Continue

200+ Video Lessons 50+ Hours of Video 200+ Excel Guides Become a master of VBA and Macros in Excel and learn how to automate all of your tasks in Excel with this online course. (No VBA experience required.) View Course () Hello Excellers welcome to another #MacroMonday blog post in my series of articles. Do you need to email a report from Excel Automatically? Maybe you want to email a colleague whenever you change a worksheet. Whatever the reason, VBA can help. In this post, we will show you how to use VBA to send an email from Excel. Let's get started!Microsoft Outlook is a separate application from Excel. So, to reference Outlook to send the email using Excel VBA, a reference to the Outlook object library is necessary for the macro to work. So, the first step is to object reference to Microsoft Outlook 16.0 Object Library. Once this step is completed, the OutlookObject is available to reference in the macro.VBA | Tools | ReferencesSelect Microsoft Outlook 16.0 Object LibraryHit OkThe second step in this macro is to insert a new module to store the VBA code. There are two options. The first option is to insert the new module into the current workbook or your personal macro workbook. As this macro is particular to this workbook, it makes sense to insert the new module there. If this code is to be used on multiple workbooks, then storing it in your personal macro workbook makes it available to re-use at any time in Excel. But, this macro will send an email update to colleagues on a regular report, so I insert the module into the current workbook. The name of this macro is SendMondayEmail.Sub SendByEmail()End SubThe End Sub line of code is automatically inserted into the code window as the macro is named. Write all other lines of code between these two lines. So, the next step is to declare any variables for this macro. This declaration means that Excel allocates some memory to store these values. In this macro, the Outlook Application is declared and the actual email. Finally, the attachment to send is declared as a string. After each of the first two Dim statements, the instance of each object is also set. This makes sense, as we are writing a macro to send emails from Excel using VBA.So, the email app will launch, and a new email will be ready to write.Dim EmailApp As Outlook.ApplicationSet EmailApp = New Outlook.ApplicationDim EmailItem As Outlook.MailItemSet EmailItem = EmailApp.CreateItem(olMainItem)Dim AttachSource As StringIt is time to write the mail components to get the recipients, the body of the message and add that ever-so-important email attachment. The first part of the mail is to TO property, which is the email address. At this stage, add any CC and blank CC if required. In this example, I have used the same mail address for demonstration purposes for the VBA code. The Subject is set as well as the email message in the body section of the email. Finally, the code for attaching the current workbook is set.EmailItem.To = "test@gmail.com"EmailItem.CC = "test@gmail.com"EmailItem.BCC = "test@gmail.com"EmailItem.Subject = "MondayReport"EmailItem.HTMLBody = "Hi," & vbNewLine & vbNewLine & "Please find attached Monday Report" & _vbNewLine & vbNewLine & _"Regards,"AttachSource = ThisWorkbook.FullNameThe last line of code (except the End Sub already entered by Excel at the beginning of the macro) is to send the email. Once Excel sends the email, the code ends. Feel free to take the code and test yourself to send your email using Excel VBA.Sub SendMondayEmail()Dim EmailApp As Outlook.ApplicationSet EmailApp = New Outlook.ApplicationDim EmailItem As Outlook.MailItemSet EmailItem = EmailApp.CreateItem(olMainItem)Dim AttachSource As StringEmailItem.To = "test@gmail.com"EmailItem.CC = "test@gmail.com"EmailItem.BCC = "test@gmail.com"EmailItem.Subject = "MondayReport"EmailItem.HTMLBody = "Hi," & vbNewLine & vbNewLine & "Please find attached Monday Report" & _vbNewLine & vbNewLine & _"Regards,"AttachSource = ThisWorkbook.FullNameEmailItem.SendEnd SubIn this blog post, we showed you how to use Excel VBA to send an email with attachment. This can be a very helpful tool if you need to regularly send emails with attachments. We hope you found this tutorial helpful and that it will make your life a little easier. If you have any questions or feedback, please don't hesitate to reach out to us. And be sure to sign up for our newsletter so you never miss another useful tip!So, if you want to read all of my macro and Excel tips then feel free to bookmark by FormulaFriday and MacroMonday pages. They are updated every week.All Macro Monday ArticlesAll Formula Friday Articles.Also please join thousands of other Excellers, whoc receive my free Excel Newsletter. Three Free Excel tips very month direct to your inbox. No Spam.Finally, if you like to watch Excel tip videos then head over to my YouTube channel with over one hundred Excel videos. Subcribe and get access immediately to my full playlist.link to How To Add A Dynamic Horizontal Target Line To An Excel Chartlink to How To Write Excel Macros For Beginners Last updated: 06th May 2024 We often use Microsoft Office Outlook application to manage emails, contacts etc. from home and office. It is arguably one of the safest and secure ways to manage confidential emails. It has many useful features. Here in this article, I am going to show you how to send emails from Excel step-by-step using VBA and Outlook. Although you can send and receive emails from Outlook directly, you can actually automate the process using a simple macro. Get Access to Microsoft Outlook in Excel using VBA There are two ways you can access Outlook properties and methods using VBA in Excel. 1) You can add a reference of Outlook object in your VBA project (This is called EarlyBinding.) Follow these steps to add a reference of the Outlook object in your VBA project In the top menu find Tools and choose References…. In the References dialog box, find Microsoft Outlook 16.0 Object Library, click the Check-box and press OK. Note: If you are using Office 2007, add Microsoft Outlook 12.0 Object library. 