Profit and Loss Data Modeling and Analysis with Microsoft PowerPivot in Excel

Every day, business professionals use Microsoft® PowerPivot in Excel® to create sophisticated data models and perform powerful, ad-hoc data analysis. Excel PivotTables and Power View reports further extend PowerPivot data models with dynamic visualizations providing new insights into data.

This article, along with the ContosoPnL.xlsx sample Excel workbook can provide business, accounting, and finance professionals help with data modeling in PowerPivot and creating dynamic reports for profit and loss analysis.

The information and samples referenced here are real-world, created by users just like you. The best information about how to use PowerPivot comes from those who use it every day to solve real-world problems. This is Community!

Writer:

Owen Duncan, Senior Writer, SQL Server BI, Microsoft Corp.

Contributors:

Howie Dickerman, Senior Program Manager, SQL Server BI, Microsoft Corp.

Scott Estes, Group Finance Manager, Microsoft Corp.

Copyright

Contents

Overview

This article describes how finance professionals at Contoso Ltd. use PowerPivot for self-service, ad-hoc profit and loss data analysis.

Along with the ContosoPnL Excel workbook, information provided here can help you understand how to create powerful and efficient PowerPivot data models to analyze key profit and loss metrics, such as:

- Aggregate, or value measures such as sums and averages for currency and headcount.
- Comparison measures such as year-over-year, year-to-date, and variance for currency and headcount.
- Performance, or ratio measures such as percentage, cost per head, and rate and volume variances. All of which calculate on different combinations of aggregate and comparison measures.

When reading through this article, be sure to use the ContosoPnL workbook alongside. There are two versions of the ContosoPnL workbook, one for Excel 2010, and one for Excel 2013. While both versions include the PowerPivot data model and a number of PivotTables already created, the Excel 2013 version also includes several example Power View sheets to show even more ways to explore data.

This article and the sample workbook were created by Business Intelligence professionals at Microsoft with extensive input from the finance and accounting community. Known best practices are used to create a simple, highly efficient data model that can scale to suit individual users, and both small and large organizations.

Prerequisites

Microsoft Excel 2010 with the PowerPivot for Excel Add-in enabled.

Or

Microsoft Excel 2013 with the PowerPivot in Excel and Power View Add-ins enabled.

Or

Microsoft SQL Server Analysis Services running in Tabular mode and SQL Server Data Tools (SSDT) or SSDT-BI for Visual Studio 2012. To use the sample workbook, use the Import from PowerPivot project template to import the ContosoPnL workbook into a Tabular model project.

You Should Already Know

You should already be familiar with how to use Excel PivotTables, PowerPivot in Excel, and Power View. This article and the samples are not intended to be a starting point for learning how to use these tools. You will save yourself a lot of frustration if you already understand fundamental concepts in PowerPivot such as data models, imported and linked data, relationships, calculated columns, measures, and perspectives. In Power View, you should be familiar with creating visualizations, selecting fields, and using filters.

> *IMPORTANT:* The information and examples provided here should in no way be assumed to be the only way to create a PowerPivot data model and analyze data. This article and the provided samples represent one approach to creating a data model and analysis that best suits the particular data in the ContosoPnL Access database.

There are many excellent resources to help you learn about PowerPivot and Power View. Be sure to refer to the Additional Resources section at the end of this article for more information about the concepts and features described here.

Samples

A ContosoPnL Excel sample workbook is included with this article. Also available as a separate download is the ContosoPnLData Access database. The database is not necessary to use the ContosoPnL workbook. All of the data in the ContosoPnLData Access database has already been imported into the ContosoPnL workbook's PowerPivot data model. The database is provided as a separate download in case you want to create your own test model and use it as a data source.

> *Note:* In order to keep the sample size to a minimum, only a small amount of data for a limited number of months and fiscal years is provided.

ContosoPnL Access database

The ContosoPnL Access database contains three tables: DimAccounts, DimExecGeography, and FactFinanceData.

FactFinanceData is a detailed listing of monthly account balances for actual, budget, forecast, actual headcount, budget headcount, and forecast headcount. DimAccounts and DimExecGeography provide account and profit center data.

ContosoPnL Excel workbook

The ContosoPnL Excel workbook contains the PowerPivot data model with data imported from the ContosoPnL Access database and an additional Date table. The workbook does not contain any Excel

worksheets that are used as a data source. The only worksheets in the workbook are for sample PivotTables. All of the data exists in the PowerPivot data model.

Tables

The ContosoPnL PowerPivot data model contains the following tables:

Finance Data – Imported from the FactFinanceData table in the ContosoPnLData Access database. The Finance Data table contains all of the fact data. All measures are created in this table. This table includes the following fields (columns):

> **[Fiscal Month]** – Month and year (i.e. July, 2013) in text data type. This column is hidden from client tools.
>
> **[Date]** – A calculated column which derives a date/time value in Date data type from [Fiscal Month]. This column is related to the [Date] column in the Date table. This column is hidden from client tools.
>
> **[Profit Center]** – Profit center number. This column is related to the [Profit Center] column in the Executive Geography table. This column is hidden from client tools.
>
> **[Account]** – Account number. Each account can have multiple profit centers. This column is related to the [Account] column in the Accounts table. This column is hidden from client tools.
>
> **[Actual]** - Monthly *actual* amounts in dollars for each profit center.
>
> **[Budget]** - Monthly *budget* amounts in dollars for each profit center.
>
> **[Forecast]** - Monthly *forecast* amounts in dollars for each profit center.
>
> **[Actual People]** – Month end *actual* number of employees for each profit center of each people account.
>
> **[Budget People]** - Month end *budget* number of employees for each profit center of each people account.
>
> **[Forecast People]** - Month end *forecast* number of employees for each profit center of each people account.

Accounts – Imported from the DimAccounts table in the ContosoPnLData Access database contains the following columns:

> **[Account]** – A unique column which contains one account number for each row. This column is related to the [Account] column in the Finance Data table.
>
> **[Line Item]** – The type of revenue or expense associated with each account.
>
> **[Group]** – The type of group associated with each account.
>
> **[Class]** – The class associated with each account. All accounts are either Expenses or Net Revenue.
>
> **[Sub Class]** – A sub classification of Class which describes the type of cost.

Executive Geography – This table, imported from the DimExecGeography table in the ContosoPnLData Access database contains the following columns:

[Profit Center] – A unique column which contains one profit center number for each row. This column is related to the [Profit Center] column in the Finance Data table.

Additional columns provide further classification of each profit center and can be used to slice metrics on different dimensions, but do not merit further definition here.

Date – A single column of dates with range from 7/1/2008 through 6/30/2015. A date table with a single column of daily dates from 7/1/2008 through 6/30/2015 was created in Excel. This column was then copied and pasted as a new table into PowerPivot. Four additional calculated columns were created to support filtering on fiscal year, fiscal quarter, fiscal month, and month.

[Date] – A unique column of Date data type contains one day per row from 7/1/2008 through 6/30/2015. This column is specified as the unique identifier for the Date table in Mark as Date Table to support working with DAX time intelligence functions. This column is hidden from client tools.

[Fiscal Year] – A calculated column of Text data type which derives a fiscal year in yyyy format. This field can then be used as a dimension to filter on fiscal year.

[Fiscal QTR] – A calculated column of Text data type which derives a fiscal quarter in yyyy-q# format. This field can then be used as a dimension to filter on fiscal quarter.

[Fiscal Month] – A calculated column of Text data type which derives a fiscal month in yyyy-p## format. This field can then be used as a dimension to filter on fiscal month.

[Month] – A calculated column of Text data type which derives a month in ##-mmm format. Along with Fiscal Year, this field can be used as a dimension to filter on year and month.

Note: Slicing or filtering on Fiscal Month is effectively the same as slicing or filtering on both Fiscal Year and Month.

Relationships

The ContosoPnL data model has the following relationships:

Table	Related Lookup Table
Finance Data[Account]	Accounts[Account]
Finance Data[Profit Center]	Executive Geography[Profit Center]
Finance Data[Date]	Date[Date]

Calculations

The ContosoPnL data model contains four calculated columns in the Date table to support time-intelligence functions and slicing and filtering on dates. The Finance Data table includes one calculated column, Date, formatted to match values in the related Date column in the Date table.

There are 79 measures, nine of which are hidden from client tools. All measures are in the Finance Data table. Hidden measures provide interim calculations that support other measures. Many of the measures are described in detail later in this article. All measure formulas are provided in the Appendix section of this article.

> *Note:* In PowerPivot, the names measure and calculated field are synonymous. Measure, in PowerPivot for Excel 2010, was renamed Calculated Field in PowerPivot in Excel 2013. We are using the name measure here.

Perspectives

The ContosoPnL workbook has the following perspectives defined in PowerPivot:

- Budget
- Forecast
- Year-over-Year (YoY)

When no perspective, or *Default Perspective*, is selected, all fields and measures are shown in the Field List.

