The goal of this workshop is to create a dynamic "dashboard" or "Report". A partial image of what we will be creating is shown on the right. It contains graphs and tables that display information relevant for a presentation or decision maker.

Our dashboard is also dynamic. If you click a company's radio button, the data in all of the charts and tables on the dashboard update to display data for the company selected. It also contains drop down lists that allow you to select the financials you wish to view (top chart only), macro buttons to display charts as either lines or columns, and a button to print the dashboard as a report.

To create our dashboard, we will use the following functions and techniques:
- Charts from Data in a Matrix Format.
- PivotTables & PivotCharts from Data in a Database Format.
- Form Tools to Create Radio Buttons.
- Validation Rules to Create a Drop Down List.
- Index() and Match() in an Array Formula to pull information from a Database.
- Choose() to Assign Text to Numbers.
- Vlookup() to pull Information from a Database.
- The Macro Recorder.
- Typing Macros in the VBA Editor.
- The Camera Tool.

About Learning VBA (Macros) in this Workshop
We will use a lot of different techniques to create our dashboard. Some of them rely on formulas to make the dashboard dynamic and some of them rely on Visual Basic macros. (A macro is programming code that can automate tasks you normally would do manually in Excel. For example, changing the chart type of all charts on your sheet.) Macros are written in Visual Basic for Applications (VBA) which is a full programming language that can takes months or even years to learn. Unfortunately, I can't teach you VBA in the timespan of this workshop. It would be like trying to teach you a foreign language in 3 hours – it's just not possible.

However, there are a few very simple things we can learn to do such as using If Then structures, For Each Next loops, and Variables. Further, we can use the Macro Recorder to build the VBA for us. I am not an expert in VBA but by examining the code generated by the recorder, Googling VBA questions, and some reading, I was able to create the dashboard we will be using in this workshop. You don't have to know it all, you just have to know enough to accomplish the task a hand! For those interested in learning Excel VBA, here are some resources and advice:
- Use the recorder to record your steps and then examine the code generated. Note however that recorder code is not always the most efficient and it can't create conditional macros but it helps a lot.
- Google your VBA questions. For example, "How do you change your Chart type using Excel VBA?" You will be surprised how many answers you can find in VBA forums that supply you with the advice and code you need.
- Read books on VBA. I found these two books by "John Walkenbach" very useful: "Excel VBA Programming for Dummies" and "Excel 2010 – Power Programming with VBA".
- Finally, one of the best ways to learn VBA is to think of something you would like to accomplish with VBA and then set out on doing it. As they say, the best way to learn to write is to start writing!
# Table of Contents

About Learning VBA (Macros) in this Workshop.......................................................................................... 1

Workshop Dashboard Overview.................................................................................................................. 4

Exercise 1: Workshop Preparation.............................................................................................................. 5
  Step A: Downloading the Excel Workshop File ....................................................................................... 5
  Step B: Displaying the Developer Tab ..................................................................................................... 5
  Step D: Adding the Camera Tool to the Ribbon ..................................................................................... 6
  Step E: Saving your Excel file Macro Enabled .......................................................................................... 7

About Macro Security.................................................................................................................................... 7
  Accessing the Security Settings .................................................................................................................. 7
  Trusted Locations........................................................................................................................................ 7

About the VBA Macro Editor......................................................................................................................... 8
  Editor Navigation ......................................................................................................................................... 8
  Inserting a Module ..................................................................................................................................... 8
  Debugging Your Code ............................................................................................................................... 9

Exercise 2: Update a Chart using Form Controls (Database Structure).................................................. 10
  How it Works ............................................................................................................................................... 10
  Part A: Creating the Small Table .............................................................................................................. 11
  Part B: Creating the Companies Options ................................................................................................. 11
  Step C: Programming the Option Buttons .............................................................................................. 12
  Step D: Use Choose() to Display Company Names Rather than Numbers ........................................... 12
  Step E: Creating the List for the Financials Drop Down ........................................................................ 13
  Step F: Creating the Drop Downs for Financials .................................................................................... 14

Pulling the Information from the Database: Array Formula with Match() & Index()............................. 15
  Match()..................................................................................................................................................... 15
  Using Match() as an Array Function (CONTROL + SHIFT + ENTER) .................................................. 15
  Using Index() to Get the Financial from the Selected Row ................................................................... 16
  Step G: Type the formula (and Don’t forget Absolute Addresses) ....................................................... 17
  Step H: Labeling our Table ...................................................................................................................... 18
  Step I: Creating the Chart ....................................................................................................................... 18
  Step J: Testing the Interactive Chart ....................................................................................................... 19
  Step K: Naming the Chart: TopChart ...................................................................................................... 19

Exercise 3: Macro to Update Chart Data Based on Company Name Selected ..................................... 20
  Step A: Creating the Profit Margin Chart ............................................................................................... 21
  Step B: Creating the Price Earnings Ratio Chart .................................................................................... 22
  About the Macro to Change the Profit Margin Data ............................................................................... 23
  Step C: Creating the Macro to Swap the Profit Margin Data ................................................................. 24
  Using an IF THEN Structure .................................................................................................................... 24
  Making our Macro More Efficient by Not Repeating Lines Unnecessarily ........................................ 25
  Making our Macro More Efficient by using Variables .......................................................................... 26
  Not Using Variables: Easier to Understand but More Typing ............................................................... 27
  Shorter Method Using Variables: Less Typing but Slightly more Difficult to Understand ............... 28
Exercise 9: Format the Report as Desired

Exercise 8: Creating a Macro Button to Print the Dashboard

Exercise 7: Adding Buttons to Change All Charts to Lines or Columns

Exercise 6: Vlookup() and the Camera Tool to Display Company Information

Exercise 5: Creating the Earnings per Share Chart (Pivot Table Chart & Macro)

Exercise 4: Macro to Update the Price Earnings Ratio Chart

Exercise 3: Creating the Earnings Per Share Chart to Match the Company Selected

Exercise 2: Copying the PivotChart to the "Dashboard" Sheet

Exercise 1: Naming the Earnings Per Share Chart Located on the Dashboard Sheet: "EPSChart"

Step A: Using Range Names for P7 and Our Database Range (A8:E15)

Step B: Writing the Vlookup()s to Populate the Table

Step C: Testing Your Vlookup() Formulas

Step D: Using the Camera Tool to Display the Table on the Dashboard Sheet

Step E: Testing the Camera Table

Step F: Testing your Macros

Step G: Making the Button to Run the ChangeChartsToLine Macro

Step H: Testing the Buttons

Exercise 4: Macro to Update the Price Earnings Ratio Chart

Understanding the Macro to Update the Price Earnings Ratio Chart

More on the Concatenation Operator

Step A: Creating the Macro to Update the PE Ratio Chart

Step B: Stringing the Macros Together with "Call"

Step C: Assign the Macro to the Radio Buttons

Step D: Testing the Macro

Exercise 5: Creating the Earnings per Share Chart (Pivot Table Chart & Macro)

Step A: Creating the PivotTable

Step B: Creating the PivotChart

Step C: Copying the PivotChart to the "Dashboard" Sheet

Step D: Naming the Earnings Per Share Chart Located on the Dashboard Sheet: "EPSChart"

Step E: Creating a Macro to Update the Earning Per share Chart to Match the Company Selected

Step F: Testing the code

Step G: Calling the "ChangePivotChartSeries" Macro

Exercise 6: Vlookup() and the Camera Tool to Display Company Information

How Vlookup() Works

Step A: Using Range Names for P7 and Our Database Range (A8:E15)

Step B: Writing the Vlookup()s to Populate the Table

Step C: Testing Your Vlookup() Formulas

Step D: Using the Camera Tool to Display the Table on the Dashboard Sheet

Step E: Testing the Camera Table

Exercise 7: Adding Buttons to Change All Charts to Lines or Columns

Step A: Alternative One - Creating the Two Macros for Line and Column Charts

Step B: Testing your Macros

Step C: Making the Button to Run the ChangeChartsToLine Macro

Step D: Test the Buttons

Exercise 8: Creating a Macro Button to Print the Dashboard

Step A: Record Printing

Step B: Making a Button to Run the PrintDashboard Macro

Step C: Testing the Macro

Step D: Assign the Macro to the Radio Buttons

Step E: Testing the Macro
Workshop Dashboard Overview

Below is the dashboard we will be making in the workshop. Its main feature is that all the charts and tables on the dashboard will update to display information for whatever company the user selects using the radio buttons.

![Dashboard Image]

This chart gets its data from the table below. Match() and Index() are to populate the table with data that corresponds to the company selected. No macros are used for this.

