that cell, write the UDF you have just created in the code (MatchByIndex) and pass the Student ID and Exam Marks of the specific student inside the parentheses of the function. As we are trying to extract the name "Finn" from his ID (105) and Marks (84), the formula becomes, In Cell F5, we have successfully retrieved the name "Finn" by passing his ID and Marks inside the function that we created in the VBA code. VBA Code Explanation Function MatchByIndex(x As Double, y As Double) Creating a new function and passing the variables inside it. You can define any name to the function. Const StartRow = 4 Our row starts from row number 4. You must provide the row number that your dataset starts from. Dim EndRow As Long Dim iRow As Long Defining the variables. With Worksheets("UDF")   EndRow = .Range("C:D").Find(What:="*", SearchOrder:=xlByRows, SearchDirection:=xlPrevious).Row Define the worksheet to work with. The name of our sheet is "UDF", you should provide the name according to your spreadsheet. Start searching in the range C:D from the first row that we defined to the last row. For iRow = StartRow To EndRow   If .Range("C" & iRow).Value = x And .Range("D" & iRow).Value = y Then   MatchByIndex = .Range("B" & iRow).Value   Exit Function   End If   Next iRow End With Start iterating from the first row to the last row. If the first value that we will pass inside the function falls inside the C column and if the second value that we will pass inside the function falls inside the D column, it will return the match from the B column. Otherwise, exit the function, end all the statements and go to the next line. MatchByIndex = "Data Not found" End Function If the previous condition doesn't get fulfilled while executing, the "Data Not Found" message will be returned and the code will leave the function. Method 3 – Implement VBA to Return MATCH Value from a Table with Multiple Data in Excel Let's see how to extract the Marks from the table shown in our dataset (Table Name: TableMatch) of a certain student by providing the Name and the ID inside the code. For our case, the Name and the ID will be Finn and 105 respectively. Steps: Open Visual Basic Editor from the Developer tab and Insert a Module in the code window. Enter the following code. Sub ReturnMatchedResultByIndex()   Dim iBook As Workbook   Dim iSheet As Worksheet   Dim iTable As Object   Dim iValue As Variant   Dim TargetName As String   Dim TargetID As Long   Dim IdColumn As Long   Dim NameColumn As Long   Dim MarksColumn As Long   Dim iCount As Long   Dim iResult As Boolean   Set iBook = Application.ThisWorkbook   Set iSheet = iBook.Sheets("Return Result")   Set iTable = iSheet.ListObjects("TableMatch")   TargetID = 105   TargetName = "Finn"   IdColumn = iTable.ListColumns("Student ID").Index   NameColumn = iTable.ListColumns("Student Name").Index   MarksColumn = iTable.ListColumns("Exam Marks").Index   iValue = iTable.DataBodyRange.Value   For iCount = 1 To UBound(iValue)   If iValue(iCount, IdColumn) = TargetID Then   If iValue(iCount, NameColumn) = TargetName Then   iResult = True   End If   End If   If iResult Then Exit For   Next iCount   If iResult Then   MsgBox "ID: " & TargetID & vbLf & "Name: " & TargetName & vbLf & "Marks: " & iValue(iCount, MarksColumn)   Else   MsgBox "ID: " & TargetID & vbLf & "Name: " & TargetName & vbLf & "Marks Not found"   End If End Sub Run the code.There is a Microsoft Excel pop-up message box showing you the Marks: 84 of ID: 105 and Name: Finn that we provided inside the code. VBA Code Explanation Dim iBook As Workbook Dim iSheet As Worksheet Dim iTable As Object Dim iValue As Variant Dim TargetName As String Dim TargetID As Long Dim IdColumn As Long Dim NameColumn As Long Dim MarksColumn As Long Dim iCount As Long Dim iResult As Boolean Defining the variables. Set iBook = Application.ThisWorkbook Set iSheet = iBook.Sheets("Return Result") Set iTable = iSheet.ListObjects("TableMatch") Setting the sheet name and the table name inside variables. TargetID = 105 TargetName = "Finn" IdColumn = iTable.ListColumns("Student ID").Index NameColumn = iTable.ListColumns("Student Name").Index MarksColumn = iTable.ListColumns("Exam Marks").Index Storing the lookup values and the lookup columns to search. iValue = iTable.DataBodyRange.Value   For iCount = 1 To UBound(iValue)   If iValue(iCount, IdColumn) = TargetID Then   If iValue(iCount, NameColumn) = TargetName Then iResult = True   End If   End If If iResult Then Exit For Next iCount This piece of code scans through from the start to the end of the subscript and if it finds the match of the defined ID and the Name in the search columns then store the result and close all the statements. Exit the iteration and go to the next part of the code. If iResult Then   MsgBox "ID: " & TargetID & vbLf & "Name: " & TargetName & vbLf & "Marks: " & iValue(iCount, MarksColumn) Else   MsgBox "ID: " & TargetID & vbLf & "Name: " & TargetName & vbLf & "Marks Not found" End If Throws the result in the MsgBox. Download Workbook VBA INDEX MATCH Based on Multiple Criteria.xlsm Related Articles Get FREE Advanced Excel Exercises with Solutions!