Mastering DAX

Exercise Book

Exercises

This document contains the exercises needed to complete the Mastering DAX Workshop. There are many exercises in this paper, some of them you will be able to cover during the course, and some others might be useful later, to fix some concepts in your mind.

Each lab is stored in a separate folder. In each lab, you will find the starting point of the lab (it might be an Excel file or a Power BI file) and the final solution.

Following the exercises, you always start with the starting point and try to solve the scenario. If you are having issues, or if you want to look at the solution, you can open our proposed solution to get some hints.

Follow the exercises with your time. There is no need to hurry. If you cannot finish one exercise, you will be able to do it later. This is why we give all the solutions. Moreover, feel free to ask the teacher for any information that you might require or to discuss different solutions you might have found.

Please note that, in the solution, we provide both measures and calculated columns. The difference between the two is very subtle: we define columns using an equal sign, like in:

```
Sales[SalesAmount] = Sales[Quantity] * Sales[Unit Price]
```

Measures, on the other hand, have a different syntax, we use ":=":

```
SalesAmount := SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
```

At the end of this paper there is a large figure of the Contoso data model that you can use to familiarize with the database. We use Contoso for most of the exercises, whereas some are built on AdventureWorks and others are based on simpler educational models but, for these exercises, looking the data model is not needed.

The most important tool for this exercise book is a white page that you use to hide the solution. Do not look at the solution, which oftentimes is in the next paragraph. Hide it and think, then look at the solution when you have at least an idea on how to proceed.

LAB 1
First Steps with DAX

This is the name of the file to use

Contoso Sales Analysis

First Steps with DAX

In this exercise, you create some calculated columns and measures using Excel to perform analysis of orders. The goal is to discover how to create calculated columns, measures and how to leverage them to solve a simple business scenario.

Open the demo file, which contains a simplified version of the Contoso database you have seen during the demo.

Reporting Sales

You want to prepare a report that shows yearly sales. Thus, you need a measure that computes the sales amount. The problem is that Sales has two columns [Quantity] and [Unit Price]; a column with the sales amount is missing.

Your task is to create a measure that computes the sales amount.

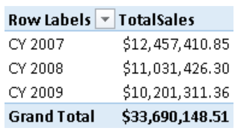
1. Start creating a calculated column that computes the sales amount, by multiplying [Quantity] and [Unit Price]. Name it [Sales Amount]

```
Sales[SalesAmount] = Sales[Quantity] * Sales[Unit Price]
```

2. Create a measure that sums the SalesAmount column, name it [TotalSales]

```
TotalSales := SUM ( Sales[SalesAmount] )
```

3. At this point, you can create a report like the following one:



Row Labels	TotalSales
CY 2007	$12,457,410.85
CY 2008	$11,031,426.30
CY 2009	$10,201,311.36
Grand Total	**$33,690,148.51**

4. Creating the measure using a calculated column is a simple task. Nevertheless, it involves the additional column SalesAmount, which is useful only for the calculation of TotalSales. Can you figure out a better way to compute TotalSales, avoiding the calculated column?
5. Using iterators, the solution is straightforward, try it out:

```
TotalSales := SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
```

6. With this definition of TotalSales, you can avoid the SalesAmount calculated column, so you can delete it.

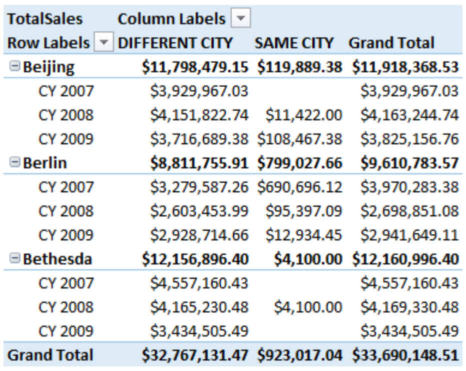
Analyze sales in the same city

The City column is present in Store and in Customer. You want to divide sales made to customers in the same city of the store, against sales to customers in a different city, evaluating customers buying in different cities.

Your task is to create a calculated column, in Sales, that checks if the city of the customer is the same as the city of the store. The only problem is that the columns are in different tables. RELATED is your best friend here:

```
Sales[SaleType] =
IF (
    RELATED ( Customer[City] ) = RELATED ( Store[City] ),
    "SAME CITY",
    "DIFFERENT CITY"
)
```

You can use the column to analyze sales, like in the following report:



TotalSales	Column Labels		
Row Labels	DIFFERENT CITY	SAME CITY	Grand Total
⊟Beijing	**$11,798,479.15**	**$119,889.38**	**$11,918,368.53**
CY 2007	$3,929,967.03		$3,929,967.03
CY 2008	$4,151,822.74	$11,422.00	$4,163,244.74
CY 2009	$3,716,689.38	$108,467.38	$3,825,156.76
⊟Berlin	**$8,811,755.91**	**$799,027.66**	**$9,610,783.57**
CY 2007	$3,279,587.26	$690,696.12	$3,970,283.38
CY 2008	$2,603,453.99	$95,397.09	$2,698,851.08
CY 2009	$2,928,714.66	$12,934.45	$2,941,649.11
⊟Bethesda	**$12,156,896.40**	**$4,100.00**	**$12,160,996.40**
CY 2007	$4,557,160.43		$4,557,160.43
CY 2008	$4,165,230.48	$4,100.00	$4,169,330.48
CY 2009	$3,434,505.49		$3,434,505.49
Grand Total	**$32,767,131.47**	**$923,017.04**	**$33,690,148.51**

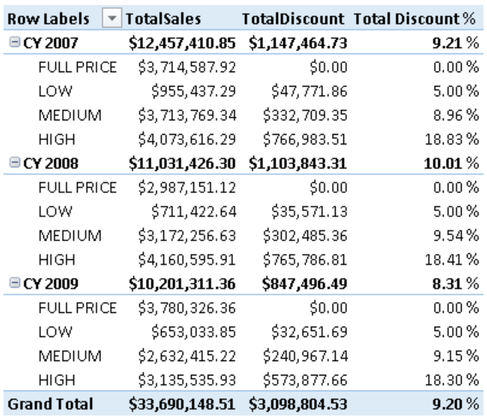
OPTIONAL: can you sort the column in a way that shows “SAME CITY” first and “DIFFERENT CITY” later?

Analyze Discounts

The solution is in the next page, don’t look at it, try to solve the problem by yourself and look at the solution only later. In the Sales table, there are [Unit Price] and [Unit Discount] columns. Your task is to build:

- One measure that reports the total discount
- One calculated column, in Sales, that contains for each row:
    - FULL PRICE if the discount is zero
    - LOW if the discount of the sale is less than or equal to 5%
    - MEDIUM if the discount of the sale is between 5% and 10%
    - HIGH if the discount percentage is higher than 10%
- One measure that computes the average discount percentage

Your goal is to be able to author a report like the following one:



Row Labels	TotalSales	TotalDiscount	Total Discount %
⊟CY 2007	**$12,457,410.85**	**$1,147,464.73**	**9.21 %**
FULL PRICE	$3,714,587.92	$0.00	0.00 %
LOW	$955,437.29	$47,771.86	5.00 %
MEDIUM	$3,713,769.34	$332,709.35	8.96 %
HIGH	$4,073,616.29	$766,983.51	18.83 %
⊟CY 2008	**$11,031,426.30**	**$1,103,843.31**	**10.01 %**
FULL PRICE	$2,987,151.12	$0.00	0.00 %
LOW	$711,422.64	$35,571.13	5.00 %
MEDIUM	$3,172,256.63	$302,485.36	9.54 %
HIGH	$4,160,595.91	$765,786.81	18.41 %
⊟CY 2009	**$10,201,311.36**	**$847,496.49**	**8.31 %**
FULL PRICE	$3,780,326.36	$0.00	0.00 %
LOW	$653,033.85	$32,651.69	5.00 %
MEDIUM	$2,632,415.22	$240,967.14	9.15 %
HIGH	$3,135,535.93	$573,877.66	18.30 %
Grand Total	**$33,690,148.51**	**$3,098,804.53**	**9.20 %**

1. First, you need to create a calculated column in Sales that contains the discount percentage:

```
Sales[DiscountPct] = DIVIDE ( Sales[Unit Discount], Sales[Unit Price] )
```

2. Using DiscountPct, you can create a new calculated column, also in Sales, with the DiscountCategory, leveraging the IF function:

```
Sales[DiscountCategory] =
IF (
    Sales[DiscountPct] = 0,
    "FULL PRICE",
    IF (
        Sales[DiscountPct] <= 0.05,
        "LOW",
        IF ( Sales[DiscountPct] <= 0.1, "MEDIUM", "HIGH" )
    )
)
```

3. The [Total Discount] measure is straightforward, using a simple SUMX:

```
TotalDiscount := SUMX ( Sales, Sales[Unit Discount] * Sales[Quantity] )
```

4. [Total Discount %] is, again, a very simple DIVIDE:

```
[Total Discount %] := DIVIDE ( [TotalDiscount], [TotalSales] )
```

5. You might notice that the, in our report, discount categories are not sorted alphabetically; they are sorted by discount percentage. To obtain this effect, you can create a new calculated column to sort the discount category:

```
Sales[DiscountCategorySort] =
IF (
    Sales[DiscountPct] = 0,
    0,
    IF (
        Sales[DiscountPct] <= 0.05,
        1,
        IF ( Sales[DiscountPct] <= 0.1, 2, 3 )
    )
)
```

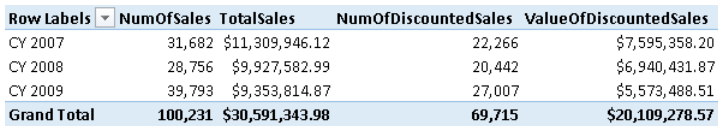
LAB 2
Table Functions

Now that you learned how to work with table functions, it is time to put them in action. You write some simple calculations that involve filtering and iteration over filtered tables.

Analyzing Discounted Sales

Count Discounted Sales

You want to produce a report that shows the number of sales versus the number of discounted sales, along with their respective values. In other words, you need to build a report like the following one:



Row Labels	NumOfSales	TotalSales	NumOfDiscountedSales	ValueOfDiscountedSales
CY 2007	31,682	$11,309,946.12	22,266	$7,595,358.20
CY 2008	28,756	$9,927,582.99	20,442	$6,940,431.87
CY 2009	39,793	$9,353,814.87	27,007	$5,573,488.51
Grand Total	**100,231**	**$30,591,343.98**	**69,715**	**$20,109,278.57**

The starting model already contains NumOfSales and TotalSales, can you figure out how to create the two measures NumOfDiscountedSales and ValueOfDiscountedSales, which compute the number and the value of only discounted sales?

1. NumOfDiscountedSales need to count only the sales where the discount is greater than zero:

```
NumOfDiscountedSales :=
COUNTROWS (
    FILTER (
        Sales,
        Sales[Unit Discount] > 0
    )
)
```

2. ValueOfDiscountedSales follows the very same pattern but, this time, instead of counting rows it sums values:

```
ValueOfDiscountedSales :=
SUMX (
    FILTER (
        Sales,
        Sales[Unit Discount] > 0
    ),
    Sales[Quantity] * Sales[Net Price]
)
```

OPTIONAL: can you write a new measure that computes the percentage (in value) of discounted sales against all the sales?

Count discounted products

The next step is more advanced and a bit harder. You want to count the number of products for which the number of discounted sales is greater than the number of full price ones. In other words, if you have three products and one of them has more discounted sales than full price ones, the result should be 1. As usual, try it by yourself before looking at the solution.

```
NumOfDiscountedProducts :=
COUNTROWS (
    FILTER (
        Product,
        COUNTROWS (
            FILTER ( RELATEDTABLE ( Sales ), Sales[Unit Discount] = 0 )
        )
        <
        COUNTROWS (
            FILTER ( RELATEDTABLE ( Sales ), Sales[Unit Discount] > 0 )
        )
    )
)
```

This expression is a bit hard to catch at first sight. Look carefully at the usage of RELATEDTABLE, which returns the sales related to the currently iterated product by the outer FILTER. Once you have a clear understanding on the formula, move further.

Now you want to count the number of products that you never sold. The pattern is very similar to the previous one, but this time it is a bit less intuitive. Try to figure out the formula by yourself, if you are in trouble, look at the solution:

```
NumOfProductsWithNoSales :=
COUNTROWS (
    FILTER (
        Product,
        COUNTROWS ( RELATEDTABLE ( Sales ) ) = 0
    )
)
```

This technique, of checking that a table is empty, is very common. In DAX 2015 you can also write it in this way:

```
NumOfProductsWithNoSales :=
COUNTROWS (
    FILTER (
        Product,
        ISEMPTY ( RELATEDTABLE ( Sales ) )
    )
)
```

LAB 3
Evaluation Contexts and CALCULATE

Finding the best customers

Best Customers

In this exercise, you make practice with evaluation contexts. The value to compute is an easy one: you want to count the number of customers who bought more than the average of customers. Thus, if on average a customer buys 1,000 USD, you want to count how many of your customers buy more than 1,000 USD.

The first measure you need to compute is the average sales per customer. The formula needs to iterate over the customers and, for each one, compute the sales. Finally, you compute the average of this value.

There are a couple of ways to write this measure. The first, a bit more intuitive:

```
AvgCustomerSale :=
AVERAGEX (
    Customer,
    SUMX (
        RELATEDTABLE ( Sales ),
        Sales[Quantity] * Sales[Net Price]
    )
)
```

The second one, much more elegant, which uses the TotalSales measure and context transition:

```
AvgCustomerSale :=
AVERAGEX (
    Customer,
    [TotalSales]
)
```

Being a measure, AvgCustomerSale computes the average taking into account any filter context under which it is evaluated, and it relies on context transition to further restrict the filter context to a single customer. Once this measure is created, it is now time to compute the other one, i.e. the number of customers who bought more than this average. Start thinking, the solution is in the next page.

The correct formula is the following one:

```
GoodCustomers :=
VAR AvgAllSelected = [AvgCustomerSale]
RETURN
    COUNTROWS (
        FILTER (
            Customer,
            [AvgCustomerSale] >= AvgAllSelected
        )
    )
```

In this case, the usage of a variable is very important to get the average of selected customers. Without variables, you can write it in this way:

```
GoodCustomers :=
COUNTROWS (
    FILTER (
        Customer,
        [AvgCustomerSale] >= CALCULATE ( [AvgCustomerSale], ALL ( Customer ) )
    )
)
```

This latter formula works fine but it does not take into account partial filtering that might happen on the Customer table. It is worth noting that this other version of the formula does not work:

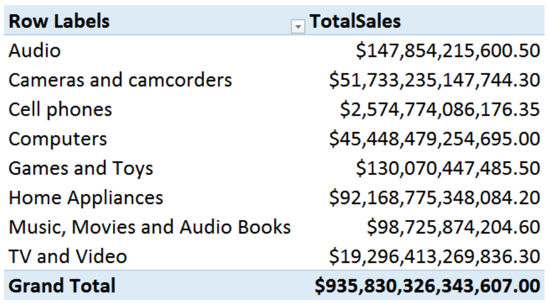
```
GoodCustomers :=
COUNTROWS (
    FILTER (
        Customer,
        [AvgCustomerSale] >= CALCULATE ( [AvgCustomerSale], ALLSELECTED () )
    )
)
```

The reason is that ALLSELECTED does not restore the correct filter context. Understanding the reason will take some time, stay tuned, because we will cover that later during the workshop.

Creating a parameter table

Parameter Table

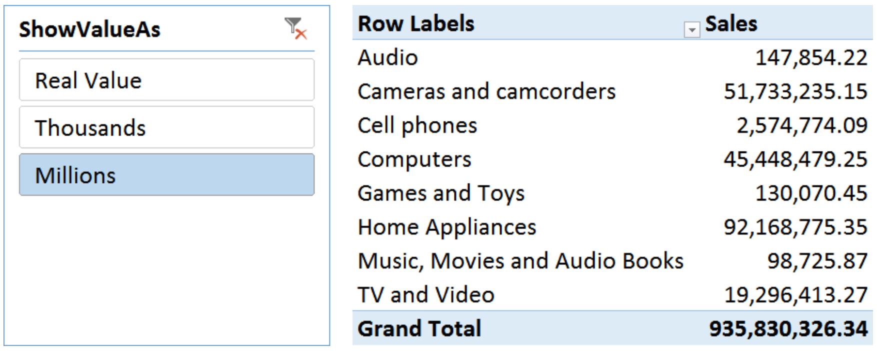
In this exercise, you create a Parameter Table. The idea of a parameter table is to create a table that is unrelated to the rest of the data model, but you use it internally in DAX expressions to modify their behavior. Imagine that you have created a report that shows the sum of sales amount, and because your company sells a large quantity of goods, the numbers shown in the report are big. Because the Contoso database does not suffer from this problem, we have modified the TotalSales measure to return total sales squared, so that numbers are bigger and the scenario is more evident. Using that column in a PivotTable, you get this result:



Row Labels	TotalSales
Audio	$147,854,215,600.50
Cameras and camcorders	$51,733,235,147,744.30
Cell phones	$2,574,774,086,176.35
Computers	$45,448,479,254,695.00
Games and Toys	$130,070,447,485.50
Home Appliances	$92,168,775,348,084.20
Music, Movies and Audio Books	$98,725,874,204.60
TV and Video	$19,296,413,269,836.30
Grand Total	**$935,830,326,343,607.00**

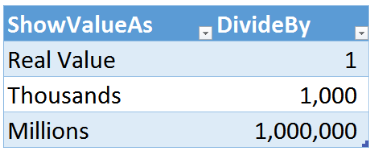
This is the starting point of the exercise. The issue with this report is that the numbers are large, and they tend to be hard to read: are they millions, billions, or trillions? Moreover, they use a lot of space in the report without delivering much information. A common request for this kind of report is to show the numbers using a different scale. For example, you might want to show the values divided by 1,000 or 1 million, so that they result in smaller numbers that still carry out the useful information.

The idea is to let the user decide which scale to use for the report using a slicer:



Row Labels	Sales
Audio	147,854.22
Cameras and camcorders	51,733,235.15
Cell phones	2,574,774.09
Computers	45,448,479.25
Games and Toys	130,070.45
Home Appliances	92,168,775.35
Music, Movies and Audio Books	98,725.87
TV and Video	19,296,413.27
Grand Total	**935,830,326.34**

1. To create this report, the first thing that you need is a table containing the values you want to show on the slicer. This is as easy as creating an Excel table and loading it into the data model. We already created if for you, in the Parameter tab there is a table named **ShowValueAs** that you need to bring into the model using Add to Data Model from the Power Pivot tab of the ribbon.



ShowValueAs	DivideBy
Real Value	1
Thousands	1,000
Millions	1,000,000

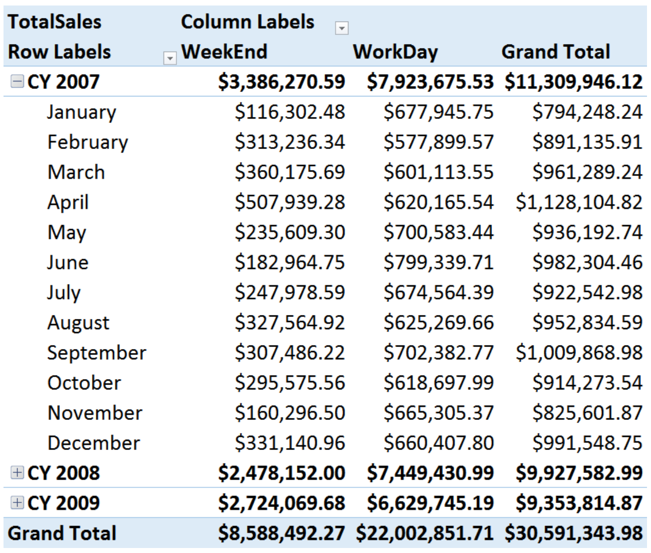
2. Obviously, you cannot create any relationship with this table in the data model, because the FactInternetSales table does not contain any column that can be used to relate to ShowValueAs. Nevertheless, once the table is in the model, you can use the ShowValueAs column as the source for a slicer. At first, you end up with a slicer that does nothing, but you can use some DAX code that will perform the last step, i.e. reading what the user has selected in the slicer and modify the content of the report accordingly, showing real numbers, thousands or millions, so that the slicer becomes useful.
3. Create a new measure Sales to perform this operation:

```
Sales :=
IF (
    HASONEVALUE ( ShowValueAs[ShowValueAs] ),
    [TotalSales] / VALUES ( ShowValueAs[DivideBy] ),
    [TotalSales]
)
```

4. Now, add a slicer to the PivotTable and look how the numbers in Sales change depending on the slicer selection. Before moving to the next exercise, look at the formula and ask yourself in which case the IF takes the ELSE branch, i.e. does not perform any division.
5. You might notice that the slicer does not sort the values in the correct order (we would like to avoid alphabetical sorting for this column and sort them by value); can you try to solve the issue?
6. Finally, does the ShowValuesAs column work if you put it on the rows or on the columns? And, do you know why it behaves as it does?

The next exercise is designed to force you thinking about filter context, row context, iterators and take confidence with the concepts you learned so far. The formulas you will write here can be solved in a much better way once you learn the usage of CALCULATE and the reason we want you to train with these formulas is to better appreciate, later, the CALCULATE function.