<table>
<thead>
<tr>
<th>ACME Financial Report</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACME</strong></td>
</tr>
<tr>
<td><strong>Revenue</strong></td>
</tr>
<tr>
<td><strong>Net Profit</strong></td>
</tr>
</tbody>
</table>

These two drop down lists allow you to decide which financial to display for the top chart only.

- Profit Margin (%)
- Price Earnings Ratio

These two charts get their information from data that is in a matrix format. The company to use is controlled by a macro.

- Earnings Per Share

This PivotChart gets its information from data that is in database format. The chart was made from a PivotTable and a macro is used to update the chart for the company selected.

This PivotChart gets its information from data that is in database format. The chart was made from a PivotTable and a macro is used to update the chart for the company selected.

This table gets its data from a series of Vlookup() formulas that pull data based on the company selected. It is made to appear here using the Camera tool.

<table>
<thead>
<tr>
<th>Company Name</th>
<th>Industry</th>
<th>Year Founded</th>
<th>Lead</th>
<th>Makes everything from anvils to portable holes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME</td>
<td>Everything</td>
<td>1891</td>
<td>R.K. Maroon</td>
<td></td>
</tr>
</tbody>
</table>

This macro button prints the dashboard as a report.
Exercise 1: Workshop Preparation

To complete the exercises in this workshop, you need to first do the following:
A) Download the Excel Workshop File.
B) Display the Developer Ribbon.
C) Add the "Camera" tool button to one of your ribbons.
D) Save the Excel file you downloaded from above as a Macro Enabled file.

**Step A: Downloading the Excel Workshop File**
To complete the exercises in this workshop, you will need to download the workshop Excel file that contains the data used by our dashboard.

a. Start your web browser and go to: www.marshall.usc.edu/computing/docs
b. Click "Microsoft Applications" from the left menu.
   c. Click "Excel" from the table of contents at the top of the page.
d. Download the Excel file "Dashboards.xlsx" located in the right most column.
   e. Open the file in Excel.

**Step B: Displaying the Developer Tab**
The optionally displayed "Developer" tab makes such actions as recording macros and accessing the VBA editor easier and is necessary to access the forms tools. To display the Developer tab:

   a. Right click any existing menu tab and select "Customize the Ribbon".
   b. If necessary, select "All Tabs" from the list.
   c. Check "Developer" then click "OK".

You should have a new tab to the right of your other tabs. Note that clicking the "Visual Basic" button will open the Visual Basic editor and you can also record and run macros from this tab.
### Step D: Adding the Camera Tool to the Ribbon

The *Camera* tool comes with Excel but is not on any of its ribbons. You can add it to any ribbon or the Quick Access Toolbar. We will make an area for it on the *Developer* tab called "Special" and place the Camera Tool there.

1. **Right** click the "**Developer**" menu tab and select "**Customize Ribbon**".

<table>
<thead>
<tr>
<th>Developer</th>
<th>Code</th>
<th>Add-Ins</th>
<th>Controls</th>
<th>XML</th>
<th>Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Developer]</td>
<td>[Code]</td>
<td>[Add-Ins]</td>
<td>[Controls]</td>
<td>[XML]</td>
<td>[Modify]</td>
</tr>
</tbody>
</table>

2. Click the **Development** tab to highlight it. (Don’t uncheck it.)

3. Click the "**New Group**" button.

4. Highlight the new group you created then click "**Rename**".

5. In Display Name, type "**Special**" then click "**OK**".

6. Set **Choose commands from** to **All Commands**.

7. Locate the "**Camera**" tool and drag it into the new area you created.

8. Click "**OK**".
**Step E: Saving your Excel file Macro Enabled**

If you create macros (a.k.a. "sub procedures") or custom functions, to save them you must save your Excel file as a Macro Enabled Workbook (XLSM). You do not lose any Excel functionality when saving as this type.

With the Excel file you downloaded open:

1. Click "File – Save As".
2. At Save as Type select: "Excel Macro-Enabled Workbook (*.xlsm)"
3. Give it a name and click "Save".

---

**About Macro Security**

One method hackers use to spread viruses is to place them in Excel files as macros. Typically, when you open a macro enabled Excel file you will see the warning shown to the right.

1. If you know it doesn’t contain any viruses, click "Enable Content" to open it. If you don’t click the button, your macros will not work.

---

**Accessing the Security Settings**

To view or change Excel’s security settings:

1. Click the "Macro Security" button on the Developer menu tab.
   
   (It's also under "File – Options – Trust Center – Trust Center Settings").

The default setting is shown and is what is recommended. This makes Excel ask you to enable content every time you open a macro enabled Excel file. Note that if you open an Excel file and closed the security warning without enabling, you can always close the Excel file and reopen it to get the Enable Content button back.

---

**Trusted Locations**

You can specify folders on your computer to be "Trusted Locations". When you open files stored in a trusted location, Excel will automatically enable the macro content for you without asking you to enable it.

1. Click the "Macro Security" button on the Developer menu tab.
2. Click "Trusted Locations".
3. Click "Add new location…".
4. Specify the folder and click "OK".
About the VBA Macro Editor

Some of the exercises in this guide utilize the VBA editor. This section offers a brief overview of the editor and some basics on debugging VBA code. While the code for your custom functions & sub procedures (a.k.a. macros) are stored in your macro enabled workbook, it is viewed, created, and edited within the VBA editor. To access the VBA editor:

1. While in Excel press: **ALT + F11** *(You can also click the "Visual Basic" button on the Developer tab.)*

**Editor Navigation**
The illustration below shows the components beginners might use within the VBA Editor. Components can be closed by clicking their X's and reopened either under "**View**" or pressing their shortcut keys. For this workshop, all the macros we created will be stored within modules.

**Inserting a Module**
Code is stored within modules. You can store all of your macros in a single module or use multiple modules. To insert a module, while in the VBA editor, click **"Insert – Module"**. Note that the recorder often creates a new module.
Debugging Your Code

When beginning, chances are you will not get your macros right the first time you test them. They might do nothing at all, do the wrong thing, or give you an error message and take you into Debugger mode.

The debugger automatically highlights the name of the macro with issues in yellow and highlights the problem for you. In this case, I spelled the name of the Ucase() function wrong. When it takes you into debugger mode, you will have to do two things to get up and running again:

a. Fix the code (or comment it out – see below).
b. Press CONTROL + SHIFT + F8 to step out of debugger mode (or "Debug – Step Out" from the menu.)

Commenting Out Code: '

If you have having problems finding the bad code, you can temporarily tell VBA to ignore it out by typing an apostrophe in front of it and then pressing CONTROL + SHIFT + F8 to turn off debugger mode. Commented code turns green and will not be read by the macro. This allows you to trouble shoot and try different things without having to erase lines of code.

Commenting Out Blocks of Code

If you need to comment out large blocks of code, typing an apostrophe in front of each line can be tedious. Follow these steps to comment out large blocks of code.

1. In the VBA editor, display the "Edit" toolbar (Right click any menu tab and select "Edit").
2. Highlight the code to be commented out.
3. Click the "Comment Block" button on the Edit toolbar.
   (To remove the comments, re-highlight then click the "Uncomment Block" button.)

```vba
' N is the name of the company
Dim V As Variant
Dim N As Variant

'If Worksheets("Dashboard").Range("P7") = "ACME" Then
  V = "=MTXFormat!$BS4:$FS4"
  N = "=MTXFormat!$AS4"
'ElseIf Worksheets("Dashboard").Range("P7") = "Mcm Corp." Then
  V = "=MTXFormat!$BS5:$FS5"
  N = "=MTXFormat!$AS5"
'ElseIf Worksheets("Dashboard").Range("P7") = "Oceanic Airlines" Then
  V = "=MTXFormat!$BS6:$FS6"
  N = "=MTXFormat!$AS6"
'ElseIf Worksheets("Dashboard").Range("P7") = "Stark Industries" Then
  V = "=MTXFormat!$BS7:$FS7"
  N = "=MTXFormat!$AS7"
```
Exercise 2: Update a Chart using Form Controls (Database Structure)

The chart shown is based on the data in the small table below it. As you are probably aware, if you change the data a chart is based on the chart will update automatically. In this example, we have provided radio buttons and drop down lists that allow the user to select what data appears in the table. When they select a different company or change which financials to display, the data in the table will be swapped out to match their choice and the chart will update.

We will accomplish this using the following tools:

- Form Tools
- Choose()
- Match()
- Index()

Note that no macros are needed in this example!

How it Works

As mentioned above, the chart gets its data points from the small table directly below it. The numbers in the small table in turn come from the database shown to the right ("DBFormat" sheet).