> *Tip:* Perspectives that include only a subset of tables, fields, and/or measures is a great way to define particular views of data, so users can focus on a particular type of analysis. When a perspective is selected in the Field List, only those tables, fields, and measures defined for the perspective are shown, making it much easier to navigate.

Working with Dates

Creating a Date table in your data model is essential to evaluating data over time. There are many different ways to create a Date table; for example, if the database you import your finance data from also has a date table, you can import it into your data model. Or, you can import date data from DateStream on Windows Azure DataMarket. You can also create a date table in Excel, and then copy and paste the date data into a new date table in PowerPivot.

If your data source regularly deletes old data and adds new data as time goes by, and your data model will refresh source data regularly, be sure to import or create a Date table in your data model that also refreshes to match the dates in your data.

Creating a Date table in PowerPivot by using Excel

Most finance data, like the data in the FactFinanceData table in the ContosoPnL Access database, includes a date column with a year and month value for each row. In order for us to be able to evaluate our finance data by year, month, or quarter, we need to create a Date table in our data model that we can then relate to the dates in the finance data.

Creating a Date table in PowerPivot is really quite simple. We must first determine what type of dates are included in our source data, and how our new Date table can relate to it. In the ContosoPnL PowerPivot data model, each row in the Finance Data table includes a date in the Fiscal Month column. Each date is in the format of month, year. The data type is text.

Now that we know what type of dates we need to relate our new Date table to, we can create a date table in Excel. Most important is that we have a date range that covers all of the potential dates in the Finance Data table.

In order to create a new Date table in the data model, in Excel, we create a single column named Date with dates ranging from 7/1/2008 through 6/30/2015. We set the data type as Date and we format as a table with headers.

Date column in Excel table

This column is then copied and pasted as a new table into PowerPivot. We then create four additional calculated columns to support slicing and filtering on fiscal year, fiscal quarter, fiscal month, and month.

Our first new column calculates a fiscal year from the values in the Date column. Our formula takes the dates from the Date column and formats a return value that we can use, for example FY13. Our fiscal year formula looks like this:

```
="FY"&RIGHT(INT((MONTH([Date])-1)/6)+YEAR([Date]),2)
```

Similarly, we can create a column that calculates and returns fiscal year and fiscal quarter with format FY13-Q1. Our Fiscal Qtr formula looks like this:

```
=[Fiscal Year] & "-Q" & FORMAT(INT((MOD(MONTH([Date])+5,12)+3)/3 ),"0")
```

For fiscal month, which we too format as month periods like FY13-P07, looks like this:

```
=[Fiscal Year] & "-P" & FORMAT(MOD(MONTH([Date])+5,12)+1,"00")
```

Finally, we create a column that calculates a month number and month name, like 01-Jul:

```
=FORMAT(MOD(MONTH([Date])+5,12)+1,"00") & "-" & FORMAT([Date],"mmm")
```

Note: Adding Fiscal Month to Report Filters or Slicers is in effect the same as adding Fiscal Year and Month. The difference is in how many items appear in the Report Filter listbox, or how many buttons appear in the Slicer.

Of these five columns we now have in the Date table, the Date column is the most important. All of the other columns are merely different representations of the dates in the Date column.

We need to select the Date table as Mark as Data Table (on the Design tab) and the Date column as the unique identifier for the Date table. This is necessary when using DAX Time Intelligence functions.

Mark as Date Table function in PowerPivot

Now that we have our Date table, we need to create a relationship between it and the Finance Data table. If we look at our Finance Data table we see there is only one column with dates, Fiscal Month. Unfortunately, the values in this column are simply a month name followed by the year, and they are in Text data type.

Fiscal Month column in the Finance Data table

We can't create a relationship between the Finance Data table and the Date table using this column. We can however create a new calculated column with the dates we need. We can use Excel's DATEVALUE function to format a new date value from the month and year text values in Fiscal Month, like this:

```
=DATEVALUE([Fiscal Month])
```

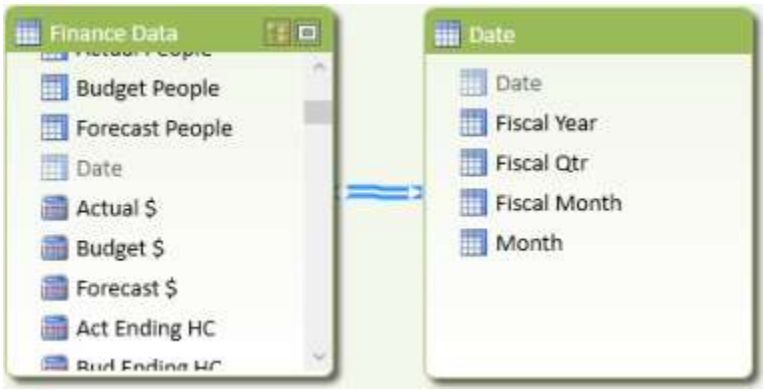
We now have two columns with dates in the Finance Data table.

Fiscal Month column and Date calculated column in the Finance Data table

Fiscal Month	Date
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM
January, 2012	1/1/2012 12:00:00 AM

After verifying the Data Type is Date, we can create a relationship between this new Date column in the Finance Data table and the Date column in the Date table.

Date relationship in PowerPivot Diagram View



Because the Fiscal Month column and our new Date column in the Finance Data table don't provide a date we can use to slice or filter on in PivotTables, we can hide both of them from client tools.

> *Resource:* Howie Dickerman, Program Manager at Microsoft, provides a great description and best practices for using DAX time intelligence functions in his blog post Time Intelligence Functions in DAX.

Currency Metrics

Calculating dollar amounts for a particular time period, organization, account, or geographical location are often the basis for any financial report. This section describes how to create measures that support basic dollar and percent calculations such as sum, year-over-year, year-to-date, and variance.

If we want to focus on currency metrics only, excluding any headcount and cost per head metrics, we must first create three very simple aggregation measures. We will call these *base* measures because they also provide interim calculations that we will use in many other measures.

> *Tip:* A best practice when creating measures in PowerPivot is to write once and re-use. Creating measures that can be used in other measures will save you a lot of time and make your measures and data model more efficient.

The most fundamental of all measures we create is the sum of all values in the Actual column in the Finance Data table:

```
Actual $:=SUM([Actual])
```

Similarly, we can create sum formulas for Budget:

```
Budget $:=SUM([Budget])
```

And Forecast:

```
Forecast $:=SUM([Forecast])
```

Now that we have these three important base measures, and we have our Date table, we can create a PivotTable with some meaningful analysis.

If we add Fiscal Year to Row Labels, and Actual $, Budget $, and Forecast $ measures to Values, we see:

Row Labels	Actual $	Budget $	Forecast $
FY12	$1,633,020,521	$0	$0
FY13	$626,511,723	$1,596,728,987	$1,695,366,594
Grand Total	**$2,259,532,244**	**$1,596,728,987**	**$1,695,366,594**

Notice there are values calculated only for FY12 and FY13? You may remember the Date table has months for fiscal years 2008 through 2015. But, because our Finance Data includes data only for fiscal years 2012 and 2013, only values for those years is returned. And, because our data does not include any values other than $0 for Budget or Forecast for fiscal year 2012, $0 is returned for those years.

From these three measures, along with the Date column from the Date table, we can create many of the other measures we need to analyze budget metrics.

For example, to create a measure that calculates the sum of Actual for the previous year, we use the CALCULATE function with the Actual $ measure we already created as its first argument, and then use the DATEADD function to filter, or shift, dates one year prior:

```
Prior $:=CALCULATE([Actual $], DATEADD('Date'[Date],-1,YEAR))
```

If we change our PivotTable to include Actual $ and Prior $ in Values, and leave Fiscal Year on Rows, we see:

Row Labels	Actual $	Prior $
FY12	$1,633,020,521	
FY13	$626,511,723	$1,633,020,521
FY14		$626,511,723
Grand Total	**$2,259,532,244**	**$2,259,532,244**

Notice Prior $ for FY13 is the same value as Actual $ for FY12, and Prior $ for FY14 is the same as Actual $ for FY13. We have no Actual data for FY11, and since we are not yet in FY14, we have no Actual data there either, so blank is returned for those periods.

> *Note:* Similar to DATEADD, we could use SAMEPERIODLASTYEAR or PARALLELPERIOD functions to return a set of dates shifted back one year. The difference is DATEADD allows us to specify the number, -1, and period, Year, to shift.

Now that we have measures that aggregate sum values for the current period and for the same period the previous year, we can begin creating other measures that compare and calculate ratios.

Calculate Year-Over-Year

Year-over-year is a measure of growth. It's really a simple calculation of Actual minus Prior. A positive result reflects an increase in actual, and a negative result reflects a decrease in actual.

To calculate year-over-year, we need a measure that subtracts Prior $ from Actual $. However, we need to modify our formula to account for the fact that revenue accounts normally have credit (negative) amounts while expenses are debit (positive) amounts.