2) By creating an instance of Outlook using CreateObject() method. (This is called LateBinding.) To get access to Outlook methods and properties, you'll have to first create an instance of Outlook in VBA. Next, initialize Outlook using CreateObject(). Dim objOutlook as Object Set objOutlook = CreateObject("Outlook.Application") I am using the 2nd method in my example. Remember: You must first configure Microsoft Office Outlook in your computer. Else, the code example that I am going to show you here will not produce the desired result. Send email from Excel Copy the macro in your VBA editor. Option Explicit Private Sub send_email Dim objOutlook As Object Set objOutlook = CreateObject("Outlook.Application") Dim objEmail As Object Set objEmail = objOutlook.CreateItem(olMailItem) With objEmail .to = "xyz@email.com" .Subject = "This is a test message from Arun Banik" .Body = "Hi there" .Send End With Set objEmail = Nothing: Set objOutlook = Nothing End Sub If everything is right, then it will send an email with a subject and a message saying "Hi there". As you can see, I have created two objects (objOutlook and objEmail), one for outlook application and another for creating email. 01) Object objOutlook: Using the CreateObject() function, I have initialized Outlook. I can now access email properties with the CreateItem() method. 02) Object objEmail: Using this object, I'll create an Outlook item. This will give access to properties such as to, body and subject etc. Using .Display property to display message before sending email Here's another important feature that you can use. You can actually display or see the email message like body, subject, attachements etc. in Outlook before sending the email. To do this simply comment the .Send property and add .Display property. Please remember, the properties are case-sensitive. With objEmail .to = "xyz@email.com" .Subject = "This is a test message from Arun Banik" .Body = "Hi there" .Display End With This will now open Ms-Outlook application and shows the message with all the parameters. You can see the To address, with the Subject and Body. Click the Send button (in outlook) to email the message to its address. Add Attachments, CC and BCC in Excel Similarly, you can test with other important properties such, CC, BCC, Attachments etc. With objEmail .To = "xyz@email.com" .CC = "arun@mail.com" .BCC = "arun@hotmail.com" .Subject = "This is a test message from Arun" .Body = "Hi there" .Attachments.Add ("e:\report.doc") .Send End With Conclusion Now you know how to send emails from Excel using VBA and Outlook. Here we also learned how to add Attachments, Blind Carbon Copy (or BCC), CC etc., using VBA. It is very simple. Try working with other properties too. ← PreviousNext → We'll use a simple dataset with a list of people, their emails, and the city they're from to send emails to them. Use Macro to Send Email from Excel: 5 Ways Method 1 – Use the Outlook Object Library to Send Email Steps: From the Developer tab, select Visual Basic. Alternatively, you can press Alt + F11. From Tools, select "References…". A new dialog box will appear. Select Microsoft Outlook 16.0 Object Library and press OK. From Insert, select Module. Insert the following code into the new module. Sub Macro_Send_Email() Dim eApp As Outlook.Application Dim eSource As String Set eApp = New Outlook.Application Dim eItem As Outlook.MailItem Set eItem = eApp.CreateItem(olMailItem) eItem.To = Range("C5").Value 'These items are optional 'eItem.CC = "[email protected]" 'etem.BCC = "[email protected]" eItem.Subject = "Sending Email using VBA from Excel" eItem.Body = "Hello," & vbNewLine & "Hope this email finds you well." & _ vbNewLine & vbNewLine & _ "Sincerely," & vbNewLine & "Exceldemy" 'If you want to attach this workbook, then uncomment these two lines from below 'Source = ThisWorkbook.FullName 'eItem.Attachments.Add Source eItem.Display 'can use .Send End Sub VBA Code Breakdown We're calling our Sub Procedure Macro_Send_Email. We're selecting our email sending address from cell C5. The email content is set in our code. VBA Display Property is used to display our email. We'll need to press Send manually to send the emails. You can use the Send Property to send emails automatically. Save and close the Module. From the Developer tab, select Macros. The Macro dialog box will appear. Select the Sub Procedure "Macro_Send_Email". Press Run. You'll get an Outlook window with the email draft. You can press Send to send the email. Read More: Send Email from Excel VBA without Outlook Method 2 – Macro for Sending Emails from a Gmail Account in Excel Steps: Follow Method 1 to bring up the References dialog box. Select Microsoft CDO for Windows 2000 Library and press OK. Go to Security from your Google Account settings. Turn on Less secure app access. As shown in method 1, bring up the Module window and insert this code. Option Explicit Sub Send_Gmail_Macro() Dim cMail As Object Dim cConfig As Object Dim sConfig As Variant Dim cSubject As String Dim cFrom As String Dim cTo As String Dim cCC As String Dim cBcc As String Dim cBody As String cSubject = "Macro to Send Gmail" cFrom = "[email protected]" cTo = "[email protected]" cBody = "Hello. This is an automated message. Please don't reply" Set cMail = CreateObject("CDO.Message") On Error GoTo Error_Handling Set cConfig = CreateObject("CDO.Configuration") cConfig.Load -1 Set sConfig = cConfig.Fields With sConfig .Item(") = 2 .Item(") = "smtp.gmail.com" .Item(") = 1 .Item(") = "[email protected]" .Item(") = "SenderGmailPassword" .Item(") = 465 .Item(") = True .Update End With With cMail Set .Configuration = cConfig End With cMail.Subject = cSubject cMail.From = cFrom cMail.To = cTo cMail.TextBody = cBody cMail.CC = cCC cMail.BCC = cBcc cMail.Send Error_Handling: If Err.Description "" Then MsgBox Err.Description End Sub VBA Code Breakdown We're calling our Sub Procedure Send_Gmail_Macro. For login credentials, you need to type your own ID and Password here. We've set out port to 465. The code automatically sends the email. Read More: How to Send Email from Excel with Body Using a Macro Method 3 – Send Email to a List of Recipients from a Column We'll send to the emails in the cell C5:C10 range. Steps: As shown in method 1, bring up the Module window and insert this code. Sub Macro_Send_Email_From_A_List() Dim pApp As Object Dim pMail As Object Dim z As Integer Dim eList As String Dim eRow As Long Set pApp = CreateObject("Outlook.Application") Set pMail = pApp.CreateItem(0) eRow = Range("C:C").SpecialCells(xlCellTypeLastCell).Row - 1 With pMail eList = "" For z = 5 To eRow If eList = "" Then eList = Cells(z, 3).Value Else eList = eList & ";" & Cells(z, 3).Value End If Next z .BCC = eList .Subject = "Hello There" .Body = "This Message is brought to you by Exceldemy." .Display 'You can use .Send here End With Set pMail = Nothing Set pApp = Nothing End Sub VBA Code Breakdown We're calling our Sub Procedure Macro_Send_Email_From_A_List. We're selecting Outlook as our Mail Application. We're