To accomplish this, the small table uses formulas to pull their numbers from the database on the "DBFormat" sheet. In this example, the user selected "Stark Industries" as the company and "EBITDA" & "Enterprise Value" as the financials. The tricky part is to pulling the correct values from the database for the company and financials selected.
Part A: Creating the Small Table

In this section we will create the small table that will display below the chart.

1. On the "Dashboard" sheet, create the table shown.
   a. Type the years in.
   b. Format as shown.

We will create the formulas to display the numbers the chart is based on later.

Part B: Creating the Companies Options

We will now create the radio buttons shown to the right. We will be placing these inside an option group so the radio buttons will work as a group. Note the following as you work:

- **Control** click an object to get the grips around it.
- **Right** click an object to view its properties or delete it.

1. From the menu, click the "Developer" tab.
2. Click "Insert" and then click "Option Group".
3. Click and drag a box in the location shown in the image to the right.
   (If you need to reselect it, **Control** click its name.)
4. Click "Insert" again.
5. Select the Form Control "Option Button".
6. Click inside the option group.
7. Highlight its default text and type "ACME".
   (If necessary, **Control** click to reselect.)
8. Repeat steps 4-7 to create the option buttons for the other companies, typing their name as appropriate. Also, be sure to create them in the order shown.
9. If you desire, you can change the name of the option group or delete it. Highlight the option groups name and either type something else or delete it.
**Step C: Programming the Option Buttons**

When options buttons are working as a group, clicking one of the option buttons outputs a number that corresponds to that option button. Typically the first option button is 1, the second is 2, and so on. In this section we will tell the buttons to store that number in cell P6.

6. (Optional) Click "Format – Group – Group" to group your selected option buttons.
7. Click in a cell to deselect your option buttons.
8. Click the different buttons and watch cell P6. It should get a different number for each button.

**Step D: Use Choose() to Display Company Names Rather than Numbers**

To pull the correct company name out of our database on the "DBFormat" tab, we need to use actual company names, not numbers. We could use Vlookup() or a nested IF() to do this but because it is a relatively small list of companies, we will use Choose(). Choose() is used to assign an item to a number based on its position on a list.

Syntax: =Choose(Index Number, Item1, Item2, Item3, etc.)

Example: =CHOOSE(2,"ACME","Mom Corp.","Oceanic Airlines","Stark Industries","Tyrell Corp.","Umbrella")

Returns Mom Corp. because Mom Corp. is in the second item (Item2) on the list.

1. In cell P7 type:
   
   =CHOOSE(P6,"ACME","Mom Corp.","Oceanic Airlines","Stark Industries","Tyrell Corp.","Umbrella")

2. Select different option buttons and both the number in P6 should the company name in P7 should change.
Step E: Creating the List for the Financials Drop Down

The other interactive part of our dynamic chart is the drop down that allows users to select the financials they wish to view. To populate the choices for the drop down list, the choices must exist as a list in cells as the image shown to the far right. We could simply type them but we already have them on our "DBFormat" sheet but there are duplicates. This technique shows you how to extract unique entries from a list.

1. Go to the "DBFormat" tab.
2. Click the letter "B" to highlight all of column B.
3. From the menu, click "Data" then the "Advanced" button.

4. Click "Copy to another location".
5. List Range should already be filled in.
6. Copy the list to G1 on the same sheet.
7. Check "Unique records only".
8. Click "OK".

Your list should now exist in G1:G5 on the "DBFormat" sheet.
**Step F: Creating the Drop Downs for Financials**

We will now create the drop down choices for our financial data.

1. Go to the "Dashboard" sheet and click cell **K21**.
2. From the menu, click the "Data" tab, then "Data Validation – Data Validation…".

![Data Validation dialog box](image)

3. At "Allow" select "List".

   ![List selection](image)

4. At "Source", select the unique company names on the "DBFormat" sheet as shown.
5. Click "OK".

You should now have a working drop down list in cell K21 on the Dashboard sheet.

6. Copy cell **K21** to cell **K22** to create the second drop down list.
7. Format K21 &K22 as desired.

Your Dashboard sheet should now have functioning ratio buttons and two dropdown lists as shown below.
Pulling the Information from the Database: Array Formula with Match() & Index()

Our next step is to pull the information the user selected using the radio buttons and the drop down lists from the database into the small table so we can later create our chart from it. To accomplish this, we will use an array formula that utilizes the Match() and Index() functions. Because this is fairly complicated, we will first discuss Match(), Index(), and array formulas on their own before we jump into our actual formula.

**Match()**
The match function returns the relative position of the item you are searching for in a range. For example, if you were looking for "Revenue" in the range G2:G5, Match() would return 3 because "Revenue" is the third item down within the specified range. (It can work horizontally too but no blocks.)

**Syntax:**  
`Match(Item to Find, Range to Search, Match Type)`

**Example:**  
`Match("Revenue",G2:G5) = 3`

Note that Match Type 0 means you are looking for an exact match.

**Using Match() as an Array Function (CONTROL + SHIFT + ENTER)**

Array functions allow you to run each cell within a range of cells through a formula one at a time. To make Excel behave this way, you must press Control + Shift + Enter after typing or editing the formula.

The formula in the image to the left returns 3 because the third row is the only row in the range that matches the formula's criteria. The array compares each cell in the given ranges to the company and returns 0 if the equation is False and 1 if the equation is True.

\[
=\text{Match}(1,(I2:I4="ACME")*(J2:J4="Enterprise Value")*(K2:K4=2013),0)
\]

I2, I3, and I4 all contain "ACME" so each comparison evaluates to true ("1" to a PC). J2 does not contain "Enterprise Value" ("0" to a PC) but J3 and J4 both do ("1" to a PC). K2 & K3 do not contain 2013 ("0" to a PC) but K4 does ("1" to a PC).

The array would evaluate rows 2, 3, & 4 in our range as follows:

- Row 2: \((1) \times (0) \times (0) = 0\)
- Row 3: \((1) \times (1) \times (0) = 0\)
- Row 4: \((1) \times (1) \times (1) = 1\)

Note that "1" is only returned when all three criteria are true; for all others, "0" is returned. This is because "0" times anything is zero. Knowing that 1 is only returned when all of our conditions are met, we can tell Match() to search for "1".

\[
=\text{Match}(1,(I2:I4="ACME")*(J2:J4="Enterprise Value")*(K2:K4=2012),0)
\]

"0" tells Match() we are looking for an exact match. (i.e. exactly "1")
Now that you understand how Match() and arrays work (hopefully), we will adjust Match() to work with the actual data rather than the example data shown above. The image below shows the "Dashboard" sheet where the user has selected "ACME" for the Company and "Enterprise Value" for the financial.

![Dashboard Sheet Image]

**TESTING:** To return the row number where company is "Umbrella", the financial is "Enterprise Value", and the year is "2012", type the formula below in any blank cell and press **Control + Shift + Enter**.

```
=INDEX(DBFormat!A2:D309,12,4)
```

Excel should return 12.

### Using Index() to Get the Financial from the Selected Row

At this point, we have a formula that will tell us which row in the given range contains the correct information. Now we just need tell Excel which cell in that row it should return. We can use Index() for this. It returns the value at the intersection of a given row and column within a range.

**Syntax:**

```
Index(Array, RowNumber, ColumnNumber)
```

- **Array** is our database range A2:D309.
- **RowNumber** is the row within the range you are after (12 in this example).
- **ColumnNumber** is the column within the range you are after (4 in this example).

```
=INDEX(DBFormat!A2:D309,12,4)
```

Note that the row and column numbers pertain to the position within the given range, not the actual worksheet row and column numbers. In other words, if you highlighted a range out in the middle of know where, its top row would be row 1 even if you were down in row 100 on the worksheet.
We are now ready to put this all together. We know we wish to return the data in row 4 so we have the column number and Match() returns the correct row number. So, we just need to nest Match() within the RowNumber part of our Index() function.

\[ \text{Match}(1, \text{DBFormat!A2:A309=Dashboard!P7})*\text{Match}(\text{DBFormat!B2:B309=Dashboard!K21})*\text{Match}(\text{DBFormat!C2:C309=Dashboard!B20}),0) \]

The nested formula is shown below. Remember to press Control + Shift + Enter after typing it.

\[ \text{Index(DBFormat!A2:D309, Match(1, DBFormat!A2:A309=Dashboard!P7)\times DBFormat!B2:B309=Dashboard!K21\times DBFormat!C2:C309=Dashboard!B20),0),4) } \]

**Step G: Type the formula (and Don’t forget Absolute Addresses)**

Unless you use range names (which we should have), absolute addresses never go away. We intend to copy our formula to the other years and down one row so we need to make sure we lock in most of the cell addresses in the formula above.


