Weston Whitaker
MBA 614 – VBA
13 April 2010

Sales Agent Scorecard Generation for Integra Telecom

Executive Summary:

Situation:

Integra Telecom, a telecommunications service provider, has a Sales Agent Scorecard that they generate every month from the data in their Microsoft Access database. The Scorecard itself is a Microsoft Excel file that is formatted in a very particular way – a way that can't automatically be done from Access. Thus, every month, a long, tedious process is undertaken to export the data and format it properly.

Solution:

This project imbedded the automation tools required to generate that Excel report (with the proper formatting) using a button located *within* the database. The user only needs to select the month the report needs to be generated for, and the location of the final Excel file. The code takes care of all the following actions:

- Exporting
- Formatting
- Rearranging
- File-naming
- Saving
- Clean-up

Integra Telecom:

Integra Telecom is a Portland, Oregon-based telecommunications service provider. They provide voice services, internet services, communications equipment, and networking solutions throughout the Mountain West. Their focus is on providing responsive, reliable, local service. According to the Portland Business Journal, Integra Telecom has ranking in the Top 100 Fastest-Growing Privately-Held Companies in 9 out of the last 10 years.

Problem Definition:

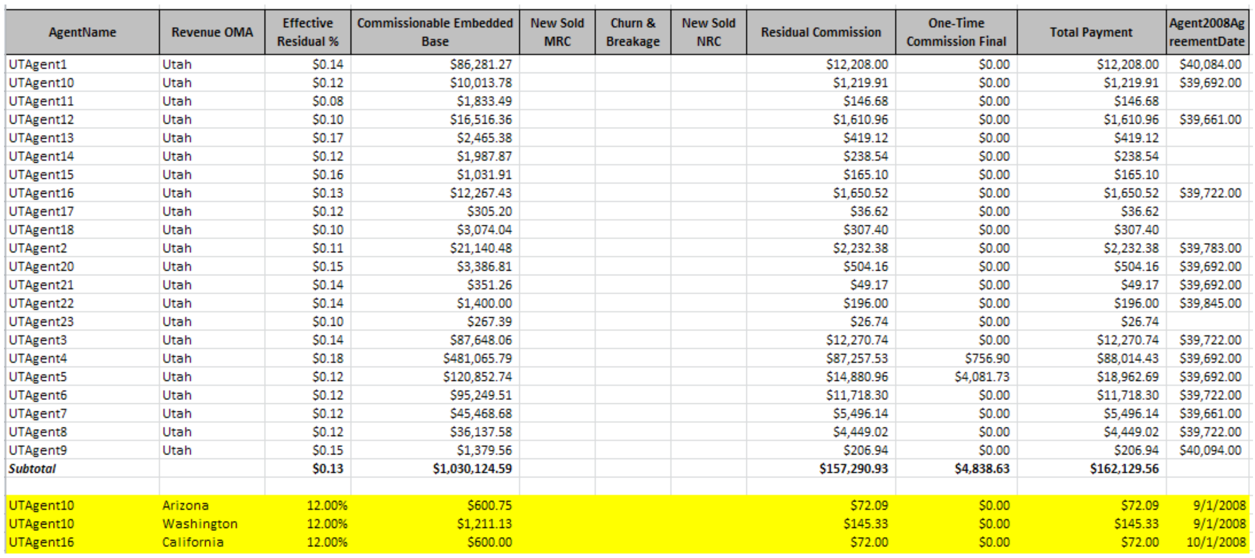
Background:

This project is in support of work being done by Scott Carlson, Business Analyst for Integra Telecom's Salt Lake City office. One of Scott's responsibilities is maintaining the "Agent Commissions" database. This is a Microsoft Access database that stores all the sales-agent commission data for the entire company. A "dummy" database has been provided for the purposes of this project – all names and numbers are fictional.

