

World of Watson Training

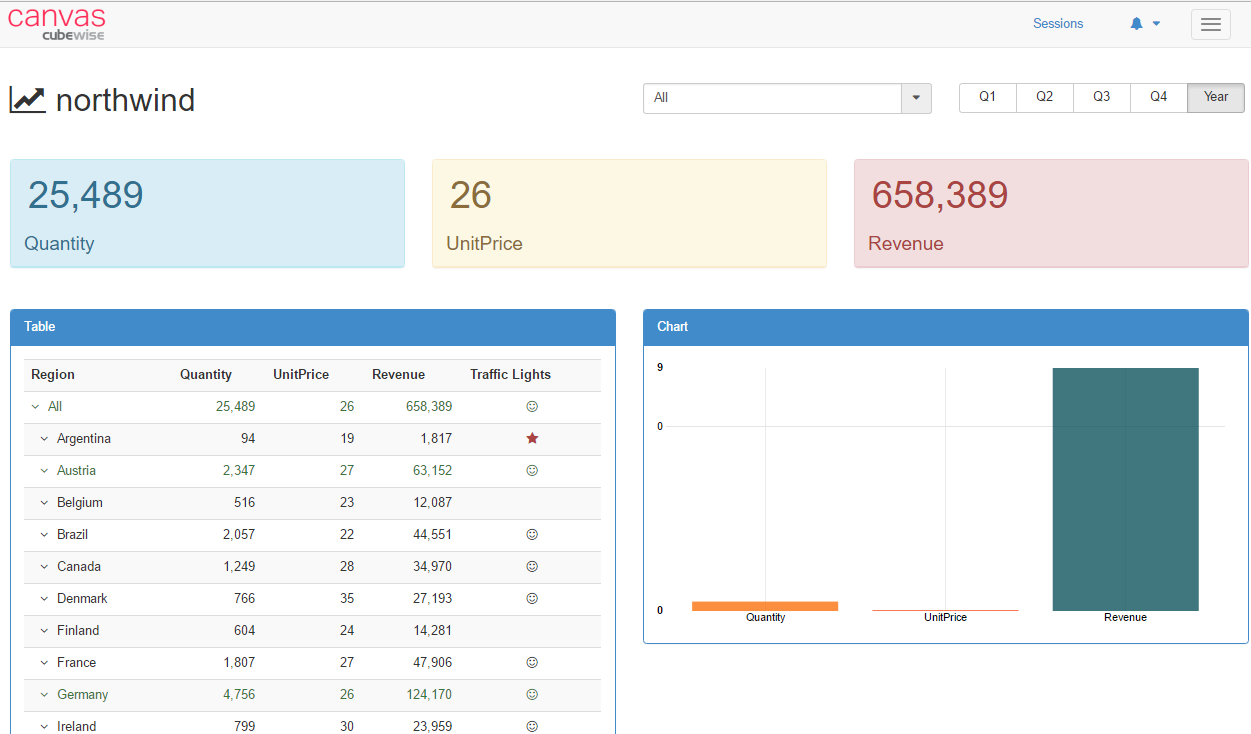


Table of Contents

[Overview 3](#_Toc464485156)

[What is Canvas 3](#_Toc464485157)

[Objective of the training 3](#_Toc464485158)

[Build the view in TM1 5](#_Toc464485159)

[Start-up the Canvas application server 7](#_Toc464485160)

[Open the canvas source editor 8](#_Toc464485161)

[Replace the string "Product selection" with a SUBNM 9](#_Toc464485162)

[Update the 3 "Value" fields 13](#_Toc464485163)

[Get the right cell intersection in TM1 13](#_Toc464485164)

[Replace the first "VALUE" panel 15](#_Toc464485165)

[Make the DBR dynamic 18](#_Toc464485166)

[Update the 2 others "VALUE" fields 22](#_Toc464485167)

[Table 25](#_Toc464485168)

[Create the rowset 26](#_Toc464485169)

[Populate the table with values 31](#_Toc464485170)

[Chart 35](#_Toc464485171)

[Advance dashboard 39](#_Toc464485172)

[Indentation 39](#_Toc464485173)

[Traffic lights 41](#_Toc464485174)

[Conditional Formatting 44](#_Toc464485175)