Note that we do need some of the cells to shift addresses when we copy. Be sure you didn’t lock everything:

- We locked the years vertically but not horizontally ($B20$).
- We only locked the financial horizontally but not vertically ($K21$).

1. Type the formula shown immediately above in cell B21 of the Dashboard sheet and press Control + Shift + Enter

2. Copy the formula left and then down to fill in the rest of the table.

3. Change the radio buttons and drop down financials to test the data table.
Step H: Labeling our Table
We now need to label our chart table.

1. Create the three links shown below.

Step I: Creating the Chart
We are now ready to create our chart.

2. From the menu, click "Insert".
3. Click the "Insert Line Chart" button and then the plain line chart button.
4. Move and resize the chart as shown.
5. We will now link the Chart Title to the company name displayed in P7.

6. Label the Y axis and apply other formatting to the chart and the sheet as desired.

**Step J: Testing the Interactive Chart**
Select different companies and financials to see the chart change.

**Step K: Naming the Chart: TopChart**
This isn't required but because we will refer to it later let's give it a meaningful name.

1. Click the chart to select it.
   - Excel 2013 → Click in the "Name" box, type: TopChart then press enter.
   - Excel 2010 → Click the "Layout" tab, click in the "Chart Name" box, type: TopChart and press enter.
Exercise 3: Macro to Update Chart Data Based on Company Name Selected

In this example, we will create the two charts shown below. One displays the Profit Margin and the other the Price Earnings Ratio for whatever company the user selects above. Rather than using another complex formula to make our updates as we did in Exercise 1, we will utilize macros to change the value series range used by the chart.

The Profit Margin Chart was created from the data on the "MTXFormat" sheet. The chart is currently displaying the legend (Name) and series (values) for the Umbrella corporation.

Macro: ChangeChartSeries_ProfitMargin
This macro will look in cell P7 to see whose company name is displayed and then adjust the chart's data series range to match. For example, the chart is currently displaying data for "Umbrella" (A9 & B9:F9) however, if the user selects "Mom Corp.", it will use A5 for the legend and B5:F5 for the series values. We will accomplish this with a VBA IF THEN statement.
Step A: Creating the Profit Margin Chart
In this step, we will create the "Profit Margin" chart shown on the previous page.

1. Click a blank cell (e.g. H4) on the "MTXFormat" sheet.
2. From the menu, click "Insert" then "Line Chart" then "2D Line Chart".
   (You should now have a blank line chart.)
3. Click the blank line chart you just created.
4. From the "Design" menu, click "Select Data".

5. Click the "Add" button to make the following settings:
   
   Series Name: A4
   Series Values: B4:F4

6. Click "OK" once.

7. Click the "Edit" button to make the following settings:
   
   Axis label Range: B3:F3

8. Click "OK".
9. Click "OK" again.

You should have a chart similar to the one shown here. Make sure your Legend and Chart Title are visible. If they are not, to the right is how you display them in Excel 2013.
10. Cut and Paste the chart to the "Dashboard" sheet.
   a. Right click the chart and select "Cut".
   b. Right click a cell on the "Dashboard" sheet and select "Paste (Use Destination Theme)".
   c. Position the chart as shown.

11. Name the Chart
    Our Macro needs to refer to the chart by name so we will name it "PMChart".
    a. Click the chart to select it.
    b. Click in the name box.
    c. Type: **PMChart** and press Enter.

12. Set the Chart Title "Profit Margin".
    a. Click the chart's Title box to select it.
    b. Click on the formula bar.
    c. Type **Profit Margin** and press enter.

---

**Step B: Creating the Price Earnings Ratio Chart**

We now need to create the Price Earnings Ratio chart. Fortunately, the steps are nearly identical creating the previous chart. The only changes will be the ranges used and the name of the chart:

1. Set the Ranges: follow the steps in "Step A"; except, use the following for the data ranges:
   - **Series Name**: A14
   - **Axis Label Range**: B13:F13
   - **Series Values**: B14:F14

2. Name the Chart: Follow the steps in "Step A"; except, name the chart: **PEChart**
3. Title the Chart: **Price Earnings Ratio**
### About the Macro to Change the Profit Margin Data

Our Profit Margin Chart is currently based on the data shown in the image below. In this step, we will create a macro to change the charts legend and data series to match the company selected in our “Select a Company” box.

![Profit Margin Chart Data](image)

### Side Exercise (Optional): See what Code the Recorder Creates

You change an existing chart’s data sources by clicking the chart and then clicking the “Select Data” button under the “Design” menu tab. If you like, turn on the recorder (use absolute referencing) and change the PMChart’s data series and name to Mom Corp’s data. Turn off the recorder and then check out the code generated (ALT + F11).

Below is the code created by the recorder.

```vba
Sub Macro4()
  ' Macro4 Macro
  ActiveSheet.ChartObjects("PMChart").Activate
  ActiveChart.SeriesCollection(1).Name = "=MTXFormat!A$5"
  ActiveChart.SeriesCollection(1).Values = "=MTXFormat!B$5:F$5"
End Sub
```

The first code line basically says: select the PMChart. It is a chart object on the current sheet. 
**ActiveSheet** is a property that specifies the current sheet. 
**ChartObjects** refers to the collection of charts. In this case “PMChart” within the collection. 
**.Activate** is a method of the chart selected.

The second line states: change the **legend** for the currently selected chart to the range A5 on the MTXFormat sheet. 
The third line states: Change the **value series** for the currently selected chart to the range B5:F5 on the MTXFormat sheet.

**SeriesCollection(1).Name** is the method used to set the name of a single series. (1 is an index number) 
**SeriesCollection(1).Values** is the method used to set the values of a single series. (1 is an index number)

### Making the Code Conditional: If Then Statements

The code needed to change the value and legend series used by a chart is shown in the box above. However, we need our code to be conditional. If the user select’s Umbrella, then we need to change both series to the data for Umbrella; if the user select’s ACME, then we need to change both series to the data for ACME. The **recorder cannot generate conditional code for us.** We must create our own sub procedure (i.e. macro) and code this one ourselves using an IF THEN statement.
1. If you are in Excel, press \texttt{ALT + F11} to toggle to the VBA editor.

If you would like a new module click “Insert – Module”. It doesn’t really matter which of our file’s modules we type the code in as long as it is on our current file’s tree. Also, you can type your macros within separate modules or place all of them within the same module. Again, it doesn’t really matter for our purposes.

2. Double click the module you would like to type your code in (Module1 for example).

3. Type the code shown to the left. We are creating a macro called “\texttt{ChangeChartSeries_ProfitMargin}”. The first line of your macro must start with “Sub” followed by the name you want to give your macro followed by (). Macros all end in “End Sub” and all of its code does between “Sub” and “End Sub”. The second line is a comment and is not read by the macro. It is a note to yourself.

\textbf{Using an IF THEN Structure}

One way to execute code conditionally is to use an IF THEN structure as shown in the image to the right. Code will only execute if the condition above it is true. “\texttt{ElseIf}” is optional, only read if the condition above it is false, and is used to test another condition. Multiple “\texttt{ElseIf}”s are allowed. “\texttt{Else}” is also optional and is only read if all other conditions are false. You may have only one “\texttt{Else}”. Note that the structure must end in “End If”.

The image below shows how we could use the IF Then structure for our code; however, ours will be slightly different.

\begin{verbatim}
Sub ChangeChartSeries_ProfitMargin()
    'Macro to change the Profit Margin Chart
    If Worksheets("Dashboard").Range("P7") = "ACME" Then
        ActiveSheet.ChartObjects("PMChart").Activate
        ActiveChart.SeriesCollection(1).Values = "=MTXFormat!$B$4:$F$4"
        ActiveChart.SeriesCollection(1).Name = "=MTXFormat!$A$4"
    ElseIf Worksheets("Dashboard").Range("P7") = "Mom Corp." Then
        ActiveSheet.ChartObjects("PMChart"). Activate
        ActiveChart.SeriesCollection(1).Values = "=MTXFormat!$B$5:$F$5"
        ActiveChart.SeriesCollection(1).Name = "=MTXFormat!$A$5"
    End If
End Sub
\end{verbatim}
Works but Long
You may notice that for every IF and ElseIf statement, we are activating the same chart. Because this is done exactly the same way for every condition, it would be more efficient to state this just once before the conditions start.

Better
Here is the redesigned structure with the chart activation line appearing before our IF structure.