Desired Output:

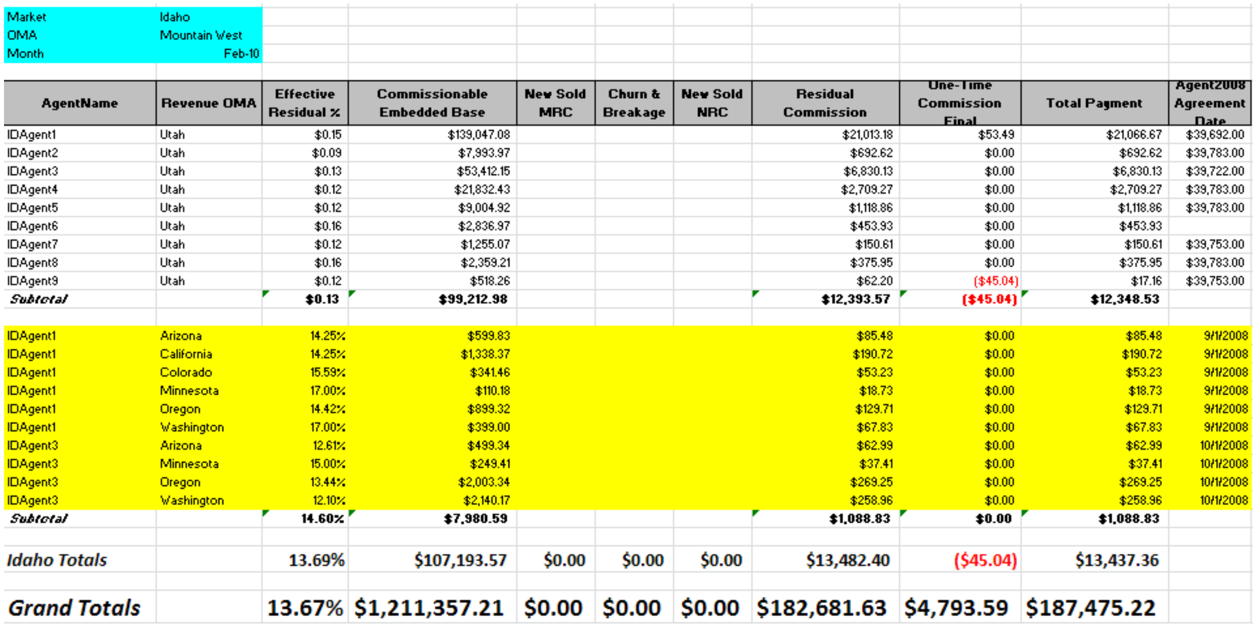
Previously, maintaining the database required a large number of manual edits, calculations, and program manipulations. Importing data, error checking, report generation, and query exports were all previously tedious, repetitive, manual operations. Moreover, the maintenance work typically had to wait until all the data had been gathered, and then must be completed a few days later for corporate-mandated reports. Thus, maintaining this database became a monthly, late-night ritual – and a prime suspect for VBA automation.

Currently, much functionality has been added by Scott Carlson. The portion of the automation dedicated to this project has to do with exporting a monthly Microsoft Excel document called the "Agent Scorecard". The Agent Scorecard is a query-based output showing the performance of all sales agents grouped by geographic regions. Figure 1 shows an example of how this Excel output document appears. Notice that the first group (unhighlighted) is a collection of Utah-based sales agent sales into Utah-based customers. They are sorted alphabetically by sales agent and subtotaled. The next section (highlighted) is similarly organized, only this section describes the Utah-based sales agent sales into non-Utah customers. This report also contains similar sections for Idaho-based sales agents and their sales into Utah-based and non-Utah-based customers. There are subtotals for each section, plus sub-totals for Utah sales agent sales and Idaho sales agent sales. There is also a grand total of all sales for all sales agents (see Figure 2).