finding the last row, which is 10 for our dataset. As our emails starts from row 5 we've input 5 as the starting value for "variable z". The emails are on the C column, so we've input 3 inside the Cells property. We're setting the email content in our code. .Display is used here to display our email. You'll need to press Send manually to send the emails. You can use .Send to send email without displaying. Save and Run the Module. All our emails addresses are displayed in BCC. Press Send to send the emails. Read More: How to Apply Macro to Send Email from Excel with Attachment Method 4 – Macro to Send a Single Sheet Using Email We'll send the Active Worksheet to someone. Steps: As shown in Method 1, bring up the Module window and insert this code. Sub Macro_Email_Single_Sheet() Dim pApp As Object Dim pMail As Object Dim zBook As Workbook Dim fxName As String 'Dim zSheet As Worksheet 'Dim shtName As String Application.ScreenUpdating = False ActiveSheet.Copy Set zBook = ActiveWorkbook fxName = zBook.Worksheets(1).Name On Error Resume Next Kill "C:\Users\Rafi\OneDrive\Desktop\Softeko\47\" & fxName On Error GoTo 0 zBook.SaveAs FileName:="C:\Users\Rafi\OneDrive\Desktop\Softeko\47\" & fxName Set pApp = CreateObject("Outlook.Application") Set pMail = pApp.CreateItem(0) With pMail .To = "[email protected]" .Subject = "Macro to Send Single Sheet via Email" .Body = "Dear RecieverName," & vbCrLf & vbCrLf & _ "Your requested file is attached" .Attachments.Add zBook.FullName .Display End With zBook.ChangeFileAccess Mode:=xlReadOnly Kill zBook.FullName zBook.Close SaveChanges:=False Application.ScreenUpdating = True Set pMail = Nothing Set pApp = Nothing End Sub VBA Code Breakdown We're calling our Sub Procedure Macro_Email_Single_Sheet. We're copying the Active Sheet and saving it as a separate Workbook. We're selecting Outlook as our Mail Application. We're setting the email content in our code. We've attached the Sheet to the email. Use ".Display" to display the email, or use ".Send" to send the email automatically. Save and Run the Module. We'll see the Sheet name in the window. Press Send to complete the task. You can open the file and verify that the code is working. Read More: Excel Macro to Send Email Automatically Method 5 – Macro to Send an Email Based on Cell Value We've added the Payment Due column to the dataset. We'll send an email to a person that lives in Obama. Steps: As shown in Method 1, bring up the Module window and insert this code. Option Explicit Sub Send_Email_Condition() Dim xSheet As Worksheet Dim mAddress As String, mSubject As String, eName As String Dim eRow As Long, x As Long Set xSheet = ThisWorkbook.Sheets("Conditions") With xSheet eRow = .Cells(.Rows.Count, 5).End(xlUp).Row For x = 5 To eRow If .Cells(x, 4) >= 1 And .Cells(x, 5) = "Obama" Then mAddress = .Cells(x, 3) mSubject = "Request For Payment" eName = .Cells(x, 2) Call Send_Email_With_Multiple_Condition(mAddress, mSubject, eName) End If Next x End With End Sub Sub Send_Email_With_Multiple_Condition(mAddress As String, mSubject As String, eName As String) Dim pApp As Object Dim pMail As Object Set pApp = CreateObject("Outlook.Application") Set pMail = pApp.CreateItem(0) With pMail .To = mAddress .CC = "" .BCC = "" .Subject = mSubject .Body = "Mr./Mrs. " & eName & ", Please pay the due amount within the next week." _ & vbNewLine & "The exact amount is attached with this email." .Attachments.Add ActiveWorkbook.FullName 'Send The File via Email .Display 'We can use .Send here too End With Set pMail = Nothing Set pApp = Nothing End Sub VBA Code Breakdown We're calling our first Sub Procedure Send_Email_Condition. The function finds the last row of the dataset. Our value starts from row 5, so we've put row 5 to the last row in our code. We called our second Sub Procedure Send_Email_With_Multiple_Condition. We're selecting Outlook as our Mail Application. The email content is set in our code. We're attaching the Excel file with the email using the Attachment method. .Display is used to display our email. Use .Send to send the email without displaying. Read More: Send Reminder Email Automatically from an Excel Worksheet Using VBA Practice Section We've added practice datasets for each method in the Excel file. Download the Practice Workbook Using Macro to send Email.xlsm Related Articles Get FREE Advanced Excel Exercises with Solutions! Welcome to our comprehensive guide where we will show you how to use Excel VBA code to automate your email tasks and streamline your workflow. With this guide, you will be able to send emails directly from your Excel spreadsheet quickly and efficiently. Are you tired of manually sending emails to your clients or coworkers? By using Excel VBA code, you can automate your email tasks and save time. Whether you need to send invoices, reports, or updates, this step-by-step guide has got you covered. Follow the guide, and you will be sending emails effortlessly with Excel VBA in no time. Key Takeaways Excel VBA code can automate your email tasks and save time. By using Excel VBA code, you can send emails directly from your Excel spreadsheet. This guide will show you step-by-step how to create an email template, add recipients, compose email body, attach files, and send emails with Excel VBA code. Following the best practices and tips provided in this guide will ensure smooth and efficient email workflows. Excel VBA code can be used to enhance email automation by scheduling automated email tasks, creating email reports, and integrating with other applications. Introduction to Excel VBA Welcome to our step-by-step guide on using Excel VBA to automate your email tasks. Before we get started, let's take a closer look at Excel VBA. VBA stands for Visual Basic for Applications and it is a programming language that can be used to automate tasks in Microsoft Excel. With VBA, you can write macros and custom functions that perform various actions within Excel, including sending emails. Excel VBA is a powerful tool that can help you streamline your workflow and save time. In the following sections, we will guide you through the process of setting up your email environment, creating an email template, adding recipients and a subject line, composing the email body, adding attachments, sending the email, enhancing email automation, and best practices and tips. Setting up the Email Environment Before we start automating email tasks in Excel, it is crucial to set up the email environment. This process consists of configuring your email account settings in Excel and installing the necessary libraries or add-ins. Here's a step-by-step guide to help you get started: 1. Enable Sending Email in Excel First, we need to give Excel permission to send emails. To do this: Open Excel and click on "File" in the upper-left corner. Select "Options" from the left-hand menu. Click on "Trust Center" in the left-hand menu. Click on "Trust Center Settings" on the right-hand side. Select "Macro Settings" in the left-hand menu. Select the option "Enable all macros" and check the box next to "Trust access to the VBA project object model." Click "OK" to save the changes. 