1.1 Executive Summary

Eastside Harley-Davidson in Bellevue, WA is the premier Harley-Davidson motorcycle dealership on the west coast. They are engaged in the sale of new and pre-owned motorcycles, parts, MotorClothes, rentals, and provide repair and maintenance service for their customer's bikes. The company compiles a daily business report from which managers base many decisions. The report is compiled from several different sources on a daily basis and often consumes 30 minutes, if not more, of an employee's time to assemble and compute the data. The sources of the daily transaction information include an Excel file which is received through e-mail, an Excel file which is manually pulled out of their in-house database, and several other figures which must be entered by hand. This VBA project minimizes, to the full extent possible, the number of steps, clicks, keystrokes, and time an employee must use to collect and analyze the information and then e-mail it to the managers group. This project also allows the user, at the beginning of each month, to reset the report with blank data as well as update the dates and goal information for each department.

1.2 Implementation Documentation

The reasoning

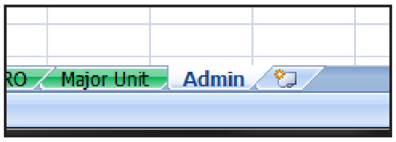
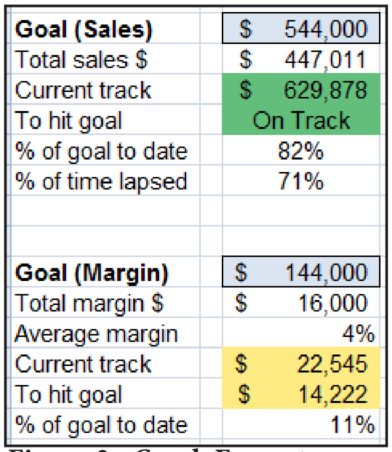


Figure 1 - Admin Tab

The first step was to simplify the UI for the end-user to make it possible for this report to be run by someone with little or no Excel experience. Consequently, this ruled out being able to bury the macros in the Developer > Macros (alt + F8) menu. To organize the chaos into a simple menu, an "Admin" sheet (see figure 1) was created at the end of the worksheets with three simple buttons to perform various functions.



Goal (Sales)	$ 544,000
Total sales $	$ 447,011
Current track	$ 629,878
To hit goal	On Track
% of goal to date	82%
% of time lapsed	71%
Goal (Margin)	$ 144,000
Total margin $	$ 16,000
Average margin	4%
Current track	$ 22,545
To hit goal	$ 14,222
% of goal to date	11%

Figure 3 - Cond. Format

The original daily business report consisted of three Excel files that were not linked. One file was the primary data for the overall store and each seperate department that was sent to all of the managers. The other two files were further breakdowns of the Parts and MotorClothes departments which were e-mailed to their respective managers. I initially considered building an Excel file that would strictly be code and would select and populate the three target Excel files with information. This proved to be conceptually much too complex, unnecessary, and a redundant use of multiple Excel files. So the first step was to consolidate, and rebuild from the ground up, all of the sheets and information from the previous three individual Excel files into a single file (see figure 2, below). Some functionality at the front-end needed to be maintained so that the user could update cells with information. Cells were colored blue that were intended to be manipulated by the user. Any other cell was not to be changed. Conditional formatting (see figure 3, right) of the "To hit track" and "To hit goal" cells of column "C" was added to enhance the overall readability of the report.