AgentName	Revenue OMA	Effective Residual %	Commissionable Embedded Base	New Sold MRC	Churn & Breakage	New Sold NRC	Residual Commission	One-Time Commission Final	Total Payment	Agent2008AgreementDate
UTAgent1	Utah	$0.14	$86,281.27				$12,208.00	$0.00	$12,208.00	$40,084.00
UTAgent10	Utah	$0.12	$10,013.78				$1,219.91	$0.00	$1,219.91	$39,692.00
UTAgent11	Utah	$0.08	$1,833.49				$146.68	$0.00	$146.68	
UTAgent12	Utah	$0.10	$16,516.36				$1,610.96	$0.00	$1,610.96	$39,661.00
UTAgent13	Utah	$0.17	$2,465.38				$419.12	$0.00	$419.12	
UTAgent14	Utah	$0.12	$1,987.87				$238.54	$0.00	$238.54	
UTAgent15	Utah	$0.16	$1,031.91				$165.10	$0.00	$165.10	
UTAgent16	Utah	$0.13	$12,267.43				$1,650.52	$0.00	$1,650.52	$39,722.00
UTAgent17	Utah	$0.12	$305.20				$36.62	$0.00	$36.62	
UTAgent18	Utah	$0.10	$3,074.04				$307.40	$0.00	$307.40	
UTAgent2	Utah	$0.11	$21,140.48				$2,232.38	$0.00	$2,232.38	$39,783.00
UTAgent20	Utah	$0.15	$3,386.81				$504.16	$0.00	$504.16	$39,692.00
UTAgent21	Utah	$0.14	$351.26				$49.17	$0.00	$49.17	$39,692.00
UTAgent22	Utah	$0.14	$1,400.00				$196.00	$0.00	$196.00	$39,845.00
UTAgent23	Utah	$0.10	$267.39				$26.74	$0.00	$26.74	
UTAgent3	Utah	$0.14	$87,648.06				$12,270.74	$0.00	$12,270.74	$39,722.00
UTAgent4	Utah	$0.18	$481,065.79				$87,257.53	$756.90	$88,014.43	$39,692.00
UTAgent5	Utah	$0.12	$120,852.74				$14,880.96	$4,081.73	$18,962.69	$39,692.00
UTAgent6	Utah	$0.12	$95,249.51				$11,718.30	$0.00	$11,718.30	$39,722.00
UTAgent7	Utah	$0.12	$45,468.68				$5,496.14	$0.00	$5,496.14	$39,661.00
UTAgent8	Utah	$0.12	$36,137.58				$4,449.02	$0.00	$4,449.02	$39,722.00
UTAgent9	Utah	$0.15	$1,379.56				$206.94	$0.00	$206.94	$40,094.00
Subtotal		**$0.13**	**$1,030,124.59**				**$157,290.93**	**$4,838.63**	**$162,129.56**	
UTAgent10	Arizona	12.00%	$600.75				$72.09	$0.00	$72.09	9/1/2008
UTAgent10	Washington	12.00%	$1,211.13				$145.33	$0.00	$145.33	9/1/2008
UTAgent16	California	12.00%	$600.00				$72.00	$0.00	$72.00	10/1/2008