The starting point of the exercise is the usual data model with products, sales and the calendar. There is a column in the Date table, named Working Day, which can be "WeekEnd" or "WorkDay" depending whether the day is a working day or not. You can easily build a PivotTable like the following one:

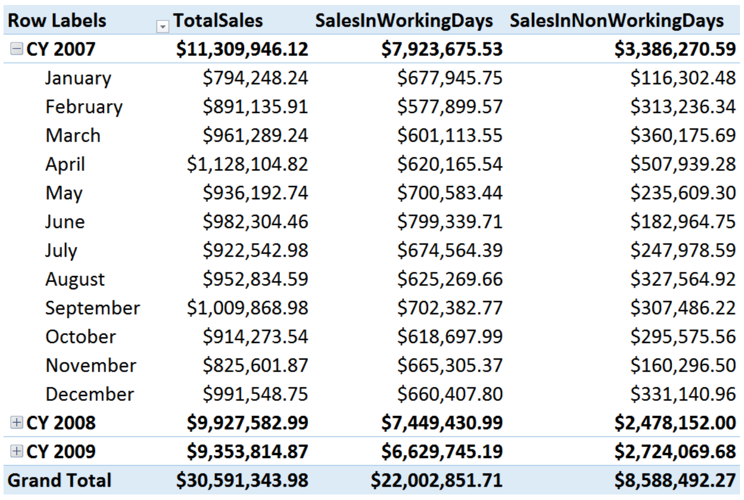


TotalSales	Column Labels		
Row Labels	WeekEnd	WorkDay	Grand Total
CY 2007	$3,386,270.59	$7,923,675.53	$11,309,946.12
January	$116,302.48	$677,945.75	$794,248.24
February	$313,236.34	$577,899.57	$891,135.91
March	$360,175.69	$601,113.55	$961,289.24
April	$507,939.28	$620,165.54	$1,128,104.82
May	$235,609.30	$700,583.44	$936,192.74
June	$182,964.75	$799,339.71	$982,304.46
July	$247,978.59	$674,564.39	$922,542.98
August	$327,564.92	$625,269.66	$952,834.59
September	$307,486.22	$702,382.77	$1,009,868.98
October	$295,575.56	$618,697.99	$914,273.54
November	$160,296.50	$665,305.37	$825,601.87
December	$331,140.96	$660,407.80	$991,548.75
CY 2008	$2,478,152.00	$7,449,430.99	$9,927,582.99
CY 2009	$2,724,069.68	$6,629,745.19	$9,353,814.87
Grand Total	$8,588,492.27	$22,002,851.71	$30,591,343.98

The exercise focus on calculation using contexts. We want to create two measures:

- SalesInWorkingDays: contains the amount sold during working days
- SalesInNonWorkingDays: contains the amount sold during non-working days

We want to be able to create a report like the following one:



Row Labels	TotalSales	SalesInWorkingDays	SalesInNonWorkingDays
CY 2007	$11,309,946.12	$7,923,675.53	$3,386,270.59
January	$794,248.24	$677,945.75	$116,302.48
February	$891,135.91	$577,899.57	$313,236.34
March	$961,289.24	$601,113.55	$360,175.69
April	$1,128,104.82	$620,165.54	$507,939.28
May	$936,192.74	$700,583.44	$235,609.30
June	$982,304.46	$799,339.71	$182,964.75
July	$922,542.98	$674,564.39	$247,978.59
August	$952,834.59	$625,269.66	$327,564.92
September	$1,009,868.98	$702,382.77	$307,486.22
October	$914,273.54	$618,697.99	$295,575.56
November	$825,601.87	$665,305.37	$160,296.50
December	$991,548.75	$660,407.80	$331,140.96
CY 2008	$9,927,582.99	$7,449,430.99	$2,478,152.00
CY 2009	$9,353,814.87	$6,629,745.19	$2,724,069.68
Grand Total	$30,591,343.98	$22,002,851.71	$8,588,492.27

The solution is not very difficult; you can do it by yourself with this hint: if you iterate over all the rows in Sales and sum only the ones where the day is a working day, then you compute SalesInWorkingDays. Testing the opposite condition, you compute SalesInNonWorkingDays.

That said, try it by yourself. If after 10 minutes you are still lost, then you can look at the solution in the next formulas:

```
SalesInWorkingDays :=
SUMX (
    Sales,
    IF (
        RELATED ( 'Date'[Working Day] ) = "WorkDay",
        Sales[Quantity] * Sales[Net Price],
        0
    )
)
```

And, for non-working days:

```
SalesInNonWorkingDays :=
SUMX (
    Sales,
    IF (
        NOT ( RELATED ( 'Date'[Working Day] ) = "WorkDay" ),
        Sales[Quantity] * Sales[Net Price],
        0
    )
)
```

Did you manage to write the formulas in this way? There are other possible solutions to the same problem, so chances are that you found a different one; you can discuss your solution with the teacher to look at it and see which one is the best one.

It is now time to start making some exercise with CALCULATE. Let us start the simple way: by using CALCULATE we rewrite the two measures created in the previous section. The two measures are SalesInWorkingDays and SalesInNonWorkingDays.

Your task is indeed an easy one: rewrite the same formulas using CALCULATE instead of the SUMX iterator. You have 10 minutes to solve this small exercise. After that, look at the solution here. Even if you fail, those 10 minutes will be well spent.

```
SalesInWorkingDays :=
CALCULATE (
    [TotalSales],
    'Date'[Working Day] = "WorkDay"
)

SalesInNonWorkingDays :=
CALCULATE (
    [TotalSales],
    NOT ( 'Date'[Working Day] = "WorkDay" )
)
```

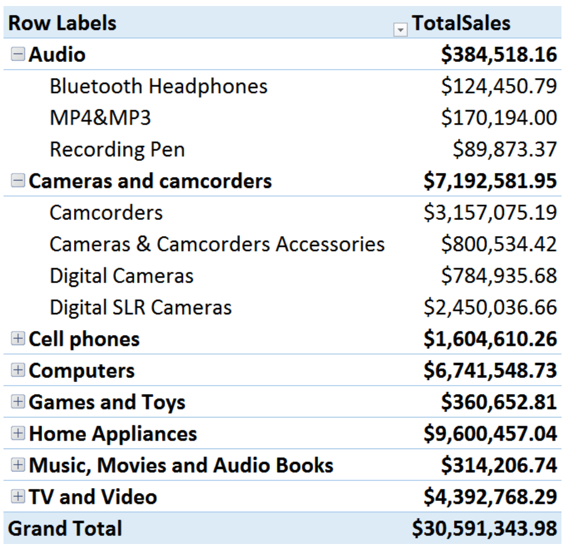
You can appreciate that these formulas are much easier to write and to understand, when compared with the previous versions, which is more complex to write without CALCULATE. It is interesting to note that the filter is on Date and you sum Sales, can you explain why it works? Hint: remember the direction of automatic propagation of the filter context and the fact that CALCULATE creates a new filter context.

Understanding CALCULATE

Basic CALCULATE

The most common scenario where CALCULATE is useful is when you need to compute ratios and percentages. Before learning more advanced topics, it is important to spend a few minutes seeing with your eyes what happens with basic CALCULATE. For educational purposes, in this model we denormalized category and subcategory in Product, using two calculated columns. The exercise would have been tougher if we did not, involving AutoExists logic, that we will cover only later.

You start with a simple model, containing categories, subcategories and total sales:

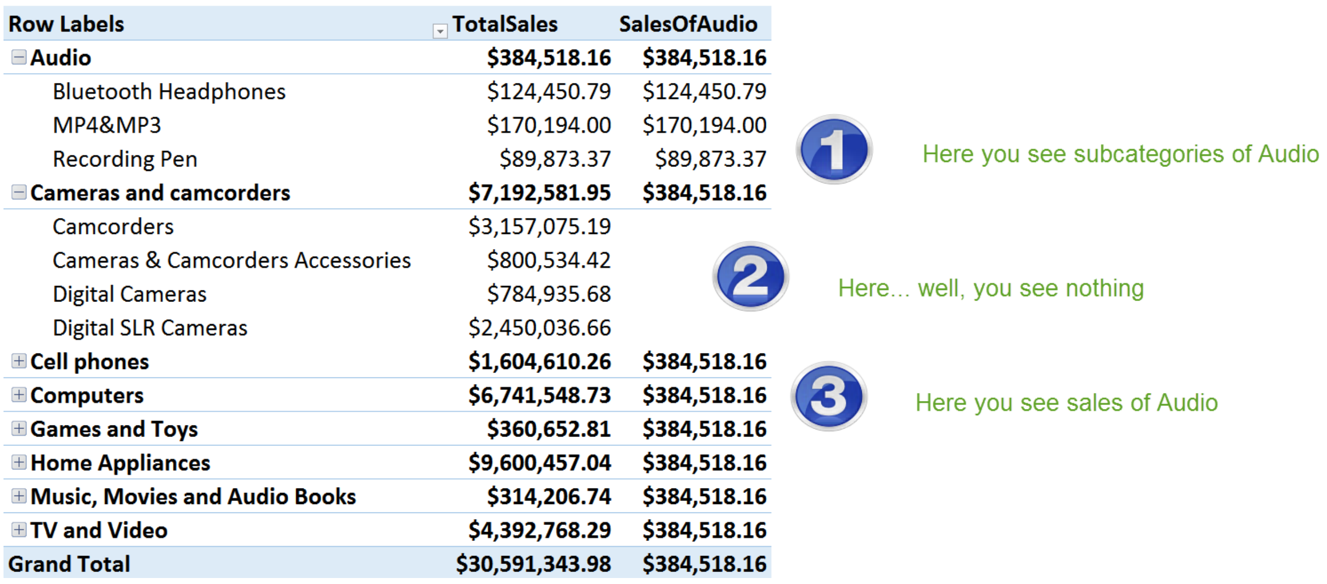


Row Labels	TotalSales
⊟ **Audio**	**$384,518.16**
Bluetooth Headphones	$124,450.79
MP4&MP3	$170,194.00
Recording Pen	$89,873.37
⊟ **Cameras and camcorders**	**$7,192,581.95**
Camcorders	$3,157,075.19
Cameras & Camcorders Accessories	$800,534.42
Digital Cameras	$784,935.68
Digital SLR Cameras	$2,450,036.66
⊞ **Cell phones**	**$1,604,610.26**
⊞ **Computers**	**$6,741,548.73**
⊞ **Games and Toys**	**$360,652.81**
⊞ **Home Appliances**	**$9,600,457.04**
⊞ **Music, Movies and Audio Books**	**$314,206.74**
⊞ **TV and Video**	**$4,392,768.29**
Grand Total	**$30,591,343.98**

1. Create a new measure with this formula:

```
SalesOfAudio :=
CALCULATE (
    [TotalSales],
    Product[Product Category] = "Audio"
)
```

Put the measure in the PivotTable, look at the result and focus on the three highlighted parts of the result:



Row Labels	TotalSales	SalesOfAudio
⊟ **Audio**	**$384,518.16**	**$384,518.16**
Bluetooth Headphones	$124,450.79	$124,450.79
MP4&MP3	$170,194.00	$170,194.00
Recording Pen	$89,873.37	$89,873.37
⊟ **Cameras and camcorders**	**$7,192,581.95**	**$384,518.16**
Camcorders	$3,157,075.19	
Cameras & Camcorders Accessories	$800,534.42	
Digital Cameras	$784,935.68	
Digital SLR Cameras	$2,450,036.66	
⊞ **Cell phones**	**$1,604,610.26**	**$384,518.16**
⊞ **Computers**	**$6,741,548.73**	**$384,518.16**
⊞ **Games and Toys**	**$360,652.81**	**$384,518.16**
⊞ **Home Appliances**	**$9,600,457.04**	**$384,518.16**
⊞ **Music, Movies and Audio Books**	**$314,206.74**	**$384,518.16**
⊞ **TV and Video**	**$4,392,768.29**	**$384,518.16**
Grand Total	**$30,591,343.98**	**$384,518.16**

1
Here you see subcategories of Audio
2
Here… well, you see nothing
3
Here you see sales of Audio

Answer these simple questions:

- Why point (3) shows the total of Audio, if Computers is on the rows?

  It is because the filter context created by the rows is overridden by the new filter context generated by CALCULATE

- Why point (2) shows nothing? How is it possible that the total of blanks results in the sum of Audio? Hint: write down on a piece of paper the filter context of the cells, and simulate the CALCULATE operation

    Because there is a filter on the product subcategory. We override the filter on category, but the filter on subcategory is still there and there is no product which is a Camera and, at the same time, an Audio.

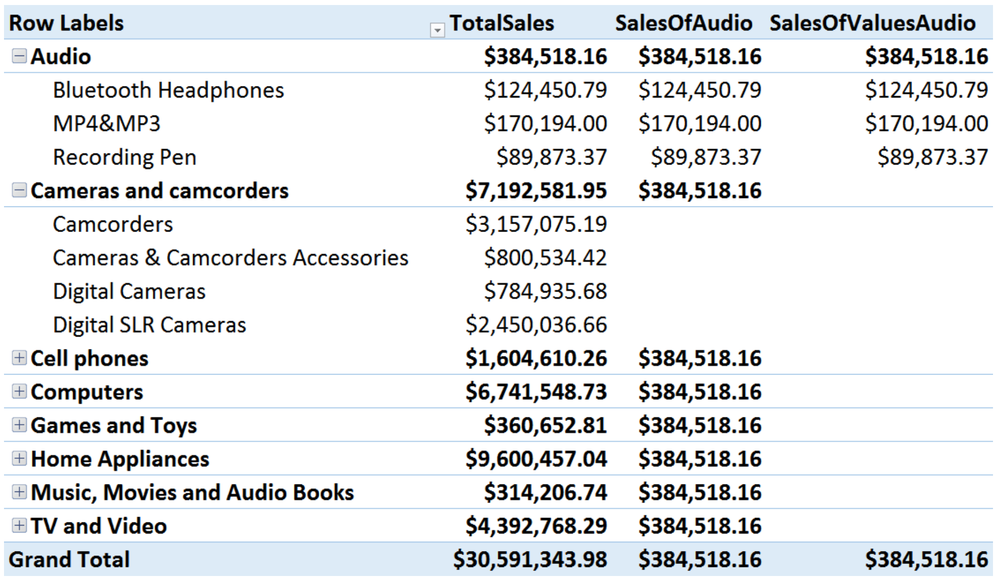
- Why, if at the subcategory level in point (2) there are blanks, at the subcategory level in point (1) there are values, and they are identical to the sales amount of the subcategories?

    Because the filter on category introduced by CALCULATE, put in and with the existing filter on subcategory, generates and empty set in (2) and a set with some elements in (1).

2. Create a new formula:

```
SalesOfValuesAudio:=
CALCULATE (
    [TotalSales],
    Product[Product Category] = "Audio",
    VALUES ( Product[Product Category] )
)
```

Take a look at the result:



Row Labels	TotalSales	SalesOfAudio	SalesOfValuesAudio
⊟ **Audio**	**$384,518.16**	**$384,518.16**	**$384,518.16**
Bluetooth Headphones	$124,450.79	$124,450.79	$124,450.79
MP4&MP3	$170,194.00	$170,194.00	$170,194.00
Recording Pen	$89,873.37	$89,873.37	$89,873.37
⊟ **Cameras and camcorders**	**$7,192,581.95**	**$384,518.16**	
Camcorders	$3,157,075.19		
Cameras & Camcorders Accessories	$800,534.42		
Digital Cameras	$784,935.68		
Digital SLR Cameras	$2,450,036.66		
⊞ **Cell phones**	**$1,604,610.26**	**$384,518.16**	
⊞ **Computers**	**$6,741,548.73**	**$384,518.16**	
⊞ **Games and Toys**	**$360,652.81**	**$384,518.16**	
⊞ **Home Appliances**	**$9,600,457.04**	**$384,518.16**	
⊞ **Music, Movies and Audio Books**	**$314,206.74**	**$384,518.16**	
⊞ **TV and Video**	**$4,392,768.29**	**$384,518.16**	
Grand Total	**$30,591,343.98**	**$384,518.16**	**$384,518.16**

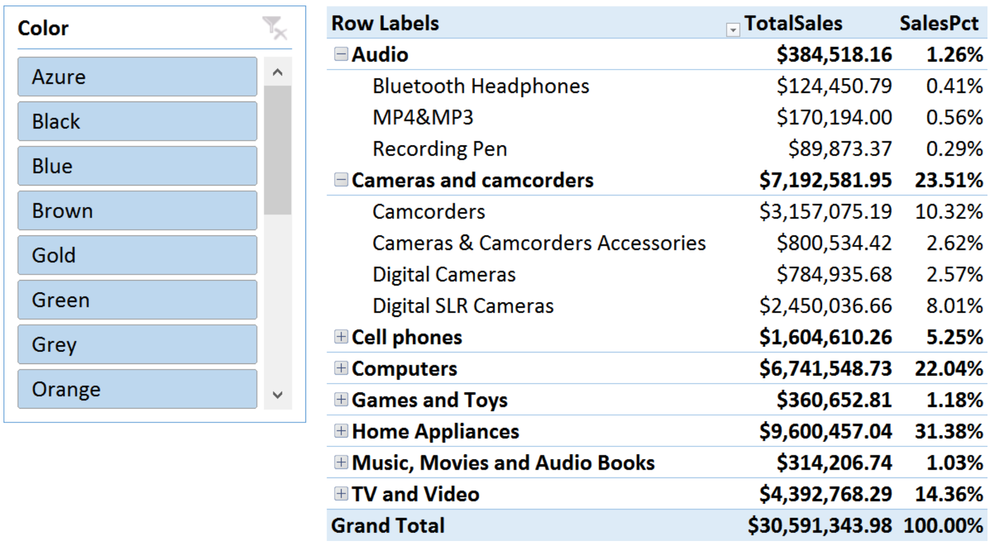
Why are the number different, now? Can you explain what happened to the row in Computers, which contained a value and now is blank? Hint: the conditions in CALCULATE are put in AND together… consider the combined effects of all the filters arguments passed to CALCULATE.

3. It is time to use CALCULATE for something more useful. You want to show the value of sales as a percentage of the grand total instead than a value. Create a SalesPct measure returning such a calculation. As usual, spend a few minutes thinking at the formula, then move on with the solution, if you did not find it.

   The formula is not very hard:

```
SalesPct :=
DIVIDE (
    [TotalSales],
    CALCULATE (
        [TotalSales],
        ALL ( Product )
    )
)
```

Now, put the field in a PivotTable and add a slicer on Color, this is what you see:

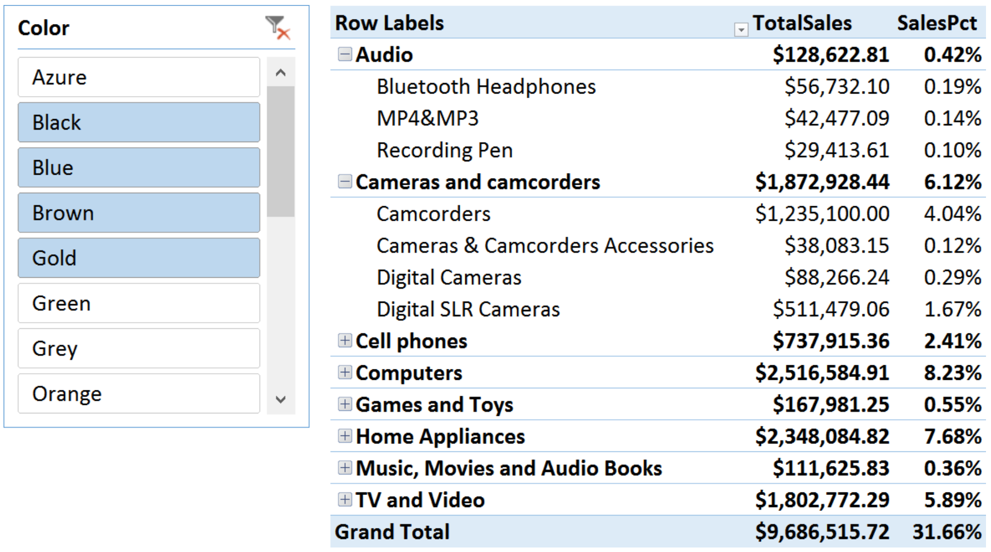


Color

- Azure
- Black
- Blue
- Brown
- Gold
- Green
- Grey
- Orange

Row Labels	TotalSales	SalesPct
⊟Audio	**$384,518.16**	**1.26%**
Bluetooth Headphones	$124,450.79	0.41%
MP4&MP3	$170,194.00	0.56%
Recording Pen	$89,873.37	0.29%
⊟Cameras and camcorders	**$7,192,581.95**	**23.51%**
Camcorders	$3,157,075.19	10.32%
Cameras & Camcorders Accessories	$800,534.42	2.62%
Digital Cameras	$784,935.68	2.57%
Digital SLR Cameras	$2,450,036.66	8.01%
⊞Cell phones	**$1,604,610.26**	**5.25%**
⊞Computers	**$6,741,548.73**	**22.04%**
⊞Games and Toys	**$360,652.81**	**1.18%**
⊞Home Appliances	**$9,600,457.04**	**31.38%**
⊞Music, Movies and Audio Books	**$314,206.74**	**1.03%**
⊞TV and Video	**$4,392,768.29**	**14.36%**
Grand Total	**$30,591,343.98**	**100.00%**

Looks fine. Now, select some colors from the slicer and look at the result:



Color

- Azure
- [✓ Black]
- [✓ Blue]
- [✓ Brown]
- [✓ Gold]
- Green
- Grey
- Orange

Row Labels	TotalSales	SalesPct
⊟Audio	**$128,622.81**	**0.42%**
Bluetooth Headphones	$56,732.10	0.19%
MP4&MP3	$42,477.09	0.14%
Recording Pen	$29,413.61	0.10%
⊟Cameras and camcorders	**$1,872,928.44**	**6.12%**
Camcorders	$1,235,100.00	4.04%
Cameras & Camcorders Accessories	$38,083.15	0.12%
Digital Cameras	$88,266.24	0.29%
Digital SLR Cameras	$511,479.06	1.67%
⊞Cell phones	**$737,915.36**	**2.41%**
⊞Computers	**$2,516,584.91**	**8.23%**
⊞Games and Toys	**$167,981.25**	**0.55%**
⊞Home Appliances	**$2,348,084.82**	**7.68%**
⊞Music, Movies and Audio Books	**$111,625.83**	**0.36%**
⊞TV and Video	**$1,802,772.29**	**5.89%**
Grand Total	**$9,686,515.72**	**31.66%**

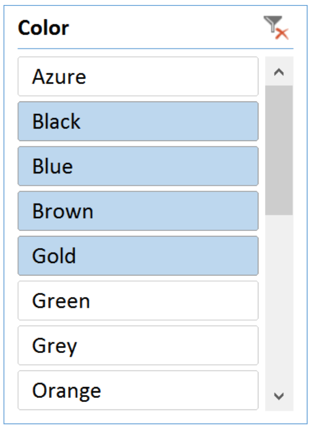
What does 31.66% means as the grand total percentage? Can you explain what happened? And, once you got the explanation, can you fix it? Take your time; it is worthwhile!

