



VBA FINAL PROJECT

BY BLAKE JONES

DECEMBER 9, 2010

EXECUTIVE SUMMARY

This VBA project improves how customer information is recorded and tracked by C-Clearly windows. C-Clearly is a small window cleaning company that a friend and I started two summers ago. Our main focus is residential customers, although we have done a few commercial jobs. We currently have 50 customers listed on a spread sheet who we've served over the last two years, and we plan on adding more to that list next spring.

Problems: Our spreadsheet is sporadically populated, often containing only the names of people and missing important information such as phone numbers and whether or not we've received payments. We are very busy over the summer, and often forget or are too tired to update our records properly at the end of each day. Also, we've had difficulty remembering important tasks such as following up with contacts, calling customers in advance of upcoming appointments, and tracking payment receipts and deposits.

VBA Solution: My program solves these problems by taking the user through a form walkthrough. The walk-through goes step by step, prompting the user to information about completed jobs, payment receipts and deposits, follow-up items, and new customers. When all new information is entered the VBA generates a 10 day calendar that lists which jobs are scheduled for each day. The calendar also includes weather forecasts pulled from weatherchannel.com. The program then searches the database and creates a list of outstanding checks and follow-up items, highlighting in red those items that are past due. Finally, the program sends a text message with a reminder to call tomorrow's appointment with the accompanying job and contact information.

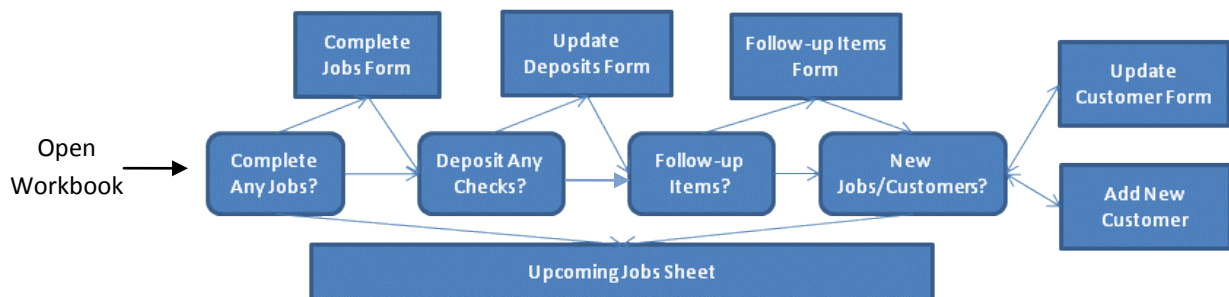
DESIGNING THE DATABASE

Before I could start writing VBA code, I had to design a database that would meet the needs of my small company. I had to choose carefully what information I included and what information I left out. I knew that once I started writing code, editing the foundation of the database could cause problems. Creating the database was more difficult and time consuming than originally expected, and I frequently had to go back and make changes as the project progressed. If I were to do this project again, I would spend more time on this step, carefully deciding what information was needed so that I would not have to go back and make changes in the middle.

	A	B	C	D	E	F	G	H	I	J	K	L
	Name	Street	City	State	Zip Cod	Phone#	Date Contacted	Recent Contacted	Date Scheduled	Time Scheduled	Current Quote	Date Complete
2	Rigby, Brennan & Melissa	27 Parley's Way	Salt Lake	UT	84201	(801) 546 6092	10/10/1997	11/28/2010	12/9/2010	10:00	\$ 120.00	12/00:00 Ar
3	Jones, Blake & Nellie	627 N 600 W Apt 10	Provo	UT	84601	(801) 540 9579	10/1/2008	12/5/2010	12/8/2010	10:00	\$ 90.00	
4	Jones, Gary & Sue	52 N Cove Lane	Layton	UT	84040	(801) 540 9579	10/18/1985	11/28/2010		9:30	\$ 200.00	12/6/2010
5	Croft, Daniel & Rowlene	709 S Angel Street	Layton	UT	84041	(801) 544 3220	6/27/2009		12/6/2010	10:00	\$ 165.00	
6	Nelson, Becky & David	730 Kensington St	Farmington	UT	84025	801 451 6827	7/1/2009		12/3/2010		\$ 100.00	
7	Harris, Thomas & Emily			UT			4/30/2010		12/4/2010		\$ 200.00	
8	McMillens, Jenine & Glen		Layton	UT	84040	801 544 5615	4/30/2010				\$ 125.00	
9	Mallards		Layton	UT	84040		5/1/2010				\$ 250.00	
10	Humphreys, Raelene & Mick	3028 E Corral Drive	Layton	UT	84040	(801) 544 8870	5/1/2010				\$ 216.00	
11	Kopsch, Sydney	Next ot Moons	Layton	UT	84040	(801) 721 8734	5/1/2010				\$ 93.00	
12	Haberstock, Diane & Bill	25 N 3175 E	Layton	UT	84040	(801) 544 0262	5/1/2010				\$ 90.00	
13	Priests, Scott & Sally		Layton	UT	84040		5/8/2009		5/12/2010		\$ 318.00	5/12/201
14	Hollands, Grant & Andrea	Next door	Lavton	UT	84040		4/30/2010		5/13/2010		\$ 250.00	5/13/201