But as you will see, we can shorten this even more…

```vba
Sub ChangeChartSeries_ProfitMargin()
    'Macro to change the Profit Margin Chart
    ActiveSheet.ChartObjects("PMChart").Activate
    If Worksheets("Dashboard").Range(“P7”) = "ACME" Then
        ActiveChart.SeriesCollection(1).Values = "=MTXFormat!B$4:F$4"
        ActiveChart.SeriesCollection(1).Name = "=MTXFormat!A$4"
    ElseIf Worksheets("Dashboard").Range(“P7”) = "Mom Corp." Then
        ActiveChart.SeriesCollection(1).Values = "=MTXFormat!B$5:F$5"
        ActiveChart.SeriesCollection(1).Name = "=MTXFormat!A$5"
    End If
End Sub
```

Code for all companies not shown.


**Making our Macro More Efficient by using Variables**

A variable is basically the name of a place holder for something that will change. A variable name cannot contain spaces, be a reserved word, and does not have to be declared (although it really should be if you want faster macros).

For each company, we are changing the range where the chart gets its data points (values) and its legend (name). We can save ourselves more typing by assigning these settings to variables (V) and (N) and then use those variables later at the end, outside of our IF structure.

**Before Using Variables**

```vba
Sub ChangeChartSeries_ProfitMargin()
'Macro to change the Profit Margin Chart

ActiveSheet.ChartObjects("PMChart").Activate

If Worksheets("Dashboard").Range("F7") = "ACME" Then
    ActiveChart.SeriesCollection(1).Values = "=MTXFormat!$B$4:$F$4"
    ActiveChart.SeriesCollection(1).Name = "=MTXFormat!$A$4"

ElseIf Worksheets("Dashboard").Range("F7") = "Mom Corp." Then
    ActiveChart.SeriesCollection(1).Values = "=MTXFormat!$B$5:$F$5"
    ActiveChart.SeriesCollection(1).Name = "=MTXFormat!$A$5"

End If

End Sub
```

**After using Variables**

```vba
Sub ChangeChartSeries_ProfitMargin()
'Macro to change the Profit Margin Chart

ActiveSheet.ChartObjects("PMChart").Activate

If Worksheets("Dashboard").Range("F7") = "ACME" Then
    V = "=MTXFormat!$B$4:$F$4"
    N = "=MTXFormat!$A$4"

ElseIf Worksheets("Dashboard").Range("F7") = "Mom Corp." Then
    V = "=MTXFormat!$B$5:$F$5"
    N = "=MTXFormat!$A$5"

End If

ActiveChart.SeriesCollection(1).Values = V
ActiveChart.SeriesCollection(1).Name = N

End Sub
```

Notice we have moved this code after the IF structure.

The value of V and N varies (hence the term variable) depending upon which IF condition is true. For example, if the user selected “Mom Corp.” then V becomes “=MTXFormat!B5:F5” and N becomes “MTXFormat!A5”. The final two lines of code set the chart series (.Values) and legend (.Name) to the ranges V and N represent.
Sub ChangeChartSeries_ProfitMargin()
' Macro to change the Profit Margin Chart

ActiveSheet.ChartObjects("PMChart").Activate

If Worksheets("Dashboard").Range("P7") = "ACME" Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$4:$F$4"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$4"
ElseIf Worksheets("Dashboard").Range("P7") = "Mom Corp." Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$5:$F$5"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$5"
ElseIf Worksheets("Dashboard").Range("P7") = "Oceanic Airlines" Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$6:$F$6"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$6"
ElseIf Worksheets("Dashboard").Range("P7") = "Stark Industries" Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$7:$F$7"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$7"
ElseIf Worksheets("Dashboard").Range("P7") = "Tyrell Corp." Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$8:$F$8"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$8"
ElseIf Worksheets("Dashboard").Range("P7") = "Umbrella" Then
    ActiveChart.SeriesCollection(1).Values = "+MTXFormat!$B$9:$F$9"
    ActiveChart.SeriesCollection(1).Name = "+MTXFormat!$A$5"
End If

End Sub
**Shorter Method Using Variables: Less Typing but Slightly more Difficult to Understand**

4. Below is the final code that we will use. Type it into the module.

```vba
Sub ChangeChartSeries_ProfitMargin()
    'Macro to change the Profit Margin Chart
    ActiveSheet.ChartObjects("PMChart").Activate

    If Worksheets("Dashboard").Range("P7") = "ACME" Then
        V = "=MTXFormat!$B$4:$F$4"
        N = "=MTXFormat!$A$4"
    End If

    ElseIf Worksheets("Dashboard").Range("P7") = "Mom Corp." Then
        V = "=MTXFormat!$B$5:$F$5"
        N = "=MTXFormat!$A$5"
    End If

    ElseIf Worksheets("Dashboard").Range("P7") = "Oceanic Airlines" Then
        V = "=MTXFormat!$B$6:$F$6"
        N = "=MTXFormat!$A$6"
    End If

    ElseIf Worksheets("Dashboard").Range("P7") = "Stark Industries" Then
        V = "=MTXFormat!$B$7:$F$7"
        N = "=MTXFormat!$A$7"
    End If

    ElseIf Worksheets("Dashboard").Range("P7") = "Tyrell Corp." Then
        V = "=MTXFormat!$B$8:$F$8"
        N = "=MTXFormat!$A$8"
    End If

    ElseIf Worksheets("Dashboard").Range("P7") = "Umbrella" Then
        V = "=MTXFormat!$B$9:$F$9"
        N = "=MTXFormat!$A$9"
    End If

    ActiveChart.SeriesCollection(1).Values = V
    ActiveChart.SeriesCollection(1).Name = N
End Sub
```
Step D: Testing the Macro

For now, to test your macro you will need to click a company on the “Dashboard” sheet and then run the macro manually. After it is working we will connect it to the company radio buttons.

1. Go to the “Dashboard” sheet.
2. Select one of the company radio buttons.
3. From the “Developer” menu tab, click the “Macros” button.
4. Click the “ChangeChartSeries_ProfitMargin” macro.
5. Click “Run”.

If there is an error, Excel will usually display the VBA editor and place you into debugger mode. If you are lucky, it will also highlight the error. Note that you will not be able to run any macros while in debugger mode. The steps to fixing this are as follows:

a. Fix the error.
b. Exit debugger mode (Control + Shift + F8) or from the menu: “Debug” – “Step Out”.
c. Try running your macro again.

Step E: Assign the Macro to the Radio Buttons

To run the macro by clicking one of the radio buttons, follow the steps below.

1. From the menu, click the “Developer” tab.
2. Make sure the “Design Mode” button is depressed.
3. Control + Click each radio button until they are all selected. (We will assign them all to the same macro.)
4. Right click one of the radio buttons and select “Assign Macro”.
5. Click the “ChangeChartSeries_ProfitMargin” macro and click “OK”.

6. Click away from the radio buttons to deselect them.
7. Click the radio buttons to test it. The Profit Margin chart should change to display data for the currently selected company.
### Exercise 4: Macro to Update the Price Earnings Ratio Chart

As with the Profit Margin chart, we want the Price Earnings Ratio chart to update when we select a different company.

We could just copy the macro we used for the Profit Margin chart (ChangeChartSeries_ProfitMargin) and make some edits. The code for this is shown below with the edits needed to make this work in red. However, there is an even shorter approach we will take as shown on the next page.

```vba
Sub ChangeChartSeries_PERatio()
' Macro to change the Price Earnings Ratio Chart

ActiveSheet.ChartObjects("PEChart").Activate

If Worksheets("Dashboard").Range("P7") = "ACME" Then
    V = "=MTXFormat!$B$14:$F$14"
    N = "=MTXFormat!$A$14"
ElseIf Worksheets("Dashboard").Range("P7") = "Mom Corp." Then
    V = "=MTXFormat!$B$15:$F$15"
    N = "=MTXFormat!$A$15"
ElseIf Worksheets("Dashboard").Range("P7") = "Oceanic Airlines" Then
    V = "=MTXFormat!$B$16:$F$16"
    N = "=MTXFormat!$A$16"
ElseIf Worksheets("Dashboard").Range("P7") = "Stark Industries" Then
    V = "=MTXFormat!$B$17:$F$17"
    N = "=MTXFormat!$A$17"
ElseIf Worksheets("Dashboard").Range("P7") = "Tyrell Corp." Then
    V = "=MTXFormat!$B$18:$F$18"
    N = "=MTXFormat!$A$18"
ElseIf Worksheets("Dashboard").Range("P7") = "Umbrella" Then
    V = "=MTXFormat!$B$19:$F$19"
    N = "=MTXFormat!$A$19"
End If

ActiveChart.SeriesCollection(1).Values = V
ActiveChart.SeriesCollection(1).Name = N

End Sub
```
Understanding the Macro to Update the Price Earnings Ratio Chart

After typing the code for each company in the previous example, it occurred to me that this wouldn’t be very efficient if we were dealing with more than just a handful of companies. The question is, how do we change the chart to match the company’s data without writing an IF statement for each company?