Anyway, here is the solution:

```
SalesPct :=
DIVIDE (
    [TotalSales],
    CALCULATE (
        [TotalSales],
        ALLSELECTED ()
    )
)
```

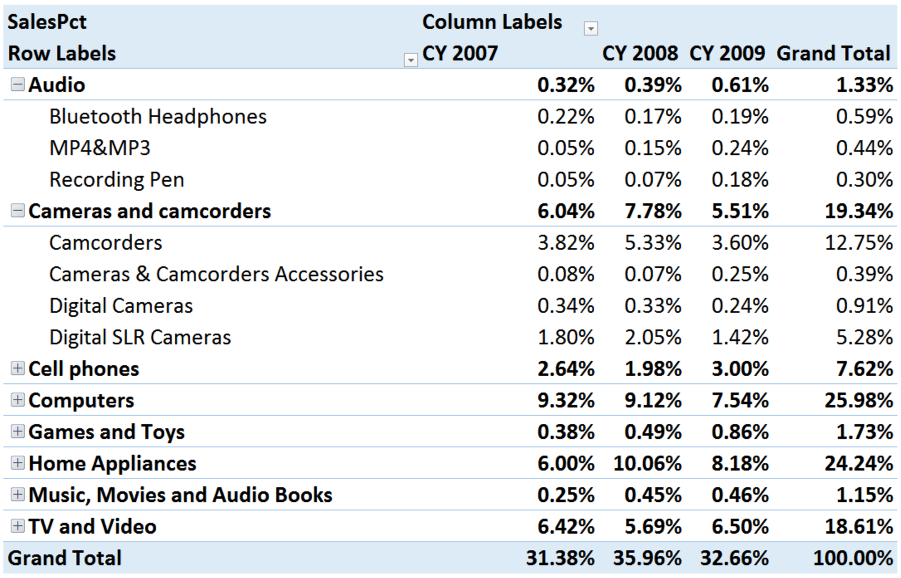
Why is it working with ALLSELECTED? *Because ALLSELECTED always returns the original filter context of the PivotTable.*

4. Ok, now remove TotalSales from the columns of the PivotTable, replacing it with the Calendar Year from Date



Color

- Azure
- Black
- Blue
- Brown
- Gold
- Green
- Grey
- Orange



SalesPct	Column Labels			
Row Labels	CY 2007	CY 2008	CY 2009	Grand Total
Audio	**0.32%**	**0.39%**	**0.61%**	**1.33%**
Bluetooth Headphones	0.22%	0.17%	0.19%	0.59%
MP4&MP3	0.05%	0.15%	0.24%	0.44%
Recording Pen	0.05%	0.07%	0.18%	0.30%
Cameras and camcorders	**6.04%**	**7.78%**	**5.51%**	**19.34%**
Camcorders	3.82%	5.33%	3.60%	12.75%
Cameras & Camcorders Accessories	0.08%	0.07%	0.25%	0.39%
Digital Cameras	0.34%	0.33%	0.24%	0.91%
Digital SLR Cameras	1.80%	2.05%	1.42%	5.28%
Cell phones	**2.64%**	**1.98%**	**3.00%**	**7.62%**
Computers	**9.32%**	**9.12%**	**7.54%**	**25.98%**
Games and Toys	**0.38%**	**0.49%**	**0.86%**	**1.73%**
Home Appliances	**6.00%**	**10.06%**	**8.18%**	**24.24%**
Music, Movies and Audio Books	**0.25%**	**0.45%**	**0.46%**	**1.15%**
TV and Video	**6.42%**	**5.69%**	**6.50%**	**18.61%**
Grand Total	**31.38%**	**35.96%**	**32.66%**	**100.00%**

5. The percentages are over the entire period. What if you want them year-by-year, so that the grand total of each column shows 100%?

   You have a couple of options here:

```
SalesPct :=
DIVIDE (
    [TotalSales],
    CALCULATE (
        [TotalSales],
        ALLSELECTED (),
        VALUES ( 'Date'[Calendar Year] )
    )
)
```

   Or, if you prefer:

```
SalesPct :=
DIVIDE (
    [TotalSales],
    CALCULATE (
        [TotalSales],
        ALLSELECTED ( Product )
    )
)
```

   Both formulas work for this specific report, but their meaning is completely different, can you spot the differences between the two formulas and explain why they provide the same result in this report?

   The first formula shows the sales for the current year, restoring all filters on all the tables in the model to the original context. The second formula shows the sales for all the products in the original filter context, retaining all the other filters. In this report, the peculiar shape of the report makes the two formula compute the same value, but if you put a filter, for example on the customer education, then the two formulas will start computing different values.

Lab 4
Working with iterators

Compute sales per working day

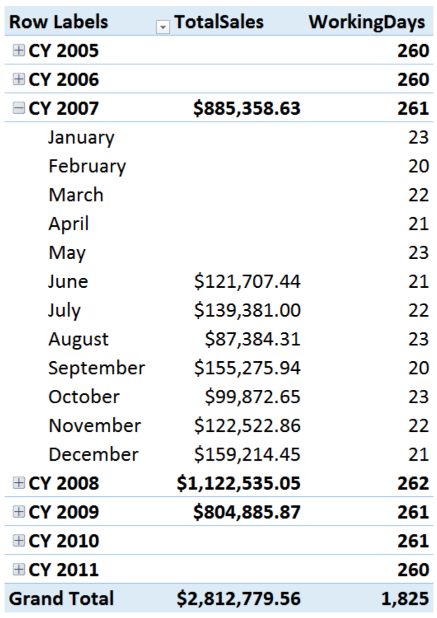
Sales in Working Days

This goal of the exercise is showing how to perform some simple computation with working days. As you are going to see, even simple calculation might hide complex scenarios, the key of which is, as usual, the understanding of evaluation contexts.

1. In the initial solution, you have a standard model with a calendar table that contains information about working and non-working days. You want to:
   - Compute the number of working days in different periods
   - Compute the value of sales amount divided by the number of working days
2. Because the calendar table contains a column indicating when a day is a working day or not, computing the number of working days is easy:

```
WorkingDays :=
SUMX (
    'Date',
    IF ( 'Date'[Working Day] = "WorkDay", 1, 0 )
)
```

With this new column, you can build a first PivotTable, like the following one:

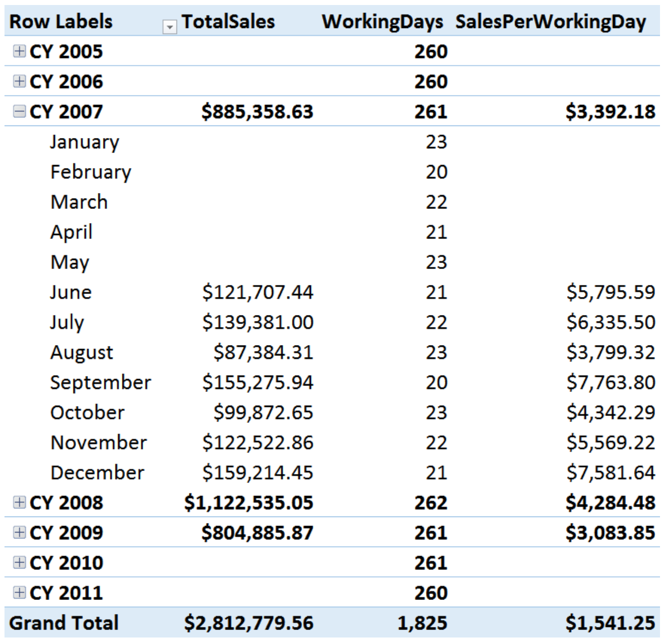


Row Labels	TotalSales	WorkingDays
⊞CY 2005		260
⊞CY 2006		260
⊟CY 2007	$885,358.63	261
January		23
February		20
March		22
April		21
May		23
June	$121,707.44	21
July	$139,381.00	22
August	$87,384.31	23
September	$155,275.94	20
October	$99,872.65	23
November	$122,522.86	22
December	$159,214.45	21
⊞CY 2008	$1,122,535.05	262
⊞CY 2009	$804,885.87	261
⊞CY 2010		261
⊞CY 2011		260
Grand Total	$2,812,779.56	1,825

3. Computing the sales amount divided by the number of working days is easy too:

```
SalesPerWorkingDay := [TotalSales] / [WorkingDays]
```

Adding it to the PivotTable produces this result:



Row Labels	TotalSales	WorkingDays	SalesPerWorkingDay
⊞ **CY 2005**		**260**	
⊞ **CY 2006**		**260**	
⊟ **CY 2007**	**$885,358.63**	**261**	**$3,392.18**
January		23	
February		20	
March		22	
April		21	
May		23	
June	$121,707.44	21	$5,795.59
July	$139,381.00	22	$6,335.50
August	$87,384.31	23	$3,799.32
September	$155,275.94	20	$7,763.80
October	$99,872.65	23	$4,342.29
November	$122,522.86	22	$5,569.22
December	$159,214.45	21	$7,581.64
⊞ **CY 2008**	**$1,122,535.05**	**262**	**$4,284.48**
⊞ **CY 2009**	**$804,885.87**	**261**	**$3,083.85**
⊞ **CY 2010**		**261**	
⊞ **CY 2011**		**260**	
Grand Total	**$2,812,779.56**	**1,825**	**$1,541.25**

The numbers look correct at the month level, but the year level and the grand total show wrong values, you can easily check it for the year 2007 and, worse, at the grand total. Before moving on, what it the reason for the wrong value?

The problem is that years like 2007 account sales for only 7 months, not for 12 months. Thus, you should compute the working days only for months where there are sales. Otherwise, the computation at the year level will always be lower than expected.

Once you detected the reason, how do you solve this?

A common first trial is the following:

```
SalesPerWorkingDay := IF ( [TotalSales] > 0, [TotalSales] / [WorkingDays] )
```

However, this does not change in any way the result. Do you understand why?

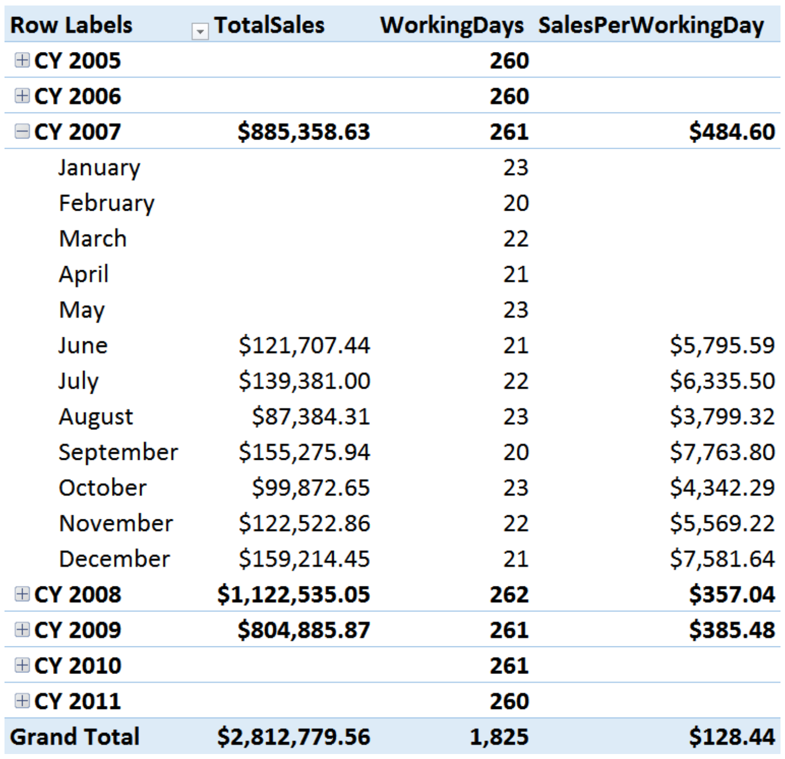
The reason is that the expression is computed at the year level and, there, [Sales] is always greater than zero. You cannot compute such an expression at the aggregate level, you need to iterate each month and repeat the expression at the month level. Thus, the correct formulation is:

```
SalesPerWorkingDay :=
DIVIDE (
    [TotalSales],
    SUMX (
        VALUES ( 'Date'[Month] ),
        IF ( [TotalSales] > 0, [WorkingDays] )
    )
)
```

4. Now, before moving on, try this equivalent expression:

```
SalesPerWorkingDay :=
DIVIDE (
    [TotalSales],
    SUMX (
        VALUES ( 'Date'[Month] ),
        IF (
            [TotalSales] > 0,
            SUMX (
                'Date',
                IF ('Date'[Working Day] = "WorkDay", 1, 0 )
            )
        )
    )
)
```

The result will be this one:



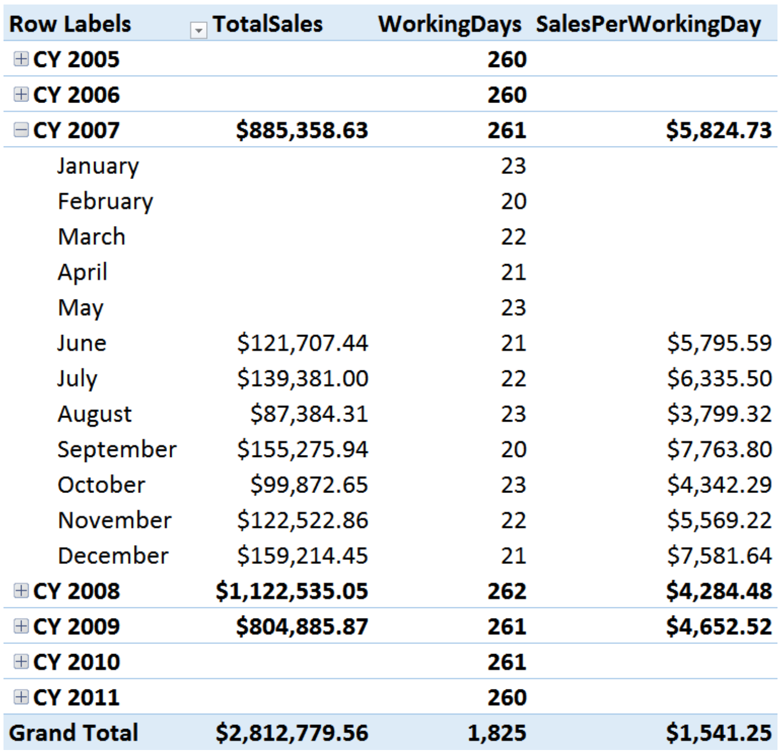
Row Labels	TotalSales	WorkingDays	SalesPerWorkingDay
⊞ **CY 2005**		**260**	
⊞ **CY 2006**		**260**	
⊟ **CY 2007**	**$885,358.63**	**261**	**$484.60**
January		23	
February		20	
March		22	
April		21	
May		23	
June	$121,707.44	21	$5,795.59
July	$139,381.00	22	$6,335.50
August	$87,384.31	23	$3,799.32
September	$155,275.94	20	$7,763.80
October	$99,872.65	23	$4,342.29
November	$122,522.86	22	$5,569.22
December	$159,214.45	21	$7,581.64
⊞ **CY 2008**	**$1,122,535.05**	**262**	**$357.04**
⊞ **CY 2009**	**$804,885.87**	**261**	**$385.48**
⊞ **CY 2010**		**261**	
⊞ **CY 2011**		**260**	
Grand Total	**$2,812,779.56**	**1,825**	**$128.44**

Can you explain why, this time, the year level is so a low number, even worse than before?

Hint: think at row and filter context, and try to compute the expressions by yourself, writing down the filter context and the row context during the full expression evaluation.

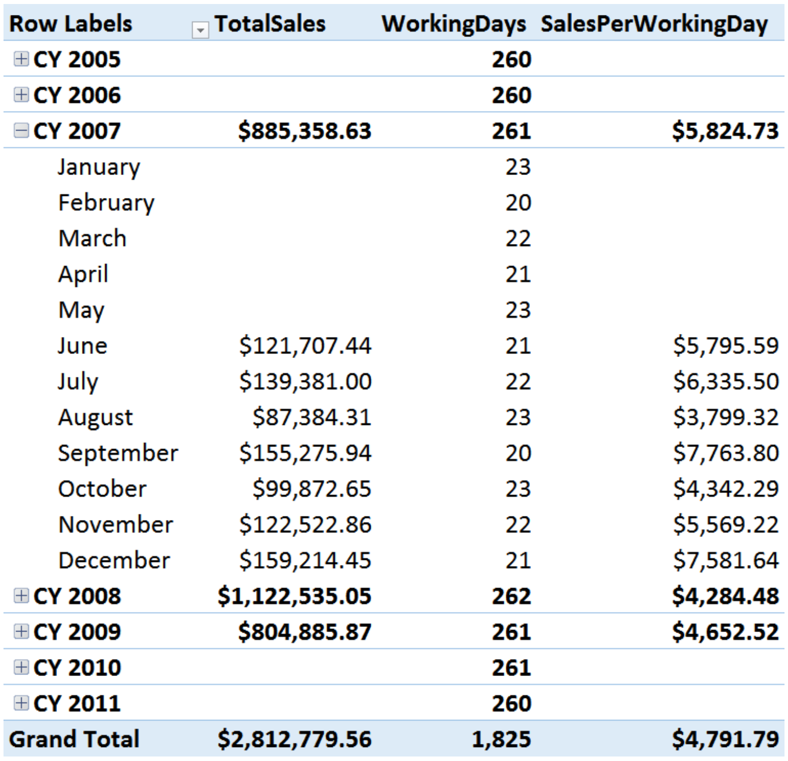
This formula, as you have noticed, does not work because of the missing context transition. We decided to show it anyway, because it is a very common error forgetting the automatic CALCULATE surrounding any measure evaluation.

5. With the correct formula, the PivotTable looks like this:



Row Labels	TotalSales	WorkingDays	SalesPerWorkingDay
⊞ **CY 2005**		**260**	
⊞ **CY 2006**		**260**	
⊟ **CY 2007**	**$885,358.63**	**261**	**$5,824.73**
January		23	
February		20	
March		22	
April		21	
May		23	
June	$121,707.44	21	$5,795.59
July	$139,381.00	22	$6,335.50
August	$87,384.31	23	$3,799.32
September	$155,275.94	20	$7,763.80
October	$99,872.65	23	$4,342.29
November	$122,522.86	22	$5,569.22
December	$159,214.45	21	$7,581.64
⊞ **CY 2008**	**$1,122,535.05**	**262**	**$4,284.48**
⊞ **CY 2009**	**$804,885.87**	**261**	**$4,652.52**
⊞ **CY 2010**		**261**	
⊞ **CY 2011**		**260**	
Grand Total	**$2,812,779.56**	**1,825**	**$1,541.25**

Even if now the year level is now correct, the grand total level is not. The reason is the same, can you write the correct formula that shows the result as it should be?



Row Labels	TotalSales	WorkingDays	SalesPerWorkingDay
⊞ **CY 2005**		**260**	
⊞ **CY 2006**		**260**	
⊟ **CY 2007**	**$885,358.63**	**261**	**$5,824.73**
January		23	
February		20	
March		22	
April		21	
May		23	
June	$121,707.44	21	$5,795.59
July	$139,381.00	22	$6,335.50
August	$87,384.31	23	$3,799.32
September	$155,275.94	20	$7,763.80
October	$99,872.65	23	$4,342.29
November	$122,522.86	22	$5,569.22
December	$159,214.45	21	$7,581.64
⊞ **CY 2008**	**$1,122,535.05**	**262**	**$4,284.48**
⊞ **CY 2009**	**$804,885.87**	**261**	**$4,652.52**
⊞ **CY 2010**		**261**	
⊞ **CY 2011**		**260**	
Grand Total	**$2,812,779.56**	**1,825**	**$4,791.79**

Try to think at the result by yourself, and then compare it with the proposed solution:

```
SalesPerWorkingDay :=
DIVIDE (
    [TotalSales],
    SUMX (
        VALUES ( 'Date'[Calendar Year] ),
        SUMX (
            VALUES ( 'Date'[Month] ),
            IF ( [TotalSales] > 0, [WorkingDays] )
        )
    )
)
```

You see that two different iterations happening: one over the year and one over the month. They are both needed to capture the month level at which the calculation of working days should happen.

Ranking Customers (Dynamic)

Ranking.xlsx

During the course, you have seen some examples of the RANKX function. This exercise makes you practice this function and learn its fundamentals.

The goal of the exercise is to create a measure that provides the ranking of customers based on profitability. By profitability, we mean simply sales minus costs.

1. We need to compute:

    - A measure that provides the profitability
    - A measure that transforms this profitability in a rank position

    You can start writing the formulas by yourself. In the following part of the exercise, you can see the solution step-by-step.
    Profit is very easy to compute, by simply subtracting costs from sales:

    ```
    Profit :=
    SUMX (
        Sales,
        Sales[Quantity] * (Sales[Net Price] - Sales[Unit Cost] )
    )
    ```

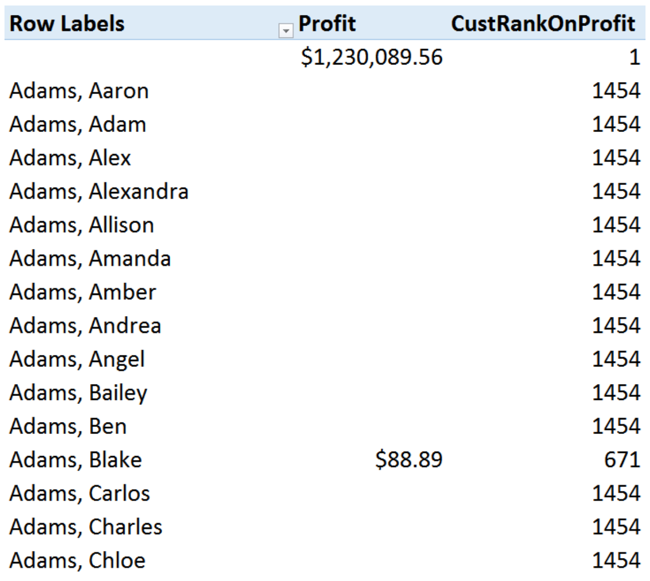
    Ranking is a bit more complex but, if you remember the topics discussed in the course, the formula is not very hard:

    ```
    CustRankOnProfit :=
    RANKX (
        ALLSELECTED ( Customer ),
        [Profit]
    )
    ```

    It is important to note a few points, here:

    - The usage of ALLSELECTED on Customer, to see all the customers visible in the query context
    - The fact that we used the [Profit] measure, which implies the automatic CALCULATE needed to fire context transition; without it, the result would not be correct

2. You can now create a PivotTable like the following one to see the fields in action but, as you see, there is a huge number of customers with no sales that make the PivotTable nearly impossible to use:



Row Labels	Profit	CustRankOnProfit
	$1,230,089.56	1
Adams, Aaron		1454
Adams, Adam		1454
Adams, Alex		1454
Adams, Alexandra		1454
Adams, Allison		1454
Adams, Amanda		1454
Adams, Amber		1454
Adams, Andrea		1454
Adams, Angel		1454
Adams, Bailey		1454
Adams, Ben		1454
Adams, Blake	$88.89	671
Adams, Carlos		1454
Adams, Charles		1454
Adams, Chloe		1454

Please note that we used the customer name on the rows.