CREATING THE WALKTHROUGH

The next step was to create the forms needed for the project. The purpose of the program is to ensure that that customer information is accurate and up to date, so I used forms to create a walkthrough system. The system forces the user to consider new information every time the workbook is opened, and gives them a uniform format to enter new information into the database. The flow of the forms is as follows:



The first form, "frmCompleteJobs", is programmed to open with the workbook. This form welcomes the user and asks if any jobs have been completed. If "Yes" is clicked, it takes the user to the Update Jobs form, "frmUpdateJobs", if no is clicked, the user is taken to the Check deposits form "frmCheckDeposits". The process continues through the Follow-up Items and Adding New Jobs/Customers forms. At any point during the walkthrough, you can click "Skip Intro" and be taken directly to the Upcoming Jobs" page of the work book.

Update Followup Activities

Did you complete any follow-up items?

Skip Intro No Yes

Job Completion

Hello Blake,

Did you complete any jobs today?

Skip Intro No Yes

Check Deposits

Did you deposit any outstanding checks?

Skip Intro No Yes

UPDATING THE DATABASE

The forms to update the database are more intensive than the walkthrough forms. First I created a form to add new customers, then two separate but similar forms to update current customer information (one to update completed jobs and one used to update general information). Finally I created forms to review and update check deposits and open followup items.

Add New Customer

The Add New Customer form, “frmAddCustomer”, was the first form created after the database was finished. It’s similar to the forms we used during our USDA assignment, but has a few elements worth noting:

It finds the last row by selecting a cell below the bottom of the datasheet and using .endlup, then selecting $r = \text{activecell.row} + 1$

The VBA declares a variable for each textbox based on the information entered. For example, “Date Scheduled” would be set as date, “Quote” as currency, etc. When the save button is clicked, textbox values are assigned to their respective variables and the variables to the database. A variable termed “Blank” is set as string and equal to the value of “-----”. All empty fields are assigned to the blank variable. This ensures information is entered into the database in its proper format so that it can be manipulated later, and that there are no blank cells.

```
'find next empty row
Cells(10000, 1).Activate
With range(ActiveCell.address)
    range(.End(xlUp), .End(xlToLeft)).Select
End With
r = ActiveCell.Row + 1
Cells(r, 1).Select
```

```
Dim blank As String
Dim quote As Currency

If Not txtquote = "" Then quote = txtquote

If txtquote = "" Then .Cells(r, 11).Value = blank Else .Cells(r, 11).Value = quote
```

Complete Jobs

The Complete Jobs form, “frmUpdateJobs”, automatically populates a list box showing the user which jobs should have been completed. You can then select a job from the list, click the “Update” button, and the information about the job will be pulled from the database. After updated information is entered in, like completion date, payment amount and payment form, the save button updates the database and clears the text boxes. Another job can be selected or the user can move on to the next step.

The list box is populated on the Initialize event, and a For statement cycles through all the rows of the database. If the scheduled date is less than or equal to the current date, and if the completed date is not less than the current date (prevents old jobs from showing up), then the customer and the scheduled date appear in the list box.

```
For r = 2 To finalrow
    If Not Cells(r, 9) = "-----" And Not Cells(r, 12) < Sheets("UpcomingJobs").Cells(4, 8).Value Then
        If Sheets("Customer Database").Cells(r, 9) <= Sheets("UpcomingJobs").Cells(4, 8).Value Then
            lstcomplete.AddItem "Scheduled " & Cells(r, 9) & " " & Cells(r, 1).Value
        End If
    End If
Next
```

The Form uses combo boxes for Payment Form and Check Deposited, as well as Option Buttons for Outside vs. Inside. This helps create consistency in the database and helps when calculating “estimated completion time” and tracking check deposits.