The PE ratio data for ACME on the MTXFormat sheet is in row 14 and ACME’s company number is 1. The PE ratio data for Mom Corp. is in row 15, Oceanic Airlines in row 16 and so on. If we add 13 to any of these company’s number, it will correspond to the row that company’s data is on. We will create a variable named RN that is equal to each company’s row number plus 13 and then use RN in the row part of the code’s address. To do this, we need to do two things:

- Set the variable RN equal to whatever value is in cell P6 on the Dashboard sheet plus 13.
- Use RN in the address of the series with the concatenation operator (&).

The final code is shown below. Note that there must be a space on both sides of the ampersand & or this will return an error. It is broken down on the next page.

Sub ChangeChartSeries_PERatio()
    RN = Worksheets("Dashboard").Range("P6").Value + 13
    ActiveSheet.ChartObjects("PEChart").Activate
    ActiveChart.SeriesCollection(1).Name = ":MTXFormat!A" & RN
    ActiveChart.SeriesCollection(1).Values = ":MTXFormat!B" & RN & ":F" & RN
End Sub
Step A: Creating the Macro to Update the PE Ratio Chart
Follow the steps below to create the code shown at the bottom of the previous page.

1. Press ALT + F11 to access the VBA editor.
2. Click below the previous macro (after its “End Sub”) and type the code shown on the previous page.
3. Test the macro by going back to the “Dashboard” sheet of Excel and run the macro (Developer – Macro).

Sub ChangeChartSeries_PERatio()

    RN = Worksheets("Dashboard").Range("P6").Value + 13

    ActiveSheet.ChartObjects("PEChart").Activate

    ActiveChart.SeriesCollection(1).Name = ",=MTXFormat!A" & RN

    ActiveChart.SeriesCollection(1).Values = ",=MTXFormat!B" & RN & ":F" & RN

End Sub

This is the name of our macro.

This sets the value of the variable RN equal to whatever company number is in cell P6 on the Dashboard sheet plus 13. If ACME is selected (1) then RN becomes 14. This corresponds to the same row number ACME PE ratio data is in.

You must activate a chart to affect it.

This specifies what cell contains the legend to use for the chart. All legends are in column A but the row number is given by the value of RN. For example, if the ACME (1) is the company selected then RN becomes 14 and the address is: MTXFormat!A14

This specifies what cells contain the series values for the company selected. The column range is always B through F but it is up to RN to specify the row number. For ACME(1), RN is 14 and the address becomes: MTXFormat!B14:F14

Macros must end in “End Sub”.

More on the Concatenation Operator
The concatenation operator (&) is used to glue together a text string whose parts come from different places. The text part of the string must be in quotes "" but the variable name cannot be. Note also there must be a space on both sides of the &.

"=MTXFormat!B" & RN & ":F" & RN

If RN is 14, this comes out to be:

=MTXFormat!B14:F14
**Step B: Stringing the Macros Together with "Call"**

We could have just added the code below just above the "End Sub" of our ChangeChartSeries_PMChart macro but we didn’t. It is its own sub procedure.

```
RN = Worksheets("Dashboard").Range("P6").Value + 13
ActiveSheet.ChartObjects("PEChart").Activate
ActiveChart.SeriesCollection(1).Name = "=MTXFormat!A" & RN
ActiveChart.SeriesCollection(1).Values = "=MTXFormat!B" & RN & ":F" & RN
```

We can however tell our ChangChartSeires_PMChart macro to run our ChangeChartSeries_PEChart once it is done running using the “Call” command.

1. Just above the “End Sub” of the ChageChartSeries_PMChart macro, type the code circled below.

```
If Self.Worksheets("Dashboard").Range("P?") Then
  V = "=MTXFormat!$B$9:$F$9"
  N = "=MTXFormat!$A$9"
  End If
ActiveChart.SeriesCollection(1).Values = V
ActiveChart.SeriesCollection(1).Name = N
Call ChangeChartSeries_PERatio
End Sub
```

**Calling Macros stored in other Modules (of the same file)**

Note that this assumes that the macro you are calling is within the same module. If it is in a different module, you must precede it with “Module”. For example, if it were in Module2: Call Module2.ChangeChartSeries_PERatio

**Testing the Macro**

1. If you click a different company on the “Dashboard” sheet, both the Profit Margin chart and the Price Earnings chart should both update to reflect the company you selected.
Exercise 5: Creating the Earnings per Share Chart (Pivot Table Chart & Macro)

In this section we will create the Earnings Per Share chart shown below. It is based on the data on the "DataForPivot" sheet and is in database format. We could build another dynamic table as we did in Exercise 1 but now that we are using macros, creating a PivotChart will be much faster. In this exercise, we will do the following:

- Create a PivotChart based on the data on the "DataForPivot" sheet.
- Use a macro to make the company displayed on the chart match the company selected by the user.
Step A: Creating the PivotTable
In this section we will create the PivotTable that our PivotChart will be based upon.

1. Click the "DataForPivot" sheet.
2. Click in any cell within the data (e.g. B4).
3. From the menu, click "Insert – PivotTable".

   Excel should have included the entire table as the Table/Range and suggests you place the PivotTable on a "New Worksheet".

4. Click "OK" to accept these settings.

5. Build the PivotTable as shown below in steps a-d.

   ![PivotTable Field List]

   a. Drag "Company" into "Report Filter".
   b. Drag "Financial" into "Legend Field" (aka "Columns").
   c. Drag "Year" into "Axis Fields" (aka "Rows").
   d. Drag "Value $" into "Σ Values".

6. Hide the Totals:
   a. Click in the PivotTable to make its menu options appear.
   b. Click the "Design" tab.
   c. Click the "Grand Totals" button and then click "Off for Rows and Columns".

   ![Off for Rows and Columns]

   Your PivotTable should resemble the image to the right.
Step B: Creating the PivotChart
We will now create the PivotChart from the data in our PivotTable

1. Click in the PivotTable to make its menu options appear.
2. From the "Options" (or "Analyze" if 2013) menu tab click the "PivotChart" button.
3. Click the "Line" category.
4. Click the 2D Line chart button then click "OK".
5. Your chart should appear on the sheet.

Note that sometimes Excel switches the legend with the x-axis and your chart won’t look right. If this has happened, click the chart, click the "Design" tab, and then click the button "Switch Row/Column".

3. Reversing the Years (Largest to Smallest)
   a. Right click one of the years and select "Sort".
   b. Select "Sort Largest to Smallest".
4. Hiding the grey field buttons on the chart.
   a. Click the PivotChart to make its menu tabs appear.
   b. Click the "Analyze" tab.
   c. Click "Field Buttons" then "Hide All".

5. Naming the PivotTable
   We don't need to name the chart but we will need to name (or at least know the name of) our PivotTable.
   a. Click anywhere within the PivotTable.
   b. From the menu, click the "Options" (or "Analyze") tab.
   c. In "PivotTable Name", type: EPS_PivotTable and press enter.

6. Naming the Sheet the PivotTable is on
   We will need to refer to our sheet name in macro code so let's give it something meaningful.
   a. Double click the sheet tab at the bottom of the screen.
   b. Type: PTSheet and press enter.

**Step C: Copying the PivotChart to the “Dashboard” Sheet**

We will now copy our PivotChart to its final destination on the "Dashboard" sheet. Note that its data is linked to the PivotTable on the "PTSheet". Any changes we make the PivotTable data will affect the chart on the PDSheet and the Dashboard sheet. (However, formatting changes made directly to the charts themselves will not affect one another.)

1. Click your PivotChart to select it.
2. Copy it to the clipboard (Control + C).
3. Click in the location on the "Dashboard" sheet where you would like to place it.
4. Paste it (Control + V).
5. Size and move the chart as desired.

**Step D: Naming the Earnings Per Share Chart Located on the Dashboard Sheet: "EPSChart"**

Because we might refer to this chart in Macro code, we should give it a meaningful name.

1. Click the "Dashboard" sheet and click the chart you just created to select it.
   - Excel 2013 → Click in the "Name" box, type: EPSChart then press enter.
   - Excel 2010 → Click the "Layout" tab, click in the "Chart Name" box, type: EPSChart and press enter.
**Step E: Creating a Macro to Update the Earning Per Share Chart to Match the Company Selected**

In this section, we will create the code to control which company’s data is used in the PivotTable based on the radio button the user selects back on the Dashboard sheet.