Figure 1: Example Excel Output Form of the Agent Scorecard



Market	Idaho
OMA	Mountain West
Month	Feb-10

AgentName	Revenue OMA	Effective Residual %	Commissionable Embedded Base	New Sold MRC	Churn & Breakage	New Sold NRC	Residual Commission	One-Time Commission Final	Total Payment	Agent2008 Agreement Date
IDAgent1	Utah	$0.15	$139,047.08				$21,013.18	$53.49	$21,066.67	$39,692.00
IDAgent2	Utah	$0.09	$7,993.97				$692.62	$0.00	$692.62	$39,783.00
IDAgent3	Utah	$0.13	$53,412.15				$6,830.13	$0.00	$6,830.13	$39,722.00
IDAgent4	Utah	$0.12	$21,832.43				$2,709.27	$0.00	$2,709.27	$39,783.00
IDAgent5	Utah	$0.12	$9,004.92				$1,118.86	$0.00	$1,118.86	$39,783.00
IDAgent6	Utah	$0.16	$2,836.97				$453.93	$0.00	$453.93	
IDAgent7	Utah	$0.12	$1,255.07				$150.61	$0.00	$150.61	$39,753.00
IDAgent8	Utah	$0.16	$2,359.21				$375.95	$0.00	$375.95	$39,783.00
IDAgent9	Utah	$0.12	$518.26				$62.20	($45.04)	$17.16	$39,753.00
Subtotal		**$0.13**	**$99,212.98**				**$12,393.57**	**($45.04)**	**$12,348.53**	
IDAgent1	Arizona	14.25%	$599.83				$85.48	$0.00	$85.48	9/1/2008
IDAgent1	California	14.25%	$1,338.37				$190.72	$0.00	$190.72	9/1/2008
IDAgent1	Colorado	15.59%	$341.46				$53.23	$0.00	$53.23	9/1/2008
IDAgent1	Minnesota	17.00%	$110.18				$18.73	$0.00	$18.73	9/1/2008
IDAgent1	Oregon	14.42%	$899.32				$129.71	$0.00	$129.71	9/1/2008
IDAgent1	Washington	17.00%	$399.00				$67.83	$0.00	$67.83	9/1/2008
IDAgent3	Arizona	12.61%	$499.34				$62.99	$0.00	$62.99	10/1/2008
IDAgent3	Minnesota	15.00%	$249.41				$37.41	$0.00	$37.41	10/1/2008
IDAgent3	Oregon	13.44%	$2,003.34				$269.25	$0.00	$269.25	10/1/2008
IDAgent3	Washington	12.10%	$2,140.17				$258.96	$0.00	$258.96	10/1/2008
Subtotal		**14.60%**	**$7,980.59**				**$1,088.83**	**$0.00**	**$1,088.83**	
Idaho Totals		13.69%	$107,193.57	$0.00	$0.00	$0.00	$13,482.40	($45.04)	$13,437.36	
Grand Totals		**13.67%**	**$1,211,357.21**	**$0.00**	**$0.00**	**$0.00**	**$182,681.63**	**$4,793.59**	**$187,475.22**	

Figure 2: Example Excel Output Form of the Agent Scorecard - showing sub-totals and grand-totals

Starting Point:

The dummy database provided already had a specified user-interface for the Agent Scorecard Export functionality. The goal is to create the Excel Spreadsheet output from the Microsoft Access database by pressing the designated button on the existing user-form.