cbodeposited.AddItem "-----" cbodeposited.AddItem "Yes" cbodeposited.AddItem "No"	cboform.AddItem "-----" cboform.AddItem "Cash" cboform.AddItem "Check"
---	--

When the form populates its list, the job and date are concatenated into one long text string. When a job is selected to update, the form finds the correct row using a Mid function to grab the name out of the string and searches the database for that name using the InStr function.

```

'-1 is the value of listindex when nothing is selected
If lstcomplete.ListIndex = -1 Then
    MsgBox "You must select a completed job"
Else
    x = Mid(lstcomplete.Text, 31, 8)

For r = 2 To finalrow
    If InStr(1, Cells(r, 1).Value, x) > 0 Then

```

Creating the Complete Jobs forms was much more difficult than anticipated. I wanted to save the data as variables set as dates, currency, etc., instead of as string so that it could be manipulated appropriately, but some cells have data, and others have the filler "----". If I used the method I did on the Add Customer Form, and set textboxes equal to declared variables, the form would crash because variables and data types sometimes didn't match. If I dimmed all variables as variant, the form wouldn't crash but would return 12:00:00 AM where dates we're cleared or left "----". To solve the problem, I creating the blank variable and dimed it as string as mentioned above. For example, if "txtDateScheduled" was populated with a date, then the variable "Date" which is dimmed as date would bet set equal to it. If

"txtDateScheduled" was left blank, than the string variable "Blank" would be set equal to it.

```

Dim blank As String
Dim quote As Currency

If Not txtquote = "" Then quote = txtquote

If txtquote = "" Then .Cells(r, 11).Value = blank Else .Cells(r, 11).Value = quote

```

Update Check Deposits and Followup on Open Items

The left screenshot shows the 'Update Deposits' form. It has a title bar 'Update Deposits' and a close button. Inside, there's a section 'Outstanding Checks' with a list box containing two items: '\$120.00 Jones, Blake & Nellie' and '\$150.00 Jones, Gary & Sue'. To the right of the list box is a 'Deposited' checkbox. At the bottom right is a 'Done' button.

The right screenshot shows the 'Follow Up Activities' form. It has a title bar 'Follow Up Activities' and a close button. Inside, there's a section 'Follow up activities' with a list box containing four items: '12/7/2010 Jones, Blake & Nellie Remind of upcoming job', '11/29/2010 Smith, Justin & Jess Call him today', '12/8/2010 Rugby, Brennan & Melissa Invite to Christmas Dinner', and '12/22/2010 Ulfenquist, Kevin & Janice Visit for Christmas'. To the right of the list box is a 'Completed' checkbox. At the bottom right is a 'Done' button.

The Update Check Deposits form, "frmUpdateDeposits", populates a list showing all the checks that have been received but not deposited. The list is updated by clicking selecting the checks that have been deposited and clicking the "deposited" button. Writing the VBA was made easier by the combo box approach used in early forms. By creating consistency, it allowed me to use a simple If cell = "No" function

```

For r = 2 To finalrow
    If Cells(r, 16) = "Check" Then
        If Cells(r, 17) = "No" Then
            lstname.AddItem "$" & Cells(r, 15).Value & ".00" & Cells(r, 1).Value
        End If
    End If
Next

```

The process was similar for the Follow-up on Open Items form. I used a For loop and IF function to search for all customers that had follow-up items. When the Item is completed the form deletes it from the list and the database.

```
'-1 is the value of listindex when nothing is selected
If lstitem.ListIndex = -1 Then
    MsgBox "You must select which items you've completed"
Else
    x = Mid(lstitem.Text, 16, 8)

    For r = 2 To finalrow
        If InStr(1, Cells(r, 1).Value, x) > 0 Then
            Sheets("Customer Database").Cells(r, 23).Value = "-----"
        Exit For
    End If
    Next
    lstitem.RemoveItem lstitem.ListIndex
End If
```

Schedule New Jobs / Update Customer Information

This form gives the user a chance to enter in a newly scheduled jobs or update customer information. It allows them to search through the database based on customer name. The search function is very similar to the one performed in class.

DESIGNING THE UPCOMING JOBS SHEET

When the information walkthrough is finished, the workbook runs the runMymacro Sub Procedure, which populates the "Upcoming Jobs" sheet and sends out the text messages. The upcoming jobs sheets has six elements: The current date, a revenue total for the year, a list of

[illegible]

The current date is a cell with the formula “=today()”. It automatically updates to the current date as soon as the workbook is opened. The revenue total is just sum of the payments received column from the database for the year.