3. Can you fix the issue, showing the ranking only for rows where there are sales? The following definition of CustRankOnProfit contains the solution:

```
CustRankOnProfit :=
IF (
    [Profit] > 0,
    RANKX (
        ALLSELECTED ( Customer ),
        [Profit]
    )
)
```

Ranking Customers (Static)

Static Ranking.xlsx

In the previous exercise, you created a ranking measure that is dynamic, i.e. it computes the ranking of the customer based upon your selection. This is fine but, sometimes, you want the ranking to be static. Said in other words: you want to identify the top customers by profitability and then look at their behavior, analyzing sales or doing other kinds of reporting.

The scenario now is different: you want a calculated column that contains the ranking of the customer based on profitability. This is an easy exercise, so we strongly suggest you trying to solve it by your own before looking at the solution.

1. The first thing you need to compute is the profitability per customer. This time it need to be static, so you need a calculated column in the Customer table.

```
Customer[Profit] =
SUMX (
    RELATEDTABLE ( Sales ),
    Sales[Quantity] * (Sales[Net Price] - Sales[Unit Cost] )
)
```

2. Then, you need the ranking calculated column:

```
Customer[RankOnProfit] = RANKX ( Customer, Customer[Profit] )
```

Lab 5
Time Intelligence

Work with YTD, MTD, PARALLELPERIOD formulas

Time Intelligence

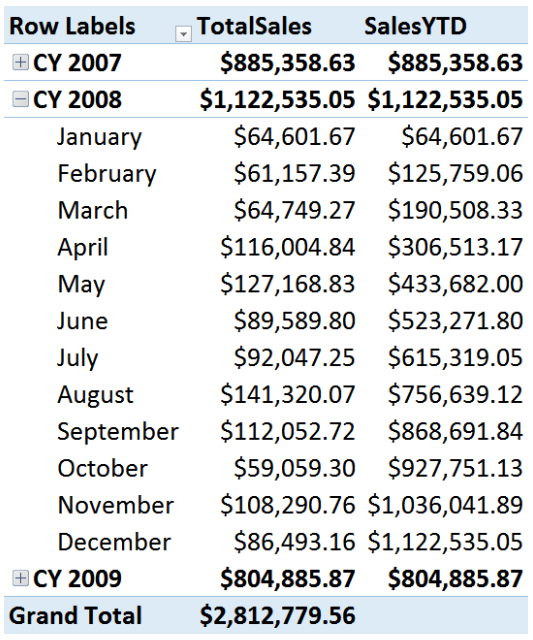
In this exercise, you are going to make some practice with the standard time intelligence functions in DAX.

You want to produce a report that shows, during the year, the comparison between the YTD of sales in the current year against the YTD of the previous year, both as a value and as a percentage.

1. You start with the classical data model, with a measure (Sales) that computes the sum of sales amount.
2. The first measure you need to write is the YTD for the current year:

```
SalesYTD := CALCULATE ( [TotalSales], DATESYTD ( 'Date'[Date] ) )
```

Using this measure, you can start to create a PivotTable:

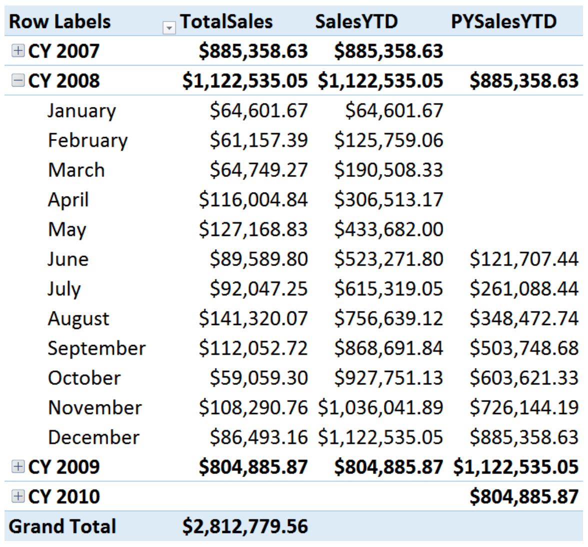


Row Labels	TotalSales	SalesYTD
⊞ CY 2007	**$885,358.63**	**$885,358.63**
⊟ CY 2008	**$1,122,535.05**	**$1,122,535.05**
January	$64,601.67	$64,601.67
February	$61,157.39	$125,759.06
March	$64,749.27	$190,508.33
April	$116,004.84	$306,513.17
May	$127,168.83	$433,682.00
June	$89,589.80	$523,271.80
July	$92,047.25	$615,319.05
August	$141,320.07	$756,639.12
September	$112,052.72	$868,691.84
October	$59,059.30	$927,751.13
November	$108,290.76	$1,036,041.89
December	$86,493.16	$1,122,535.05
⊞ CY 2009	**$804,885.87**	**$804,885.87**
Grand Total	**$2,812,779.56**	

3. SalesYTD shows the YTD of the current year, you need to compare it with the YTD of the previous year. This requires you to write a new formula that computes YTD for the previous year.

   There are many ways to express the formula. Our proposal is the following:

```
PYSalesYTD:=
CALCULATE (
    [TotalSales],
    SAMEPERIODLASTYEAR ( DATESYTD ( 'Date'[Date] ) )
)
```



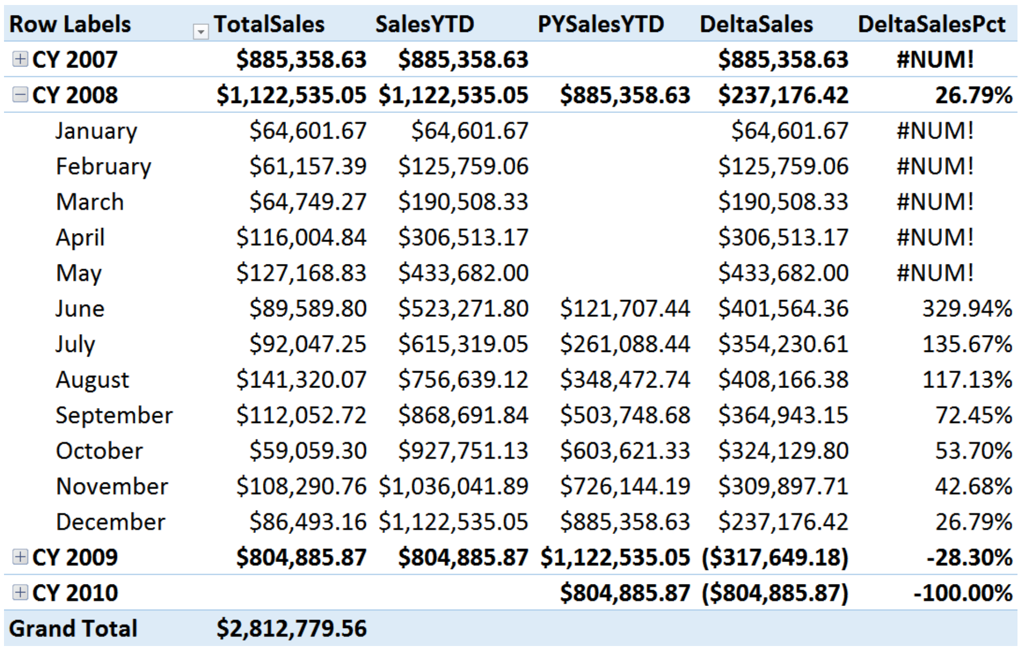
Row Labels	TotalSales	SalesYTD	PYSalesYTD
CY 2007	**$885,358.63**	**$885,358.63**	
CY 2008	**$1,122,535.05**	**$1,122,535.05**	**$885,358.63**
January	$64,601.67	$64,601.67	
February	$61,157.39	$125,759.06	
March	$64,749.27	$190,508.33	
April	$116,004.84	$306,513.17	
May	$127,168.83	$433,682.00	
June	$89,589.80	$523,271.80	$121,707.44
July	$92,047.25	$615,319.05	$261,088.44
August	$141,320.07	$756,639.12	$348,472.74
September	$112,052.72	$868,691.84	$503,748.68
October	$59,059.30	$927,751.13	$603,621.33
November	$108,290.76	$1,036,041.89	$726,144.19
December	$86,493.16	$1,122,535.05	$885,358.63
CY 2009	**$804,885.87**	**$804,885.87**	**$1,122,535.05**
CY 2010			**$804,885.87**
Grand Total	**$2,812,779.56**		

You might have found a different formulation, in that case, look at the differences and discuss them with the teacher.

4. Now that you have both values, it is easy to compute the difference, both in value and as a percentage.

```
DeltaSales    := [SalesYTD] - [PYSalesYTD]
DeltaSalesPct := [DeltaSales] / [PYSalesYTD]
```

You need to format the measures in a proper way in order to obtain the desired report:

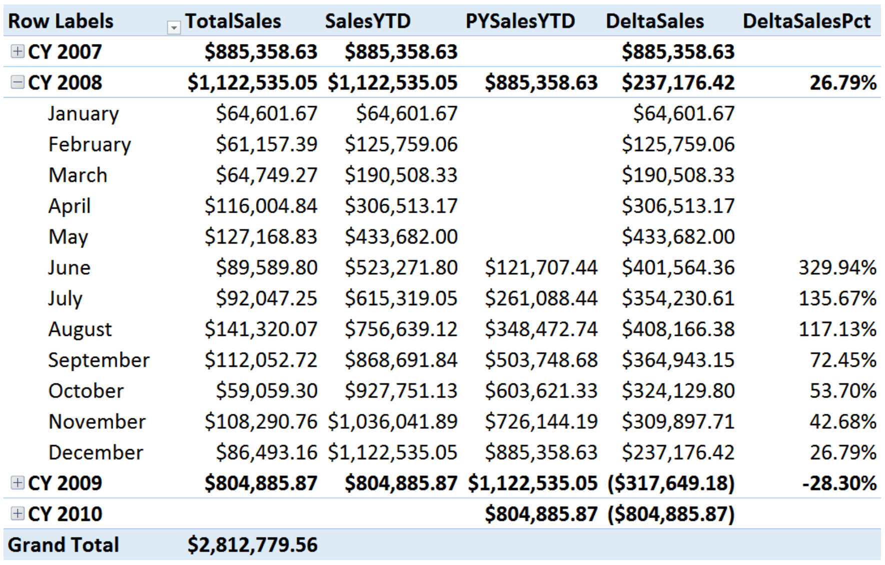


Row Labels	TotalSales	SalesYTD	PYSalesYTD	DeltaSales	DeltaSalesPct
CY 2007	**$885,358.63**	**$885,358.63**		**$885,358.63**	**#NUM!**
CY 2008	**$1,122,535.05**	**$1,122,535.05**	**$885,358.63**	**$237,176.42**	**26.79%**
January	$64,601.67	$64,601.67		$64,601.67	#NUM!
February	$61,157.39	$125,759.06		$125,759.06	#NUM!
March	$64,749.27	$190,508.33		$190,508.33	#NUM!
April	$116,004.84	$306,513.17		$306,513.17	#NUM!
May	$127,168.83	$433,682.00		$433,682.00	#NUM!
June	$89,589.80	$523,271.80	$121,707.44	$401,564.36	329.94%
July	$92,047.25	$615,319.05	$261,088.44	$354,230.61	135.67%
August	$141,320.07	$756,639.12	$348,472.74	$408,166.38	117.13%
September	$112,052.72	$868,691.84	$503,748.68	$364,943.15	72.45%
October	$59,059.30	$927,751.13	$603,621.33	$324,129.80	53.70%
November	$108,290.76	$1,036,041.89	$726,144.19	$309,897.71	42.68%
December	$86,493.16	$1,122,535.05	$885,358.63	$237,176.42	26.79%
CY 2009	**$804,885.87**	**$804,885.87**	**$1,122,535.05**	**($317,649.18)**	**-28.30%**
CY 2010			**$804,885.87**	**($804,885.87)**	**-100.00%**
Grand Total	**$2,812,779.56**				

5. There are a few issues with this formula:
   - In 2007, the previous year sales is zero, thus the result is #NUM.
   - Starting from September 2009, there are no sales, so you will probably want to hide the percentages of growth, because they would lead to wrong insights.

Can you fix those issues by yourself? No solution this time! Let us try to see what happens when you are on your own!

However, if you really want the solution, it is in the final workbook of the course. The result you want to obtain is the following:



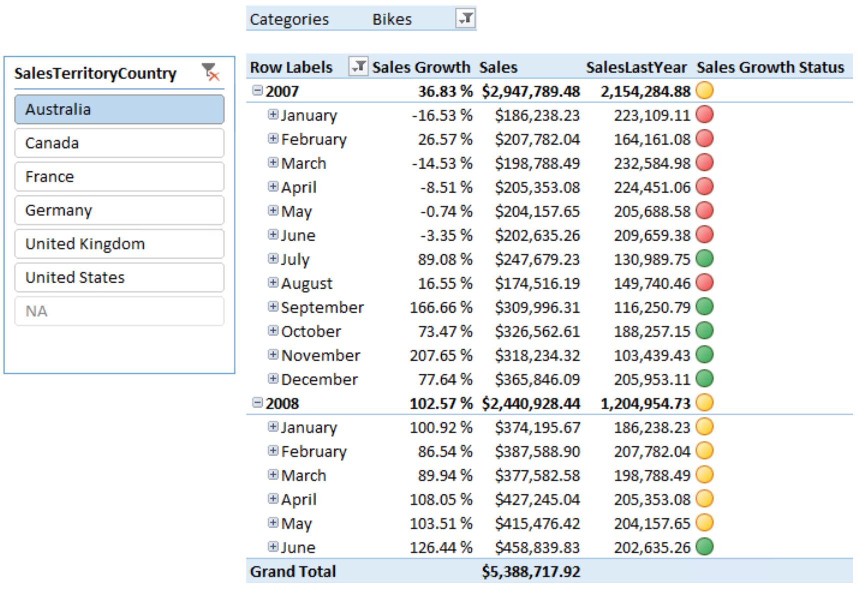
Row Labels	TotalSales	SalesYTD	PYSalesYTD	DeltaSales	DeltaSalesPct
CY 2007	**$885,358.63**	**$885,358.63**		**$885,358.63**	
CY 2008	**$1,122,535.05**	**$1,122,535.05**	**$885,358.63**	**$237,176.42**	**26.79%**
January	$64,601.67	$64,601.67		$64,601.67	
February	$61,157.39	$125,759.06		$125,759.06	
March	$64,749.27	$190,508.33		$190,508.33	
April	$116,004.84	$306,513.17		$306,513.17	
May	$127,168.83	$433,682.00		$433,682.00	
June	$89,589.80	$523,271.80	$121,707.44	$401,564.36	329.94%
July	$92,047.25	$615,319.05	$261,088.44	$354,230.61	135.67%
August	$141,320.07	$756,639.12	$348,472.74	$408,166.38	117.13%
September	$112,052.72	$868,691.84	$503,748.68	$364,943.15	72.45%
October	$59,059.30	$927,751.13	$603,621.33	$324,129.80	53.70%
November	$108,290.76	$1,036,041.89	$726,144.19	$309,897.71	42.68%
December	$86,493.16	$1,122,535.05	$885,358.63	$237,176.42	26.79%
CY 2009	**$804,885.87**	**$804,885.87**	**$1,122,535.05**	**($317,649.18)**	**-28.30%**
CY 2010			**$804,885.87**	**($804,885.87)**	
Grand Total	**$2,812,779.56**				

Build a KPI using ALLSELECTED

KPI

This exercise guides you through the creation of a simple KPI, showing how to define and use them. The DAX code is simple, the goal is to be acquainted with the concept of KPI and learn how to use them.

The starting point is the data model you used so far: dates, products, sales territories. The goal is to build a report that shows in a graphical way which periods show an increase in sales for specific countries, by comparing their growth to the average growth in the same country (or whatever selection you want to perform). The result is this report:



SalesTerritoryCountry
- Australia
- Canada
- France
- Germany
- United Kingdom
- United States
- NA

Categories: Bikes

Row Labels	Sales Growth	Sales	SalesLastYear	Sales Growth Status
2007	**36.83 %**	**$2,947,789.48**	**2,154,284.88**	🟡
January	-16.53 %	$186,238.23	223,109.11	🔴
February	26.57 %	$207,782.04	164,161.08	🔴
March	-14.53 %	$198,788.49	232,584.98	🔴
April	-8.51 %	$205,353.08	224,451.06	🔴
May	-0.74 %	$204,157.65	205,688.58	🔴
June	-3.35 %	$202,635.26	209,659.38	🔴
July	89.08 %	$247,679.23	130,989.75	🟢
August	16.55 %	$174,516.19	149,740.46	🔴
September	166.66 %	$309,996.31	116,250.79	🟢
October	73.47 %	$326,562.61	188,257.15	🟢
November	207.65 %	$318,234.32	103,439.43	🟢
December	77.64 %	$365,846.09	205,953.11	🟢
2008	**102.57 %**	**$2,440,928.44**	**1,204,954.73**	🟡
January	100.92 %	$374,195.67	186,238.23	🟡
February	86.54 %	$387,588.90	207,782.04	🟡
March	89.94 %	$377,582.58	198,788.49	🟡
April	108.05 %	$427,245.04	205,353.08	🟡
May	103.51 %	$415,476.42	204,157.65	🟡
June	126.44 %	$458,839.83	202,635.26	🟢
Grand Total		**$5,388,717.92**		

1. First, you need to create the measures needed to compute the KPI. You need Sales, SalesLastYear and Sales Growth in percentage. The formulas are easy, you can try them by yourself. If you feel confident you can write them, and then check with the solution. This is the basic result:

```
SalesLastYear := CALCULATE (
    [TotalSales],
    SAMEPERIODLASTYEAR ( 'Date'[Date] )
)

SalesGrowth := ( [TotalSales] - [SalesLastYear] ) / [SalesLastYear]
```

2. However, if you followed all the recommendation learned so far, you might have computed them in the correct way, because SalesLastYear is not computing correct values for the second year, because the first year is incomplete. The correct formulation for SalesLastYear is the following:

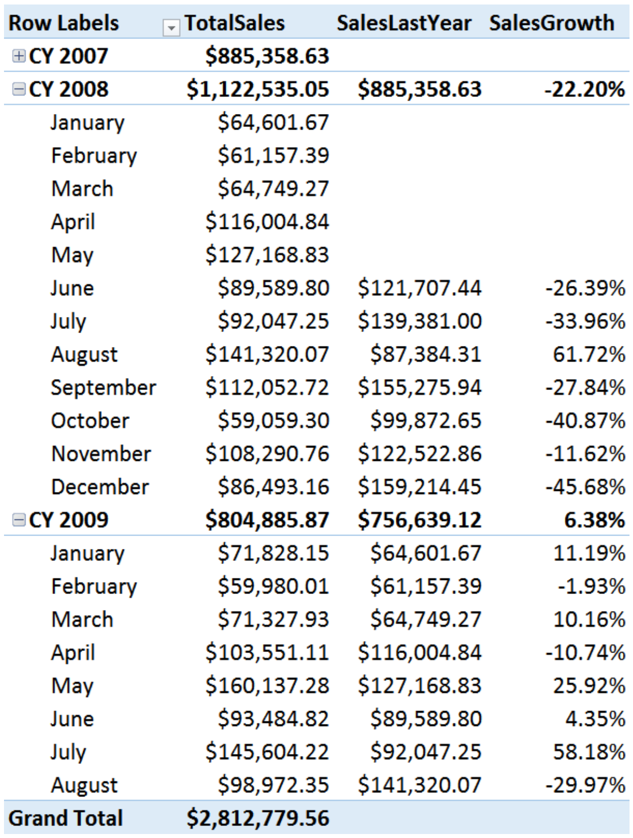
```
SalesLastYear:=
IF (
    HASONEVALUE ( 'Date'[Calendar Year] ),
    SUMX (
        VALUES ( 'Date'[Month] ),
        IF (
             [TotalSales] > 0,
             CALCULATE ( [TotalSales], SAMEPERIODLASTYEAR( 'Date'[Date] ) )
        )
    )
)
```

   It is useful to spend some minutes looking at the formula, because it shows many patterns we have used so far. It also makes the difference between writing the first formula that returns a number and knowing exactly when and how the formula might fail, anticipating possible errors.