### Recorder Code

**If you were to record yourself changing the pivot table’s Company filter, you would get something similar to the code shown to the right. It selects the sheet, sets the company filter to show all, and then makes the visibility of all companies except the one we selected “False”.

```vba
Sub Macro5()
    Sheets("PTSheet").Select
    ActiveSheet.PivotTables("EPS_PivotTable").PivotFields("Company").CurrentPage = "(All)"
    With ActiveSheet.PivotTables("EPS_PivotTable").PivotFields("Company")
        .PivotItems("Mom Corp.").Visible = False
        .PivotItems("Oceanic Airlines").Visible = False
        .PivotItems("Stark Industries").Visible = False
        .PivotItems("Tyrell Corp.").Visible = False
        .PivotItems("Umbrella").Visible = False
        .PivotItems("Wonka").Visible = False
    End With
End Sub
```

### First Attempt at Code

After looking at what the recorder did, I created the IF structure for each company as shown to the right. It works but it is a lot of typing and doesn’t adapt itself if you add more companies to your database. You would have to remember to append the code or you would get the wrong answer.

```vba
Sub ChangePivotChartSeries1()
    Sheets("PTSheet").Select
    ActiveSheet.PivotTables("EPS_PivotTable").PivotFields("Company").CurrentPage = "(All)"

    'ACME
    If Worksheets("Dashboard").Range("P7").Value = "ACME" Then
        With Sheets("PTSheet").PivotTables("EPS_PivotTable").PivotFields("Company")
            .PivotItems("Mom Corp.").Visible = False
            .PivotItems("Oceanic Airlines").Visible = False
            .PivotItems("Stark Industries").Visible = False
            .PivotItems("Tyrell Corp.").Visible = False
            .PivotItems("Umbrella").Visible = False
            .PivotItems("Wonka").Visible = False
        End With
    End If

    'Umbrella
    ElseIf Worksheets("Dashboard").Range("P7").Value = "Umbrella" Then
        With Sheets("PTSheet").PivotTables("EPS_PivotTable").PivotFields("Company")
            .PivotItems("Mom Corp.").Visible = False
            .PivotItems("Acme").Visible = False
            .PivotItems("Oceanic Airlines").Visible = False
            .PivotItems("Stark Industries").Visible = False
            .PivotItems("Tyrell Corp.").Visible = False
            .PivotItems("Umbrella").Visible = False
            .PivotItems("Wonka").Visible = False
        End With
    End If

    Sheets("Dashboard").Select
End Sub
```

To fit this page, I omitted the code for Mom Corp., Oceanic Airlines, Stark Industries, Tyrell Corp., and Wonka. Their structure is the same as the companies shown: set "Value" to the company and make the "Visible" of all other companies "False".

### Final Code (Next Page)

After Googling VBA forums and some trial and error, I discovered code that is much shorter, adaptable, and most importantly, seems to work. It is shown on the next page.
Typing the Code
1. Press **ALT + 11** to toggle to the VBA editor.
2. In a module, type the code shown below.

```vba
Sub ChangePivotChartSeries()
    CO = Worksheets("Dashboard").Range("P7").Value
    Worksheets("PTSheet").PivotTables("EPS_PivotTable").PivotFields("Company").ClearAllFilters
    Worksheets("PTSheet").PivotTables("EPS_PivotTable").PivotFields("Company").CurrentPage = CO
End Sub
```

How it Works

**Step F: Testing the code**
We will test the code in this section.
1. Click the "Dashboard" sheet and select one of the Companies from the radio buttons we created.
2. Click the "Developer" menu tab and click the "Macros" button.
3. Click the macro "ChangePivotChartSeries" then click run.
4. Debug as needed. (Remember, Control + Shift + F8 usually stops the debugger.)

**Step G: Calling the "ChangePivotChartSeries" Macro**
Clicking the Company radio buttons runs the "ChangeChartSeries_ProfitMargin" macro. After it runs, we need it to run the "ChangePivotChartSeries" macro.

1. Press **ALT + F11** to access the VBA editor.
2. View the "ChangeChartSeries_ProfitMargin" macro by double clicking Module1.
3. At the bottom of the code just above "End Sub", type: **Call**
   
   ```vba
   Call Module3.ChangePivotChartSeries
   End Sub
   ```
   
   **Mine was in a different module (Module3) so I had to proceed the macro name with the name of the module as shown.**
4. If you click the company radio buttons, our PivotChart on the Dashboard sheet should now update.

Exercise 6: Vlookup() and the Camera Tool to Display Company Information

This exercise covers how to change which company information is displayed in the table at the bottom of our dashboard based upon which company radio button the user selects. The easiest method of doing this is with VLOOKUP(). No macros are needed in this example! We will also use the Camera tool.

![Dashboard Sheet]

Camera
We will use the Camera Tool to make the table on the Company-Info sheet display on the Dashboard sheet.

![CompanyInfo Sheet]

Vlookup()
Vlookup() populates these cells with the information in the table below based upon the company name displayed in cell P7 on the Dashboard sheet.

The company displayed in P7 changes when the user selects its ratio button. (See Exercise 1).
How Vlookup() Works

Vlookup is designed to allow you to display specific information from a database based upon a criteria. For example, "Display the Industry for Tyrell Corp". As it pertains to this exercise, here is how we will use Vlookup():

=Vlookup(Company you are looking for, Database Range, Column # Item is in, False)

This is the cell address that contains the name of the company whose data we wish to return (i.e. P7)

This is the range of cells that includes the names of the companies whose data we wish to return and the data to be returned (i.e. A9:E15)

This is the column number of the item you wish to return. For example, "Industry" is the 2nd column in the range of our database so we would use 2.

False means we want an "Exact" match.

For Example:
If we wanted to display the Industry (column 2) for the company displayed in cell P7 on the Dashboard sheet, we would type the following:

=VLOOKUP(Dashboard!P7,CompanyInfo!A9:E15,2,FALSE)

Note however to make this simpler, we will give cells P7 and the range A9:E15 range names. Range names are useful because they are typically shorter, easier to understand, and if you copy them their references are absolute.

Why Use Index() & Match() when we could have Used Vlookup()?

If you are wondering why we didn't use Vlookup() way back in Exercise 2, there are a couple of reasons:

- I wanted to show you different approaches you can use.
- Vlookup() as we are using it here does not allow you to search by multiple criteria.
**Step A: Using Range Names for P7 and Our Database Range (A8:E15)**

Range names allow you to use a word in a formula to represent a cell address (think of them as variables). We don’t have to use range names for this to work but there will be less room for error if we do.

1. Click the "Dashboard" sheet and then click in cell **P7**.
2. Click in the Name Box, type: **Company** then press enter. *(You just named P7 "Company")*

![Dashboard Sheet](image)

3. Click the "CompanyInfo" sheet and then highlight **A9:E15**.
4. Click in the Name Box, type: **CoData** then press enter. *(You just named the range "CoData")*

![CompanyInfo Sheet](image)
**Step B: Writing the Vlookup()s to Populate the Table**

This section covers the Vlookup()s used to populate our table on the CompanyInfo sheet.

1. Click on the "CompanyInfo" sheet and type formulas shown in the image below.

```
1 =VLOOKUP(Company,CoInfo,1,FALSE)
2 Company Name =VLOOKUP(Company,CoInfo,1,FALSE)
3 Industry =VLOOKUP(Company,CoInfo,2,FALSE)
4 Year Founded =VLOOKUP(Company,CoInfo,3,FALSE)
5 Lead =VLOOKUP(Company,CoInfo,4,FALSE)
6 =VLOOKUP(Company,CoInfo,5,FALSE)
```

Just in case the image is fuzzy, here are the formulas for each cell:
- Cell A1: =VLOOKUP(Company,CoInfo,1,FALSE)
- Cell B2: =VLOOKUP(Company,CoInfo,1,FALSE)
- Cell B3: =VLOOKUP(Company,CoInfo,2,FALSE)
- Cell B4: =VLOOKUP(Company,CoInfo,3,FALSE)
- Cell B5: =VLOOKUP(Company,CoInfo,4,FALSE)
- Cell C2: =VLOOKUP(Company,CoInfo,5,FALSE)

Note that our range names refer to the following cells:
- "Company" is the range name for this address: Dashboard!P7
- "CoInfo" is the range name for this address: CompanyInfo!A9:E15

**Step C: Testing Your Vlookup() Formulas**

If you go to the "Dashboard" tab and select a different company from the radio buttons, your table on the "CompanyInfo" tab should update to reflect information about the company you selected.
Step D: Using the Camera Tool to Display the Table on the Dashboard Sheet

In this section, we will use the Camera Tool to display the company information table on our Dashboard sheet. We could have just created it there in the first place but I decided to use the Camera Tool for three reasons:

- The column widths of my company info table are different than the widths used on my Dashboard sheet.
- I can apply Picture Styles to a Camera object.
- Unlike a picture, Camera objects are linked to their data source. If I change the data in the table on the CompanyInfo sheet, camera created table on the Dashboard sheet will update to match.