The VBA for the list of outstanding checks is located under the Sub “pullBoxitems” in the “runMymacro” module. It’s fairly simple. I used one large For loop to populate both lists so VBA would only have to search through the database once. I used an *IF cells(r,17) = “No”* statement to find customers with outstanding checks, then copied and pasted their information to “Upcoming Jobs” sheet, row “x”, where x = 10 for the first loop and then x = x + 1 for each successive loop. I did the same thing for outstanding items, using the variable y, setting y =

```

Sub pullBoxItems()
Dim finalrow As Integer
Dim r As Integer
Dim x As Integer
Dim y As Integer

x = 10
y = 25
Sheets("Customer Database").Select
Cells(2, 9).Select
ActiveCell.End(xlDown).Select
finalrow = ActiveCell.Row

For r = 2 To finalrow
    'pull checks
    If Sheets("Customer Database").Cells(r, 17).Value = "No" Then
        Cells(r, 1).Select
        Selection.Copy
        Sheets("UpcomingJobs").Select
        Cells(x, 5).Select
        ActiveSheet.Paste
        Sheets("Customer Database").Select
        Cells(r, 15).Select
        Selection.Copy
        Sheets("UpcomingJobs").Select
        Cells(x, 9).Select
        ActiveSheet.Paste
        x = x + 1
    End If

    'Pull activities
    If Not Sheets("Customer Database").Cells(r, 23).Value = "-----" Then
        If Not Sheets("Customer Database").Cells(r, 23).Value = "" Then
            'name
            Sheets("Customer Database").Select
            Cells(r, 1).Select
            Selection.Copy
            Sheets("UpcomingJobs").Select
            Cells(y, 5).Select
            ActiveSheet.Paste
            'activity
            Sheets("Customer Database").Select
            Cells(r, 23).Select
            Selection.Copy
            Sheets("UpcomingJobs").Select
            Cells(y, 8).Select
            ActiveSheet.Paste
            'date
            Sheets("Customer Database").Select
            Cells(r, 24).Select
            Selection.Copy
            Sheets("UpcomingJobs").Select
            Cells(y, 13).Select
            ActiveSheet.Paste

            If Cells(y, 13) < Cells(4, 8) Then Cells(y, 13).Font.Color = -16776961 _
            Else Cells(y, 13).Font.ColorIndex = xlAutomatic
            y = y + 1
        End If
    End If
Next

```


25 for the first loop and then $y = y + 1$ for each successive loop. VBA also highlights past due outstanding items in red.

Populating Weather Forecast

The program populates the calendar with a 10-day weather forecast from weatherchannel.com.

Originally, I tried using the Macro recorder to write the VBA, but it was too time consuming to copy and paste so many times. I simplified it by breaking the procedure into two parts. I used the recorder to pull

<pre>Sub pasteToday() Cells.Find(What:=range("Date"), After:=ActiveCell, LookIn:=xlFormulas, LookAt _ :=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _ False, SearchFormat:=False).Activate Selection.Copy Sheets("UpcomingJobs").Select range("Q7").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(-1, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(0, -1).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(3, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(5, 0).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(3, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(-3, 1).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select</pre>	<pre>Sub findNextday() 'startObjectCOP Dim r As Integer For r = 1 To 9 Sheets("WeatherSheet").Select Cells.Find(What:=Date + r, After:=ActiveCell, LookIn:=xlFormulas, _ LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _ MatchCase:=False, SearchFormat:=False).Activate Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(-4, 2).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(-1, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(0, -1).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(3, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy Sheets("UpcomingJobs").Select ActiveCell.Offset(5, 0).range("A1").Select ActiveSheet.Paste Sheets("WeatherSheet").Select ActiveCell.Offset(3, 0).range("A1").Select Application.CutCopyMode = False Selection.Copy</pre>
---	---

the table, and wrote the code to copy and paste myself. I broke the copy and paste code into two parts. The first part, sub PasteToday (), Inserts the “current date” value into a find function, selects and copies the date, and pastes it back on the “UpcomingJobs” sheet at absolute reference Cells(7,17). It then uses relative references to fill in the rest of the weather information for that day. Rather than repeat the PasteToday () code ten times, I used a loop to fill in days 2 through 10, using all relative references.