2. Configure Email Account Settings Next, we need to configure the email account settings. Depending on your email provider, the steps may vary slightly. Here's a general guide: Open Excel and click on "File" in the upper-left corner. Select "Options" from the left-hand menu. Click on "Advanced" in the left-hand menu. Scroll down to the "General" section and find the "Web Options" button. Click on it. Select the "Custom" tab and click on the "Custom Level" button. Scroll down to the "Scripting" section and find "Active scripting." Select "Enable" and click "OK". Return to the Excel options window and click on "Email Options." Fill in your email details, including the email address, type of account, incoming and outgoing server information, and your login credentials. Test the account by clicking the "Test Account Settings" button. Once everything is configured correctly, click "OK" to save the changes. 3. Install Necessary Libraries or Add-Ins Finally, we may need to install additional libraries or add-ins, depending on the email provider and Excel version. The most common library for email automation is the Microsoft Outlook Object Library, which provides access to Outlook emails and settings. To install this library: Open Excel and click on "File" in the upper-left corner. Select "Options" from the left-hand menu. Click on "Add-Ins" in the left-hand menu. In the "Manage" dropdown list at the bottom of the window, select "COM Add-ins" and click "Go". Check the box next to "Microsoft Outlook Object Library" and click "OK". Once you have completed these steps, your email environment will be set up, and you're ready to start automating email tasks using Excel VBA. Creating an Email Template Creating an email template in Excel VBA is crucial for streamlining and automating email tasks. Having a well-designed template saves time and effort, especially when sending similar emails frequently. Here is a step-by-step guide on how to create a customizable email template within Excel: Open Microsoft Excel and create a new workbook. Click on the Developer tab on the ribbon. Click on Visual Basic to open the Visual Basic Editor. Click on Insert and select Userform to create a new userform. Drag and drop controls, such as text boxes, buttons, and labels to create your desired layout for the email template. Double-click each control and enter the appropriate code to control its behavior. For example, you can use the Text property to set the default text in a text box. Save the userform as a template for future use. Remember to include all essential elements when creating an email template. These may include a clear subject line, a professional email greeting, broken down content sections, and a closing sign-off. It's helpful to add placeholders for dynamic data such as names, dates and amounts using the code functionality of Excel VBA. Our sample code can help you get started and improve your template's versatility. Sample Code for Email Template 'Displays the user form to create a new email Sub create_new_email() Dim new_email As Object Set new_email = UserForms.Add("Email_Template") new_email.Show End Sub Adding Recipients and Subject Line After creating the email template, the next step is to add recipients and a subject line. This will ensure that your emails are personalized and can grab the attention of the recipient. Excel VBA allows you to dynamically add recipients from an Excel range to the email and specify the subject line using code. Below is a step by step guide: Firstly, open the Excel worksheet that contains the email addresses you intend to add to the email. Next, create a range of these email addresses. Then, declare a recipient variable in the VBA code like this: Dim recipient As String This variable will be used to store the recipient email addresses that will be added to the email. Use the VBA code below to add recipients to the email: For Each cell In [recipient_range] recipient = cell.Value Set objOutlookRecip = objOutlookMsg.Recipients.Add(recipient) objOutlookRecip.Type = olTo Next The VBA code above adds the recipient email addresses from the range you created in step 2 to the email. It loops through each cell in the range and sets the email recipient as the value of the cell. You can modify this code to suit your specific requirements. Finally, add a subject line to the email using the VBA code below: objOutlookMsg.Subject = "Your subject line here" Replace the text "Your subject line here" with the actual subject line you want to use for the email. By following these simple steps, you can add recipients and subject line to your email to make it more effective and personalized. Composing the Email Body The body of your email is where you can convey your message effectively. To compose an email body using Excel VBA, follow these steps: Define the email message: Begin by defining the message you want to communicate in your email body. This could be a simple text message or a more detailed message with formatting. Add formatting to the email: You can add formatting to the email body using VBA code such as italicizing or bolding text, changing the font color, aligning text, and more. Include dynamic data: You can even include dynamic data from your Excel spreadsheet in the email body. For example, you can insert names, email addresses, or other data that is specific to each recipient. Keep your email message clear and concise. Avoid large blocks of text or overwhelming the recipient with too many details. Use formatting to break up the text and make key points stand out. "The key to writing a successful email body is to keep it informative, engaging, and personalized." By following these steps, you can compose an email body that effectively communicates your message and engages your recipient. Adding Attachments Sending email attachments is a common need in many automated workflows. Fortunately, Excel VBA makes it easy to add attachments to the emails you send directly from your spreadsheet. Here's