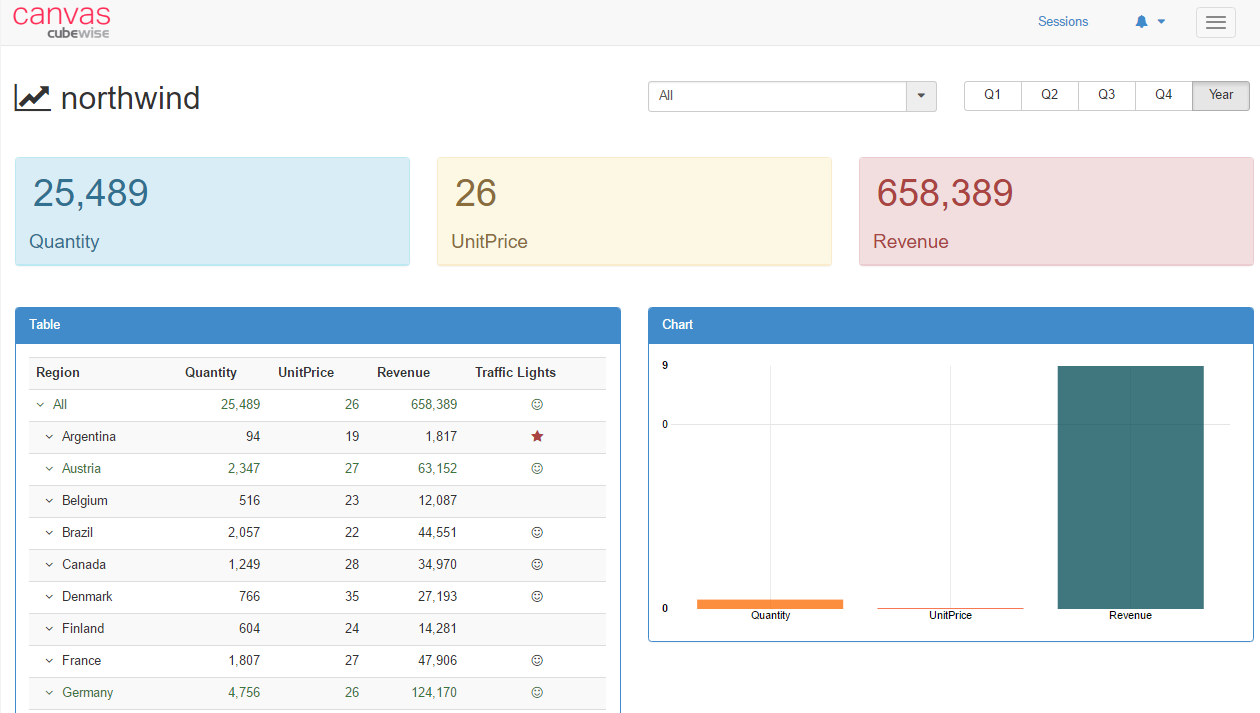
# Overview

## What is Canvas

Canvas is a framework to allow you to build modern web based user interfaces on top of TM1. These interfaces can be read-only dashboards, reporting or complicated planning applications that require sophisticated workflow, data entry and validation. It is designed and built by TM1 people for TM1 people and as you will see in this tutorial it uses common TM1 concepts, such as:  DBRs, SUBNMs, subsets, picklists, annotations, etc. It uses the wonderful REST API to access the content in your TM1 server so it is fast and can be used in geographically disperse locations or cloud environments.