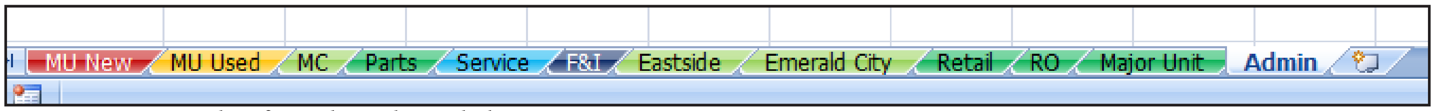
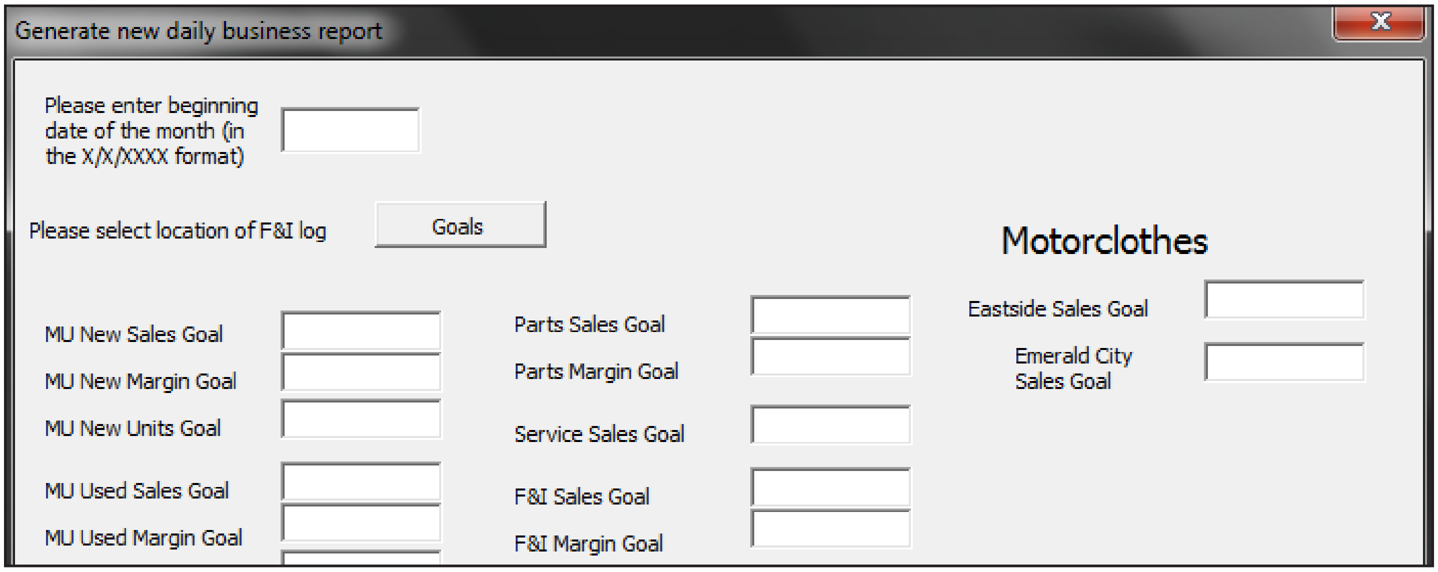


Figure 2 - Example of combined excel sheets



Generate new daily business report

Please enter beginning date of the month (in the X/X/XXXX format)

Please select location of F&I log

Motorclothes

MU New Sales Goal		Parts Sales Goal		Eastside Sales Goal	
MU New Margin Goal		Parts Margin Goal		Emerald City Sales Goal	
MU New Units Goal		Service Sales Goal			
MU Used Sales Goal		F&I Sales Goal			
MU Used Margin Goal		F&I Margin Goal			

Figure 5 - Generate new daily business report form

Generate a new report

Generating a new report for the month appeared to be the first function prudent to develop. By clicking the "Admin" sheet and then the "Generate new daily business report button, (see figure 4, right)" one could pull up a user form with which to create a new, properly formatted set of worksheets with the correct dates, goals, and blank data cells. The user form (see figure 5, above) consisted of 18 fields designated for specific store goals. These goals came from various sources, including but not limited to, verbal input and e-mail. Consequently, their entry could not be automated with VBA. Many of the main department goals could be automattically inputted into the form by clicking the "Goals" command button on the user form and navigating to the annual store goals Excel file. This allowed for the analyzation and extraction of the majority of the goals for the store, leaving only a few department-specific monthly goals for manual entry. It was also important to have the proper dates show up on the report, so a text box was created in which the first day of the month being reported was to be inputted. An invalid date would not be accepted and a message box would inform the user of such. Occassionally not all of the goals were known by the first day of the month so their values could be left blank. Finally, executing the form erased all previous entry data from the sheets, creating a new, blank daily business report.