3. SalesGrowth, too, needs a small adjustment, because it needs to account for partial years in the same way:

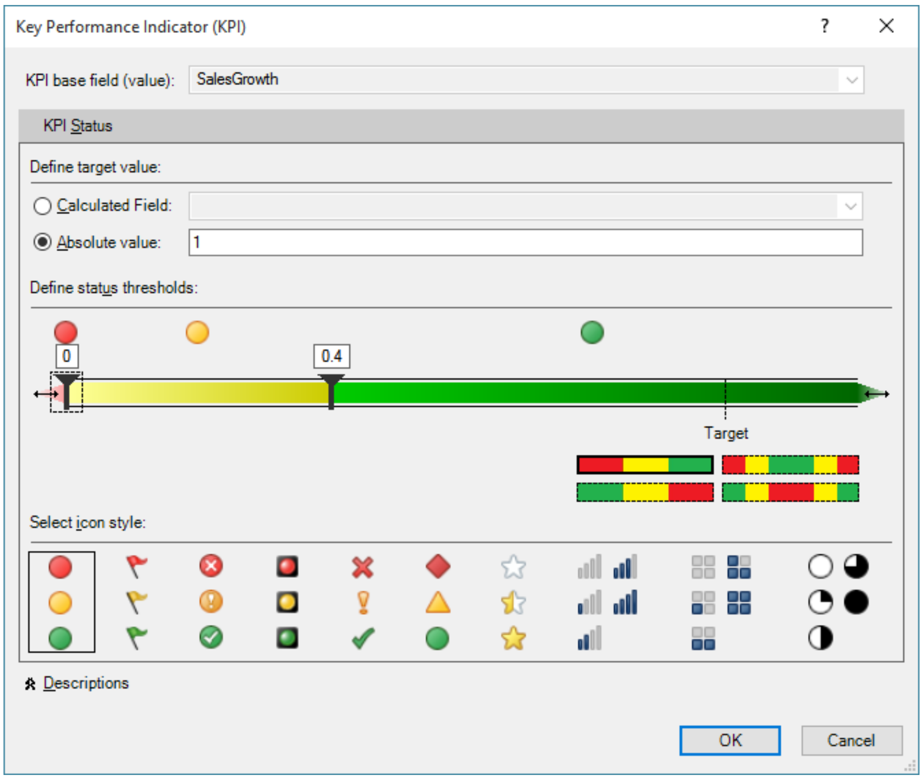
```
SalesGrowth:=
DIVIDE (
    SUMX (
        VALUES ( 'Date'[Month] ),
        IF ( [SalesLastYear] > 0, [TotalSales] )
    ) - [SalesLastYear],
    [SalesLastYear]
)
```

4. With these three measures, build this simple PivotTable, checking that the values are correct:



Row Labels	TotalSales	SalesLastYear	SalesGrowth
⊞CY 2007	$885,358.63		
⊟CY 2008	$1,122,535.05	$885,358.63	-22.20%
January	$64,601.67		
February	$61,157.39		
March	$64,749.27		
April	$116,004.84		
May	$127,168.83		
June	$89,589.80	$121,707.44	-26.39%
July	$92,047.25	$139,381.00	-33.96%
August	$141,320.07	$87,384.31	61.72%
September	$112,052.72	$155,275.94	-27.84%
October	$59,059.30	$99,872.65	-40.87%
November	$108,290.76	$122,522.86	-11.62%
December	$86,493.16	$159,214.45	-45.68%
⊟CY 2009	$804,885.87	$756,639.12	6.38%
January	$71,828.15	$64,601.67	11.19%
February	$59,980.01	$61,157.39	-1.93%
March	$71,327.93	$64,749.27	10.16%
April	$103,551.11	$116,004.84	-10.74%
May	$160,137.28	$127,168.83	25.92%
June	$93,484.82	$89,589.80	4.35%
July	$145,604.22	$92,047.25	58.18%
August	$98,972.35	$141,320.07	-29.97%
Grand Total	$2,812,779.56		

5. Now it is time to create the KPI based on the Sales Growth measure. First, we make a KPI that will show red if growth is negative, yellow if the growth is less than 40% and green if it is more than 40%. In order to do this, use the new KPI button and fill the invoked dialog box in this way:



Key Performance Indicator (KPI)

KPI base field (value): SalesGrowth

KPI Status

Define target value:

Calculated Field:

[✓ Absolute value:] 1

Define status thresholds:

0

0.4

Target

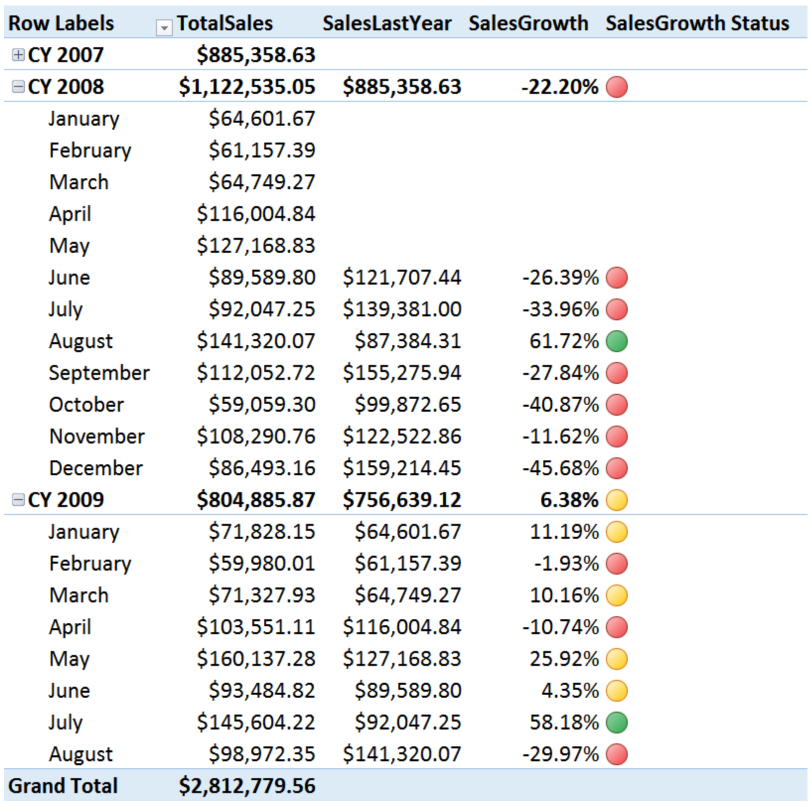
Select icon style:

Descriptions

OK

Cancel

6. You can then add the KPI status to the PivotTable, which looks like this:



Row Labels	TotalSales	SalesLastYear	SalesGrowth	SalesGrowth Status
⊞CY 2007	$885,358.63			
⊟CY 2008	$1,122,535.05	$885,358.63	-22.20%	🔴
January	$64,601.67			
February	$61,157.39			
March	$64,749.27			
April	$116,004.84			
May	$127,168.83			
June	$89,589.80	$121,707.44	-26.39%	🔴
July	$92,047.25	$139,381.00	-33.96%	🔴
August	$141,320.07	$87,384.31	61.72%	🟢
September	$112,052.72	$155,275.94	-27.84%	🔴
October	$59,059.30	$99,872.65	-40.87%	🔴
November	$108,290.76	$122,522.86	-11.62%	🔴
December	$86,493.16	$159,214.45	-45.68%	🔴
⊟CY 2009	$804,885.87	$756,639.12	6.38%	🟡
January	$71,828.15	$64,601.67	11.19%	🟡
February	$59,980.01	$61,157.39	-1.93%	🔴
March	$71,327.93	$64,749.27	10.16%	🟡
April	$103,551.11	$116,004.84	-10.74%	🔴
May	$160,137.28	$127,168.83	25.92%	🟡
June	$93,484.82	$89,589.80	4.35%	🟡
July	$145,604.22	$92,047.25	58.18%	🟢
August	$98,972.35	$141,320.07	-29.97%	🔴
Grand Total	$2,812,779.56			

7. Now, for the more complex part of the KPI: what we want to highlight is not if the growth in a specific month was more than 40%, but if it was more than the average. Thus, you need a measure that computes the average growth or, in other words, the average of the year.

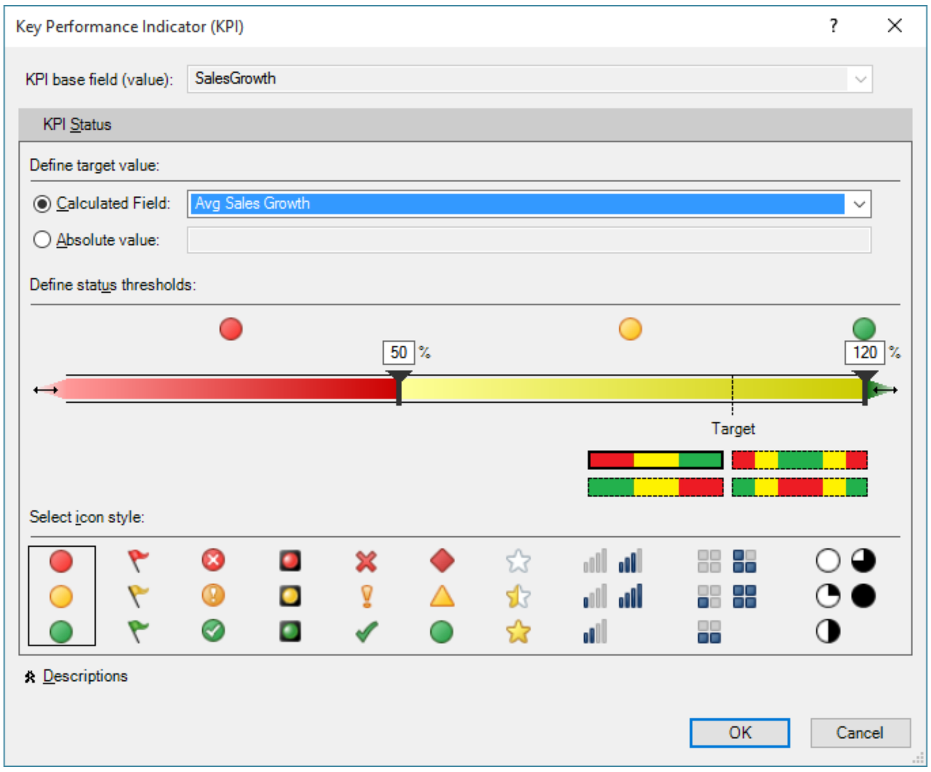
   Can you figure out the formula by yourself? It is not very easy, yet not very hard, at this point. You need to compute the Sales Growth for the current year, i.e. remove all filters from the months and leave only the ones for the year.

   Again, there are many ways to express the formula. Our proposal is the following:

```
[Avg Sales Growth] :=
CALCULATE (
    [SalesGrowth],
    ALLSELECTED (),
    VALUES ( 'Date'[Calendar Year] )
)
```

   Note the usage of ALLSELECTED without parameters to grab the query context and the usage of VALUES to push the filter of the year inside CALCULATE. Remember that the two filters are put in an AND condition, before being applied to the data model.

8. Now that you have the average growth, you can compare the growth of the selected period with the average growth, configuring the KPI window in this way:


Key Performance Indicator (KPI)

KPI base field (value): SalesGrowth

KPI Status

Define target value:

(●) Calculated Field: Avg Sales Growth

( ) Absolute value:

Define status thresholds:

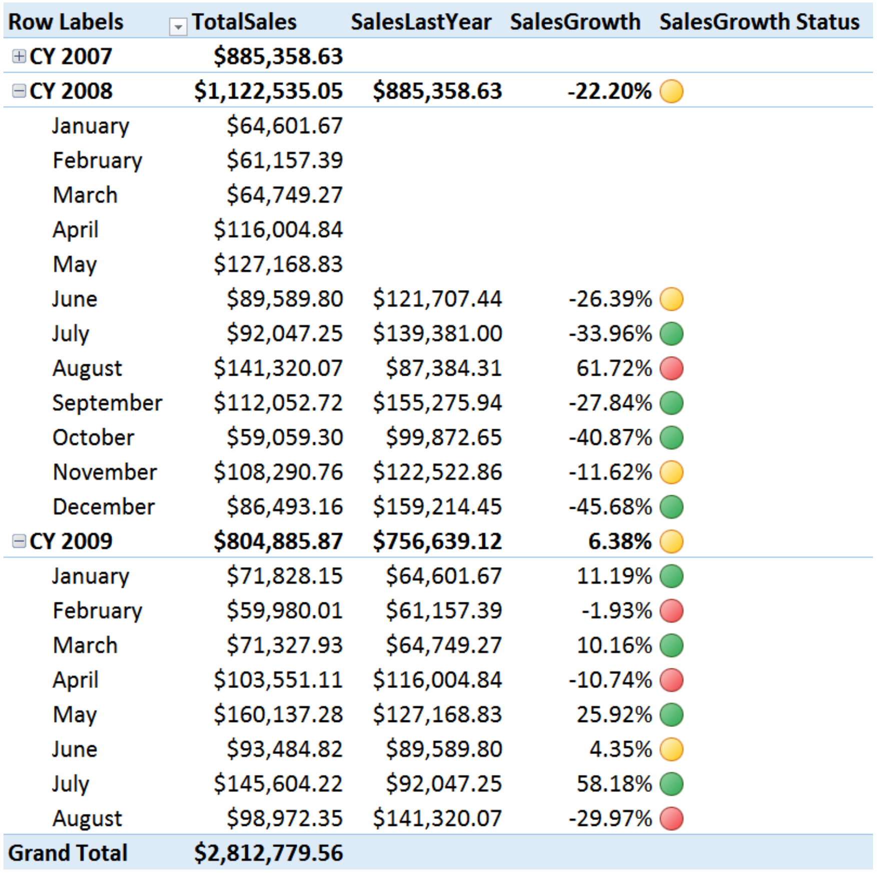
50 %

120 %

Select icon style:

This KPI says: if the growth is less than 50% of the average, then show red. If it is between 50% and 120%, then it is average (yellow). If it is more than 120%, it is outstanding (green). Unfortunately, this calculation yields to wrong numbers, because, when both the growth and the average growth are negative, their ratio will be positive.

9. In fact, the PivotTable now shows these values:



Row Labels	TotalSales	SalesLastYear	SalesGrowth	SalesGrowth Status
CY 2007	**$885,358.63**			
CY 2008	**$1,122,535.05**	**$885,358.63**	**-22.20%**	🟡
January	$64,601.67			
February	$61,157.39			
March	$64,749.27			
April	$116,004.84			
May	$127,168.83			
June	$89,589.80	$121,707.44	-26.39%	🟡
July	$92,047.25	$139,381.00	-33.96%	🟢
August	$141,320.07	$87,384.31	61.72%	🔴
September	$112,052.72	$155,275.94	-27.84%	🟢
October	$59,059.30	$99,872.65	-40.87%	🟢
November	$108,290.76	$122,522.86	-11.62%	🟡
December	$86,493.16	$159,214.45	-45.68%	🟢
CY 2009	**$804,885.87**	**$756,639.12**	**6.38%**	🟡
January	$71,828.15	$64,601.67	11.19%	🟢
February	$59,980.01	$61,157.39	-1.93%	🔴
March	$71,327.93	$64,749.27	10.16%	🟢
April	$103,551.11	$116,004.84	-10.74%	🔴
May	$160,137.28	$127,168.83	25.92%	🟢
June	$93,484.82	$89,589.80	4.35%	🟡
July	$145,604.22	$92,047.25	58.18%	🟢
August	$98,972.35	$141,320.07	-29.97%	🔴
Grand Total	**$2,812,779.56**			

You can see that September 2008, which shows -27.84% of growth is green, even if its growth is less than the average growth of the year, which is -22.20%. Using KPI with percentage is always challenging, in this case you can solve it by creating a KPI Sales Growth measure with this code:

```
KPISalesGrowth := IF ( [SalesGrowth] <> 0, [SalesGrowth] - [Avg Sales Growth] )
```

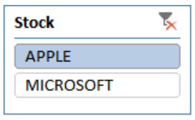
Then, you base your KPI on this measure, which computes the difference between the growth and the average growth, leading to a better representation.

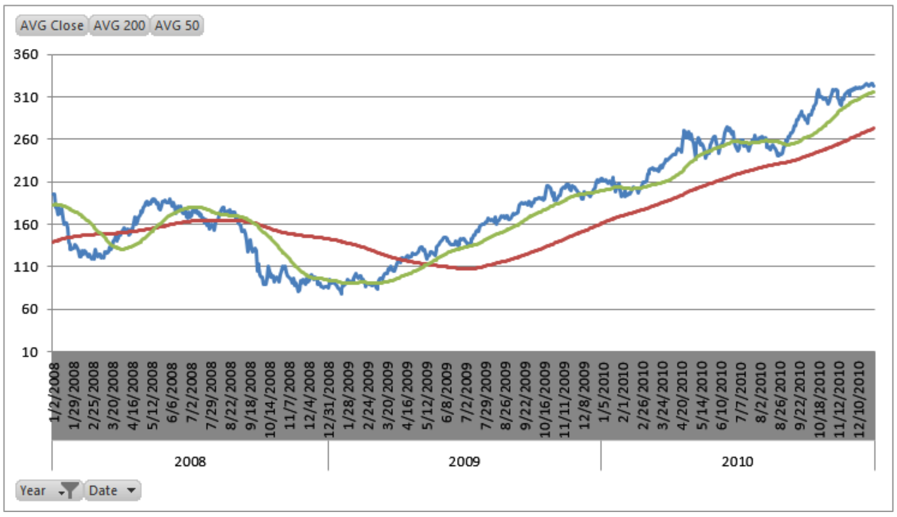
Stock Exchange

Stocks

This exercise is based on stock information, thus it does not use the Contoso database. We start from an Excel table containing stock prices and we want to compute a moving average of the price of a stock. Moreover, after having done that, we will create an interactive PivotChart to compare the performances of two different stocks.

The result should be something like this:

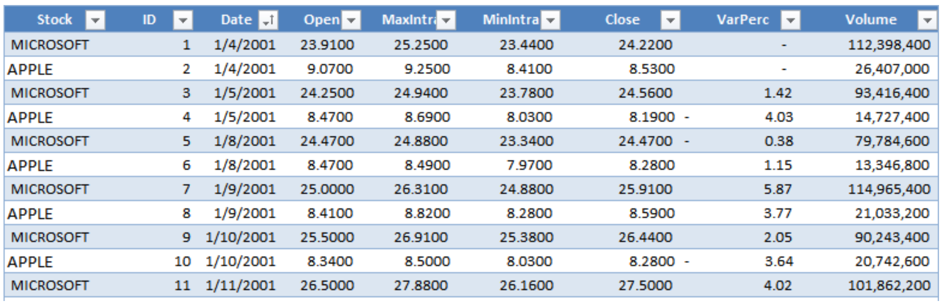




In the report we provide the analysis of last three years of the Apple stock with two moving averages (one on 50 periods, the other on 200 periods).

The interesting part of the exercise is that, this time, we will not leverage the time intelligence functions to provide the moving average. We will need to use a different technique to solve the scenario.

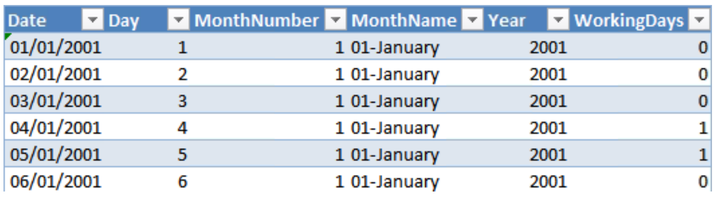
1. Take a look at the data in the Fixings sheet: the source Excel table is the following:



Stock	ID	Date	Open	MaxIntra	MinIntra	Close	VarPerc	Volume
MICROSOFT	1	1/4/2001	23.9100	25.2500	23.4400	24.2200	-	112,398,400
APPLE	2	1/4/2001	9.0700	9.2500	8.4100	8.5300	-	26,407,000
MICROSOFT	3	1/5/2001	24.2500	24.9400	23.7800	24.5600	1.42	93,416,400
APPLE	4	1/5/2001	8.4700	8.6900	8.0300	8.1900	- 4.03	14,727,400
MICROSOFT	5	1/8/2001	24.4700	24.8800	23.3400	24.4700	- 0.38	79,784,600
APPLE	6	1/8/2001	8.4700	8.4900	7.9700	8.2800	1.15	13,346,800
MICROSOFT	7	1/9/2001	25.0000	26.3100	24.8800	25.9100	5.87	114,965,400
APPLE	8	1/9/2001	8.4100	8.8200	8.2800	8.5900	3.77	21,033,200
MICROSOFT	9	1/10/2001	25.5000	26.9100	25.3800	26.4400	2.05	90,243,400
APPLE	10	1/10/2001	8.3400	8.5000	8.0300	8.2800	- 3.64	20,742,600
MICROSOFT	11	1/11/2001	26.5000	27.8800	26.1600	27.5000	4.02	101,862,200

The interesting columns, for this exercise, are only the stock name, the date and the value of Close. Other information might be useful for other purposes; we left them there because the original data source contained them. This table is called Fixing.

2. The second worksheet contains a standard calendar table with just a few columns:



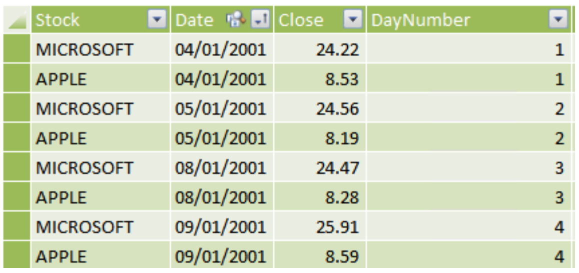
Date	Day	MonthNumber	MonthName	Year	WorkingDays
01/01/2001	1	1	01-January	2001	0
02/01/2001	2	1	01-January	2001	0
03/01/2001	3	1	01-January	2001	0
04/01/2001	4	1	01-January	2001	1
05/01/2001	5	1	01-January	2001	1
06/01/2001	6	1	01-January	2001	0

3. Link these tables into PowerPivot. For the sake of simplicity, you can delete from the Fixing table the useless columns, leaving only Stock, Date and Close.
4. It is now time to compute the moving average over 50 days. We do not want to compute the average over 50 days but over 50 "working" days, where "working" means that there is data available for that stock.