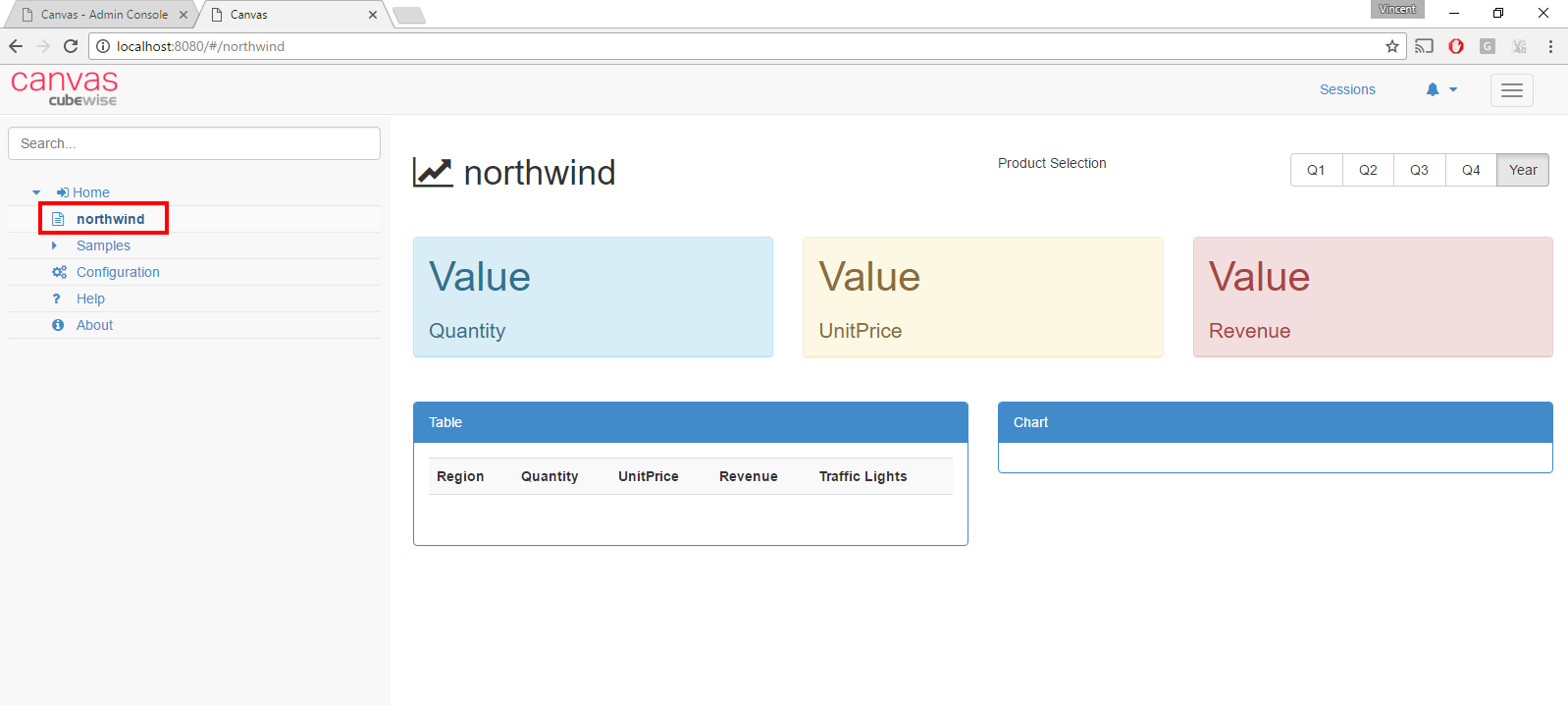
## Objective of the training

This objective of this training is to build the following TM1 dashboard with Canvas:

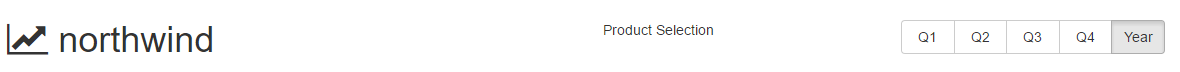


We're going to start from a template called northwind, you can find it in the pre-built menu in the left pane under Home:

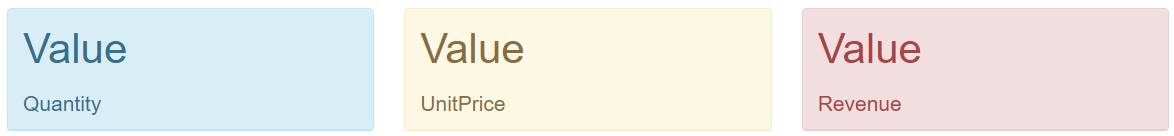
<http://localhost:8080/#/northwind>



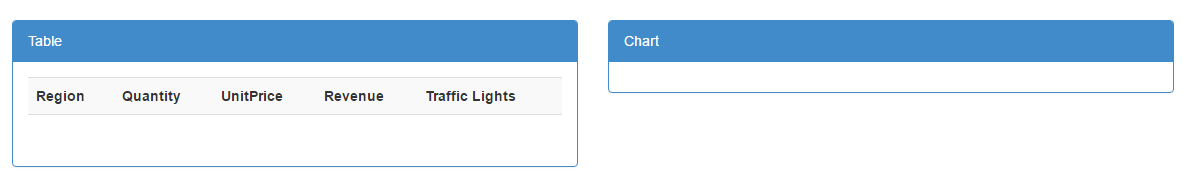
There are 3 main sections in this template. The first section at the top has a Product selection and 5 buttons to update the month:



The second section in the middle contains 3 panels:



and the last section is broken into 2 blocs, one table and one chart:

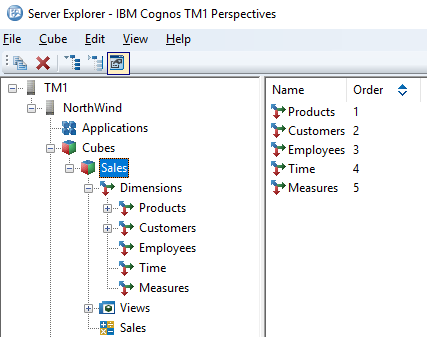


We're going to fill out all these sections, one Product selection at the top, then we're going to replace all 3 “Value” fields with dynamic values from a TM1 cube.

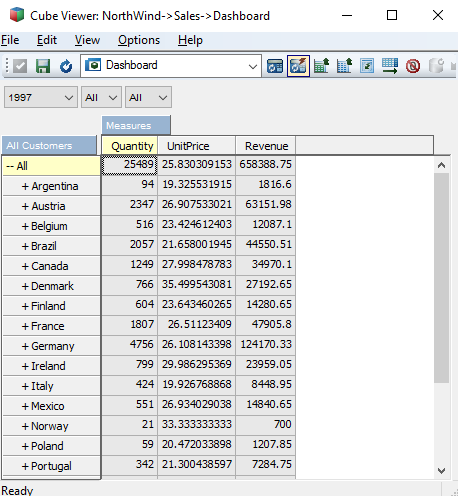
Then we'll populate the cross-tab table and finally to round out the tutorial by building an interactive chart connected to TM1.

# Build the view in TM1

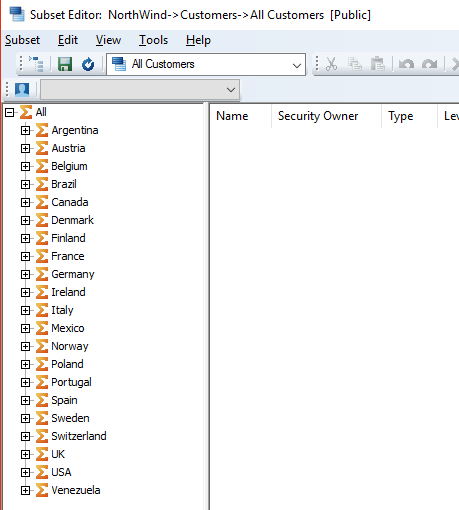
The first step of the training is to build a view from the Sales cube:



Open the Sales cube, and build the following view called “Dashboard”:



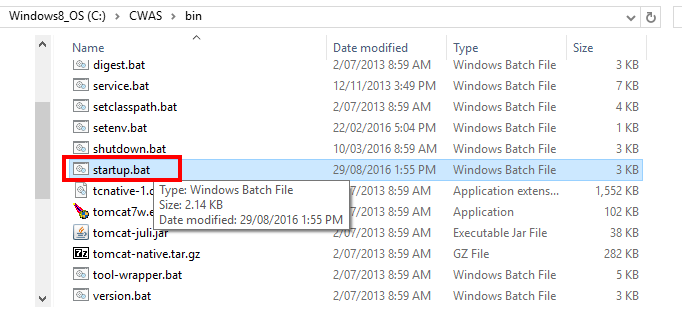
Do not forget to build the Customers subset called “All Customer”:

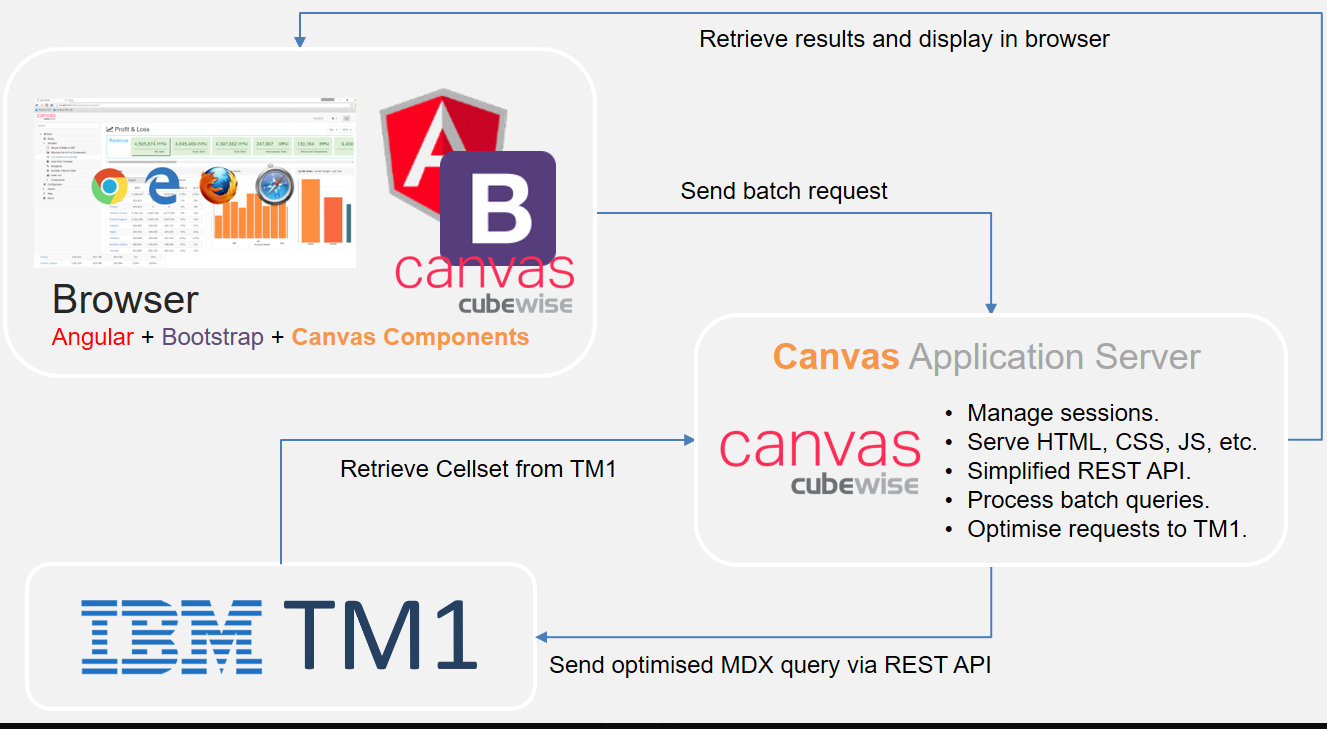


# Start-up the Canvas application server

Canvas uses a Apache Tomcat server, which is different to an Apache HTTP Server. Tomcat is a Java application server that can be used for lots of purposes including serving HTML pages, CSS, etc..

To start-up the Tomcat server, you need to double click on C:\CWAS\bin\startup.bat:

 Canvas uses a combination of client side components and an application server that simplifies requests to the TM1 server:



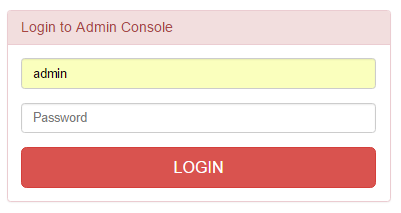
# Open the canvas source editor

Canvas has a Source Editor where you can update the HTML code of the Canvas pages. You can access the Source Editor through the following url:

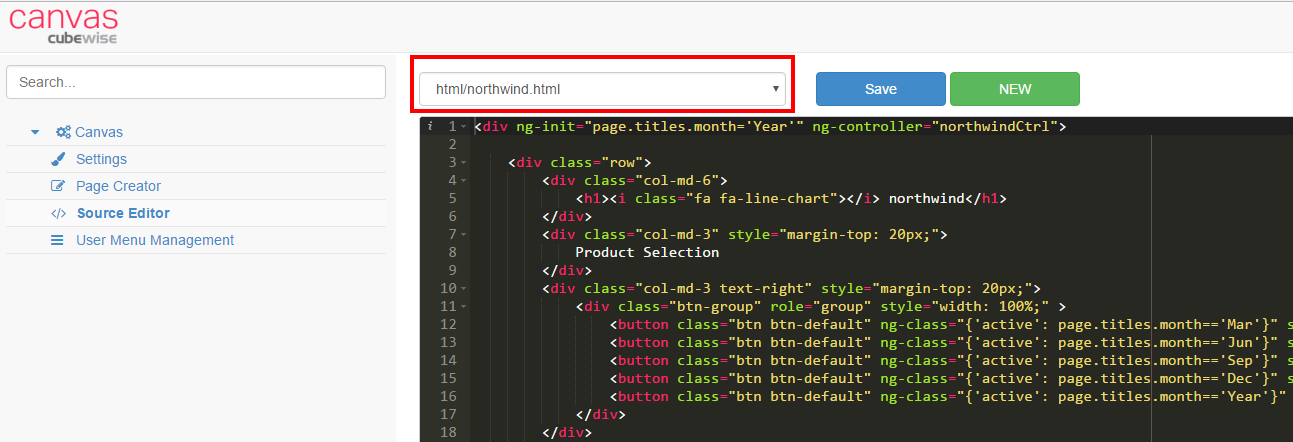
http://<servername>:<portnumber>/admin#/editor

In our example: <http://localhost:8080/admin#/editor>

The Source Editor is part of the Canvas Admin section, you will need to log in using the admin login and password:



Once you logged in just select the dashboard from the drop-down list (html/northwind.html):



The best way to continue the training is to have two web browser sessions open:

* One with the dashboard (http://localhost:8080/#/northwind)
* One with Source Editor (http://localhost:8080/admin#/page-creator).

# Replace the string "Product selection" with a SUBNM

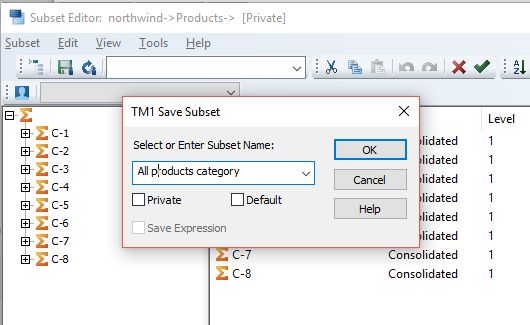
The first step we need to do is to replace the string "Product Selection" with a list of product elements coming from a TM1 subset:



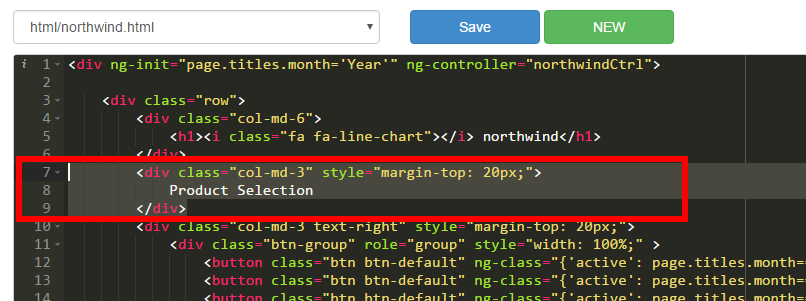
Maintaining your Canvas reports and data entry forms through TM1 subsets will reduce the time spent maintaining and updating these assets. Any changes in a subset will automatically be reflected in Canvas.

Let’s create the following subset in TM1:

1. Open the Products dimension
2. Save the subset as “All products category”:



The code which shows the "product selection" string in the dashboard can be found at the top of the HTML file:



<div class="col-md-3" style="margin-top: 20px;">

Product Selection

</div>

Let's replace the string “Product Selection” with a SUBNM formula:

<tm1-ui-subnm tm1-instance=" "

tm1-dimension=" "

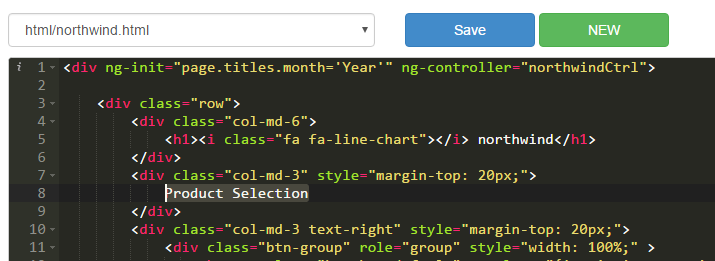
tm1-subset=" "

tm1-show-hierarchy=" "

ng-model=" ">

</tm1-ui-subnm>

Let's copy the tm1-ui-subnm HTML code above, go back to the Source Editor and replace the "Product Selection" with the SUBNM you just copied:



Update the parameters as below:

**tm1-instance** : northwind

**tm1-dimension**: Products

**tm1-subset**: All products category

**ng-model**: page.titles.product

Be careful HTML is case sensitive.