Figure 4 - Button example

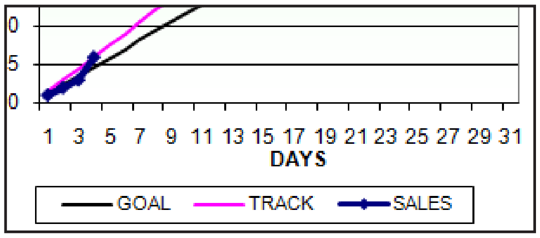
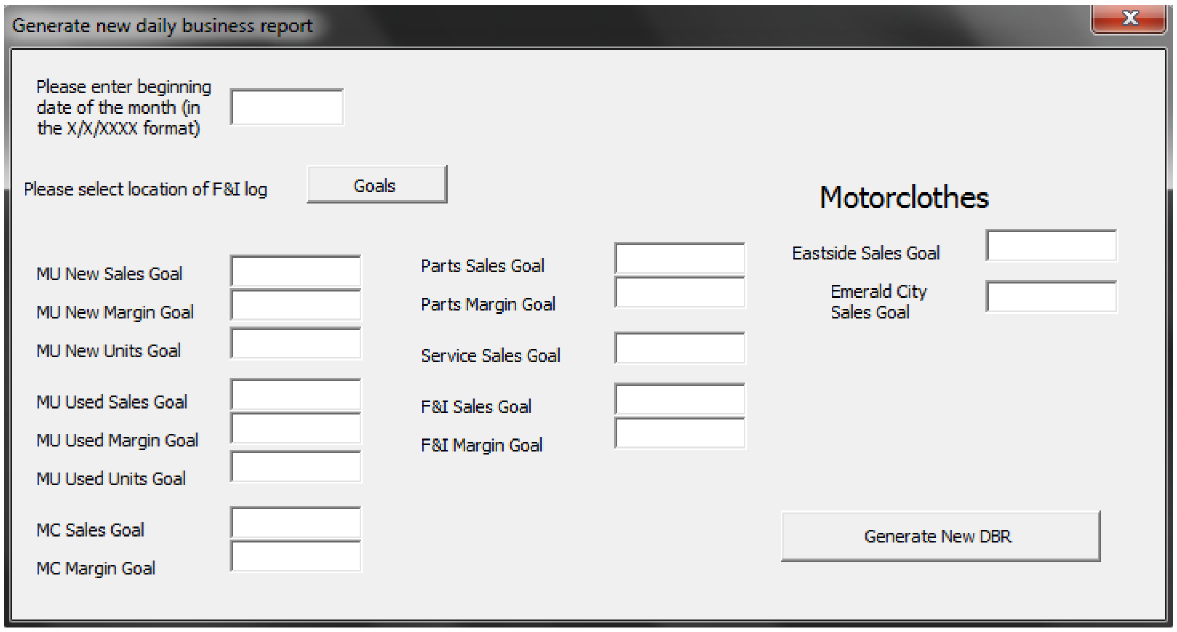


Figure 6 - Dynamic dates

There were several interesting functions that occurred as a result of generating a new report. First, the dates of each sheet extended to the correct number of days given the month of the the date initially entered. This happens across all sheets. It also sets the correct number of days to each of the 21 graphs and also updates data of each "goal" and "tracking" cell (see figure 6). The lines in the graphs represent the goal and also the general direction the department is headed for the month. Finally, the current track, daily sales to hit goal, % of goal to date, and % of time expired figures in column "A" of each sheet are also properly calculated off the number of days counted from the date originally entered into the "Generate new daily business report."