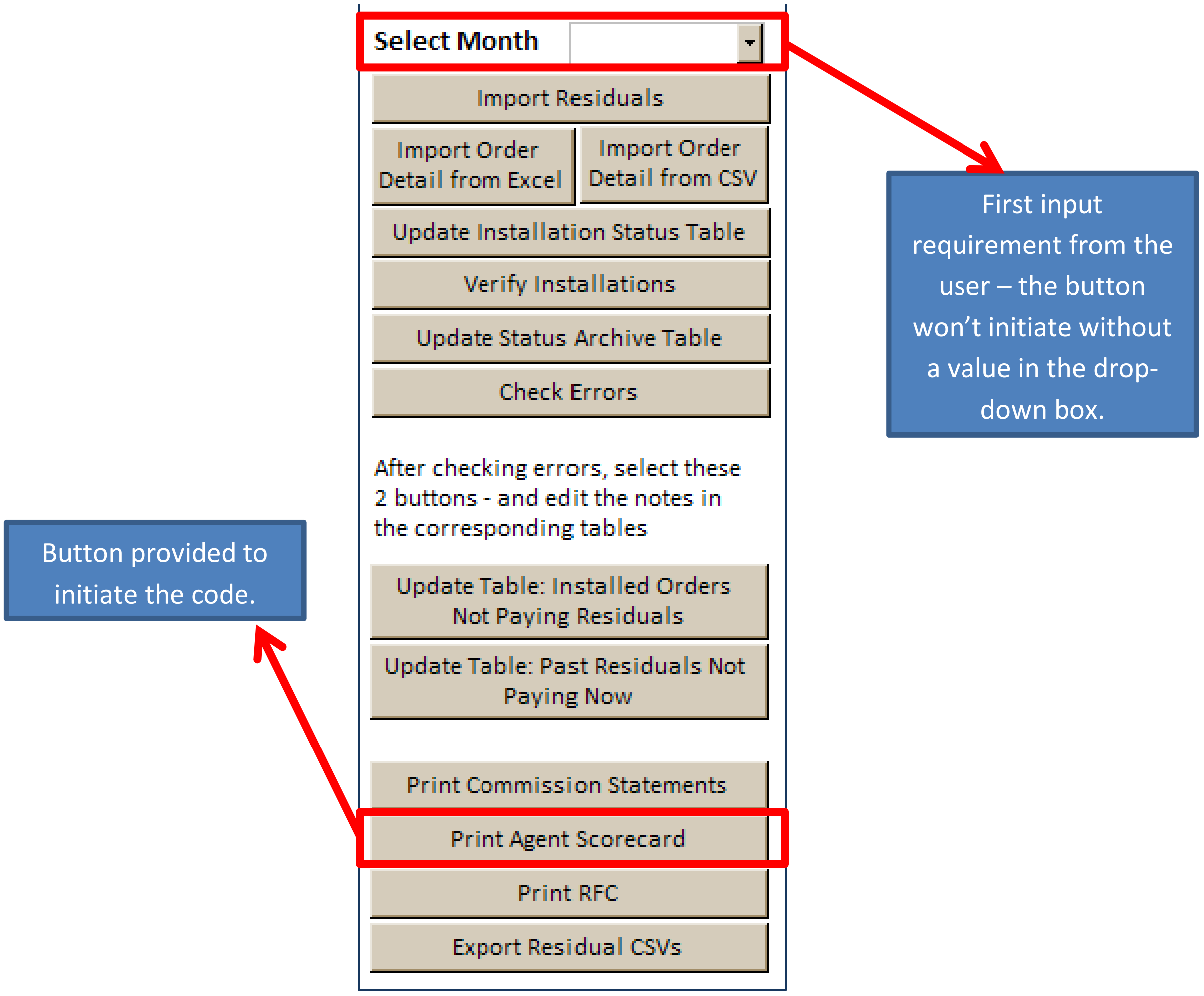

Select Month
Import Residuals
Import Order Detail from Excel
Import Order Detail from CSV
Update Installation Status Table
Verify Installations
Update Status Archive Table
Check Errors
After checking errors, select these 2 buttons - and edit the notes in the corresponding tables
Update Table: Installed Orders Not Paying Residuals
Update Table: Past Residuals Not Paying Now
Print Commission Statements
Print Agent Scorecard
Print RFC
Export Residual CSVs
First input requirement from the user – the button won't initiate without a value in the drop-down box.
Button provided to initiate the code.

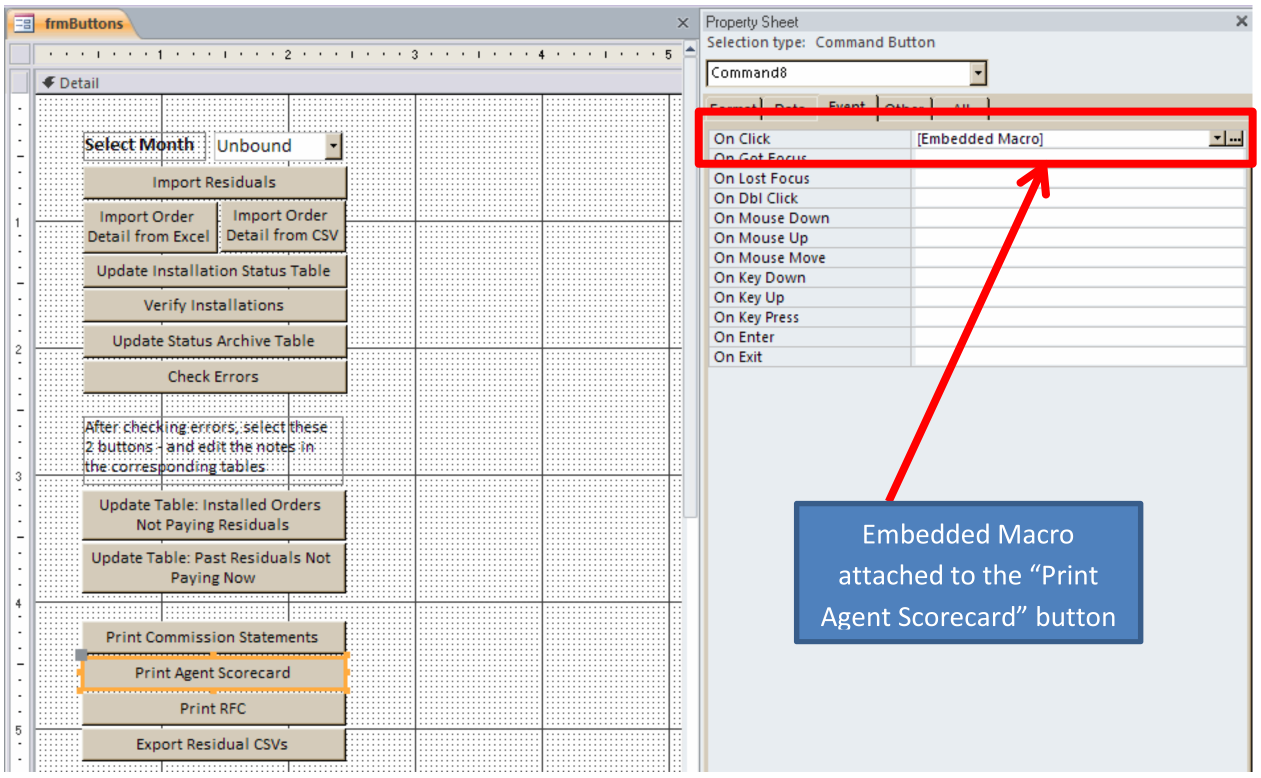
Figure 3: Existing Access Database User-form - includes "Print Agent Scorecard" button designated for this project