   Different stocks might have different dates where they are recorded. If, for whatever reason, the data about Microsoft contained holes, then the values for the date range of Microsoft would be different from the date range of Apple. Each stock needs its date ranges correctly computed.
5. Define a new calculated column named DayNumber, in the Fixing table, which assigns an incremental number to each date, taking care of the fact that different stocks need different counters. The formula is the following:

```
=COUNTROWS (
    FILTER (
        Fixing,
        Fixing[Date] <= EARLIER ( Fixing[Date] )
        && Fixing[Stock] = EARLIER ( Fixing[Stock] )
  )
)
```

   We use a technique similar to the one used to calculate the working days but, this time, we added a condition on the stock column. In this way, the number is assigned with different values for different stocks. Your table should now look like this:



Stock	Date	Close	DayNumber
MICROSOFT	04/01/2001	24.22	1
APPLE	04/01/2001	8.53	1
MICROSOFT	05/01/2001	24.56	2
APPLE	05/01/2001	8.19	2
MICROSOFT	08/01/2001	24.47	3
APPLE	08/01/2001	8.28	3
MICROSOFT	09/01/2001	25.91	4
APPLE	09/01/2001	8.59	4

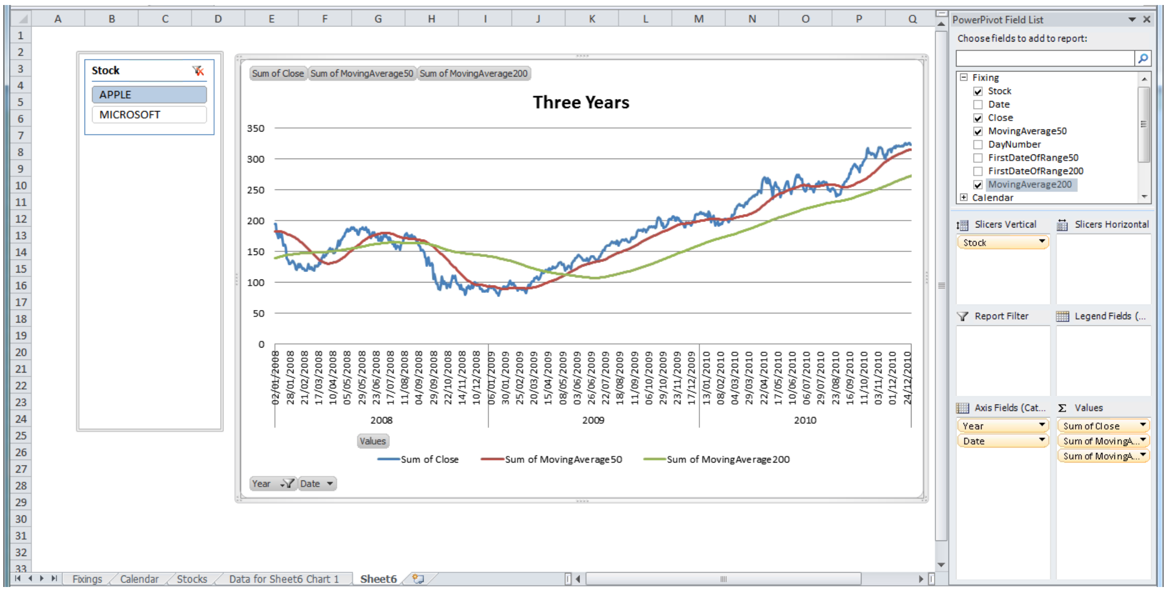
6. We can easily compute, for each date, the first date of the range of 50 days. It will be the only row that has a DayNumber equals to the current one less 50. Using CALCULATE, it is easy to compute this date:

```
FirstDateOfRange50=CALCULATE (
    VALUES ( Fixing[Date] ),
    FILTER (
        Fixing,
        Fixing[DayNumber] = EARLIER ( Fixing[DayNumber] ) - 50
        && Fixing[Stock] = EARLIER ( Fixing[Stock] )
     )
)
```

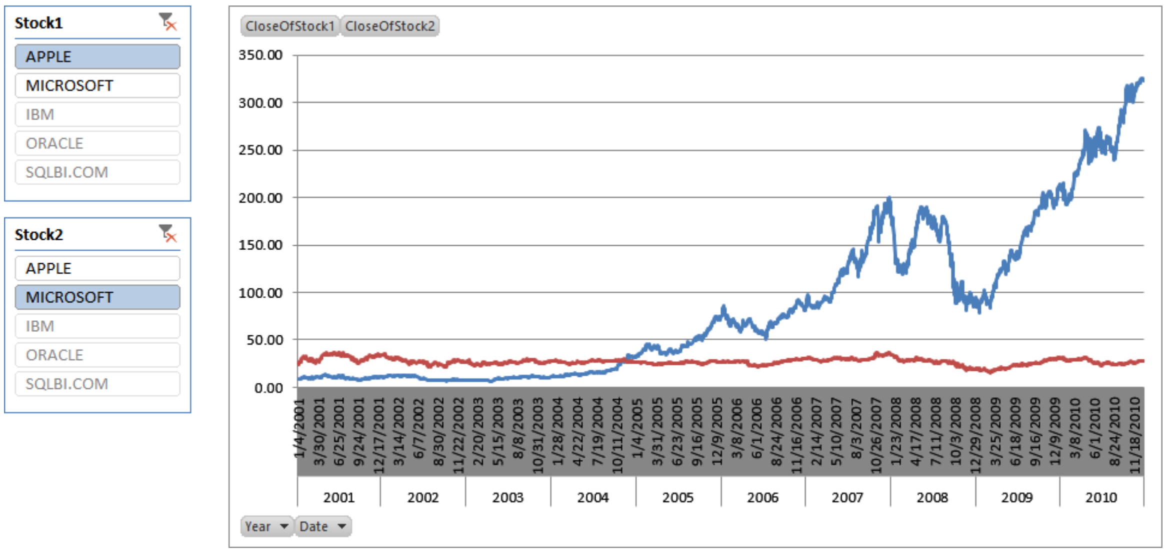
7. The following step is to compute the moving average over 50 days which, again, can be easily computed leveraging CALCULATE since now we know exactly where to start to compute the average:

```
MovingAverage50=CALCULATE (
    AVERAGE ( Fixing[Close] ),
    FILTER(
        Fixing,
        Fixing[Date] >= EARLIER ( Fixing[FirstDateOfRange50] )
        && Fixing[Date] <= EARLIER ( Fixing[Date] )
        && Fixing[Stock] = EARLIER ( Fixing[Stock] )
    )
)
```

8. Repeat the previous steps to create a moving average over 200 dates; just changing 50 into 200 will solve the problem.
9. The final step to complete the first part of the exercise is to create a PivotChart like this:



10. You should note that we use the Calendar table to select years and dates. This will raise the need to set proper relationships between the Fixing and the Calendar table.
11. At this point, it is interesting to create a new chart that let you compare two stocks on the same graph, like this:

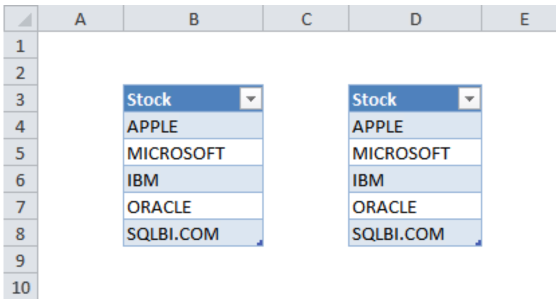


The second graph is a bit more challenging. We want to use two slicers to select two different stocks and then showing their graph into a single PivotChart to make some sort of comparison.

It is clear that the slicers used to select the stocks to show cannot be based on the Stock column of the Fixing table because, doing so, we would filter the Fixing table. On the other hand, we want to use those slicers to collect the user selection and then, inside the DAX formulas, we will decide what kind of filtering to define.

Thus, we need two tables to hold the names of the stocks that will serve as the sources for the slicers.

12. Create two Excel tables and link them in PowerPivot, calling them Stock1 and Stock2:



	A	B	C	D	E
1					
2					
3		Stock		Stock	
4		APPLE		APPLE	
5		MICROSOFT		MICROSOFT	
6		IBM		IBM	
7		ORACLE		ORACLE	
8		SQLBI.COM		SQLBI.COM	
9					
10					

13. Create a PivotChart, add the two tables as slicers, add year and date to the axis and get prepared to study the final formula.

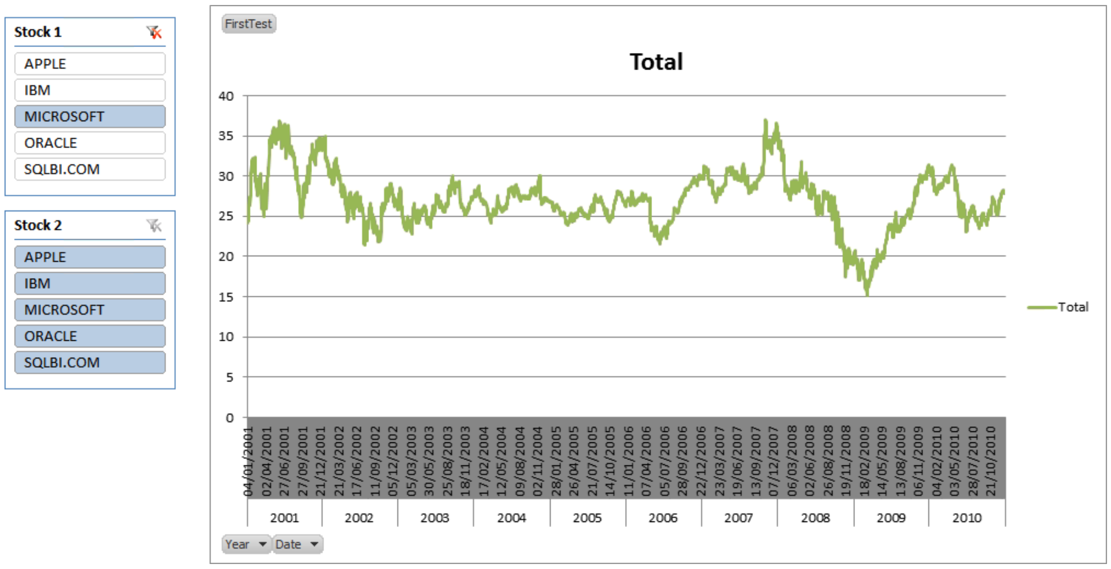
14. We already know that, using VALUES(), we can query the values available in the current filter context for a column. Thus, querying VALUES (Stock1[Stock]), we will be able to get the list of values selected in the slicer for Stock1. This simple observation will make the formula much clearer:

```
IF (
    COUNTROWS ( VALUES ( Stock1[Stock] ) ) > 1,
    BLANK (),
    CALCULATE (
        AVERAGE ( Fixing[Close] ),
        Fixing[Stock] = VALUES ( Stock1[Stock] )
    )
)
```

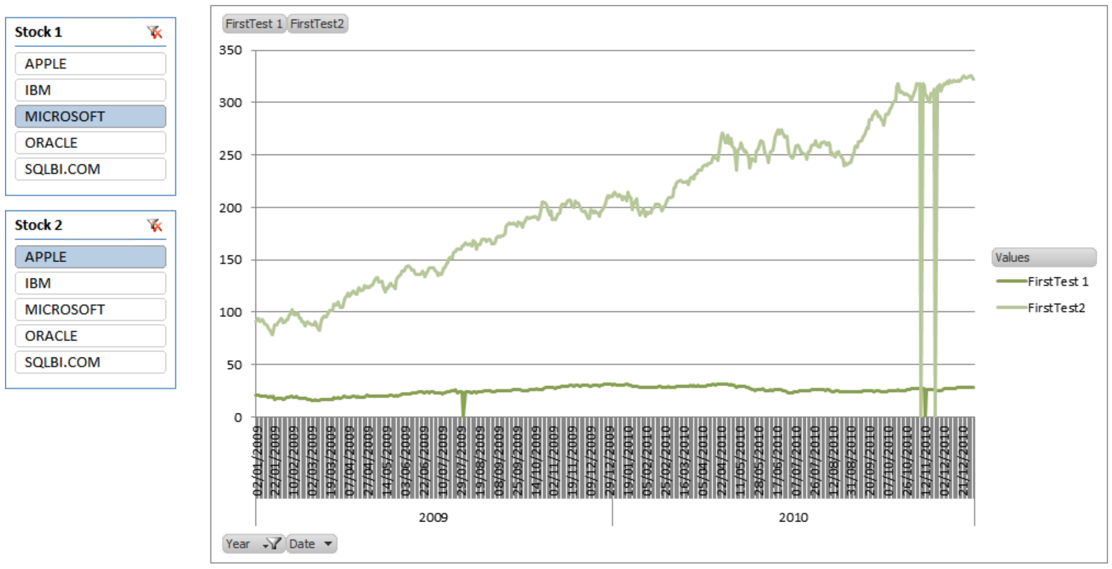
First, we check that the slicer selects exactly one stock. If this is the case, we create a new filter context where the stock in the Fixing table equals the stock selected in the slicer and, in this context, we evaluate the AVERAGE of Fixing[Close].

Depending on the value selected in the slicer, this formula will compute the value of any stock. Use it to define a new measure and, changing the name of the table, create a similar measure for Stock2.

15. If you now put one of these measures in the PivotChart, you will get this:



16. Unfortunately, when you put the second measure and select another stock with the slicer, the chart will be much less appealing. In the following picture we selected just a couple of years, to make the issue more evident.



17. There are some spikes down to zero for both stocks. Note that spikes appear if and only if two stocks are selected, otherwise they do not appear inside the graph. The reason is that, in the data sources, the value of Microsoft and Apple is not recorded for every day. There are some holes, for whatever reason, and they do not appear in the same date for the two stocks.

    Thus, when you select only one stock, these dates disappear from the chart, because the value of the measure is BLANK. However, when you insert both stocks in the chart, if one stock has a value in a specific date and the other contains BLANK, the value of the latter is considered zero and the date still appears in the chart, to show the value of the existing stock.

    Even if it can be explained, this behavior is still not very nice. We definitely need to find a way to avoid it.

18. The solution is to hide the value of Stock1 if Stock2 is not present and, of course, do the contrary, too. Hide the value of Stock2 if Stock1 is not present.

19. This consideration leads us to the following formula:

```
=IF (
    COUNTROWS ( VALUES ( Stock1[Stock] ) ) > 1 || COUNTROWS ( VALUES ( Stock2[Stock] ) ) > 1,
    BLANK (),
    IF (
        ISBLANK (
            CALCULATE (
                AVERAGE ( Fixing[Close] ),
                Fixing[Stock] = VALUES ( Stock2[Stock] )
            )
        ),
        BLANK (),
        CALCULATE (
            AVERAGE ( Fixing[Close] ),
            Fixing[Stock] = VALUES ( Stock1[Stock] )
        )
    )
)
```

20. Using this formula, and a corresponding one for Stock2, we get the result, in which the dates that do not contain both stock values are removed from the chart, leading to a much cleaner representation of the stock price.

LAB 6
DAX as a Query Language

Simple DAX queries

Querying

This set of queries requires Power BI Designer, you will use DAX Studio to connect to the Contoso database in Power BI Designer and then write some queries.

1. Use a simple EVALUATE statement to retrieve the products

```
EVALUATE
    Product
```

2. Use CALCULATETABLE to show only products of one category (Audio).

```
EVALUATE
CALCULATETABLE (
    Product,
    'Product Category'[Category] = "Audio"
)
```

3. Use FILTER and ADDCOLUMNS to show only product categories with at least 100 products.

```
EVALUATE
FILTER (
    ADDCOLUMNS (
        'Product Category',
        "NumOfProducts", CALCULATE ( COUNTROWS ( Product ) )
    ),
    [NumOfProducts] > 100
)
```

4. Use FILTER, ADDCOLUMNS and context transition to compute the number of categories for which there are more than 10 products that sold more than 2000 USD. Each product need to have more than 2000 USD of sales.

```
EVALUATE
FILTER (
    ADDCOLUMNS (
        'Product Category',
        "NumOfHighSalesProducts", CALCULATE (
            COUNTROWS (
                FILTER (
                    Product,
                    CALCULATE (
                        SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
                    )
                        > 2000
                )
            )
        )
    ),
    [NumOfHighSalesProducts] > 10
)
```

5. Try SUMMARIZE to show the total sales amount of products grouped by color and brand.

```
EVALUATE
SUMMARIZE (
    Product,
    Product[Color],
    Product[Brand],
    "Total Sales", SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
)
```

6. Find all the products that did not sell anything in 2007.

```
EVALUATE
FILTER (
    Product,
    ISEMPTY (
        CALCULATETABLE ( Sales, 'Date'[Calendar Year] = "CY 2007" )
    )
)
```

7. Find all the products that sold more than the average in 2007

```
DEFINE
    MEASURE Sales[Amount] =
        SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
EVALUATE
CALCULATETABLE (
    VAR
        AverageSales = AVERAGEX ( Product, [Amount] )
    RETURN
        FILTER ( Product, [Amount] > AverageSales ),
    'Date'[Calendar Year] = "CY 2007"
)
```

8. Find the top ten customers over all time (order by total sold over time)

```
DEFINE
    MEASURE Sales[Amount] =
        SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
EVALUATE
TOPN (
    10,
    ADDCOLUMNS ( Customer, "Sales", [Amount] ),
    [Amount]
)
```

9. Find, for each year, the top three customers (order by total sold over the year) returning, for each customer, the calendar year, the customer code, the customer name and the sales in that year.

```
EVALUATE
GENERATE (
    VALUES ( 'Date'[Calendar Year] ),
    CALCULATETABLE (
        TOPN (
            3,
            ADDCOLUMNS (
                SELECTCOLUMNS (
                    Customer,
                    "Code", Customer[Customer Code],
                    "Name", Customer[Company Name]
                ),
                "Sales", CALCULATE (
                    SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
                )
            ),
            [Sales]
        )
    )
)
```

LAB 7
Advanced Filter Context

Distinct Count of Product Attributes

Distinct Count

This exercise aims to test your ability to use and understand cross table filtering. The formulas are neither long nor too complex, once you look at the solution but understanding them correctly and – most important – learning how to write them, is not easy. For this reason, we suggest you not to look at the solution immediately, but to try to write them by yourself and then, only after a few minutes of trials, look at the solution if you are facing some difficulties.

1. If you need to create a distinct count of customers, this is very easy indeed, thanks to the DISTINCTCOUNT function. You can easily create a NumOfCustomers measure, as in:

   ```
   NumOfCustomers := DISTINCTCOUNT ( OnlineSales[CustomerKey] )
   ```

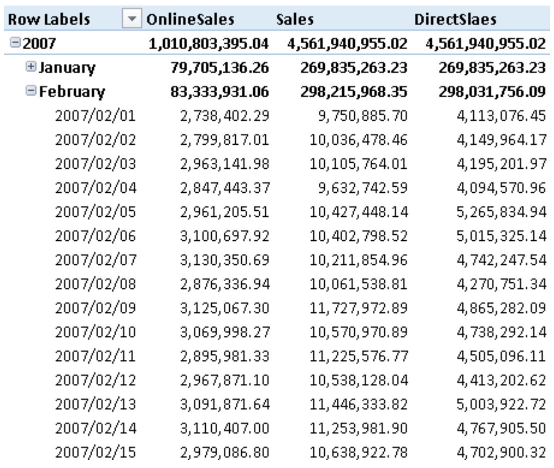
2. Unfortunately, as easy as this formula is, it is wrong. If Customer is a slowly changing dimension, then you should not compute the distinct count of the surrogate key, but compute the distinct count of the customer code (or name), which is stored in the Customer table. For this example, we will use the Customer[FullName] to count the distinct customers.
3. In the next lines we will show the formula. The key to solve is to understand that you need to count the number of customer names for only the customers that are in the OnLineSales table. Do you know of a method to propagate the filter from the fact table (OnlineSales) to the Customer table? Cross-table filtering, is the key… Stop reading and think at the solution.
4. The formula, as we said in advance, is very easy:

   ```
   NumOfScdCustomers := CALCULATE ( DISTINCTCOUNT ( Customer[FullName] ), OnlineSales )
   ```

   Understanding it takes some time, to grab its meaning think at the expanded OnlineSales table and check whether it will filter Customer or not and, if it filters them, which customers it is filtering.
5. The interesting part of this small formula is that you can use it to compute the distinct count of any attribute of the customer (or product, or any other) table or – more generally speaking – of any dimension tied to a fact table, without having to work on the data model.
6. Try, for example, to compute the distinct count of ColorName or of BrandName of the Product table and put them in a PivotTable.

The same technique can be used to perform much more complex calculations. These kind of calculations will become clearer when we will cover many-to-many relationships but, for now, we can already start playing with them.

1. You want to compare sales performed online versus direct sales. The information is stored in two different tables. You already know that, by filtering the product category, for example, you can see how that category sold in the different tables. But what if you want to compute a much more challenging value, i.e. check the direct sales of the products that have been sold online. This time, the filter cannot be created on the Product table. What you need to compute is the direct sales of the products that have been sold online.
2. Imagine, for example, that you filter one month. You need to compute the list of products that are sold online in that month and, using this list of products, filter the Sales table. Again, the solution is very compact and elegant but, understanding how it works, requires some time.
3. In the next figure you can see the final result you need to achieve, where:
    a. OnlineSales is the sales of products online
    b. Sales is the direct sales of products
    c. DirectSales is the amount of direct sales of only the products which have been sold online



Row Labels	OnlineSales	Sales	DirectSlaes
⊟2007	**1,010,803,395.04**	**4,561,940,955.02**	**4,561,940,955.02**
⊞January	**79,705,136.26**	**269,835,263.23**	**269,835,263.23**
⊟February	**83,333,931.06**	**298,215,968.35**	**298,031,756.09**
2007/02/01	2,738,402.29	9,750,885.70	4,113,076.45
2007/02/02	2,799,817.01	10,036,478.46	4,149,964.17
2007/02/03	2,963,141.98	10,105,764.01	4,195,201.97
2007/02/04	2,847,443.37	9,632,742.59	4,094,570.96
2007/02/05	2,961,205.51	10,427,448.14	5,265,834.94
2007/02/06	3,100,697.92	10,402,798.52	5,015,325.14
2007/02/07	3,130,350.69	10,211,854.96	4,742,247.54
2007/02/08	2,876,336.94	10,061,538.81	4,270,751.34
2007/02/09	3,125,067.30	11,727,972.89	4,865,282.09
2007/02/10	3,069,998.27	10,570,970.89	4,738,292.14
2007/02/11	2,895,981.33	11,225,576.77	4,505,096.11
2007/02/12	2,967,871.10	10,538,128.04	4,413,202.62
2007/02/13	3,091,871.64	11,446,333.82	5,003,922.72
2007/02/14	3,110,407.00	11,253,981.90	4,767,905.50
2007/02/15	2,979,086.80	10,638,922.78	4,702,900.32