Let's first look at how we can calculate year-over-year for expense accounts

For expenses, if Actual $ is higher than Prior $, our result should reflect a positive number for an increase in growth:

Expense account

Actual $	Prior $	YoY $
100,000	50,000	50,000

If Actual $ is lower than Prior $, our result should reflect a negative number for a decrease in growth:

Expense account

Actual $	Prior $	YoY $
25,000	50,000	-25,000

For expense accounts, a formula that subtracts Prior $ from Actual $, ([Actual $]-[Prior $]), does the job just fine. However, for revenue accounts, this simple formula won't really work.

In our data, revenue is stored as a credit. It's a negative number. A revenue value of $(100,000) means we really had $100,000 in revenue.

Because values are stored as negative numbers, we can't use a simple ([Actual $] - [Prior $]) calculation in our formula. For example, if we have revenue of $(100,000) this year and $(50,000) last year, our formula would be:

$(100,000)
-
$(50,000)

$(50,000)

This gives us a result like this:

Revenue account

Actual $	Prior $	YoY $
-100,000	-50,000	-50,000

Because this is for a revenue account, we don't really want our result to be negative. Remember, our revenue values are stored as credits, or negative numbers. Our result, 50,000, is actually an increase in revenue, so we really want to show this as a positive number. This requires a very slight, and difficult to see change to the calculation in our formula: -([Actual $] - [Prior $]).

Do you see it? All we have to do is add a minus sign in front of our calculation. But, of course, this won't work for an expense account.

We can solve this by using a CALCULATE function in our formula to filter on account class, like this:

```
YoY $:=
        IF(
           CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Actual $]-[Prior $]),
        ([Actual $]-[Prior $])
)
```

Let's look at this formula more carefully.

First, we create a formula that calculates the difference between Actual and Prior.

([Actual $]-[Prior $])

We want to test if this is for a revenue account or an expense account. We can use an IF function to test for a logical condition, like this,

IF(
CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Actual $]-[Prior $]),
([Actual $]-[Prior $])

This test states:

For each row in Finance Data, if the value in the Class column in the Accounts table contains "Net Revenue", then return a result by subtracting values from Actual $ by values from Prior $, and put a minus sign in front of it. If the value in the related Class column does not contain "Net Revenue", return the result.

Year-over-year can also be expressed as a ratio in percent. Now that we have our YoY $ measure, we can easily create another measure to calculate year-over-year in percent:

```
YoY %:=IF([Prior $], [YoY $]/ABS([Prior $]),BLANK())
```

To prevent division by 0, we use an IF function to test if the result of the divisor, Prior $, is not 0. If the condition is true (any non-zero value will be interpreted as true), then divide YoY $ by the absolute (ABS) value of Prior $. If the condition is false (or zero, which is interpreted as false), then return a blank.

Let's look at our new measures in a PivotTable with some real numbers. If we add Class and Sub Class to Row Filters, Fiscal Year to Report Filter and select FY13, then add Actual $, Prior $, YoY $, and YoY % to Values, we see:

Note: Our data includes data for only the first half of FY13.

Fiscal Year	FY13

Row Labels	Actual $	Prior $	YoY $	YoY %
Expenses	**$626,721,063**	**$1,633,147,683**	**($1,006,426,620)**	**-61.6 %**
Hosting & Online Infrastructure	$218,600	($418,711)	$637,311	152.2 %
Manufacturing & Distribution	$0	$5,021,091	($5,021,091)	-100.0 %
Marketing	$5,728,550	$24,650,925	($18,922,375)	-76.8 %
Non-People Expenses	($39,170,879)	($77,353,068)	$38,182,189	49.4 %
People	$450,498,200	$1,164,917,899	($714,419,699)	-61.3 %
Procured People	$209,446,645	$516,329,549	($306,882,904)	-59.4 %
Revenue Driven	($39)	($0)	($38)	-8845.2 %
Royalties	($14)		($14)	
Net Revenue	**($7,525)**	**$485,033**	**$492,558**	**101.6 %**
Billed Revenue	($210,109)	($238,471)	($28,362)	-11.9 %
Deferred Revenue	$210,760	$34,711	($176,048)	-507.2 %
Finished Goods Revenue	$2,024	$50,708	$48,684	96.0 %
Product Revenue Adj	$0	$1,239,893	$1,239,893	100.0 %
Services Revenue	($10,198)	($601,807)	($591,609)	-98.3 %
Grand Total	**$626,713,539**	**$1,633,632,717**	**$1,006,919,178**	**61.6 %**

Since calculations for Budget year-over-year and Forecast year-over-year are similar, we can easily create measures for budget year-over-year:

```
Bud YoY $:=
        IF(
           CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Budget $]-[Prior $]),
        ([Budget $]-[Prior $])
)
```

And forecast year-over-year:

```
Fcst YoY $:=
        IF(
           CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Forecast $]-[Prior $]),
         ([Forecast $]-[Prior $])
)
```

Other year-over-year measures we create are:

- Bud YoY %
- Fcst YoY %

Formulas for these measures are provided in the Appendix.

Calculate Variance

Variance to Budget

We can create variance measures using formulas similar to year-over-year. We need only change the measures specified as to what we want to subtract and what we want to subtract from. We also don't need to test for account class, so our formula is even more simple. To calculate variance to budget, our formula looks like this:

```
VTB $:=[Budget $]-[Actual $]
```

And to calculate variance to budget in percent:

```
VTB %:=IF(
            [Budget $], [VTB $]/ABS([Budget $]),
          BLANK()
)
```

If we create a simple PivotTable with Fiscal Year on Row Labels, and Actual $, Budget $, VTB $, and VTB % in Values, we see:

Row Labels	Actual $	Budget $	VTB $	VTB %
FY12	$1,633,020,521	$0	$1,633,020,521	
FY13	$626,511,723	$1,596,728,987	($970,217,264)	-60.8 %
Grand Total	**$2,259,532,244**	**$1,596,728,987**	**$662,803,257**	**41.5 %**

Variance to Forecast and Forecast Variance to Budget

To create a variance to forecast, we only have to change [Budget $] to [Forecast $], like this:

```
VTF $:=[Forecast $]-[Actual $]
```

And to calculate forecast variance to budget:

```
Fcst VTB $:=[Budget $]-[Forecast $]
```

Other variance measures we create are:

- VTF %
- Fcst VTB %

Formulas for these measures are provided in the Appendix.

Calculate Year-to-Date

When analyzing data on a month level, you might want to calculate a result that includes a starting balance from the beginning of a period, like fiscal year, up to a later period in time. Calculating year-to-date is quite simple when using DAX time intelligence functions. In fact, DAX has the TOTALYTD function for just such a measure.

Let's return to our first measure, Actual $, which is the sum of all dollar values in the Actual column, depending on the filter context.

If we create a simple PivotTable and add Fiscal Year to Report Filter and select FY13 only, and then add Month to Row Labels, and Actual $ to Values, we see:

Fiscal Year	FY13

Row Labels	**Actual $**
01-Jul	$111,927,774
02-Aug	$71,913,423
03-Sep	$141,548,940
04-Oct	$130,550,235
05-Nov	$135,520,342
06-Dec	$35,051,009
07-Jan	$0
08-Feb	$0
09-Mar	$0
10-Apr	$0
11-May	$0
12-Jun	$0
Grand Total	**$626,511,723**

By using the TOTALYTD function, we can create a measure that calculates a sum total for Actual from the beginning of the fiscal year, up through the last whole period specified in the filter context. Our measure formula looks like this:

```
YTD Actual $:=TOTALYTD([Actual $], 'Date'[Date], ALL('Date'), "6/30")
```

Let's look at this formula more carefully.

First, we create a formula that returns a year-to-date sum total of Actual:

TOTALYTD([Actual $], 'Date'[Date]**)**

In order to calculate the sum of Actual over a period, we must specify a date column required by the TOTALYTD function. We use the Date column in the Date table we created:

TOTALYTD([Actual $], **'Date'[Date]**

Next, we specify ALL('Date') as a filter argument to clear any filters that may be applied and return all rows in the Date column:

TOTALYTD([Actual $], 'Date'[Date], **ALL('Date')**

Finally, we specify the year end date. This tells the TOTALYTD function what date to start from. The default is 12/31, but in our case, we want to calculate over a fiscal year, so we specify 6/30:

TOTALYTD([Actual $], 'Date'[Date], ALL('Date'), **"6/30"**)

Now, if we add YTD Actual $ to our PivotTable, we see:

Fiscal Year	FY13

Row Labels	**Actual $**	**YTD Actual $**
01-Jul	$111,927,774	$111,927,774
02-Aug	$71,913,423	$183,841,197
03-Sep	$141,548,940	$325,390,137
04-Oct	$130,550,235	$455,940,372
05-Nov	$135,520,342	$591,460,714
06-Dec	$35,051,009	$626,511,723
07-Jan	$0	$626,511,723
08-Feb	$0	$626,511,723
09-Mar	$0	$626,511,723
10-Apr	$0	$626,511,723
11-May	$0	$626,511,723
12-Jun	$0	$626,511,723
Grand Total	**$626,511,723**	**$626,511,723**

Notice the values returned in YTD Actual $ for each month are the sum of each prior month in fiscal year 2013. For example, the YTD Actual $ for 05-Nov ($596, 460, 714) is the sum of Actual $ for 01-Jul through 05-Nov.