The agent scorecard data was already collected in a query in the database labeled "qryAgentScorecard2". The "Print Agent Scorecard" button was provided with the functionality to open the agent scorecard query in table format. The button also required a month selection from the "Select Month" combo box before opening the query (since the query was based on the selected month).

Solution:

The "Submacro":

The first step in developing the code for this project was to explore the built-in functionality of Microsoft Access. In the design view of the provided user form, the property sheet for the "Print Agent Scorecard" button shows a link to a place where a macro can be "embedded" into the button on the form (see Figure 4).


frmButtons
Detail
Select Month
Unbound
Import Residuals
Import Order Detail from Excel
Import Order Detail from CSV
Update Installation Status Table
Verify Installations
Update Status Archive Table
Check Errors
After checking errors, select these 2 buttons - and edit the notes in the corresponding tables
Update Table: Installed Orders Not Paying Residuals
Update Table: Past Residuals Not Paying Now
Print Commission Statements
Print Agent Scorecard
Print RFC
Export Residual CSVs
Property Sheet
Selection type: Command Button
Command8
Format
Data
Event
Other
All
Embedded Macro attached to the "Print Agent Scorecard" button

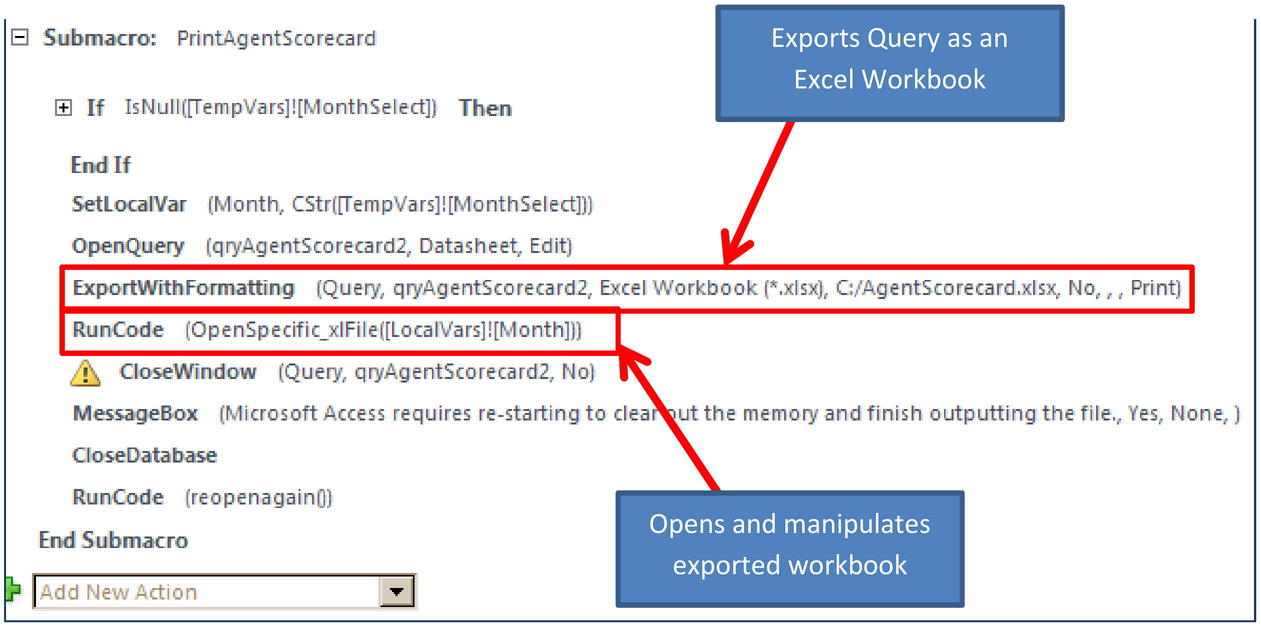
On Click	[Embedded Macro]
On Got Focus	
On Lost Focus	
On Dbl Click	
On Mouse Down	
On Mouse Up	
On Mouse Move	
On Key Down	
On Key Up	
On Key Press	
On Enter	
On Exit	