Generate new daily business report

Please enter beginning date of the month (in the X/X/XXXX format)

Please select location of F&I log

Motorclothes

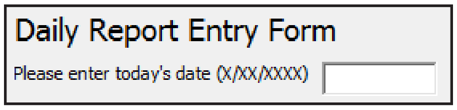
MU New Sales Goal	
MU New Margin Goal	
MU New Units Goal	
MU Used Sales Goal	
MU Used Margin Goal	
MU Used Units Goal	
MC Sales Goal	
MC Margin Goal	
Parts Sales Goal	
Parts Margin Goal	
Service Sales Goal	
F&I Sales Goal	
F&I Margin Goal	
Eastside Sales Goal	
Emerald City Sales Goal	

Figure 7 - DBR Entry Form

Filling in the details

Once a new, blank report was generated, the next step was creating a simple form to populate each of the fields on a daily basis (see figure 7, above). This consisted of a date for the current day the data was being entered, and

19 text boxes which would populate the respective cells across the worksheets. The first feature of the "Daily Report Entry Form" was the date entry (see figure 8). The date was validated conditionally upon whether or not it existed as a date in the month being reported. The form would inform the user if an invalid date had been submitted upon execution of the user form.



Daily Report Entry Form

Please enter today's date (X/XX/XXXX)

Figure 8 - DBR Form

The next logical step in the creation of the form was selection of the F&I log. The F&I log contained month-to-date data of bike sales and financing information. Clicking this button required a validated date in the date text box previously mentioned. Failure to do so resulted in a reminder to enter a date. Once done, clicking the F&I log opened a file selection dialogue that allowed the user to navigate to the F&I log, an Excel file that would have been received via e-mail at the end of the previous business day. Cells in the F&I log containing the previous days' bike sales, including new and used, as well as F&I information would have their cells filled with color for easy review if desired. The calculated figures would be automatically copied into the seven designated text boxes on the user form and the F&I log would close.

The next step was to then analyze the previous day's end-of-day report for more numbers. Processing this report did not require a date to be entered as it only contained one day's data exported as an Excel file from the in-house database. Just like the F&I log, it processed a selected file, highlighting cells from 22 specific categories in four categories (columns) for later review if warranted, placed the calculated figures into the proper parts department and other text boxes of the user form, and saved and closed the end-of-day report. The same procedure, with its own command button, applied to Emerald City, a small shop owned by the dealership.

The final step was manual entry of data into the remaining text boxes. This data came from disparate singular reports generated from the in-house database and therefore the process could not be automated. Execution of the form, or clicking "Make It So (see figure 9)," placed the data from each text box into their respective locations across the sheets. If a text box was left empty it was added to the report as a 0. Each graph and calculation across all sheets were dynamically updated with each entry.

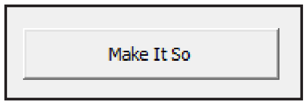


Figure 9 - Make It So

Sending it out

Finally, the "E-mail to managers" button (see figure 10) on the "Admin" sheet allowed the user to send the current daily report out to the managers group with the click of a button, assuming that the Microsoft Outlook Object Library has been referenced in Excel.

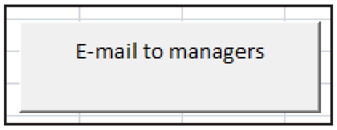


Figure 10 - E-mail

1.3 Conceptual Difficulties

Strings vs Numbers

One of the first difficulties faced in this project was during the process of coding the "Generate new daily business report" button. Numbers were to be initially entered into the various text boxes as numerical dollar figure goals for each department. Unfortunately, Excel reports an error in the actual spreadsheet cell stating that the number had been formatted as text. After extensive searching, I learned that the solution became as simple as adding a "* 1" to the end of the text box variable (see figure 11), automatically converting it to a number.

```
Sheets("MU New").Range("c10").ClearContents
Sheets("MU New").Range("c10") = txtNewSalesGoal * 1
```