a step-by-step guide on how to add both single and multiple attachments: Single Attachment To add a single attachment to your email, use the following code: Attachments.Add "C:\Path\To\File.xlsx" This code specifies the file path of the attachment as a string argument. You can replace this with the file path of your own attachment. That's it! Your attachment will now be included in the email. Multiple Attachments If you need to add multiple attachments to your email, use the following code: Attachments.Add "C:\Path\To\File1.xlsx" Attachments.Add "C:\Path\To\File2.pdf" As you can see, simply add a new line for each additional attachment, using the same format as for a single attachment. This will add each file path to the email's list of attachments and include them with the email. Adding attachments with Excel VBA can greatly enhance the usefulness of your automated email workflows. However, be mindful of file size limitations and ensure that the attachments are relevant to the email's content. Sending the Email Finally, it's time to send your email! With all the necessary components in place, using Excel VBA to send email is a straightforward process. Here's a step-by-step guide: Create a new subroutine or function in your VBA editor Define your email object using the Dim statement and set it to create a new message Specify your recipient(s) using the To property of the email object. You can use a cell range if you want to send emails to multiple recipients at once Add a subject line using the Subject property Compose your email body using the Body property or add a pre-defined html for the body part If you want to attach a file, use the Attachments property to specify the file path Use the .Send method to send your email Don't forget to handle potential errors and exception situations and run some test emails before sending to ensure that everything works as expected. Note: For advanced users, you can use macros to set up delayed email scheduling, so that you can send emails at specified times. "Sending emails using Excel VBA code is a game-changer when it comes to automating your workflow. It not only saves time but also ensures accuracy while sending emails in bulk. " Enhancing Email Automation After mastering the basics of sending emails with Excel VBA, there are several ways to further enhance the automation process. In this section, we will explore advanced techniques that will help you streamline your workflow and optimize your email tasks. Scheduling Automated Email Tasks One way to take email automation to the next level is to schedule tasks ahead of time. This can be done by combining Excel VBA with the task scheduler built into Windows. By doing this, you can set up an automated email task to run at a specific time, whether hourly, daily, or weekly. This way, you can reduce the amount of time spent manually sending emails and focus on other important tasks. Creating Email Reports Another way to enhance email automation is by generating email reports. With Excel VBA, you can create reports that provide real-time information on email performance, including the number of emails sent, delivery rates, and open rates. By analyzing this data, you can improve your email strategy and achieve better engagement with your recipients. Email Metric Definition Open Rate The percentage of email recipients that opened your email. Click-Through Rate The percentage of email recipients that clicked a link within your email. Bounce Rate The percentage of email addresses that were undeliverable. Integrating with Other Applications Excel VBA can also be used in conjunction with other applications to streamline workflows and enhance automation. For example, you can integrate VBA code with CRM software to automatically send follow-up emails to contacts or pull data from other applications to populate email templates. Tip: When integrating with other applications, always ensure that the data is accurate and in the correct format. Inaccurate or incomplete data can lead to errors and decrease the effectiveness of your email automation. By implementing these advanced techniques, you can take your email automation to the next level and achieve greater efficiency in your workflows. Experiment with different scenarios and find the combination of techniques that works best for your specific needs. Best Practices and Tips Now that you know how to automate your email tasks using Excel VBA, it's time to optimize your workflow. Here are some best practices and tips to improve your email automation: 1. Keep it Simple When designing your email template, keep it simple and easy to read. Use a clear and concise subject line and limit the body of the email to the essential information. This will ensure that your recipients can quickly and easily understand the purpose of the email. 2. Error Handling When sending emails, it's essential to include error handling to ensure the successful delivery of emails. Use debugging techniques to identify and fix errors in your VBA code. Test your email template with a small group of recipients to ensure that everything is working correctly. 3. Adhere to Email Etiquette Make sure that you follow proper email etiquette in your messages. Avoid using all caps, excessive punctuation, or slang vocabulary. Be polite and professional in your emails, and ensure that your tone is appropriate for the context of the message. 4. Optimize Performance To optimize the performance of your email automation, ensure that your VBA code is optimized for speed and efficiency. Avoid using unnecessary loops or functions that could slow down the process. Additionally, ensure that your email account settings are configured correctly to avoid delays or errors in sending emails. 