Figure 4: Design View of the user form - with the "Print Agent Scorecard" button highlighted - notice the Embedded Macro shown on the Property Sheet

The embedded macros in Access have quite a bit of functionality. It is fairly easy to export a query to a spreadsheet, but it is impossible to manipulate that spreadsheet. Thus, it became necessary to write additional code to manipulate the exported query data into the format desired. Figure 5 shows a copy of the Access SubMacro interface with some of the key functionality. An important feature of the “subMacro” is the “Run Code” option. This is where code written in a module using the VBA editor can be called.

Manipulating Excel from Access:

One of the biggest challenges in writing the code to manipulate Excel was “translating” it back to the Access program. It was necessary to enable the “Microsoft Excel 14.0 Object Library” Reference. The important code elements in manipulating Excel from Access are setting the correct objects, and then running the code with those objects.



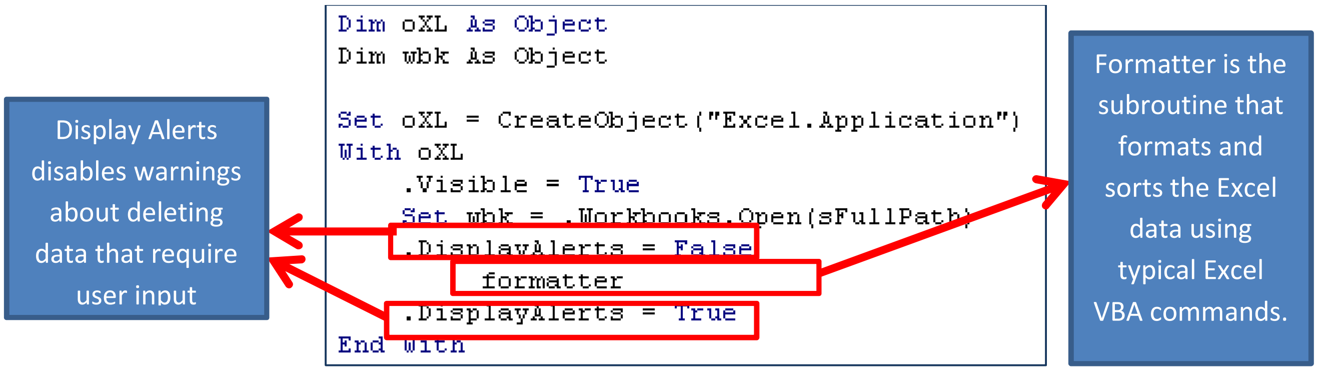
```
Submacro:   PrintAgentScorecard

    If   IsNull([TempVars]![MonthSelect])   Then

    End If
    SetLocalVar   (Month, CStr([TempVars]![MonthSelect]))
    OpenQuery   (qryAgentScorecard2, Datasheet, Edit)
    ExportWithFormatting   (Query, qryAgentScorecard2, Excel Workbook (*.xlsx), C:/AgentScorecard.xlsx, No, , , Print)
    RunCode   (OpenSpecific_xlFile([LocalVars]![Month]))
    CloseWindow   (Query, qryAgentScorecard2, No)
    MessageBox   (Microsoft Access requires re-starting to clear out the memory and finish outputting the file., Yes, None, )
    CloseDatabase
    RunCode   (reopenagain())
End Submacro
```