Adding the Camera Tool

The Camera Tool comes with Excel but does not appear on any ribbons. Before proceeding, follow the "Workshop Preparation" section at the beginning of this handout to add the Camera tool to one of your ribbons.

2. Click the "Camera" button. (We put it either on your "Developer" tab or on the "Quick Access Toolbar").
3. Click the "Dashboard" sheet.
4. Click where you want the table to appear (try A44).
5. Resize the table as desired.

Step E: Testing the Camera Table

If you go to the "Dashboard" tab and select a different company from the radio buttons, your table on the "CompanyInfo" tab should update AND your table on the "Dashboard" sheet should update as well.
Exercise 7: Adding Buttons to Change All Charts to Lines or Columns

In this section, we will create two macro buttons to change all of the charts on the Dashboard sheet to either Line charts or Column charts.

What the Recorder Returns
If you were to turn on the macro recorder and record yourself changing the PEChart to a line chart, you would get the code shown to the right.

Step A: Alternative One - Creating the Two Macros for Line and Column Charts
Now that we know the code, we could create the two macros below; however, there is a shorter method shown on the next page you might like to use instead.

1. Press ALT + F11 to toggle to the VBA editor.
2. In a module, type both macros shown below. (See the next page for an alternative approach).

Change the Charts to Line Charts

```vba
Sub ChangeChartsToLine()
    ActiveSheet.ChartObjects("PMChart").Activate
    ActiveChart.ChartType = xlLine

    ActiveSheet.ChartObjects("PEChart").Activate
    ActiveChart.ChartType = xlLine

    ActiveSheet.ChartObjects("TopChart").Activate
    ActiveChart.ChartType = xlLine

    ActiveSheet.ChartObjects("EPSChart").Activate
    ActiveChart.ChartType = xlLine

    Range("K16").Select
End Sub
```

Change the Charts to Column Charts

```vba
Sub ChangeChartsToColumn()
    ActiveSheet.ChartObjects("PMChart").Activate
    ActiveChart.ChartType = xlColumnClustered

    ActiveSheet.ChartObjects("PEChart").Activate
    ActiveChart.ChartType = xlColumnClustered

    ActiveSheet.ChartObjects("TopChart").Activate
    ActiveChart.ChartType = xlColumnClustered

    ActiveSheet.ChartObjects("EPSChart").Activate
    ActiveChart.ChartType = xlColumnClustered

    Range("K16").Select
End Sub
```
**Step A: Alternative Two - Coding Using a For Each …Next Loop**

Rather than naming and changing each chart individually, you can tell VBA to change all charts on the sheet by using a *For Each ...Next loop*. This type of loop can be used to affect each object in a collection of objects. Once all of the objects have been affected, the loop terminates. For our purposes, the loop is defined as follows:

For Each *item* In *collection*
    *Lines of code*
Next *item*

**item** – This is a variable name you are assigning each individual item in the collection of objects. It can be any name.

**collection** – This is the object you wish to affect.

**Next** - This causes the next item in the collection to be evaluated.

If you would like to use this approach instead of the one on the previous page:

1. Press **ALT + F11** to toggle to the VBA editor.
2. In a module, type both macros shown below.

<table>
<thead>
<tr>
<th>Sub ChangeChartsToLine()</th>
<th>Sub ChangeChartsToColumn()</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Each Cht In ActiveSheet.ChartObjects</td>
<td>For Each Cht In ActiveSheet.ChartObjects</td>
</tr>
<tr>
<td>Cht.Chart.ChartType = xlLine</td>
<td>Cht.Chart.ChartType = xlColumnClustered</td>
</tr>
<tr>
<td>Next Cht</td>
<td>Next Cht</td>
</tr>
<tr>
<td>End Sub</td>
<td>End Sub</td>
</tr>
</tbody>
</table>

Notes:
- Cht is a variable name we made up that represents the currently selected chat in the loop.
- We really should declare our Cht variable on the 2nd line like so:  *Dim Cht As ChartObject*

**Step B: Testing your Macros**

1. Click the "Dashboard" sheet.
2. From the menu, click the "Developer" tab and then the "Macros" button.
3. Click the "ChangeChartToColumn" macro and then click "Run".  *(All charts on the sheet should be Column.)*
4. Repeat steps 1-3 for the "ChangeChartToLine" macro.  *(All charts on the sheet should be Line.)*
Step C: Making the Button to Run the ChangeChartsToColumn Macro
In this section we will make the button to run the ChangeChartToColumn macro.

1. Click the "Dashboard" sheet.
2. From the menu, click the "Developer" tab.
3. Click "Insert" then the form control "Button". (Not the ActiveX Control one.)
4. Click near cell K16 to place that button in that vicinity.
5. Click the "ChangeChartsToColumn" macro.
6. Click "OK".
7. Right click the button and select "Edit text".
8. Type "Column Charts" for the text on the button.
9. Size / Move the button as desired. *

* If you are in "Design Mode" you should be able to Control + Click the button to select it; otherwise, you will have to right click it to get its grips.

Step D: Making the Button to Run the ChangeChartsToLine Macro
1. Repeat the steps above to make a button for the "ChangeChartsToLine" macro.

Step E: Test the Buttons
1. Click each button. 
   (One should change all charts to Column Charts and the other should change all charts to Line Charts.)
Exercise 8: Creating a Macro Button to Print the Dashboard

In this final exercise, we will create a button to print out our Dashboard. Fortunately, the Macro Record is pretty good at capturing print settings so we will use it. Before we turn on the recorder, some advice:

- Use Absolute Referencing when recording.
- Make sure you click the "Dashboard" tab even though you might be already on it.
- Even though some setting may already set to how you intend to print, record yourself clicking them anyway. This will ensure that everything prints how you want it to in case someone changed your regular settings.

Step A: Record Printing
1. Click the "Dashboard" tab.
2. From the menu, click the "Developer" tab.
3. Make sure "Use Relative References" is NOT highlighted.
4. Click "Record Macro".

5. Name the macro: PrintDashboard
6. Store it in: "This Workbook".
7. Click "OK" to begin recording.

(Be very careful what you click.)

8. Click the "Dashboard" tab.
9. Highlight the area of your dashboard you would like to print. (Typically A1:K55)

10. From the menu, click the "Layout" tab.

11. Click "Print Area" then "Set Print Area".

12. Click a cell to un-highlight.

13. Click "Orientation" then "Portrait".

![Record Macro](image1.png)
![Use Relative References](image2.png)

![Dashboard Tab](image3.png)
![Print Area](image4.png)
![Set Print Area](image5.png)
![Clear Print Area](image6.png)

![Print Screen](image7.png)
14. Set Scale to Fit to "1 Page" for both Width and Height.

15. Click "Margins" and select the desired margins.

16. Make any other desired layout settings.

17. Click "File – Print".

18. Make any desired printer settings. *
19. Click "Print".

* Note that you probably won't be able to print from the workshop computers in Marshall's labs.

20. Click back on the "Developer" tab.
21. Click "Stop Recording".

**Step B: Making a Button to Run the PrintDashboard Macro**
1. Click the "Dashboard" sheet.
2. From the menu, click the "Developer" tab.
3. Click "Insert" then the form control "Button". (NOT the ActiveX Control one.)
4. Click near cell K11 to place that button in that vicinity.

5. Click the "PrintDashboard" macro.
6. Click "OK".
7. Right click the button and select "Edit text".
8. Type "Print Report" for the text on the button.
9. Size / Move the button as desired. *

* If you are in "Design Mode" you should be able to Control + Click the button to select it; otherwise, you will have to right click it to get its grips.

---

**Exercise 9: Format the Report as Desired**

**Title the Report**
2. Type the following formula: P7&" Financial Report
   (Because we named P7 "Company", you can also type: =Company&" Financial Report)

**Format the Report**
Feel free to format your report as desired but be very cautious of the following:

- Avoid inserting or deleting columns and rows. A lot of our code refers to specific cell addresses and if they change, the code could mess up (especially the macros). (Using range names rather than cell addresses in macros is a good practice because it avoids this problem.)
- Don’t change the names of your sheets.
- Avoid moving cells that contain references that macros, vlookup, match(), or index() use.