Here again, we can create similar measures for year-to-date budget and year-to-date forecast. We only have to copy our formula, change the measure name, and the expression measure specified.

Other year-to-date measures we create are:

- Prior YTD Actual $
- YTD Budget $
- YTD Forecast $

Formulas for these measures are provided in the Appendix.

DAX time intelligence also includes TOTALQTD and TOTALMTD functions. TOTALMTD is not really useful when analyzing data on a month level like we are doing here, but TOTALQTD may be useful if you need to aggregate values from the beginning of a quarter.

Create Your Own:

In this quick lesson, you create a quarter to date measure.

1. In the Finance Data table, in the Measure Grid, click on the empty cell below Prior YTD Actual $.
2. In the formula, type the following formula, and then press enter.

```
QTD Actual $:=TOTALQTD([Actual $], 'Date'[Date],ALL('Date'))
```

3. Right click on the new measure, then click Format, and then select Currency, Decimal Places = 0.
4. Create a PivotTable like the one above, and then add your new measure to see the results.

Budget Remaining and Budget Attainment

Other useful measures that calculate on year-to-date are budget remaining, which calculates the full year budget minus year-to-date actual:

```
Budget Remaining $:=CALCULATE([Budget $], ALL('Finance Data'[Date]))-[YTD Actual $]
```

And budget attainment, which calculates the amount of full year budget in percent we have already spent:

```
Budget Attain %:=IF(
                    [Budget $], [YTD Actual $]/CALCULATE([Budget $],
                        ALL('Finance Data'[Date])),
                    BLANK()
                    )
```

If we add Budget Remaining $ and Budget Attain % to our PivotTable, we see:

Fiscal Year	FY13

Row Labels	Budget $	YTD Actual $	Budget Attain %	Budget Remaining $
01-Jul	$122,732,561	$111,927,774	91.20 %	$10,804,787
02-Aug	$65,776,009	$183,841,197	279.50 %	($118,065,188)
03-Sep	$146,391,213	$325,390,137	222.27 %	($178,998,924)
04-Oct	$136,686,794	$455,940,372	333.57 %	($319,253,578)
05-Nov	$135,740,529	$591,460,714	435.73 %	($455,720,185)
06-Dec	$137,601,100	$626,511,723	455.31 %	($488,910,622)
07-Jan	$229,466,959	$626,511,723	273.03 %	($397,044,764)
08-Feb	$107,120,311	$626,511,723	584.87 %	($519,391,412)
09-Mar	$75,022,491	$626,511,723	835.10 %	($551,489,231)
10-Apr	$130,400,150	$626,511,723	480.45 %	($496,111,573)
11-May	$127,136,562	$626,511,723	492.79 %	($499,375,161)
12-Jun	$182,654,307	$626,511,723	343.00 %	($443,857,415)
Grand Total	**$1,596,728,987**	**$626,511,723**	**39.24 %**	**$970,217,264**

Other measures we create are:

- Forecast Remaining $
- Forecast Attain %

Formulas for these measures are provided in the Appendix.

Headcount Metrics

Headcount (HC) metrics help us analyze the number of employees in the company at the end of a period. Headcount data is primarily used to track changes in the employee base and is a key driver in the total employee cost.

Much like with the measures we create for currency metrics, most headcount measures can be derived from a number of base measures. However, unlike our base dollar measures where we could simply sum values for Actual, Budget, and Forecast, to create our base measures for headcount we need a couple of additional measures. These are ending headcount and average headcount. Once we have these, we can then create these same base measures for prior year, budget, and forecast. Then we can create year-over-year, and variance measures much like we did for currency metrics. We also need a measure that calculates a total headcount over time.

> *Resource:* We are starting to delve into more complex DAX formulas, especially when it comes to understanding filter context and how it applies when pivoting on different data. Rob Collie, in his book DAX Formulas for PowerPivot (http://www.powerpivotpro.com/the-book/), provides *the most clear understanding* of how PowerPivot evaluates filter context in Chapter 7. The "Golden Rules" of DAX Measures.

Calculate Ending Headcount

We are going to start with ending headcount. A little later, you will see why this is our most important headcount measure.

Unlike a typical sum calculation, which returns an aggregation of all values in a column regardless of the of period, headcount is much like product inventory, in that at the end of a period, a fiscal quarter for example, the number of employees is not the sum total of all three months in the quarter. It is, in-fact, the number of employees at the end of the last month in the quarter.

Let's take a look at a simple PivotTable where Actual People is added to Values:

Fiscal Year	FY13

Row Labels	Sum of Actual People
FY13-Q1	**13363**
01-Jul	4460
02-Aug	4495
03-Sep	4408
Grand Total	**13363**

By simply adding Actual People to Value, we create an implicit measure that simply sums all of the values in the Actual People column. As you can see, it's doing its job, but at the end of FY13-Q1, we don't really have 13,363 employees, so Sum of Actual People isn't really giving us the result we need.

What we do have is 4, 408 employees, which is the real ending headcount for the FY13-Q1. So, what we need is a measure that gives us the actual ending headcount for any given period.

To calculate the actual ending headcount for any period in a given filter context, we need to be a bit more creative when writing our formula. We can describe the result we need by using our everyday language. In effect, we need to:

> *Return the sum of Actual People for each month, but for the quarter, return the result from the last month of the quarter, and for the fiscal year, return the result from the last month of the fiscal year.*

We can create a formula that does just this:

```
Act Ending HC:=
CALCULATE(
        SUM('Finance Data'[Actual People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Actual People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Actual People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

This formula looks rather intimidating, so let's look at it more carefully.

First, we create a formula that returns the sum of the Actual People column, calculated for all accounts:

> **CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))**

We want to test if this is 0 or a non-zero number. If it is zero, we want to return BLANK(). If it is non-zero, we want to return the sum. This makes the formula look like this:

```
IF(
   CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))=0,
   BLANK(),
   CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))
)
```

Now, we want to find the last date for which this sum is not blank. In other words, what is the last month in which we have non-zero number of people. For this we use the LASTNONBLANK function, like this:

```
LASTNONBLANK(
    'Finance Data'[Date],
    IF(
      CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))=0,
      BLANK(),
      CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))
    )
)
```

Now that we have the last date for which we have non-blank sum of people, we want to calculate the sum of people on that date.

To do this, we use the LASTNONBLANK result as a filter argument in CALCULATE, like this:

```
CALCULATE(
   SUM('Finance Data'[Actual People]),
   LASTNONBLANK(
      'Finance Data'[Date],
      IF(
         CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))=0,
         BLANK(),
         CALCULATE(SUM('Finance Data'[Actual People]),ALL(Accounts))
      )
   ),
   ALL(Accounts)
)
```

Now, if we change our PivotTable, removing Actual People from Values, and replacing it with our new Act Ending HC measure, we get the results we want.

Fiscal Year	FY13

Row Labels	Act Ending HC
FY13-Q1	**4,408**
01-Jul	4,460
02-Aug	4,495
03-Sep	4,408
Grand Total	**4,408**

Creating a formula like this can be confusing. It helps if you start with what you need most, in this case, sum of Actual People. You can then build out your formula taking into account what the variables in the data might be; like we did here by using a conditional IF function. You can then apply any filters needed to return the result you need.

Just like with many of the other base measures we have created, we only need to make slight changes to make this formula work for budget ending headcount,

```
Bud Ending HC:=
CALCULATE(
        SUM('Finance Data'[Budget People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Budget People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Budget People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

And forecast ending headcount,

```
Fcst Ending HC:=
CALCULATE(
        SUM('Finance Data'[Forecast People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Forecast People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Forecast People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

To create an actual ending headcount for the prior year, we simply need to take the Act Ending HC measure, and shift back one year prior:

```
Prior Ending HC:=CALCULATE([Act Ending HC], DATEADD('Date'[Date],-1,YEAR))
```

Calculate Average Headcount

We also need a measure that calculates average headcount. This is the average monthly headcount of any given selection of months. If the selection is one quarter, then the average headcount is the sum of the monthly headcount amounts divided by the number of months in the selection. We will also use this measure as an argument to a number of other measures we will need.

This formula returns the average number of actual ending headcount by using the AVERAGEX function to calculate a mean average of actual ending headcount over all unique periods.