Exports Query as an Excel Workbook
Opens and manipulates exported workbook
Add New Action

Figure 5: The Microsoft Access SubMacro Interface - based on drop down boxes to select code functionality, not typing code to create specific functionality



```
Dim oXL As Object
Dim wbk As Object

Set oXL = CreateObject("Excel.Application")
With oXL
    .Visible = True
    Set wbk = .Workbooks.Open(sFullPath)
    .DisplayAlerts = False
        formatter
    .DisplayAlerts = True
End With
```

Display Alerts disables warnings about deleting data that require user input
Formatter is the subroutine that formats and sorts the Excel data using typical Excel VBA commands.

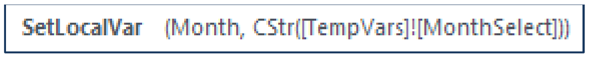
Figure 6: Code allowing for the manipulation of Excel from Access modules

The specific formatting code was found by recording a macro of a user formatting Excel in the desired way. Due to the potential for differing numbers of sales entries (because the month could change), multiple "place-holder" variables were utilized – allowing for easy, dynamic defining of the beginning and end of output sections. This streamlined the alphabetizing of the entries and the insertion of the correct formula references.

Challenges:

Temporary Variables

The database provided for this project uses a temporary variable to identify the month selected for filtering the query data. Temporary variables are difficult to deal with due to the ambiguous nature of their scope. Much of the formatting required in the Excel Spreadsheet (including naming the file) required the use of the selected month. Defining a new non-temporary variable in the submacro ended up being a suitable way to pass the temporary variable values into the module-level code (see Figure 7). Notice that a type-conversion was required to get the information in the right format to use in the module-level code.

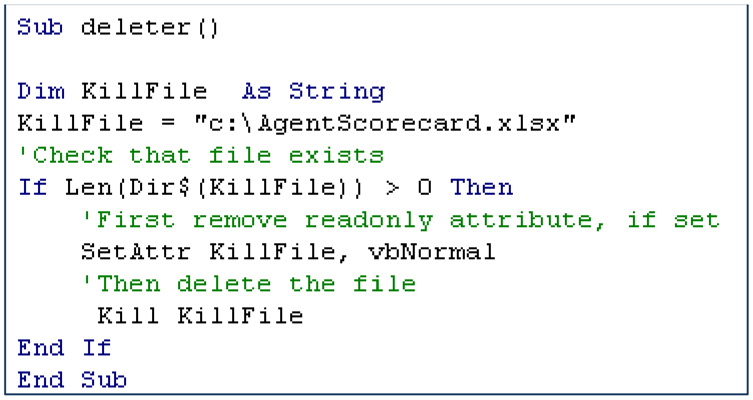


```
SetLocalVar    (Month, CStr([TempVars]![MonthSelect]))
```

Figure 7: Creating a non-temporary variable that can pass information to module-level code

Clean-up:

One of the unique challenges of this project was the need to clean the code up after use. The Excel file Access first exported was located in a different place and named differently than the final output file. Thus, it became necessary to delete an Excel file from Access VBA code. This was done as follows:



```
Sub deleter()

Dim KillFile  As String
KillFile = "c:\AgentScorecard.xlsx"
'Check that file exists
If Len(Dir$(KillFile)) > 0 Then
    'First remove readonly attribute, if set
    SetAttr KillFile, vbNormal
    'Then delete the file
     Kill KillFile
End If
End Sub
```

Figure 8: Kill-file code - deletes the unnecessary files created during the processing of this code