I downloaded the pictures from weatherchannel.com and saved them to the same folder as the workbook. I gave each picture the same name that appears for its description online and in the calendar. For example, the picture for “sunny” weather has the address “C:\Users\Blake\Documents\Accounting 4\VBA\VBA Project\sunny.gif”. I used the function “ThisWorkbook.path” in my loop and saved the pictures to the same folder as the workbook. This allowed me to move the workbook without having to change the code.

```
Sub insertPictures()  
  
Dim weather As String  
Dim c As Integer  
Dim address As String  
  
Sheets("UpcomingJobs").Select  
For c = 16 To 43 Step 3  
  
weather = Cells(12, c)  
  
If Cells(12, c) = "Rain / Snow Showers" Then weather = "Rain or Snow Showers"  
If Cells(12, c) = "Rain / Snow" Then weather = "Rain or Snow Showers"  
  
ActiveSheet.Pictures.Insert (ThisWorkbook.path & "\" & weather & ".gif")  
With ActiveSheet.Shapes(ActiveSheet.Shapes.Count)  
.LockAspectRatio = msoFalse  
.Height = 50  
.Width = 98  
.Left = Cells(8, c).Left  
.Top = Cells(8, c).Top  
End With  
  
Next  
  
End Sub
```




Clearing old pictures was a challenge. I decided it was simplest to create a do-loop that cleared all shapes from the worksheet until the shape count equaled zero, however this method also deleted my buttons. This is there are no buttons to easily access the user forms from the worksheet and is one of the short comings of the project. I can solve the problem by inserting the buttons onto a ribbon later.

```
'clear pictures
Do Until ActiveSheet.Shapes.Count = 0
ActiveSheet.Shapes(ActiveSheet.Shapes.Count).Select
Selection.Delete
Loop
```

(Sometimes the code freezes because it finds a weather condition that I haven't downloaded a picture for. When that happens I go online, download the new picture, and save it under the proper name. Hopefully soon I'll have all possible weather conditions accounted for.)

Populating Upcoming Jobs

Populating the table with upcoming jobs uses a double For loop, If statements, and the In string function. The difficult part of pulling in job information is that some days have multiple jobs. To solve the problem, I created variable named "extraJob" and set it initially equal to one. If extraJob equals one and the loop finds a job, it populates the first part of the table and sets extraJob equal to two. If extraJob equals two, it populates the bottom part, and sets extraJob equal to three. We generally only have time for two jobs in one day, so If a third job is found, a message box stating that you have too many jobs for one day is displayed.

Tonight	T-Dec	Wed	8-Dec	Thu	9-Dec	F
	High N/A COP 10%		High 42° COP 20%		High 41° COP 20%	
Partly Cloudy		Mostly Cloudy		Mostly Cloudy		
Liljenquist, Kevin & Janice		Jones, Blake & Nellie		Rigby, Breanna & Melissa		
Time: 10:00		Time: 10:00		Time: 10:00		
117 E Magnolia		627 N 600 W Apt 10		27 Parley's Way		
Oviedo FL 32765		Provo UT 84601		Salt Lake UT 84201		
Regular 30	Hard to Reach 2	Window Wells 2	Regular 8	Hard to Reach 0	Window Wells 0	
Quote ECT 1.82	###	Quote ECT 0.67	\$ 30.00	Quote ECT 1.43	###	
Time:		Time:		Time:		
				Rigby, Breanna & Melissa		
				Time: 0.41667		
				27 Parley's Way		
				Salt Lake UT 84201		
Regular 30	Hard to Reach 2	Window Wells 2	Regular 8	Hard to Reach 0	Window Wells 0	
Quote ECT 1.82	###	Quote ECT 0.67	\$ 30.00	Quote ECT 1.43	###	