The code will look like:

<div class="col-md-3" style="margin-top: 20px;">

<tm1-ui-subnm tm1-instance="northwind"

tm1-dimension="Products"

tm1-subset="All products category"

tm1-show-hierarchy="false"

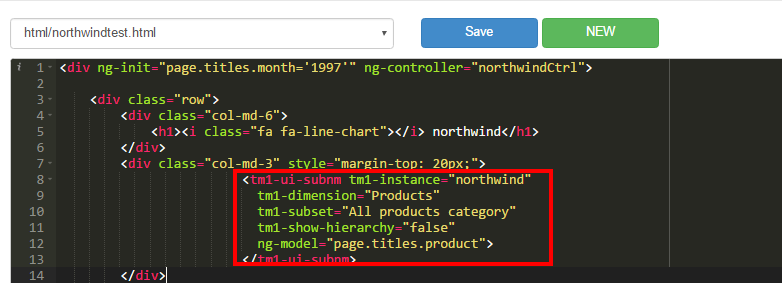
ng-model="page.titles.product">

</tm1-ui-subnm>

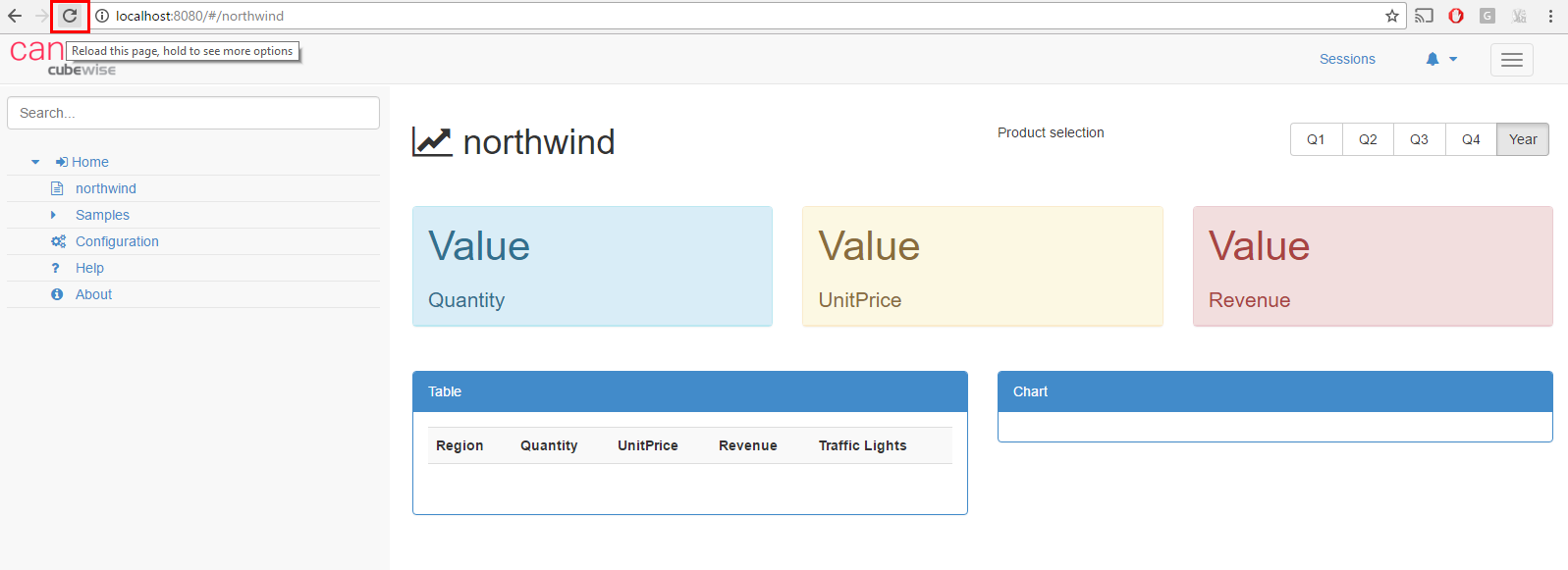
</div>

We use [ng-model](http://www.w3schools.com/angular/angular_model.asp) to store the value of the list in a variable called "page.titles.product". By storing the value of the SUBNM as a variable, we will be able to use this value in other page items, the values in our dashboard will be updated depending on the value of this SUBNM.