5. Test and Experiment Don't be afraid to test and experiment with different scenarios and templates. Use A/B testing techniques to determine the most effective email format and content. Continuously improve your email automation to ensure that it meets your evolving needs and requirements. "Automating your email tasks using Excel VBA can save significant amounts of time and streamline your workflow. By following these best practices and tips, you can ensure that your email automation is efficient, professional, and effective." Conclusion Congrats! You have successfully mastered the art of using Excel VBA code for sending emails. You can now streamline your workflow and save valuable time by automating your email tasks directly from your Excel spreadsheet. Remember to keep practicing and experimenting with different scenarios to fully utilize the power of Excel VBA in your email automation. By following the step by step guide provided in this article, you have learned how to: Set up the email environment in Excel VBA Create customizable email templates Add recipients, subject lines, and email bodies dynamically Attach files to your emails effortlessly Send the email using Excel VBA code Enhance the automation further with advanced techniques Implement email best practices and tips for optimal performance Now that you have this knowledge, you are well on your way to becoming an Excel VBA email automation expert. Happy coding! Excel VBA stands for Visual Basic for Applications and is a programming language used to automate tasks in Microsoft Excel. It allows users to write macros and custom functions to perform various actions within Excel, including sending emails. With Excel VBA code, you can automate the process of sending emails directly from your Excel spreadsheet. By writing code, you can streamline your workflow and save time by sending personalized emails to multiple recipients with ease. Yes, before sending emails from Excel, you need to set up the email environment. This involves configuring your email account settings within Excel and installing any necessary libraries or add-ins. We will guide you through this process step by step. Absolutely! In fact, creating an email template in Excel is a great way to standardize your email format and ensure consistency. We will show you how to create a customizable email template within Excel and provide sample code to get you started. Adding recipients and a subject line to your emails is crucial for effective communication. In Excel VBA, we will demonstrate how to dynamically add recipients from an Excel range and specify a subject line using code. This will enable you to send personalized emails to multiple recipients effortlessly. Composing the email body in Excel VBA offers various methods to convey your message effectively. We will guide you through different techniques, such as including text, formatting, and even dynamic data from your Excel spreadsheet. Absolutely! Sometimes, you may need to attach files when sending emails. In Excel VBA, we will show you how to add attachments effortlessly, whether it's a single file or multiple files. Sending the email is the final step in the process. We will demonstrate how to use Excel VBA code to send the email directly from your spreadsheet. We will also cover error handling and provide troubleshooting tips to ensure successful email delivery. Absolutely! Once you have mastered the basics of sending emails with Excel VBA, there are several ways to enhance automation further. We will explore advanced techniques such as scheduling automated email tasks, creating email reports, and integrating with other applications. In order to optimize your email automation with Excel VBA, we will share best practices and useful tips. You will learn how to improve performance, handle errors gracefully, and adhere to email etiquette. Implementing these best practices will ensure smooth and efficient email workflows. Congratulations on completing this comprehensive guide on Excel VBA code for sending emails! By now, you should have a solid understanding of how to automate your email tasks using VBA in Excel. Remember to practice and experiment with different scenarios to fully utilize the power of Excel VBA in email automation. VBA code provides a straightforward and efficient way to send emails directly from Excel. Utilizing the Outlook application can automate composing and sending emails, saving time and streamlining your communication tasks. By leveraging VBA code to send emails from Excel, you can automate repetitive tasks, improve efficiency, and ensure accurate and timely communication with your recipients. It provides seamless integration between Excel and Outlook, offering a convenient solution for email automation. In the provided screenshot, we observe that upon running the VBA code to send emails, Outlook is launched automatically. The email's recipients, subject line, and body are populated per the code, streamlining the process and offering a time-saving solution for sending bulk emails. VBA send Email from Excel enables you to automate the process of sending emails from Excel, allowing you to efficiently communicate with multiple recipients. To send emails from Excel using VBA, you need to enable Outlook scripting and set a reference to the Microsoft Outlook Object Library. The key steps in VBA send Email from Excel include creating a new instance of Outlook, creating a new email item, setting the email properties such as recipients, subject, and body, and finally sending the email using the ".Send" method. VBA send email provides flexibility in managing email properties, such as specifying recipients in the To, CC, and BCC fields, setting the subject and body of the email, and even attaching files to the email using the Attachments property. To send emails from Excel using VBA, follow the below steps. You must initialize variables to store important information, such as the Outlook application object, mail item object, email subject, body, and recipient's email address. Then, create an instance of the Outlook application and a new mail item object. By setting its relevant properties, such as subject, body, and recipients, you can customize the email's content. Additionally, you can attach files to the email using the "Attachments.Add" method. 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. Setting a reference to the Microsoft Office Library is an essential step when working with VBA code in Excel to utilize the features of Microsoft Office applications like Outlook. This reference allows us to access the necessary objects and methods provided by the library to interact with Outlook seamlessly. Here's a step-by-step guide on how to set the reference: Open the Visual Basic Editor (VBE) in Excel by pressing Alt+F11. In the VBE window, go to the "Tools" menu and select "References" from the dropdown menu. It will open the References dialog box. In the References dialog box, scroll down, and locate the entry named "Microsoft Office XX.0 Object Library," where "XX.0" represents the version number of Microsoft Office installed on your computer. For example, it could be "Microsoft Office 16.0 Object Library" for Office 2016. Check the checkbox next to the "Microsoft Office XX.0 Object Library" entry to select it. Click the "OK" button to close the References dialog box. By