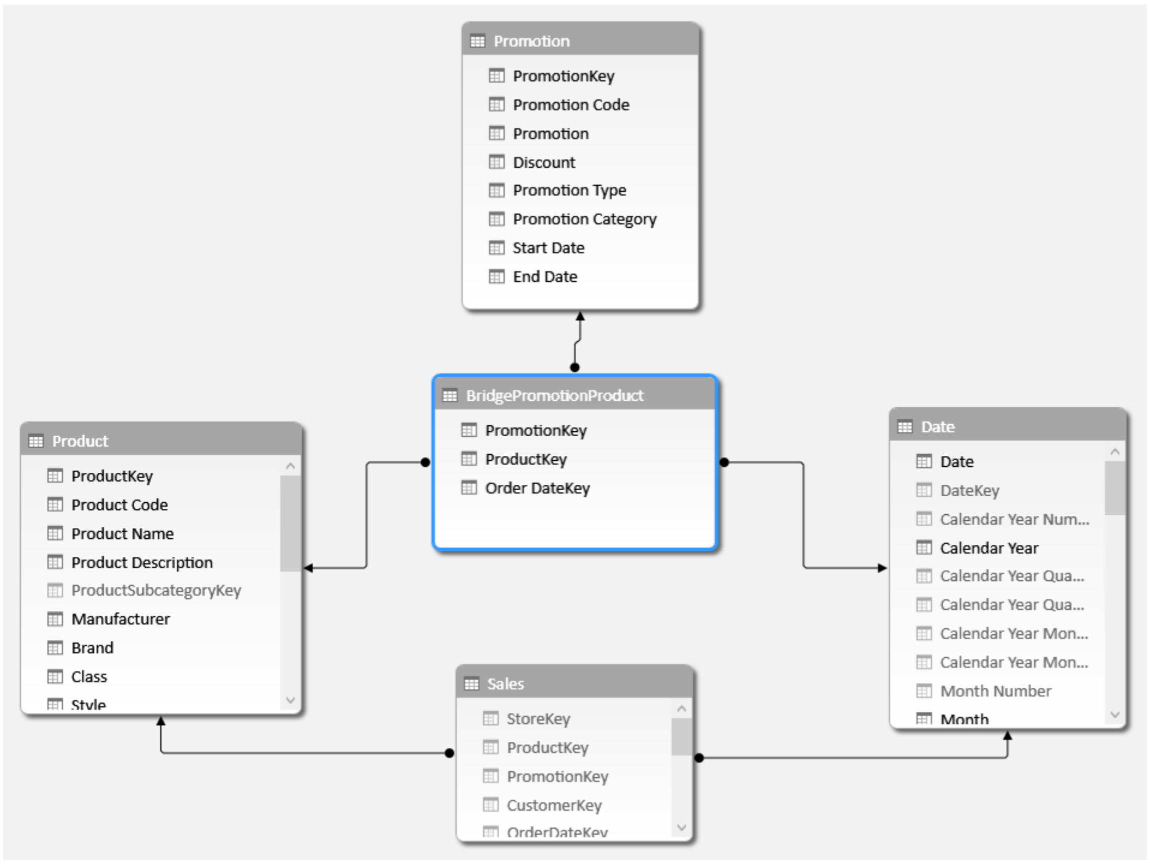
4. Your task is to write the three formulas. OnlineSales and Sales are straightforward, DirectSales is much more challenging. Go for it. The answer, as usual, is following, but your exercise is in finding it by yourself.

```
Sales       := SUM ( Sales[SalesAmount] )
OnlineSales := SUM ( OnlineSales[SalesAmount] )
DirectSales := CALCULATE (
                   SUM ( Sales[SalesAmount] ),
                   CALCULATETABLE ( Product, OnlineSales )
               )
```

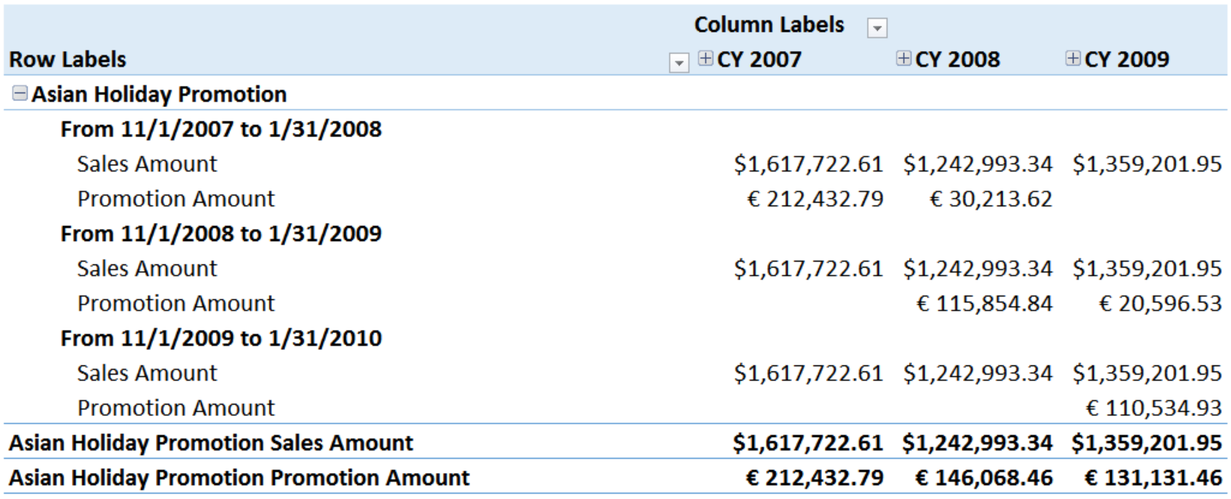
Multiple many-to-many relationships

Many-to-many

This exercise is aimed to test your understanding of table expansion for many-to-many relationships. You have a data model with a bridge table that links products and dates, meaning that, in the give date, the product was on sale with a specific promotion:



Your task is to write a measure that computes the sale amount of only the product that, in a given period of time, were on sale for the given promotion. In other words, you need to produce a report like the following one:



	Column Labels		
Row Labels	**CY 2007**	**CY 2008**	**CY 2009**
Asian Holiday Promotion			
From 11/1/2007 to 1/31/2008			
Sales Amount	$1,617,722.61	$1,242,993.34	$1,359,201.95
Promotion Amount	€ 212,432.79	€ 30,213.62	
From 11/1/2008 to 1/31/2009			
Sales Amount	$1,617,722.61	$1,242,993.34	$1,359,201.95
Promotion Amount		€ 115,854.84	€ 20,596.53
From 11/1/2009 to 1/31/2010			
Sales Amount	$1,617,722.61	$1,242,993.34	$1,359,201.95
Promotion Amount			€ 110,534.93
Asian Holiday Promotion Sales Amount	**$1,617,722.61**	**$1,242,993.34**	**$1,359,201.95**
Asian Holiday Promotion Promotion Amount	**€ 212,432.79**	**€ 146,068.46**	**€ 131,131.46**

Where Sales Amount represents the total sales of the period, Promotion Amount is the amount of only the product in the specific promotion and period.

As usual, the formula is very short, finding is a harder task. Start thinking and… don’t look at the solution too early!

```
Promotion Amount :=
CALCULATE (
    SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] ),
    BridgePromotionProduct
)
```

As you might notice from the formula, the expanded version of the bridge table contains both products and dates which, in turn, are contained in Sales. Thus, it is enough to put the bridge table as a filter parameter of CALCULATE to obtain the solution of this multiple many-to-many relationship.

You might have found a different solution, like the following one:

```
Wrong Promotion Amount :=
CALCULATE (
    SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] ),
    CALCULATETABLE ( Product, BridgePromotionProduct ),
    CALCULATETABLE ( 'Date', BridgePromotionProduct )
)
```

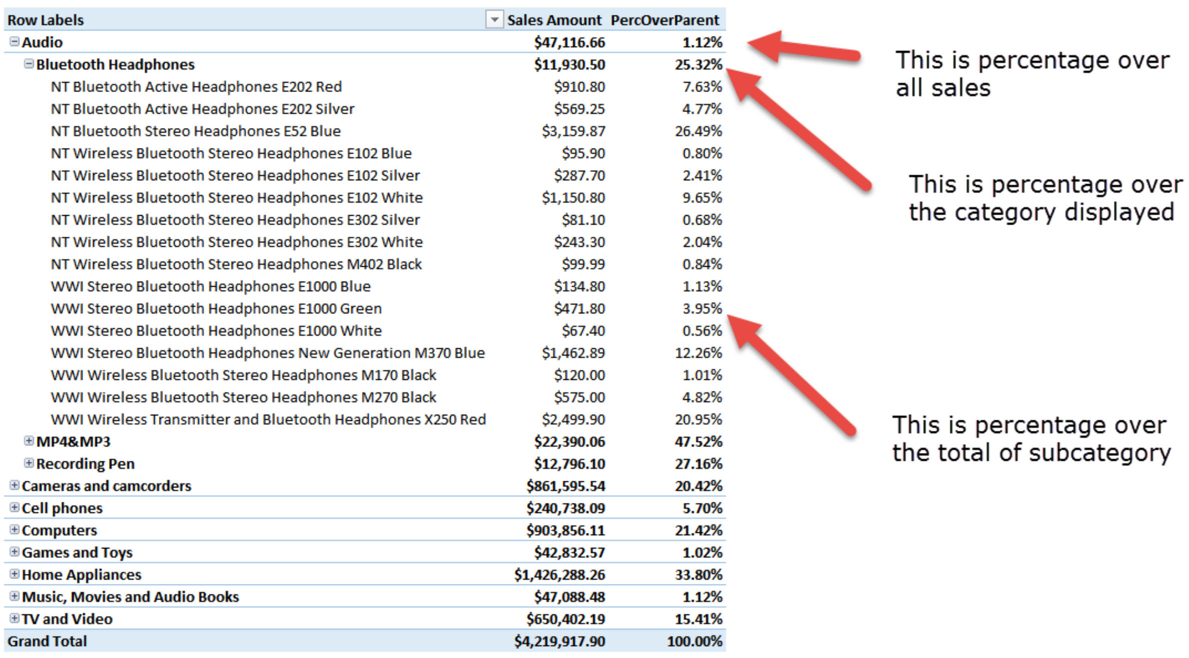
Can you figure what is wrong with this code? You can put it in the pivot table to see that the results produced are wrong but… why? Remember, the teacher is there to help you understanding this.

Lab 8
Hierarchies

Ratio Over Parent

This exercise is intended to let you gain familiarity on working on hierarchies.

Using our usual data model, you want to compute a measure, named PercOverParent, which computes the percentage of sales over the parent in the Categories hierarchy of the Product table. The result need to produce a measure like this one:



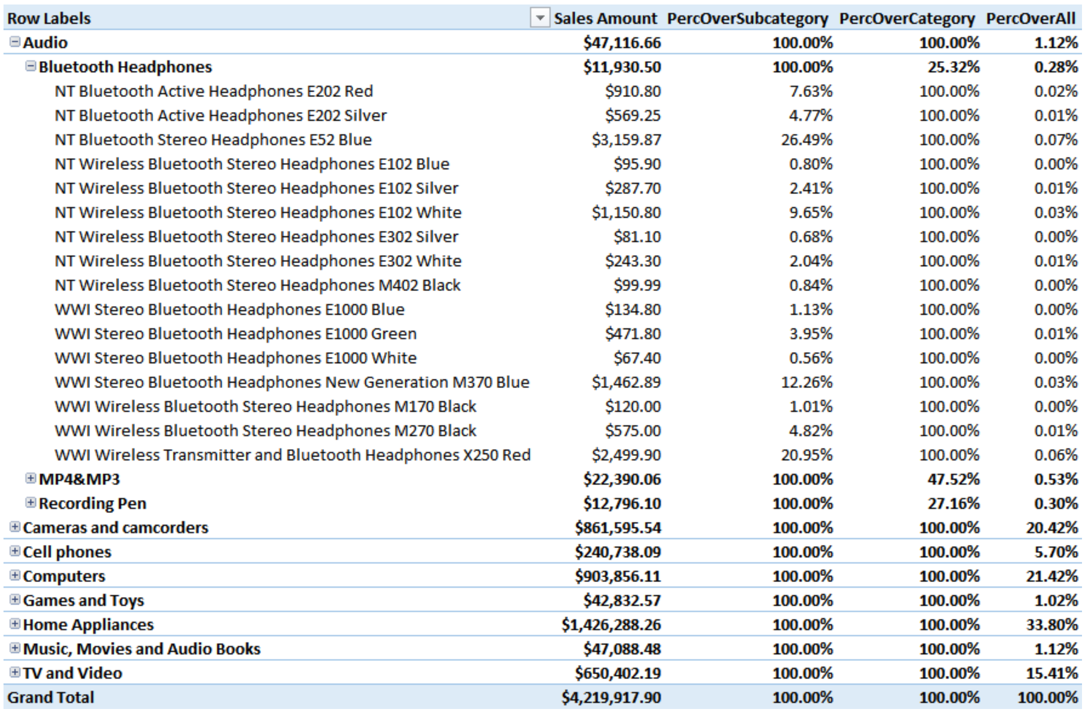
Row Labels	Sales Amount	PercOverParent
⊟Audio	$47,116.66	1.12%
⊟Bluetooth Headphones	$11,930.50	25.32%
NT Bluetooth Active Headphones E202 Red	$910.80	7.63%
NT Bluetooth Active Headphones E202 Silver	$569.25	4.77%
NT Bluetooth Stereo Headphones E52 Blue	$3,159.87	26.49%
NT Wireless Bluetooth Stereo Headphones E102 Blue	$95.90	0.80%
NT Wireless Bluetooth Stereo Headphones E102 Silver	$287.70	2.41%
NT Wireless Bluetooth Stereo Headphones E102 White	$1,150.80	9.65%
NT Wireless Bluetooth Stereo Headphones E302 Silver	$81.10	0.68%
NT Wireless Bluetooth Stereo Headphones E302 White	$243.30	2.04%
NT Wireless Bluetooth Stereo Headphones M402 Black	$99.99	0.84%
WWI Stereo Bluetooth Headphones E1000 Blue	$134.80	1.13%
WWI Stereo Bluetooth Headphones E1000 Green	$471.80	3.95%
WWI Stereo Bluetooth Headphones E1000 White	$67.40	0.56%
WWI Stereo Bluetooth Headphones New Generation M370 Blue	$1,462.89	12.26%
WWI Wireless Bluetooth Stereo Headphones M170 Black	$120.00	1.01%
WWI Wireless Bluetooth Stereo Headphones M270 Black	$575.00	4.82%
WWI Wireless Transmitter and Bluetooth Headphones X250 Red	$2,499.90	20.95%
⊞MP4&MP3	$22,390.06	47.52%
⊞Recording Pen	$12,796.10	27.16%
⊞Cameras and camcorders	$861,595.54	20.42%
⊞Cell phones	$240,738.09	5.70%
⊞Computers	$903,856.11	21.42%
⊞Games and Toys	$42,832.57	1.02%
⊞Home Appliances	$1,426,288.26	33.80%
⊞Music, Movies and Audio Books	$47,088.48	1.12%
⊞TV and Video	$650,402.19	15.41%
Grand Total	$4,219,917.90	100.00%

This is percentage over all sales
This is percentage over the category displayed
This is percentage over the total of subcategory

As you can see from the figure, the same measure shows different calculations, depending on the level of the hierarchy that is visible in the PivotTable.

In order to solve this scenario, we suggest you do the following:

1. Create three different measures: PercOverSubCategory, PercOverCategory and PercOverAll using the knowledge you gained so far. This will let you create a PivotTable like the following one:



Row Labels	Sales Amount	PercOverSubcategory	PercOverCategory	PercOverAll
⊟ **Audio**	**$47,116.66**	**100.00%**	**100.00%**	**1.12%**
⊟ **Bluetooth Headphones**	**$11,930.50**	**100.00%**	**25.32%**	**0.28%**
NT Bluetooth Active Headphones E202 Red	$910.80	7.63%	100.00%	0.02%
NT Bluetooth Active Headphones E202 Silver	$569.25	4.77%	100.00%	0.01%
NT Bluetooth Stereo Headphones E52 Blue	$3,159.87	26.49%	100.00%	0.07%
NT Wireless Bluetooth Stereo Headphones E102 Blue	$95.90	0.80%	100.00%	0.00%
NT Wireless Bluetooth Stereo Headphones E102 Silver	$287.70	2.41%	100.00%	0.01%
NT Wireless Bluetooth Stereo Headphones E102 White	$1,150.80	9.65%	100.00%	0.03%
NT Wireless Bluetooth Stereo Headphones E302 Silver	$81.10	0.68%	100.00%	0.00%
NT Wireless Bluetooth Stereo Headphones E302 White	$243.30	2.04%	100.00%	0.01%
NT Wireless Bluetooth Stereo Headphones M402 Black	$99.99	0.84%	100.00%	0.00%
WWI Stereo Bluetooth Headphones E1000 Blue	$134.80	1.13%	100.00%	0.00%
WWI Stereo Bluetooth Headphones E1000 Green	$471.80	3.95%	100.00%	0.01%
WWI Stereo Bluetooth Headphones E1000 White	$67.40	0.56%	100.00%	0.00%
WWI Stereo Bluetooth Headphones New Generation M370 Blue	$1,462.89	12.26%	100.00%	0.03%
WWI Wireless Bluetooth Stereo Headphones M170 Black	$120.00	1.01%	100.00%	0.00%
WWI Wireless Bluetooth Stereo Headphones M270 Black	$575.00	4.82%	100.00%	0.01%
WWI Wireless Transmitter and Bluetooth Headphones X250 Red	$2,499.90	20.95%	100.00%	0.06%
⊞ **MP4&MP3**	**$22,390.06**	**100.00%**	**47.52%**	**0.53%**
⊞ **Recording Pen**	**$12,796.10**	**100.00%**	**27.16%**	**0.30%**
⊞ **Cameras and camcorders**	**$861,595.54**	**100.00%**	**100.00%**	**20.42%**
⊞ **Cell phones**	**$240,738.09**	**100.00%**	**100.00%**	**5.70%**
⊞ **Computers**	**$903,856.11**	**100.00%**	**100.00%**	**21.42%**
⊞ **Games and Toys**	**$42,832.57**	**100.00%**	**100.00%**	**1.02%**
⊞ **Home Appliances**	**$1,426,288.26**	**100.00%**	**100.00%**	**33.80%**
⊞ **Music, Movies and Audio Books**	**$47,088.48**	**100.00%**	**100.00%**	**1.12%**
⊞ **TV and Video**	**$650,402.19**	**100.00%**	**100.00%**	**15.41%**
Grand Total	$4,219,917.90	100.00%	100.00%	100.00%

2. The next step is to detect the level at which you are in the hierarchy, using the ISFILTERED function, and show the correct measure at the correct level.
3. Do it by yourself, asking help to the teacher if you need some. The solution comes in the next paragraph but, as with any other exercise, it is better if you try to solve it by yourself.

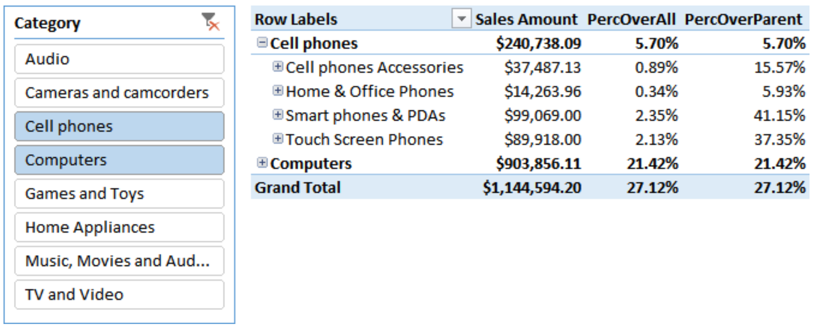
```
PercOverSubcategory:=
DIVIDE (
    [Sales Amount],
    CALCULATE ( [Sales Amount], ALL ( Product[Product Name] ) )
)

PercOverCategory:=
DIVIDE (
    [Sales Amount],
    CALCULATE ( [Sales Amount], ALL ( Product[Subcategory] ) )
)

PercOverAll:=
DIVIDE (
    [Sales Amount],
    CALCULATE ( [Sales Amount], ALL ( Product ) )
)

PercOverParent :=
IF (
    ISFILTERED ( Product[Product Name] ),
    [PercOverSubcategory],
    IF (
        ISFILTERED ( Product[Subcategory] ),
        [PercOverCategory],
        [PercOverAll]
    )
)
```

4. Now, add a slicer for the Category and look at the results:

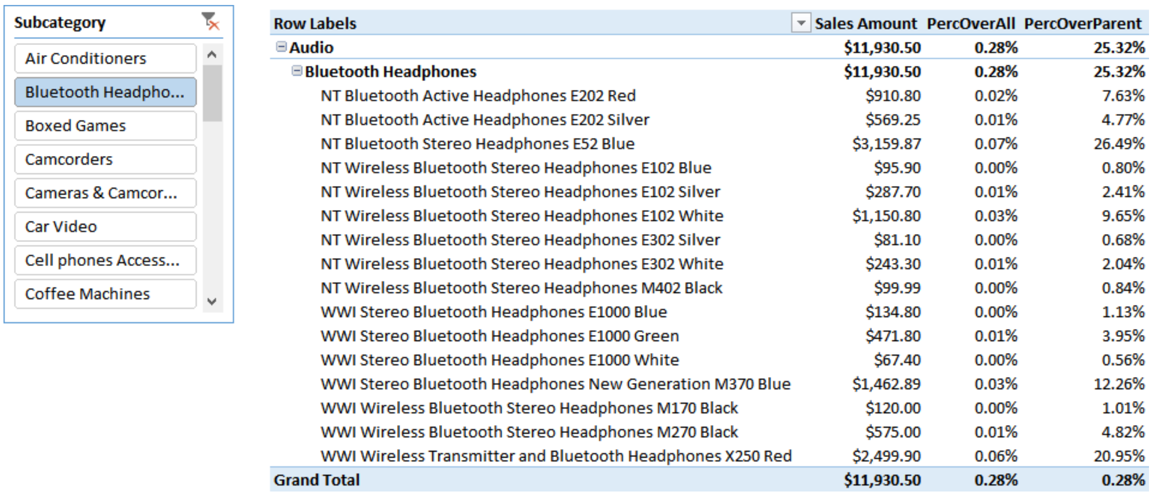


Category
Audio
Cameras and camcorders
Cell phones
Computers
Games and Toys
Home Appliances
Music, Movies and Aud...
TV and Video

Row Labels	Sales Amount	PercOverAll	PercOverParent
⊟Cell phones	$240,738.09	5.70%	5.70%
⊞Cell phones Accessories	$37,487.13	0.89%	15.57%
⊞Home & Office Phones	$14,263.96	0.34%	5.93%
⊞Smart phones & PDAs	$99,069.00	2.35%	41.15%
⊞Touch Screen Phones	$89,918.00	2.13%	37.35%
⊞Computers	$903,856.11	21.42%	21.42%
Grand Total	$1,144,594.20	27.12%	27.12%

The values of PercOverAll (and PercOverParent) are wrong. Can you fix them?