```
Act Avg HC:=AVERAGEX( VALUES('Finance Data'[Fiscal Month]), [Act Ending HC])
```

Let's look at this formula more carefully.

First, we want to calculate the mean average of actual ending headcount for all rows in the Finance Data table. For this, we use the AVERAGEX function with the Finance Data table in the table argument, and Act Ending HC as our expression to be evaluated,

AVERAGEX('Finance Data', [Act Ending HC])

This, however, won't give us the result we need because it is including every row in the Finance Data table.

The AVERAGEX function requires a table or a table expression as its first argument. If we specify the Finance Data table in our table argument, our expression, [Act Ending HC], is evaluated for each row in the table, and the mean average is calculated from that. Remember, Act Ending HC is not just a sum total of all people in the Actual People column. It is the ending headcount for a period. So, what we need is calculate our mean average of ending headcount over all unique periods.

Instead of specifying the Finance Data table as our table, we can use the VALUES function to return a single column table with the distinct count of unique values from the [Fiscal Month] column, like this:

AVERAGEX(**VALUES('Finance Data'[Fiscal Month])**, [Act Ending HC])

Now we have our mean average headcount for all unique periods in the Finance Data table.

Again, we can create average headcount measures for budget and forecast using pretty much the same formula, with only changes in the name and expression. You're starting to get the idea here.

Other measures we create are:

- Bud Avg HC
- Fcst Avg HC
- Prior Avg HC

Formulas for these measures are provided in the Appendix.

Calculate Count of Months

This measure is very important. We will use it as an argument in several headcount measure formulas and also in a number of cost per head measure formulas that we will look at later.

This measure returns the distinct count of months in [Fiscal Month] where the value in Actual is other than 0. This is important because there are rows in the Finance Data table that have to be ignored because they contain zeros. Because this measure might be evaluated for a single month, or any number of months, and given any set of rows in the table, we need to know how many months of data we actually have.

To calculate the distinct count of actual months, we create a formula like this:

```
CountOfActualMonths:=CALCULATE(DISTINCTCOUNT('Finance Data'[Fiscal Month]),'Finance Data'[Actual]<>0)
```

Other measures we create are:

- CountOfBudgetMonths
- CountOfFcstMonths

Formulas for these measures are provided in the Appendix.

Calculate Total Headcount

This formula calculates total headcount over time. We don't really need the result by itself. We will however use this as an interim measure in other measures, in-order to calculate an annualized cost per head. In most cases, this measure can be hidden from client tools.

```
Actual HC Total:=[Act Avg HC]*[CountOfActualMonths]
```

Other measures we create are:

- Budget HC Total
- Forecast HC Total

Formulas for these measures are provided in the Appendix.

Calculate Headcount Year-over-Year and Variance

Now that we have all of our base headcount measures, we can create year-over-year and variance measures. Much like with our budget measures, we can simply re-use the base measures we already created to subtract and what we want to subtract from.

To calculate year-over-year actual ending headcount,

```
YoY Ending HC:=[Act Ending HC]-[Prior Ending HC]
```

Variance to budget ending headcount,

```
VTB Ending HC:=[Bud Ending HC]-[Act Ending HC]
```

And variance to forecast ending headcount,

```
VTF Ending HC:=[Fcst Ending HC]-[Act Ending HC]
```

Take a moment to look at other headcount measures in the measure grid under the Actual People, Budget People, and Forecast People columns. Much like with our budget measures, our headcount measures follow a pattern.

Other measures we create are:

- YoY Avg HC
- YoY HC
- VTB Avg HC
- VTB HC
- VTF Avg HC
- VTF HC
- Bud YoY Ending HC
- Bud YoY Avg HC
- Bud YoY HC
- Fcst VTB Ending HC
- Fcst VTB Avg HC
- Fcst VTB HC
- Fcst YoY Ending HC
- Fcst YoY Avg HC
- Fcst YoY HC

Formulas for these measures are provided in the Appendix.

Create Your Own:

In this quick lesson, you create a variance to budget ending headcount in percent measure.

1. In the Finance Data table, in the Measure Grid, click on the empty cell below VTB HC.
2. In the formula, type the following formula, and then press enter.

   ```
   VTB Ending HC %:=IF([Bud Ending HC], [VTB Ending HC]/ABS([Bud Ending HC]), BLANK())
   ```

3. Right click on the new measure, then click Format, and then select Number, Format: Percentage, and Decimal Places = 2.
4. Create a PivotTable with your new measure and see the results.

Cost per Head Metrics

Now that we have the budget and headcount measures, we can begin creating measures that calculate cost per head.

The most important metrics we want when analyzing cost per head (CPH) are annualized cost per head. This is the full year average cost of an employee for any given selection of time. The cost per head represents the cost burden of having one employee on a full year basis. In a traditional price x quantity equation, the cost per head is the price equivalent and the headcount is the quantity equivalent, with the resulting multiplication being the company's total employee cost.

In order to calculate annualized cost per head, we first need to create several interim measures we will use in our annualized cost per head measures. For now, let's only look at these for Actual. By now you've seen measures created for Budget and Forecast are nearly identical.

Calculate Actual People Cost

The first measure we create calculates an actual people cost.

Since we only want to aggregate a sum total for Actual $ for expense accounts with a Sub Class value of People, we use a CALCULATE function with the Actual $ measure as the expression to be evaluated, and for the first filter argument we use a FILTER function to return a table with rows only with a Sub Class value of People, like this:

```
Actual People Cost $:=CALCULATE([Actual $],  FILTER('Finance Data', RELATED(Accounts[Sub Class])="People"))
```

Other measures we create are:

- Budget People Cost $
- Fcst People Cost $

Formulas for these measures are provided in the Appendix.

Calculate Annualized Actual People Cost

The next measure we create calculates an annualized actual people cost:

```
Anlzd Actual People Cost $:=
        IF(
            [CountOfActualMonths], [Actual People Cost $]* 12/[CountOfActualMonths],
         BLANK()
        )
```

Let's look at this formula more carefully.

First, we create a formula that returns a multiplier for annualizing the Actual People Cost $ by dividing 12 by the distinct count of months in the current context, like this:

12/[CountOfActualMonths]

Next, we want to multiply the Actual People Cost $ by our multiplier to get an annualized Actual People Cost $, like this:

[Actual People Cost $]*12/[CountOfActualMonths]

Finally, we want to make sure our formula doesn't divide by 0, so we wrap it in an IF statement:

IF(

[CountOfActualMonths], [Actual People Cost $]* 12/[CountOfActualMonths],

BLANK()

Other measures we create are:

- Anlzd Actual People Cost $
- Anlzd Fcst People Cost $

Formulas for these measures are provided in the Appendix.

Since each of these measures on their own do not provide a useful metric we can use in a report, we can hide them from client tools. These measures will only be provided as arguments along with other measures to calculate annualized cost per head.

Calculate Annualized Cost per Head

Now that we have measures that return an annualized people cost, and measures that returns an average headcount, we can create a measure that returns an annualized actual cost per head:

```
Act Anlzd CPH:=IF([Act Avg HC], [Anlzd Actual People Cost $]/[Act Avg HC], BLANK())
```

This measure states, divide the value from Anlzd Actual People Cost $ by the value from Act Avg HC. Like other measures where we are performing a division, we wrap it in an IF statement to prevent division by 0.

From our Act Anlzd CPH measure, we can create a similar measure for Budget:

```
Bud Anlzd CPH:=IF([Bud Avg HC], [Anlzd Budget People Cost $]/[Bud Avg HC], BLANK())
```

And, for Forecast:

```
Fcst Anlzd CPH:=IF([Fcst Avg HC], [Anlzd Fcst People Cost $]/[Fcst Avg HC], BLANK())
```

We also need to create a measure that calculate an actual annualized cost per head for the previous year. While we don't really need to create year-over-year cost per head measures, we will need to use this measure when calculating rate and volume variance.

```
Prior Act Anlzd CPH:=CALCULATE([Act Anlzd CPH], DATEADD('Date'[Date],-1, Year))
```

Calculate Rate and Volume Variance

Now that we have measures that calculate annualized cost per head and measures that calculate headcount totals, we can create rate variance measures that calculate what portion of a currency variance is caused by differences in cost per head, and we can create volume measures that calculate how much of the currency variance is driven by fluctuation in headcount.

For example, in regards to people costs (employee payroll, travel, benefits, etc.), if Budget $ is $1,000,000 and Actual $ is $900,000, the total variance to budget is $100,000. A rate variance measure calculates what portion of that $100,000 is driven by the difference in cost per head between Actual and Budget. Our volume variance measure will calculate what portion of that $100,000 is driven by the difference in headcount. Added together, the rate variance and volume variance equal the total variance in people costs.

Calculate Rate Variance

Our variance to budget rate formula looks like this:

```
VTB Rate:=([Bud Anlzd CPH]/12-[Act Anlzd CPH]/12)*[Actual HC Total]
```

Let's look at this formula more carefully.

In the first part of this formula, we are calculating the difference (or variance) in cost per head. We want to subtract the result from Bud Anlzd CPH by the result from Act Anlzd CPH. But first, because these results are annualized, we want to divide them by twelve to return them to a monthly value, like this:

[Bud Anlzd CPH]/12

And

[Act Anlzd CPH]/12

Now, to calculate the variance from these two values, we subtract the monthly value for Act Anlzd CPH from the monthly value for Bud Anlzd CPH:

([Bud Anlzd CPH]/12-[Act Anlzd CPH]/12)

This, however, will not get us the result we need. To get the correct rate, we need to multiply by the value returned from Actual HC Total, like this:

([Bud Anlzd CPH]/12-[Act Anlzd CPH]/12)*[Actual HC Total]

Calculate Volume Variance

To calculate our volume variance, we multiply the result from variance to budget for headcount, VTB HC, with the result from our budget annualized cost per head, Bud Anlzd CPH. Because this result is annualized, just like with our rate formula, we want to divide by twelve to return it to a monthly value. This gives us our VTB Volume formula:

```
VTB Volume:=[VTB HC]*[Bud Anlzd CPH]/12
```

Let's create a PivotTable with our new rate and volume measures. If we add Fiscal Year and Sub Class to Report Filters, and then select FY13, and People respectively, then add Fiscal QTR and Fiscal Month to Row Labels, and then add Act Anlzd CPH, Bud Anlzd CPH, VTB Rate, VTB Volume, and VTB $ to Values, we see:

Fiscal Year	FY13
Sub Class	People

Row Labels	Act Anlzd CPH	Bud Anlzd CPH	VTB Rate	VTB Volume	VTB $
FY13-Q1	**$203,451**	**$210,744**	**$8,121,022**	**($1,545,454)**	**$6,575,568**
FY13-P01	$218,653	$242,596	$8,898,967	($505,409)	$8,393,558
FY13-P02	$83,531	$71,600	($4,469,283)	($190,933)	($4,660,216)
FY13-P03	$310,357	$320,347	$3,669,788	($827,563)	$2,842,225
FY13-Q2	**$198,424**	**$259,977**	**$69,466,795**	**$1,299,883**	**$70,766,678**
FY13-P04	$250,489	$251,157	$251,839	($83,719)	$168,120
FY13-P05	$256,882	$240,663	($6,064,434)	$641,768	($5,422,667)
FY13-P06	$88,432	$287,867	$75,253,579	$767,646	$76,021,225
FY13-Q3		**$251,510**		**$288,544,349**	**$288,544,349**
FY13-P07		$477,847		$183,015,576	$183,015,576
FY13-P08		$175,968		$66,251,871	$66,251,871
FY13-P09		$101,294		$39,276,902	$39,276,902
FY13-Q4		**$251,152**		**$292,781,016**	**$292,781,016**
FY13-P10		$205,446		$79,730,015	$79,730,015
FY13-P11		$205,823		$80,185,367	$80,185,367
FY13-P12		$342,364		$132,865,634	$132,865,634
Grand Total	**$200,921**	**$243,621**	**$95,740,685**	**$562,926,925**	**$658,667,612**

In this table, for FY13-Q1, our variance to budget is $6,575,568 (remember, we have filtered on only those expense accounts with a Sub Class value of People), our budget annualized cost per head is $210,744, and our actual annualized cost per head is $203,451. Our VTB Rate measure calculates that $8,121,022 of the variance to budget (VTB $) is caused by the difference in cost per head, and $-1,545, 454 is caused by the difference in headcount.

If we add our result for VTB Rate to our result for VTB Volume, we get $6,575,568, the same value as returned by VTB $ (when filtered by Sub Class People).

Calculate Year-Over-Year Rate and Volume Variance

We only have to make slight changes to our formula in order to create a year-over-year rate measure:

```
YoY Rate:=([Act Anlzd CPH]/12-[Prior Act Anlzd CPH]/12)*[Actual HC Total]
```

And to create a year-over-year volume measure:

```
YoY Volume:=[YoY HC]*[Prior Act Anlzd CPH]/12
```

Other rate and volume measures we create are:

- VTF Rate
- VTF Volume
- Bud YoY Rate
- Bud YoY Volume
- Fcst VTB Rate
- Fcst VTB Volume

Formulas for these measures are provided in the Appendix.

We now have all of the measures we need to perform some in-depth analysis. There are literally hundreds of different ways to explore the data. Take some time with the ContosoPnL workbook by creating some PivotTables and reports, and slicing and dicing different metrics.

Our data model is also very efficient. We imported over 70 megabytes of data from the ContosoPnL database. Because of PowerPivot's ability to compress the data, and because we created efficient measures to perform our calculations, our workbook is only seven megabytes.

Our workbook can be uploaded to a PowerPivot Gallery in SharePoint, so we can share it with other users. We can configure it to automatically refresh the data, so users are always analyzing the latest data. Users will not need to edit the data model because all of the fields and all of the measures are already there. Users can create their own reports based on the data model using any of the fields and measures we've included.

Be sure to check the Additional Resources section later in this article for links to other valuable PowerPivot and Power View resources.

Did this paper and the samples help you?

Please give us your feedback at sqlfback@microsoft.com.

Appendix

DAX Formulas in the ContosoPnL Data Model

Calculated Columns

Fiscal Year

Table: Date

Description: For each row, create a fiscal year value in the format of FY13 from values in the Date column.

DAX Formula:

```
="FY"&RIGHT(INT((MONTH([Date])-1)/6)+YEAR([Date]),2)
```

Fiscal Qtr

Table: Date

Description: For each row, create a fiscal quarter value in the format of FY13-Q1 from values in the Date column.

DAX Formula:

```
=[Fiscal Year] & "-Q" & FORMAT(INT((MOD(MONTH([Date])+5,12)+3)/3 ),"0")
```

Fiscal Month

Table: Date

Description: For each row, create a fiscal month value in the format of FY13-P01 from values in the Date column.

DAX Formula:

```
=[Fiscal Year] & "-P" & FORMAT(MOD(MONTH([Date])+5,12)+1,"00")
```

Month

Table: Date

Description: For each row, create a month value in the format of 01-Jul from values in the Date column.

DAX Formula:

```
=FORMAT(MOD(MONTH([Date])+5,12)+1,"00") & "-" & FORMAT([Date],"mmm")
```

Date

Table: Finance Data

Description: For each row, create a data value in the format of 07/01/13 12:00:00 from values in the Fiscal Month column.

DAX Formula:

```
=DATEVALUE([Fiscal Month])
```

Currency Measures

Actual $

Name: Actual $

Description: Calculate the sum total for all rows in the Actual column.

DAX Formula:

```
Actual $:=SUM([Actual])
```

Budget $

Description: Calculate the sum total for all rows in the Budget column.

DAX Formula:

```
Budget $:=SUM([Budget])
```

Forecast $

Description: Calculate the sum total for all rows in the Forecast column.

DAX Formula:

```
Forecast $:=SUM([Forecast])
```

Prior $

Description: Calculate the sum total for all rows in the Actual column for the prior year.

DAX Formula:

```
 Prior $:=CALCULATE([Actual $], DATEADD('Date'[Date],-1,YEAR))
```

YTD Actual $

Description: Year running sum of [Actual $] from the end of the prior fiscal year end date up to last date in the current period.

DAX Formula:

```
YTD Actual $:=TOTALYTD([Actual $], 'Date'[Date],ALL('Date'), "6/30")
```

YTD Budget $

Description: Year running sum of [Budget $] from the end of the prior fiscal year end date up to last date in the current period.

DAX Formula:

```
YTD Budget $:=TOTALYTD([Budget $], 'Date'[Date],ALL('Date'), "6/30")
```

YTD Forecast $

Description: Year running sum of [Forecast $] from the end of the prior fiscal year end date up to last date in the current period.

DAX Formula:

```
YTD Forecast $:=TOTALYTD([Forecast $], 'Date'[Date],ALL('Date'), "6/30")
```

Prior YTD Actual $

Description: [YTD Actual $] for the same period prior year.

DAX Formula:

```
Prior YTD Actual $:=CALCULATE([YTD Actual $], DATEADD('Date'[Date],-1,YEAR))
```

Budget Remaining $

Description: Calculates budget dollars remaining. [YTD Budget $] minus [YTD Actual $].

DAX Formula:

```
Budget Remaining $:=CALCULATE([Budget $], ALL('Finance Data'[Date]))-[YTD Actual $]
```

Forecast Remaining $

Description: Calculates forecast dollars remaining. [YTD Forecast $] minus [YTD Actual $].

DAX Formula:

```
Forecast Remaining $:=CALCULATE([Forecast $], ALL('Finance Data'[Date]))-[YTD Actual $]
```

Budget Attain %

Description: Calculates budget spent in percent. [YTD Actual $] divided by [YTD Budget $].

DAX Formula:

```
Budget Attain %:=IF(
                    [Budget $], [YTD Actual $]/CALCULATE([Budget $],
                        ALL('Finance Data'[Date])),
                    BLANK()
                    )
```

Forecast Attain %

Description: Calculates forecast spent in percent. [YTD Actual $] divided by [YTD Forecast $].

DAX Formula:

```
Forecast Attain %:=IF(
                    [Forecast $], [YTD Actual $]/CALCULATE([Forecast $],
                        ALL('Finance Data'[Date])),
                    BLANK()
                    )
```

YoY $

Description: Calculates the difference in Actual dollars between the current period and the same period last year. [Actual $] minus [Prior $].

DAX Formula:

```
YoY $:=
          IF(
             CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Actual $]-[Prior $]),
          ([Actual $]-[Prior $])
 )
```

YoY %

Description: Calculates the difference in percent between the current period Actual $ from Actual $ for the same period last year. [YoY $] divided by [Prior $].

DAX Formula:

```
YoY %:=IF([Prior $], [YoY $]/[Prior $],BLANK())
```

VTB $

Description: Calculates the difference in dollars between Budget $ and Actual $. [Budget $] minus [Actual $].

DAX Formula:

```
VTB $:=[Budget $]-[Actual $]
```

VTB %

Description: Calculates the difference in percent between Budget $ and Actual $. [VTB $] divided by [Budget $].

DAX Formula:

```
VTB %:=IF([Budget $], [VTB $]/[Budget $], BLANK())
```

VTF $

Description: Calculates the difference in dollars between Forecast $ and Actual $. [Forecast $] minus [Actual $].

DAX Formula:

```
VTF $:=[Forecast $]-[Actual $]
```

VTF %

Description: Calculates the difference in percent between Forecast $ and Actual $. [VTF $] divided by [Forecast $].

DAX Formula:

```
VTF %:=IF([Forecast $], [VTF $]/[Forecast $], BLANK())
```

Bud YoY $

Description: Calculates the difference in budget dollars between the current period and the same perios last year. [Budget $] minus [Prior $].

DAX Formula:

```
Bud YoY $:=
        IF(
           CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Budget $]-[Prior $]),
        ([Budget $]-[Prior $])
)
```

Bud YoY %

Description: Calculates the difference in percentage between the current period Budget YoY $ from Prior $. [Bud YoY $] divided by [Prior $].

DAX Formula:

```
Bud YoY %:=IF([Prior $], [Bud YoY $]/[Prior $],BLANK())
```

Fcst VTB $

Description: Calculates the difference in forecast dollars between the current period and the same period last year. [Budget $] minus [Forecast $].

DAX Formula:

```
Fcst VTB $:=[Budget $]-[Forecast $]
```

Fcst VTB %

Description: Calculates the difference in percent between Budget $ and Forecast $. [Fcst VTB $] divided by [Budget $].

DAX Formula:

```
Fcst VTB %:=IF([Budget $], [Fcst VTB $]/[Budget $], BLANK())
```

Fcst YoY $

Description: Calculates the difference in dollars between Forecast $ from Actual $ for the same period last year. [Forecast $] minus [Prior $].

DAX Formula:

```
Bud YoY $:=
        IF(
           CONTAINS(Accounts, Accounts[Class], "Net Revenue"), -([Forecast $]-[Prior $]),
        ([Forecast $]-[Prior $])
)
```

Fcst YoY %

Description: Calculates the difference in percent between Forecast $ from Actual $ for the same period last year. [Fcst YoY $] divided by [Prior $].

DAX Formula:

```
Fcst YoY %:=IF([Prior $],[Fcst YoY $]/[Prior $],BLANK())
```

Headcount Measures

CountOfActualMonths

Description: Distinct count of fiscal months with [Actual] values other than 0. This measure is hidden from client tools.

DAX Formula:

```
CountOfActualMonths:=CALCULATE(DISTINCTCOUNT('Finance Data'[Fiscal Month]),'Finance Data'[Actual]<>0)
```

CountOfBudgetMonths

Description: Distinct count of fiscal months with [Budget] values other than 0. This measure is hidden from client tools.

DAX Formula:

```
CountOfBudgetMonths:=CALCULATE(DISTINCTCOUNT('Finance Data'[Fiscal Month]),'Finance Data'[Budget]<>0)
```

CountOfFcstMonths

Description: Distinct count of fiscal months with [Forecast] values other than 0. This measure is hidden from client tools.

DAX Formula:

```
CountOfFcstMonths:=CALCULATE(DISTINCTCOUNT('Finance Data'[Fiscal Month]),'Finance Data'[Forecast]<>0)
```

Actual HC Total

Description: [Act Avg HC] multiplied by [CountOfActualMonths].

DAX Formula:

```
Actual HC Total:=[Act Avg HC]*[CountOfActualMonths]
```

Budget HC Total

Description: [Bud Avg HC] multiplied by [CountOfBudgetMonths].

DAX Formula:

```
Budget HC Total:=[Bud Avg HC]*[CountOfBudgetMonths]
```

Forecast HC Total

Description: [Fcst Avg HC] multiplied by [CountOfFcstMonths].

DAX Formula:

```
Forecast HC Total:=[Fcst Avg HC]*[CountOfFcstMonths]
```

Act Ending HC

Description: Actual headcount for the last period in the filtered data set.

DAX Formula:

```
Act Ending HC:=
CALCULATE(
        SUM('Finance Data'[Actual People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Actual People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Actual People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

Bud Ending HC

Description: Budget headcount for the last period in the filtered data set.

DAX Formula:

```
Act Ending HC:=
CALCULATE(
        SUM('Finance Data'[Budget People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Budget People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Budget People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

Fcst Ending HC

Description: Forecast headcount for the last period in the filtered data set.

DAX Formula:

```
Act Ending HC:=
CALCULATE(
        SUM('Finance Data'[Forecast People]),
        LASTNONBLANK(
            'Finance Data'[Date],
             IF(
                CALCULATE(SUM('Finance Data'[Forecast People]), ALL(Accounts))=0,
                BLANK(),
                CALCULATE(SUM('Finance Data'[Forecast People]), ALL(Accounts))
             )
        ),
        ALL(Accounts)
)
```

Act Avg HC

Description: Mean average of [Act Ending HC] for each unique value in 'Finance Data'[Fiscal Month].

DAX Formula:

```
Act Avg HC:=AVERAGEX( VALUES('Finance Data'[Fiscal Month]), [Act Ending HC])
```

Bud Avg HC

Description: Mean average of [Bud Ending HC] for each unique value in 'Finance Data'[Fiscal Month].

DAX Formula:

```
Bud Avg HC:=AVERAGEX( VALUES('Finance Data'[Fiscal Month]), [Bud Ending HC])
```

Fcst Avg HC

Description: Mean average of [Fcst Ending HC] for each unique value in 'Finance Data'[Fiscal Month].

DAX Formula:

```
Fcst Avg HC:=AVERAGEX( VALUES('Finance Data'[Fiscal Month]), [Fcst Ending HC])
```

Prior HC Total

Description: [Actual HC Total] for the same period prior year.

DAX Formula:

```
Prior HC Total:=CALCULATE([Actual HC Total],DATEADD('Date'[Date],-1,Year),All(Accounts))
```

Prior Ending HC

Description: [Act Ending HC] for the last period in the filtered data set for the same period prior year.

DAX Formula:

```
Prior Ending HC:=CALCULATE([Act Ending HC], DATEADD('Date'[Date],-1,YEAR))
```

Prior Avg HC

Description: [Act Avg HC] for the same period prior year.

DAX Formula:

```
Prior Avg HC:=[Prior HC Total]/CALCULATE(DISTINCTCOUNT('Finance Data'[Fiscal Month]),All(Accounts),'Finance Data'[Actual People]<>0)
```

YoY Ending HC

Description: [Act Ending HC] minus [Prior Ending HC].

DAX Formula:

```
YoY Ending HC:=[Act Ending HC]-[Prior Ending HC]
```

YoY Avg HC

Description: [Act Avg HC] minus [Prior Avg HC].

DAX Formula:

```
YoY Avg HC:=[Act Avg HC]-[Prior Avg HC]
```

YoY HC

Description: [Actual HC Total] minus [Prior HC Total].

DAX Formula:

```
YoY HC:=[Actual HC Total]-[Prior HC Total]
```

VTB Ending HC

Description: [Bud Ending HC] minus [Act Ending HC].

DAX Formula:

```
VTB Ending HC:=[Bud Ending HC]-[Act Ending HC]
```

VTB Avg HC

Description: [Bud Avg HC] minus [Act Avg HC].

DAX Formula:

```
VTB Avg HC:=[Bud Avg HC]-[Act Avg HC]
```

VTB HC

Description: [Budget HC Total] minus [Actual HC Total].

DAX Formula:

```
VTB HC:=[Budget HC Total]-[Actual HC Total]
```

VTF Ending HC

Description: [Fcst Ending HC] minus [Act Ending HC].

DAX Formula:

```
VTF Ending HC:=[Fcst Ending HC]-[Act Ending HC]
```

VTF Avg HC

Description: [Fcst Avg HC] minus [Act Avg HC].

DAX Formula:

```
VTF Avg HC:=[Fcst Avg HC]-[Act Avg HC]
```

VTF HC

Description: [Forecast HC Total] minus [Actual HC Total].

DAX Formula:

```
VTF HC:=[Forecast HC Total]-[Actual HC Total]
```

Bud YoY Ending HC

Description: [Bud Ending HC] minus [Prior Ending HC].

DAX Formula:

```
Bud YoY Ending HC:=[Bud Ending HC]-[Prior Ending HC]
```

Bud YoY Avg HC

Description: [Bud Avg HC] minus [Prior Avg HC].

DAX Formula:

```
Bud YoY Avg HC:=[Bud Avg HC]-[Prior Avg HC]
```

Bud YoY HC

Description: [Budget HC Total] minus [Prior HC Total].

DAX Formula:

```
Bud YoY HC:=[Budget HC Total]-[Prior HC Total]
```

Fcst VTB Ending HC

Description: [Bud Ending HC] minus [Fcst Ending HC].

DAX Formula:

```
Fcst VTB Ending HC:=[Bud Ending HC]-[Fcst Ending HC]
```

Fcst VTB Avg HC

Description: [Bud Avg HC] minus [Fcst Avg HC].

DAX Formula:

```
Fcst VTB Avg HC:=[Bud Avg HC]-[Fcst Avg HC]
```

Fcst VTB HC

Description: [Budget HC Total] minus [Forecast HC Total].

DAX Formula:

```
Fcst VTB HC:=[Budget HC Total]-[Forecast HC Total]
```

Fcst YoY Ending HC

Description: [Fcst Ending HC] minus [Prior Ending HC].

DAX Formula:

```
Fcst YoY Ending HC:=[Fcst Ending HC]-[Prior Ending HC]
```

Fcst YoY Avg HC

Description: [Fcst Avg HC] minus [Prior Avg HC].

DAX Formula:

```
Fcst YoY Avg HC:=[Fcst Avg HC]-[Prior Avg HC]
```

Fcst YoY HC

Description: [Forecast HC Total] minus [Prior HC Total].

DAX Formula:

```
Fcst YoY HC:=[Forecast HC Total]-[Prior HC Total]
```

Cost per Head Measures

Act Anlzd CPH

Description: [Anlzd Actual People Cost $] divided by [Act Avg HC].

DAX Formula:

```
Act Anlzd CPH:=IF([Act Avg HC], [Anlzd Actual People Cost $]/[Act Avg HC], BLANK())
```

Bud Anlzd CPH

Description: [Anlzd Budget People Cost $] divided by [Bud Avg HC].

DAX Formula:

```
Bud Anlzd CPH:=IF([Bud Avg HC], [Anlzd Budget People Cost $]/[Bud Avg HC], BLANK())
```

Fcst Anlzd CPH

Description: [Anlzd Fcst People Cost $] divided by [Fcst Avg HC].

DAX Formula:

```
Fcst Anlzd CPH:=IF([Fcst Avg HC], [Anlzd Fcst People Cost $]/[Fcst Avg HC], BLANK())
```

Prior Act Anlzd CPH

Description: [Act Anlzd CPH] for the same period last year.

DAX Formula:

```
Prior Act Anlzd CPH:=CALCULATE([Act Anlzd CPH], DATEADD('Date'[Date],-1, Year))
```

YoY Rate

Description: [Act Anlzd CPH] divided by 12, minus [Prior Act Anlzd CPH] divided by 12, multiplied by [Actual HC Total].

DAX Formula:

```
YoY Rate:=([Act Anlzd CPH]/12-[Prior Act Anlzd CPH]/12)*[Actual HC Total]
```

YoY volume

Description: [YoY HC] multiplied by [Prior Act Anlzd CPH] divided by 12.

DAX Formula:

```
YoY Volume:=[YoY HC]*[Prior Act Anlzd CPH]/12
```

VTB Rate

Description: [Bud Anlzd CPH] divided by 12, minus [Act Anlzd CPH] divided by 12, multiplied by [Actual HC Total].

DAX Formula:

```
VTB Rate:=([Bud Anlzd CPH]/12-[Act Anlzd CPH]/12)*[Actual HC Total]
```

VTB Volume

Description: [VTB HC] multiplied by [Bud Anlzd CPH] divided by 12.

DAX Formula:

```
VTB Volume:=[VTB HC]*[Bud Anlzd CPH]/12
```

VTF Rate

Description: [Fcst Anlzd CPH] divided by 12, minus [Act Anlzd CPH] divided by 12, multiplied by [Actual HC Total].

DAX Formula:

```
VTF Rate:=([Fcst Anlzd CPH]/12-[Act Anlzd CPH]/12)*[Actual HC Total]
```

VTF Volume

Description: [VTF HC] multiplied by [Fcst Anlzd CPH] divided by 12.

DAX Formula:

```
VTF Volume:=[VTF HC]*[Fcst Anlzd CPH]/12
```

Bud YoY Rate

Description: [Bud Anlzd CPH] divided by 12, minus [Prior Act Anlzd CPH] divided by 12, multiplied by [Budget HC Total].

DAX Formula:

```
Bud YoY Rate:=([Bud Anlzd CPH]/12-[Prior Act Anlzd CPH]/12)*[Budget HC Total]
```

Bud YoY Volume

Description: [Bud YoY HC] multiplied by [Prior Act Anlzd CPH] divided by 12.

DAX Formula:

```
Bud YoY Volume:=[Bud YoY HC]*[Prior Act Anlzd CPH]/12
```

Fcst VTB Rate

Description: [Bud Anlzd CPH] divided by 12, minus [Fcst Anlzd CPH] divided by 12, multiplied by [Forecast HC Total].

DAX Formula:

```
Fcst VTB Rate:=([Bud Anlzd CPH]/12-[Fcst Anlzd CPH]/12)*[Forecast HC Total]
```

Fcst VTB Volume

Description: [Fcst VTB HC] multiplied by [Bud Anlzd CPH] divided by 12.

DAX Formula:

```
Fcst VTB Volume:=[Fcst VTB HC]*[Bud Anlzd CPH]/12
```

Fcst YoY Rate

Description: [Fcst Anlzd CPH] divided by 12, minus [Prior Act Anlzd CPH] divided by 12, multiplied by [Forecast HC Total].

DAX Formula:

```
Fcst YoY Rate:=([Fcst Anlzd CPH]/12-[Prior Act Anlzd CPH]/12)*[Forecast HC Total]
```

Fcst YoY Volume

Description: [Fcst YoY HC] multiplied by [Prior Act Anlzd CPH] divided by 12.

DAX Formula:

```
Fcst YoY Volume:=[Fcst YoY HC]*[Prior Act Anlzd CPH]/12
```

Actual People Cost $

Description: [Actual $] filtered by Accounts[Sub Class] with a value of "People". This measure is hidden from client tools.

DAX Formula:

```
Actual People Cost $:=CALCULATE([Actual $],  FILTER('Finance Data', RELATED(Accounts[Sub Class])="People"))
```

Budget People Cost $

Description: [Budget $] filtered by Accounts[Sub Class] with a value of "People". This measure is hidden from client tools.

DAX Formula:

```
Budget People Cost $:=CALCULATE([Budget $], FILTER('Finance Data', RELATED(Accounts[Sub Class])="People"))
```

Fcst People Cost $

Description: [Forecast $] filtered by Accounts[Sub Class] with a value of "People". This measure is hidden from client tools.

DAX Formula:

```
Fcst People Cost $:=CALCULATE([Forecast $], FILTER('Finance Data', RELATED(Accounts[Sub Class])="People"))
```

Anlzd Actual People Cost $

Description: [Actual People Cost $] multiplied by 12, divided by [CountOfActualMonths]. This measure is hidden from client tools.

DAX Formula:

```
Anlzd Actual People Cost $:=
        IF(
            [CountOfActualMonths], [Actual People Cost $]* 12/[CountOfActualMonths],
         BLANK()
        )
```

Anlzd Budget People Cost $

Description: [Budget People Cost $] multiplied by 12, divided by [CountOfBudgetMonths]. This measure is hidden from client tools.

DAX Formula:

```
Anlzd Budget People Cost $:=
        IF(
            [CountOfBudgetMonths], [Budget People Cost $]* 12/[CountOfBudgetMonths],
         BLANK()
        )
```

Anlzd Fcst People Cost $

Description: [Fcst People Cost $] multiplied by 12, divided by [CountOfFcstMonths]. This measure is hidden from client tools.

DAX Formula:

```
Anlzd Fcst People Cost $:=
        IF(
            [CountOfFcstMonths], [Fcst People Cost $]* 12/[CountOfFcstMonths],
         BLANK()
        )
```

Additional Resources

DAX Resource Center Wiki on Microsoft TechNet is a great resource for all things DAX. This site is updated frequently by BI professionals, sharing the latest and greatest info about DAX.

Calculations in PowerPivot in Excel 2013 on Office.com has a lot of great DAX information, including the DAX Reference.

Power View: Explore, visualize, and present your data on Office.com is a great resource for learning about Power View in Excel 2013.

Visualizing Data with Microsoft Power View (ISBN: 0071780823 / 9780071780827) By Brian Larson, Mark Davis, Dan English, Paul Purington shows how to effectively analyze and communicate complex information through elegant interactive reports.