setting this reference, you enable VBA to recognize and use the objects and methods available in the Microsoft Office Library, including those specific to Outlook. It allows you to write VBA code that interacts with Outlook, such as sending emails or manipulating Outlook items. Setting the reference to the Microsoft Office Library ensures that your VBA code can access the necessary tools and functions required to integrate Excel with Outlook seamlessly, opening a wide range of possibilities for automating tasks and enhancing productivity. To send emails using VBA, you need to utilize the capabilities of Microsoft Outlook. Here's an example of a VBA code that sends an email from Excel using Outlook. Please refer to the "VBA Send Email from Excel" Excel workbook. Step 1: Open the Visual Basic Editor (VBE) by pressing the Alt + F11 keys in Excel or clicking on the Developer tab in the ribbon and then click on the Visual Basic button. In the VBE, click on the Insert menu and select Module to create a new module. Step 2: In the new module, start the sub-procedure named "SendEmail" and declare the necessary variables and objects. For this example, we declare two variables ("OutlookApp" and "OutlookMail") as objects. These objects will be used to interact with Outlook and create an email. Step 3: Create a new instance of Outlook. The above line of code creates a new instance of Outlook and assigns it to the "OutlookApp" variable. It allows us to access Outlook's features and functionality. Step 4: Create a new email using the Set statement. This line of code creates a new email item using the "CreateItem" method of the "OutlookApp" object. The argument "(0)" specifies that we want to create an email. Step 5: Set the properties of the email. This code sets various properties of the email using the "With" statement. Step 6: Enter the recipient's email address. Replace "recipient@example.com" with the actual email address of the recipient. This sets the "To" field of the email. Step 7: Enter any additional recipients in the CC field. If you want to carbon copy (CC) anyone on the email, replace "cc@example.com" with the actual email address. It sets the "CC" field of the email. Step 8: Enter any additional recipients in the BCC field. If you want to blind carbon copy (BCC) anyone on the email, replace "bcc@example.com" with the actual email address. It sets the "BCC" field of the email. Step 9: Set the subject of the email. Replace "Test Email" with the desired subject for your email. It sets the subject field of the email. Step 10: Set the body of the email. Replace the text within the quotation marks with the desired content for the email body. It sets the body text of the email. Step 11: Display the email for review before sending. This line of code displays the email in Outlook, allowing you to review the content, recipients, subject, and any other details before sending it. If you want to send the email, just replace ".Display" with ".Send" in the VBA code. Step 12: Clean up the Outlook objects. After sending the email or closing the email window, it's essential to release the Outlook objects from memory to free up system resources. Setting the objects to Nothing ensures proper cleanup. Step 13: Customize and test the code. Explanation: Customize the code by modifying the email recipients, subject, body, and any other properties to suit your specific needs. Once you're done, run the code to send the email from Excel using VBA. Here is the full code: Sub SendEmail() Dim OutlookApp As Object Dim OutlookMail As Object ' Create a new instance of Outlook Set OutlookApp = CreateObject("Outlook.Application") ' Create a new email Set OutlookMail = OutlookApp.CreateItem(0) ' Set the properties of the email With OutlookMail .To = "recipient@example.com" .CC = "cc@example.com" .BCC = "bcc@example.com" .Subject = "Test Email" .Body = "This is a test email sent from Excel." .Display ' Use .Send instead of .Display to send the email without displaying it End With ' Clean up Set OutlookMail = Nothing Set OutlookApp = Nothing End Sub While sending emails from Excel using VBA, there are a few vital considerations to keep in mind. When using VBA to send an email from Excel with a signature, the signature might not be included in the email body. By default, the email body generated through VBA does not include the user's Outlook signature. Therefore, you need to handle the signature explicitly in your VBA code. When using VBA send email from Excel range, specify the range of cells containing the email data accurately. Ensure that the range includes all the necessary information, including recipients' email addresses, subject, and body content. Ensure the data is organized to align with the desired email layout. When using VBA send email from Excel using the Oft template, it's important to declare and create the necessary Outlook objects, such as "Outlook.Application" and "Outlook.MailItem". When using VBA send email from Excel with attachment, ensure that the file paths are accurate and accessible. Consider the file size limitations and the impact on email delivery and recipient's inbox. 1. How to send bulk emails from Outlook Using Excel VBA? To send bulk emails from Outlook using VBA Send email from Excel, follow these steps:1) First, prepare an Excel spreadsheet with recipient email addresses, subject, and body.2) Enable the "Microsoft Outlook Object Library" reference in VBA.3) Write a VBA subroutine to loop through the spreadsheet and send emails. To do this, you can set a range of email addresses in the worksheet using the following code:"Set Recipients = Sheets("Sheet1").Range("A2:A10") ' Update the sheet name and range as needed"4) Declare Outlook application and mail item objects using "CreateObject".5) Iterate through each recipient using a loop. This can be done using the following code:For Each Recipient In Recipients ' Create a new email Set OutlookMail = OutlookApp.CreateItem(0)6) Run the code to send bulk emails.7) Review and refine the process based on feedback. We have Employee Restructured Salary data in Excel, and we want to send automatic emails using Outlook. Opening Microsoft Visual Basic and Inserting Code in Module Opening the VBA Window: Using Keyboard Shortcuts: Press Alt + F11. Using the Developer Tab: In an Excel worksheet, go to the Developer Tab and select Visual Basic. Using the Worksheet Tab: Go to any worksheet, right-click on its name and choose View Code from the Context Menu. Inserting a Module in Microsoft Visual Basic: Using Sheet Options: Select a