Figure 11 - Convert numerical string to number

Dates, Dates, Dates

The next conceptual problem came when attempting to code the "Daily Report Entry Form." One of the objectives was to validate the date to ensure it was a date that already existed on the spreadsheet. This process turned out to be excessively difficult as the potential combinations of strings, numbers, and dates and their comparisons can create a multitude of VBA errors. Eventually, a solution was worked out where if the text box "value" was compared against the current dates in the cells then it would identify a correct date. However, incorrect dates continued to

```
Sub validateDate()
validDate = False

Dim dateTester As String

dateTester = txtDateEntry.Value
Sheets("MU New").Select
On Error GoTo thereisanError
For x = 2 To 32
  If dateTester = Cells(x, 5) Then
    validDate = True
    Exit Sub
  End If
Next

thereisanError:
Exit Sub

End Sub
```

Figure 12 - Date validation

```
Sub OpenPartsReport()

PartsFN = Application.GetOpenFilename(FileFilter:="Excel Files (*.xls), *.xls", Title:="Please select a file")
If PartsFN = False Then
' They pressed Cancel
  MsgBox "Please select a file"
  Exit Sub
Else
  Workbooks.Open Filename:=PartsFN
End If

partsSub

End Sub
```

Figure 13 - GetOpenFilename example

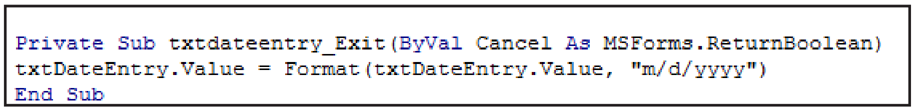
throw runtime errors and therefore an “On Error Goto” statement was added in which the sub-procedure would simply be exited upon the first invalid date (see figure 12). It was a seemingly elegant solution to a very technical problem as Excel has numerous methods of handling dates.

File > Open!

The next big challenge was to add the ability to allow the user to select a single Excel file to be analyzed. I met with the TA for ideas but found no simple solution. I needed something that would open just a single file and allow it to be manipulated by VBA. After turning to Google for quite some time, I came across the Application. GetOpenFilename solution (see figure 13). I was able to tweak the code to be implemented into my project in two places and it worked wonderfully.

Format that date

Another conceptual objective was to have uniformity of date formatting across the worksheets. Most every cell depends upon the dates in one form or another. The idea was that if everything followed the (X)X/(X)X/XXXX format, where (X) is never entered as a 0, then hopefully future errors with the dates would never be an issue. Since dates across all worksheets are propagated from first of the month date in cell E2 of the “MU New” sheet, it seemed prudent to ensure that it would be in the format as mentioned. The “Generate new daily business report” user form has a text box in which to enter the date of the first of the month. Upon exiting that box, the format of the date was automatically converted to the proper formatting (see figure 14).



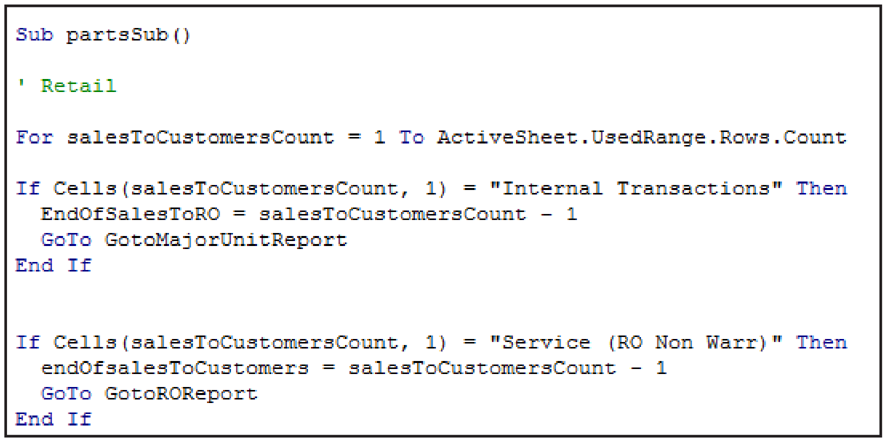
```
Private Sub txtdateentry_Exit(ByVal Cancel As MSForms.ReturnBoolean)
txtDateEntry.Value = Format(txtDateEntry.Value, "m/d/yyyy")
End Sub
```