5. Now, add a slicer for the Subcategory, remove any filter from the category and add a filter on the Subcategory, filtering only Bluetooth Headphones. The numbers will be as shown:



Subcategory
Air Conditioners
Bluetooth Headpho...
Boxed Games
Camcorders
Cameras & Camcor...
Car Video
Cell phones Access...
Coffee Machines

Row Labels	Sales Amount	PercOverAll	PercOverParent
⊟Audio	$11,930.50	0.28%	25.32%
⊟Bluetooth Headphones	$11,930.50	0.28%	25.32%
NT Bluetooth Active Headphones E202 Red	$910.80	0.02%	7.63%
NT Bluetooth Active Headphones E202 Silver	$569.25	0.01%	4.77%
NT Bluetooth Stereo Headphones E52 Blue	$3,159.87	0.07%	26.49%
NT Wireless Bluetooth Stereo Headphones E102 Blue	$95.90	0.00%	0.80%
NT Wireless Bluetooth Stereo Headphones E102 Silver	$287.70	0.01%	2.41%
NT Wireless Bluetooth Stereo Headphones E102 White	$1,150.80	0.03%	9.65%
NT Wireless Bluetooth Stereo Headphones E302 Silver	$81.10	0.00%	0.68%
NT Wireless Bluetooth Stereo Headphones E302 White	$243.30	0.01%	2.04%
NT Wireless Bluetooth Stereo Headphones M402 Black	$99.99	0.00%	0.84%
WWI Stereo Bluetooth Headphones E1000 Blue	$134.80	0.00%	1.13%
WWI Stereo Bluetooth Headphones E1000 Green	$471.80	0.01%	3.95%
WWI Stereo Bluetooth Headphones E1000 White	$67.40	0.00%	0.56%
WWI Stereo Bluetooth Headphones New Generation M370 Blue	$1,462.89	0.03%	12.26%
WWI Wireless Bluetooth Stereo Headphones M170 Black	$120.00	0.00%	1.01%
WWI Wireless Bluetooth Stereo Headphones M270 Black	$575.00	0.01%	4.82%
WWI Wireless Transmitter and Bluetooth Headphones X250 Red	$2,499.90	0.06%	20.95%
Grand Total	$11,930.50	0.28%	0.28%

Can you spot the problem? The PercOverParent and PercOverAll show different numbers at the category level. Why that? How can you fix it?

Try out a few ideas, and discuss them with the teacher.

LAB 9
Advanced Relationships

Static Segmentation

Static Segmentation

This first exercise is aimed to show you how to solve the banding scenario, you have already seen during the course how to write the DAX code that performs banding, now it is your turn to perform the same operation, solving a couple of issues too.

The starting point is the data model with sales, products and dates. You already have a segment configuration table available as an Excel table, containing the various segments.

SegmentCode	Segment	MinPrice	MaxPrice
1	VERY LOW	0	20
2	LOW	20	30
3	MEDIUM	30	250
4	HIGH	250	500
5	VERY HIGH	500	99999

1. Write a formula that brings, in the sales table, the band name. You need to use CALCULATE and set a filter that shows, in the band table, the only row for which the product price is between the boundaries. Try it by yourself, before looking at the solution.

```
Sales[PriceSegment] =
CALCULATE (
    VALUES ( PriceSegments[Segment] ),
    FILTER (
        PriceSegments,
        AND (
            PriceSegments[MinPrice] <= Sales[Net Price],
            PriceSegments[MaxPrice] > Sales[Net Price]
        )
    )
)
```

2. Using this formula, you can now produce a PivotTable like the following:

TotalSales	Column Labels			
Row Labels	CY 2007	CY 2008	CY 2009	Grand Total
HIGH	$240,380.90	$299,952.10	$192,072.86	$732,405.86
LOW	$4,842.63	$8,548.86	$13,590.07	$26,981.56
MEDIUM	$137,622.05	$242,537.15	$196,632.11	$576,791.31
VERY HIGH	$496,388.18	$559,077.91	$392,629.17	$1,448,095.26
VERY LOW	$6,124.88	$12,419.03	$9,961.67	$28,505.58
Grand Total	**$885,358.63**	**$1,122,535.05**	**$804,885.87**	**$2,812,779.56**

You can see that the segments are not sorted correctly, HIGH is the first, VERY LOW is the last one. Power Pivot used alphabetical sorting, which is clearly the wrong choice in this case. Solve the issue by bringing the SegmentCode in the fact table, using a similar formula, and then setting the Sort By Column option on PriceSegment. At the end, your PivotTable should look like this:

TotalSales	**Column Labels**			
Row Labels	**CY 2007**	**CY 2008**	**CY 2009**	**Grand Total**
VERY LOW	$6,124.88	$12,419.03	$9,961.67	$28,505.58
LOW	$4,842.63	$8,548.86	$13,590.07	$26,981.56
MEDIUM	$137,622.05	$242,537.15	$196,632.11	$576,791.31
HIGH	$240,380.90	$299,952.10	$192,072.86	$732,405.86
VERY HIGH	$496,388.18	$559,077.91	$392,629.17	$1,448,095.26
Grand Total	**$885,358.63**	**$1,122,535.05**	**$804,885.87**	**$2,812,779.56**

3. At this point, you can modify the configuration table and, in order to make some test, you enter a wrong one, where the segments overlap, like in the following figure:

SegmentCode	Segment	MinPrice	MaxPrice
1	VERY LOW	0	20
2	LOW	20	100
3	MEDIUM	30	250
4	HIGH	250	500
5	VERY HIGH	500	99999

4. After having modified the table, you will see that the calculated column PriceSegment in Sales shows errors. The problem is that the VALUES () call returned more than one row, due to overlapping segments. Can you think at a way to solve the issue, showing the message “Wrong Configuration” if an error is detected? Take your time, and think at the solution, which, as usual, we provide here:

```
Sales[PriceSegment] =
CALCULATE (
    IFERROR (
        VALUES ( PriceSegments[Segment] ),
        "Wrong Configuration"
    ),
    FILTER (
        PriceSegments,
        AND (
            PriceSegments[MinPrice] <= Sales[Net Price],
            PriceSegments[MaxPrice] > Sales[Net Price]
        )
    )
)
```

Obviously, a similar update is needed for the column you use for sorting.

5. With this new formula, now in presence of an error, you get a result, not meaningful, but that at least shows the error, highlighted in the next figure:

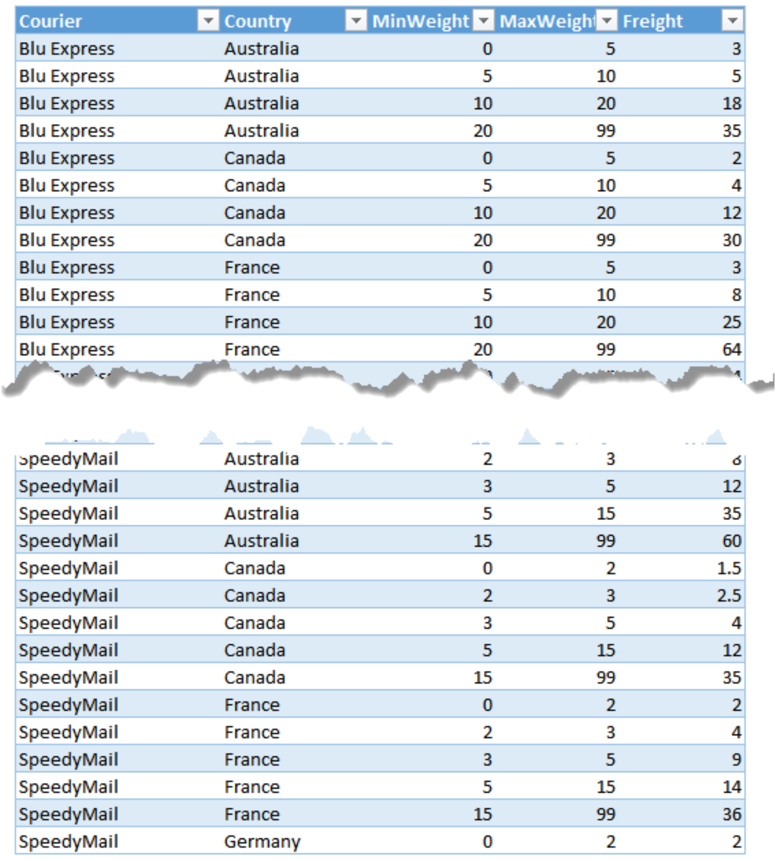
TotalSales	**Column Labels**			
Row Labels	**CY 2007**	**CY 2008**	**CY 2009**	**Grand Total**
Wrong Configuration	$22,325.51	$40,277.83	$50,623.82	$113,227.17
VERY LOW	$6,124.88	$12,419.03	$9,961.67	$28,505.58
LOW	$4,842.63	$8,548.86	$13,590.07	$26,981.56
MEDIUM	$115,296.53	$202,259.32	$146,008.28	$463,564.14
HIGH	$240,380.90	$299,952.10	$192,072.86	$732,405.86
VERY HIGH	$496,388.18	$559,077.91	$392,629.17	$1,448,095.26
Grand Total	**$885,358.63**	**$1,122,535.05**	**$804,885.87**	**$2,812,779.56**

6. At this point, you can modify the configuration table, update the linked table and you will see the report showing different values, depending on the specific configuration you created.

Courier Simulation

This lab uses a technique, similar to the previous one, in order to solve a simulation scenario. Imagine you are in charge of handling AdventureWorks shipments. You need to choose the courier that will handle your shipments for the next year and you ask some couriers the cost of shipment.

You collect information about the shipments proposals in a table, which we preloaded in the solution:



Courier	Country	MinWeight	MaxWeigh	Freight
Blu Express	Australia	0	5	3
Blu Express	Australia	5	10	5
Blu Express	Australia	10	20	18
Blu Express	Australia	20	99	35
Blu Express	Canada	0	5	2
Blu Express	Canada	5	10	4
Blu Express	Canada	10	20	12
Blu Express	Canada	20	99	30
Blu Express	France	0	5	3
Blu Express	France	5	10	8
Blu Express	France	10	20	25
Blu Express	France	20	99	64
SpeedyMail	Australia	2	3	8
SpeedyMail	Australia	3	5	12
SpeedyMail	Australia	5	15	35
SpeedyMail	Australia	15	99	60
SpeedyMail	Canada	0	2	1.5
SpeedyMail	Canada	2	3	2.5
SpeedyMail	Canada	3	5	4
SpeedyMail	Canada	5	15	12
SpeedyMail	Canada	15	99	35
SpeedyMail	France	0	2	2
SpeedyMail	France	2	3	4
SpeedyMail	France	3	5	9
SpeedyMail	France	5	15	14
SpeedyMail	France	15	99	36
SpeedyMail	Germany	0	2	2

Each courier states the price of shipping an order of a specific weight in a country. The weight ranges are different for each courier and country, and this makes it hard comparing them. Thus, you want to follow a simulation path. You simulate the adoption of each courier on the past orders, to grab an idea of what happened in the past with the different couriers.

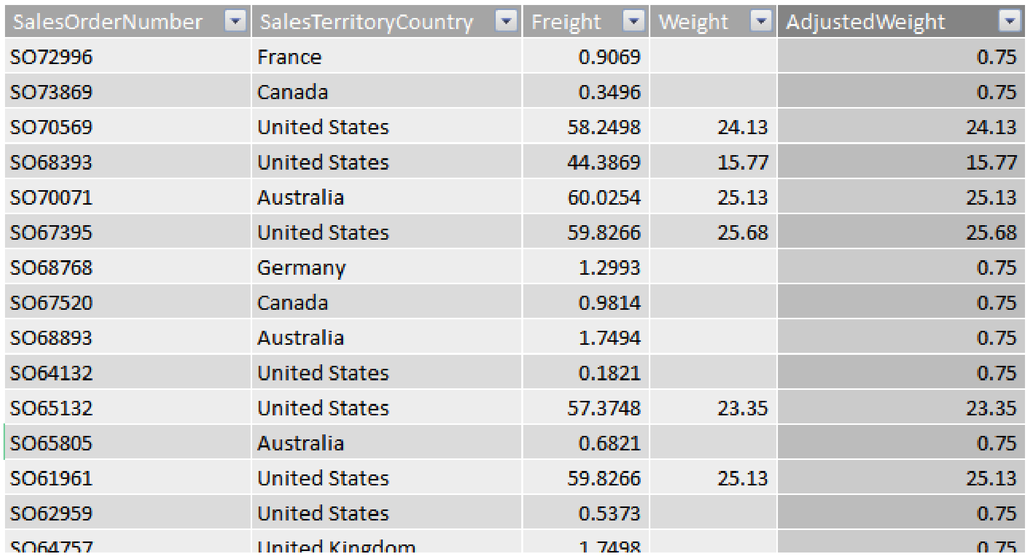
We have created a table (Orders) using this SQL query:

```sql
SELECT
    SalesOrderNumber,
    SalesTerritoryCountry,
    SUM(Freight) AS Freight,
    SUM(OrderQuantity * Weight) AS Weight
FROM
    FactInternetSales
    LEFT OUTER JOIN dbo.DimSalesTerritory
        ON dbo.FactInternetSales.SalesTerritoryKey = dbo.DimSalesTerritory.SalesTerritoryKey
    LEFT OUTER JOIN dbo.DimProduct
        ON dbo.FactInternetSales.ProductKey = dbo.DimProduct.ProductKey
GROUP BY
    SalesOrderNumber,
    SalesTerritoryCountry
```

1. There is a small issue with the result. In fact, as you can see, the weight is missing for several rows. You can easily solve it by adopting a default weight when it is unknown, through the creation of a calculated column AdjustedWeight as follows:

```
AdjustedWeight = IF ( Orders[Weight] = 0, 0.75, Orders [Weight] )
```

The resulting Orders table looks like the following:



SalesOrderNumber	SalesTerritoryCountry	Freight	Weight	AdjustedWeight
SO72996	France	0.9069		0.75
SO73869	Canada	0.3496		0.75
SO70569	United States	58.2498	24.13	24.13
SO68393	United States	44.3869	15.77	15.77
SO70071	Australia	60.0254	25.13	25.13
SO67395	United States	59.8266	25.68	25.68
SO68768	Germany	1.2993		0.75
SO67520	Canada	0.9814		0.75
SO68893	Australia	1.7494		0.75
SO64132	United States	0.1821		0.75
SO65132	United States	57.3748	23.35	23.35
SO65805	Australia	0.6821		0.75
SO61961	United States	59.8266	25.13	25.13
SO62959	United States	0.5373		0.75
SO64757	United Kingdom	1.7498		0.75

2. The second problem you face is the fact that there is no way to create a relationship between this table and the Excel table containing the courier proposals. In fact, in the courier proposals table there are two boundaries for the weight, while in this new table there is a single column Weight. Moreover, for each order there are many rows matching the courier, namely one row for each courier.

   If you take, for example, the first order (S044204), it has a weight of 20 and there are many rows in the configuration table: one for Blue Express and one for Speedy Mail. Thus, the way of trying to create a relationship is going to fail, because you do not have a one-to-many relationship. You need to use DAX in order to solve this scenario.

   While the problem has no solution at the aggregated level, if you focus on a single order and on a single courier, then you can compute the freight by searching for the matching row using a pattern similar to that of the static segmentation.

   In the static segmentation pattern, you only had to search for a band name using the boundaries. Here the formula is slightly more complex because the condition need to be set on both the weigth boundaries and the country name. In fact, both the weight and the country define the freight.

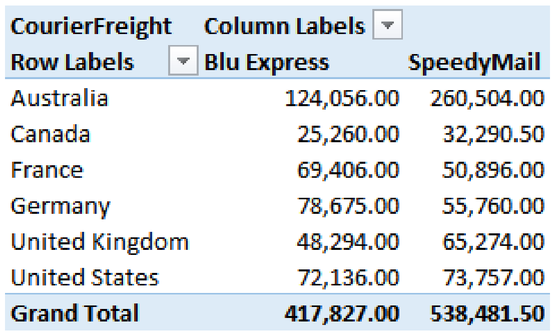
   With this idea in mind, read and get the sense out of the following formula:

```
CourierFreight :=
IF (
    HASONEVALUE ( Couriers[Courier] ),
    SUMX (
        Orders,
        CALCULATE (
            VALUES ( Couriers[Freight] ),
            FILTER (
                Couriers,
                Couriers[MinWeight] < Orders[AdjustedWeight]
                && Couriers[MaxWeight] >= Orders[AdjustedWeight]
                && Couriers[Country] = Orders[SalesTerritoryCountry]
            )
        )
    )
)
```

The interesting parts of the formula are:

- The initial check with HASONEVALUE to compute the value only when a single courier is visible in the filter context.
- SUMX over orders is useful because it iterates each order. By performing the following calculation for a single order, you can then search for the freight of that specific order.
- CALCULATE (VALUES (Couriers[Freight]), …) is a pattern that – in reality – means: "search for the value of the freight in the couriers table, in the row that satisfies the CALCULATE condition, i.e. that has min and max weight that contain the weight of the current order". Thus, we are using CALCULATE to search for a value.
- The condition inside FILTER guarantees that a single row will be visible in the Couriers table, making the VALUES invocation return a correct number.

Using this measure, you can easily build a PivotTable like the following:



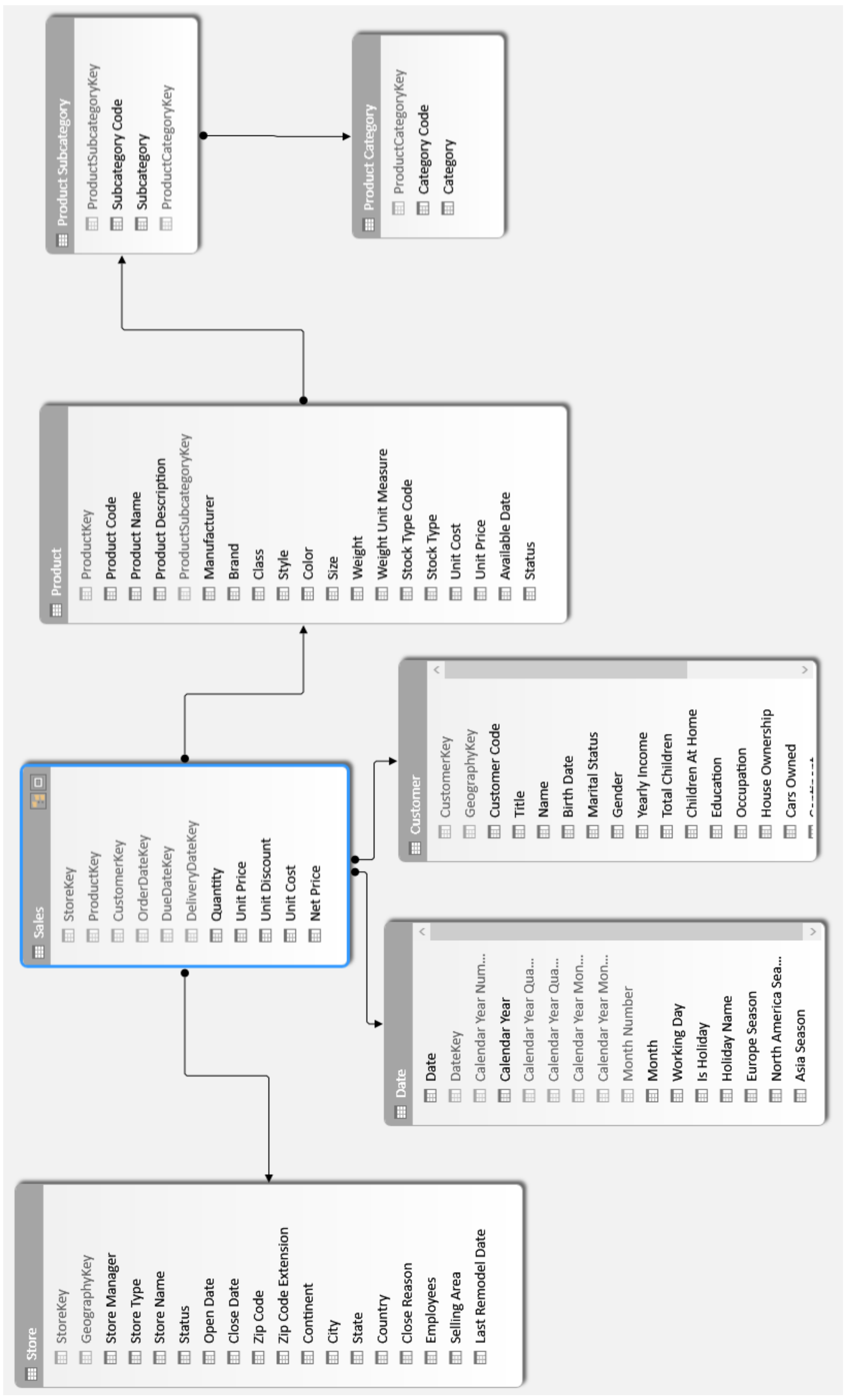
CourierFreight	Column Labels	
Row Labels	Blu Express	SpeedyMail
Australia	124,056.00	260,504.00
Canada	25,260.00	32,290.50
France	69,406.00	50,896.00
Germany	78,675.00	55,760.00
United Kingdom	48,294.00	65,274.00
United States	72,136.00	73,757.00
Grand Total	**417,827.00**	**538,481.50**

The PivotTable makes it easy to see that Blu Express is by far your best option, even if it is higher in some countries, it provides the overall best price.

The exercise – at this point – is solved. You might note, at this point, that we are missing date information. If you want to see the different expenses in different years, then you will need to update the query that generates the Orders table, bring the DateKey column in the Excel table first, in the PowerPivot table next and create a correct relationship between the date of the order and the calendar table.

It is interesting that, with this exercise, you mixed most of the advanced techniques learned during the course: data modeling, DAX for simple calculated columns, missing relationships, an advanced DAX expression that uses most of the techniques you have learned.

Contoso Data Model



Product Subcategory
ProductSubcategoryKey
Subcategory Code
Subcategory
ProductCategoryKey
Product Category
ProductCategoryKey
Category Code
Category
Product
ProductKey
Product Code
Product Name
Product Description
ProductSubcategoryKey
Manufacturer
Brand
Class
Style
Color
Size
Weight
Weight Unit Measure
Stock Type Code
Stock Type
Unit Cost
Unit Price
Available Date
Status
Sales
StoreKey
ProductKey
CustomerKey
OrderDateKey
DueDateKey
DeliveryDateKey
Quantity
Unit Price
Unit Discount
Unit Cost
Net Price
Customer
CustomerKey
GeographyKey
Customer Code
Title
Name
Birth Date
Marital Status
Gender
Yearly Income
Total Children
Children At Home
Education
Occupation
House Ownership
Cars Owned
Date
Date
DateKey
Calendar Year Num...
Calendar Year
Calendar Year Qua...
Calendar Year Qua...
Calendar Year Mon...
Calendar Year Mon...
Month Number
Month
Working Day
Is Holiday
Holiday Name
Europe Season
North America Sea...
Asia Season
Store
StoreKey
GeographyKey
Store Manager
Store Type
Store Name
Status
Open Date
Close Date
Zip Code
Zip Code Extension
Continent
City
State
Country
Close Reason
Employees
Selling Area
Last Remodel Date