Worksheet, then right-click on it, select Insert from the Context Menu, then choose Module. Using the Toolbar: Select Insert from the Toolbar and choose Module. Method 1 – Using Excel VBA Macro to Automatically Send Email from Outlook to the Selected Recipients Steps: Go to the Insert tab. Select Shapes. Select any of the offered shapes (i.e., Rectangular: Rounded Corners). Drag the Plus Icon wherever you want to insert the Shape as shown in the image below. Choose a preferred Shape Fill and Outline Color, then right-click on the shape. Click on Edit Text to insert text. Open Microsoft Visual Basic and insert a Module. Paste the following Macro in the Module. Sub ExcelToOutlookSR() Dim mApp As Object Dim mMail As Object Dim SendToMail As String Dim MailSubject As String Dim mMailBody As String For Each r In Selection SendToMail = Range("C" & r.Row) MailSubject = Range("F" & r.Row) mMailBody = Range("G" & r.Row) Set mApp = CreateObject("Outlook.Application") Set mMail = mApp.CreateItem(0) With mMail .To = SendToMail .Subject = MailSubject .Body = mMailBody .Display ' You can use .Send End With Next r End Sub 1 – starts the macro procedure by declaring the variables as Object and String. 2 – runs a VBA FOR loop for each row in the selection to assign Email's Send To, Subject, and Body using row entries. 3 – assigns variables. 4 – performs the VBA With statement to populate Outlook items such as Send To, Mail Subject, etc. Here the macro only executes the Display command to bring out Outlook with an email draft. However, if Send command is used in place or after Display, Outlook will send the created email to the selected recipients. 5 – finishes the VBA FOR loop. Return to the worksheet. Right-click on the Shape and select Assign Macro from the Context Menu options. Select the Macro ExcelToOutlookSR under the Macro name and choose the Macro in option as This Workbook. Click on OK. In the worksheet, select one or multiple employees, then click on the Shape. Excel prompts Outlook to launch and creates or sends emails to the selected employees. Since we selected two employees, Outlook generated two different email drafts ready to be sent. Use the Send command to automatically send emails from Excel to Outlook using cell entries. Read More: How to Send Excel File to Email Automatically Method 2 – Sending Email Automatically from Excel to Outlook Depending on Specific Cell Value We have Quarterly Sales Data as depicted below. After achieving a target (i.e., Sales > 2,000), we will automatically prompt Outlook to send an email from Excel to an assigned email. Use the following macro code in a VBA Module. Option Explicit Dim Rng As Range Sub Worksheet_Change(ByVal mRng As Range) On Error Resume Next If mRng.Cells.Count > 1 Then Exit Sub Set Rng = Intersect(Range("F17"), mRng) If Rng Is Nothing Then Exit Sub If IsNumeric(mRng.Value) And Target.Value > 2000 Then Call ExcelToOutlook End If End Sub Sub ExcelToOutlook() Dim mApp As Object Dim mMail As Object Dim mMailBody As String Set mApp = CreateObject("Outlook.Application") Set mMail = mApp.CreateItem(0) mMailBody = "Greetings Sir" & vbNewLine & vbNewLine & _ "Our outlet has quarterly Sales more than the target." & vbNewLine & _ "It's a confirmation mail." & vbNewLine & vbNewLine & _ "Regards" & vbNewLine & _ "Outlet Team" On Error Resume Next With mMail .To = "[email protected]" .CC = "" .BCC = "" .Subject = "Notification on Achieving Sales Target" .Body = mMailBody .Display 'or you can use .Send End With On Error GoTo 0 Set mMail = Nothing Set mApp = Nothing End Sub 1 – assigns a cell (i.e., F17) within a range to execute the VBA IF statement. If the statement results in True, the macro calls another macro for execution. 2 – declares variable types and assign them to populate Outlook's entries. 3 – performs VBA With statement to assign variables to email entries. Use the Send command instead of Display in case you directly want to send emails without reviewing them. The recipient email is inserted within the macro. Use alternative methods in case you want an automatic insertion of the recipient's email id. 4 – clears certain variables. Use the F5 key to run the macro. Excel fetches Outlook with a draft email created automatically as shown. You can click on Send or auto-send using the Send command in the macro. Read More: Automatically Send Emails from Excel Based on Cell Content Method 3 – Creating VBA Macro to Send Email with the Active Worksheet from Excel by Outlook Insert the following macro in a Module. Function ExcelOutlook(mTo, mSub As String, Optional mCC As String, Optional mBd As String) As Boolean On Error Resume Next Dim mApp As Object Dim rItem As Object Set mApp = CreateObject("Outlook.Application") Set rItem = mApp.CreateItem(0) With rItem .To = mTo .CC = "" .Subject = mSub .Body = mBd .Attachments.Add ActiveWorkbook.FullName .Display 'or you can use .Send End With Set rItem = Nothing Set mApp = Nothing End Function Sub OutlookMail() Dim mTo As String Dim mSub As String Dim mBd As String mTo = "[email protected]" mSub = "Quarterly Sales Data" mBd = "Greetings Sir" & vbNewLine & vbNewLine & _ "Kindly find Outlet's Quarterly Sales data attached with this mail." & vbNewLine & _ "It's a notification mail." & vbNewLine & vbNewLine & _ "Regards" & vbNewLine & _ "Outlet Team" If ExcelOutlook(mTo, mSub, , mBd) = True Then MsgBox "Successfully created the Mail draft or Sent" End If End Sub 1 – declares and set the variables. 2 – assigns the commands using the VBA With statement. Use the Display or Send command for reviewing or direct sending emails respectively. 3 – clears the previously set variables. 4 – assigns the VBA With commands with texts. 5 – executes the VBA Custom Function. Press F5, and you'll get a draft email to review similar to the below image. Read More: How to Send Multiple Emails from Excel Spreadsheet Method 4 – Sending an Automatic Email from Excel to Outlook Using the HYPERLINK Function Use the following formula in cell H5. =HYPERLINK("MailTo:"&C5&"?Subject="&F5&"&cc="&D2&"&body="&G5,"Click Here") The HYPERLINK function takes "MailTo:"&C5&"?Subject="&F5&"&cc="&D2&"&body="&G5 as link_location, and "Click Here" as friendly_name. Hit Enter to paste the link. Click on the link. Excel takes you to Outlook. You'll see all the Outlook entries are filled with assigned data from Excel. Click on Send. Drag the Fill Handle to apply the formula to other cells and repeat sending. Read More: How to Send an Editable Excel Spreadsheet by Email Download the Excel Workbook Send Automatic Email.xlsm Related Articles