Figure 14 - Date formatting

Sections, Categories, and a Few GoTos

Another great challenge in this project was processing the end-of-day report. Initially it was difficult to conceptually grasp what needed to be calculated. Essentially there were three sections to the report for the Parts department: retail sales, repair order sales, and major unit sales. Each of the three had a shared list of sales categories that are specific to the Parts department. Any or all of those sections and categories may or may not be present in a given day’s report, but, if present, they had to be accounted for accordingly. Daily service dollars were also calculated from the report on a separate set of categories spanning all three sections, if present. It was a daunting task.

The problem was eventually solved with the implementation of a series of If... Then...GoTo statements (see figre 15). The report begins the first of the three components of the end-of-day report and, if present, performs the GoTo upon hitting the beginning of the second, or third, section. The report will go straight to the second section, or third, if either the first or second components are not present in the end-of-day report. The daily service dollars calculation was insensi-



```
Sub partsSub()

' Retail

For salesToCustomersCount = 1 To ActiveSheet.UsedRange.Rows.Count

If Cells(salesToCustomersCount, 1) = "Internal Transactions" Then
  EndOfSalesToRO = salesToCustomersCount - 1
  GoTo GotoMajorUnitReport
End If


If Cells(salesToCustomersCount, 1) = "Service (RO Non Warr)" Then
  endOfsalesToCustomers = salesToCustomersCount - 1
  GoTo GotoROReport
End If
```

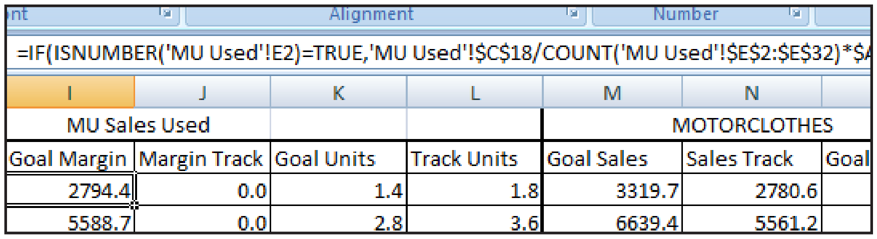
Figure 15 - GoTo example

tive to which component was being analyzed so it runs through the entire report looking for specific categories regardless of the component. As an added bonus, all the cells that get selected also receive a cell fill color and the file is saved so that a manager can review the end-of-day report to ensure the program ran properly if desired.

Other features

Plot Data

There were a couple of other neat features that are worth mentioning but not VBA related. The data points for each graph were dynamically calculated on a hidden sheet called "Plot Data." There were over 2000 formula cells on that sheet used for calculating data points on the other worksheets (see figure 15).



```
=IF(ISNUMBER('MU Used'!E2)=TRUE,'MU Used'!$C$18/COUNT('MU Used'!$E$2:$E$32)*$A
```

I	J	K	L	M	N	
MU Sales Used				MOTORCLOTHES		
Goal Margin	Margin Track	Goal Units	Track Units	Goal Sales	Sales Track	Goal
2794.4	0.0	1.4	1.8	3319.7	2780.6	
5588.7	0.0	2.8	3.6	6639.4	5561.2	

Figure 15 - Plot data example

Parts and MotorClothes

As initially mentioned in section 1.2, the report was consolidated from three different Excel files down to one. So now the numbers for the MotorClothes sheet are calculated off of the sales and margin data for the MotorClothes salespeople. The old method was to run the numbers for the MotorClothes salespeople in a single Excel file and then manually copy the sums over to a separate report for the managers group. It was also a similar process for the Parts department.

1.4 Initial Feedback

"Outstanding job." - Todd, General Manger

"I'm REALLY glad you're involved in this. Thanks!" - Sebastian, Parts Manager

"James has done an amazing job with the report. He has really simplified the data aggregation process." - Joe, IT Manager

"The report you built is a beautiful thing." - Joe, IT Manager