Formatting the Table

I formatted the "Upcoming Jobs" sheet and the Customer Database using Macro Recorder, and set macro to run at the end of the sub procedure. Also, I insert a formula to calculate the estimated completion time for the job based on the number of regular panes, hard to reach panes, window wells, and whether it's outside, inside, or both.

```
Sub InsertFormula()
Dim r As Integer
Dim finalrow As Integer
Dim Time1 As Double
Dim Time2 As Double

Sheets("Customer Database").Select
Cells(2, 1).Select
ActiveCell.End(xlDown).Select
finalrow = ActiveCell.Row

For r = 2 To finalrow

If Not Cells(r, 19) = "-----" And Not Cells(r, 20) = "-----" And Not Cells(r, 21) = "-----" _
And Not Cells(r, 22) = "-----" Then

Time1 = (Cells(r, 19) * 2.5 + Cells(r, 20) * 2 + Cells(r, 21) * 5 + 20) / 60
Time2 = (Cells(r, 19) * 1.25 + Cells(r, 20) * 2 + Cells(r, 21) * 5 + 20) / 60

If Cells(r, 22) = "In & Out" Then Cells(r, 13) = Time1 Else Cells(r, 13) = Time2

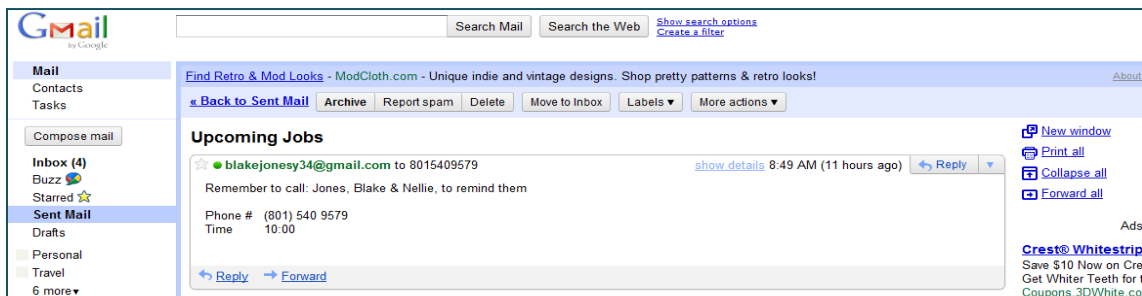
End If

Next

End Sub
```

SENDING THE REMINDER

At the end of the project, VBA will send a text message to my phone reminding me to call our jobs for the next day. The text includes the name, time, and phone number for tomorrow's jobs. I used the class example as a foundation to automate sending text through email. Writing the message was difficult because I didn't include the customers phone number as part of the information on the calendar. This means I had to pull the name from the next day on the calendar, find that name in the database, and then find the corresponding phone number. To simplify the code I used a Sub function to find and return the number text, inside a Sub function designed to build return the text of the message, inside a Sub Procedure that emails the message. Sub functions are an aspect of VBA I've struggled with, and this was a great chance for me gain a better understanding of them.



```
Function message() As String
    Sheets("UpcomingJobs").Select
    If Cells(14, 19).Value = "" Then message = "You have no jobs tomorrow, go golfing" _
    Else If Cells(26, 19).Value = "" Then _
        message = "Remember to call: " & Cells(14, 19).Value & ", to remind them" & vbNewLine & _
            "" & vbNewLine & _
            "Phone # " & FindNumber1 & vbNewLine & _
            "Time " & Sheets("UpcomingJobs").Cells(15, 20).Text _
    Else _
        message = "Remember to call: " & Sheets("UpcomingJobs").Cells(14, 19).Value & ", to remind them" & vbNewLine & _
            "" & vbNewLine & _
            "Phone # " & FindNumber1 & vbNewLine & _
            "Time " & Sheets("UpcomingJobs").Cells(27, 20).Text & vbNewLine & _
            vbNewLine & _
            "Remember to call: " & Sheets("UpcomingJobs").Cells(26, 19).Value & ", to remind them" & vbNewLine & _
            "" & vbNewLine & _
            "Phone # " & FindNumber2 & vbNewLine & _
            "Time " & Sheets("UpcomingJobs").Cells(15, 20).Text _
    End Function

Sub sendReminder()
    Dim subject As String
    Dim d As Integer

    subject = "Upcoming Jobs"

    If sendGMail("blakejonesy34", "baseball85", _
        message, "8015409579@txt.att.net", subject) Then d = 1
End Sub
```

```
Function FindNumber1() As String
    Dim finalrow As Integer
    Dim r As Integer
    Dim Calander1 As String
    Dim Database1 As String

    Calander1 = Sheets("UpcomingJobs").Cells(14, 19).Value

    Sheets("Customer Database").Select
    Cells(2, 1).Select
    ActiveCell.End(xlDown).Select
    finalrow = ActiveCell.Row

    For r = 2 To finalrow
        Database1 = Sheets("Customer Database").Cells(r, 1).Value
        If InStr(1, Database1, Calander1) > 0 Then
            FindNumber1 = Cells(r, 6).Value
            Exit For
        End If
    Next
End Function
```

APPENDIX A – UPCOMING JOBS CALANDER