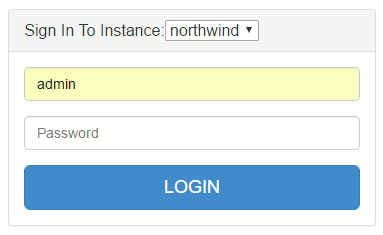
Click the save button:



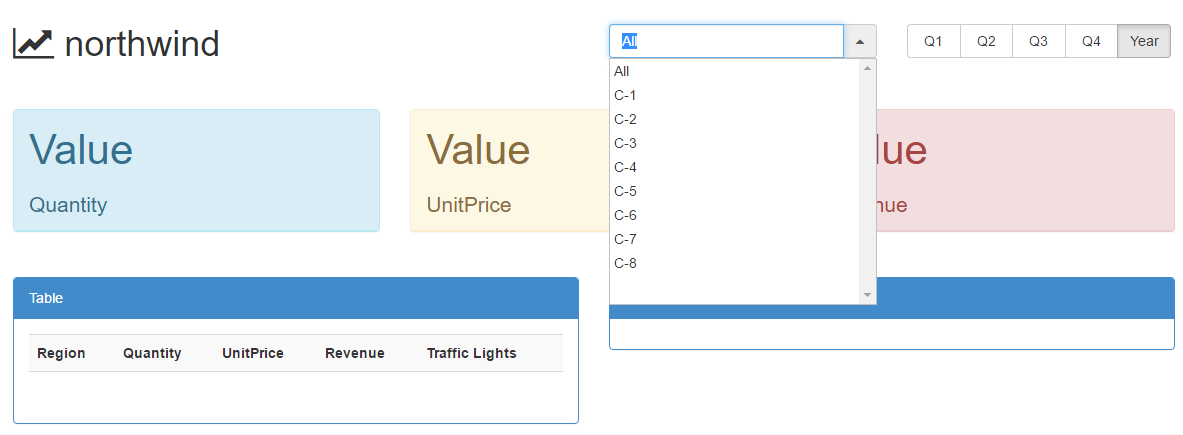
Now go to the dashboard and press refresh:



A log in page should pop-up, login is admin with no password:

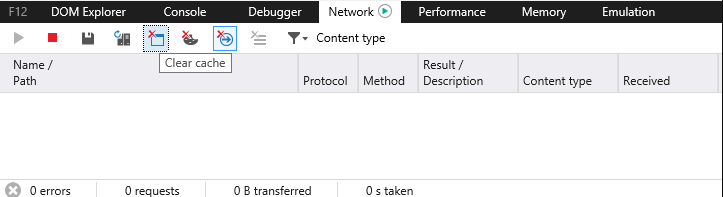


Click login and you should now be able to see the dropdown list:

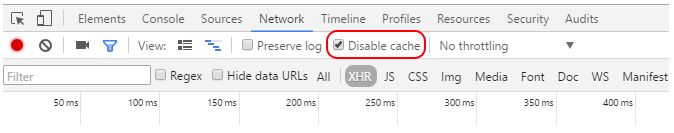


If you cannot see the change, just make sure that you have [disable cache](http://www.canvasfortm1.com/help-content/2016/6/14/disable-cache) in your web browser:

With Internet Explorer, press F12 and go to Network and click clear cache:

****

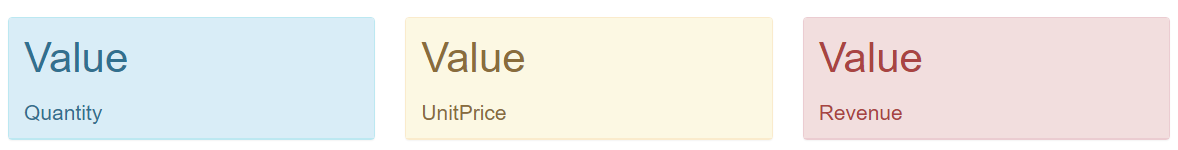
With Google Chrome press F12 and go to Network and click disable cache:



Every time you make a change to your HTML page, if you cannot see the change, clear the cache.

# Update the 3 "Value" fields

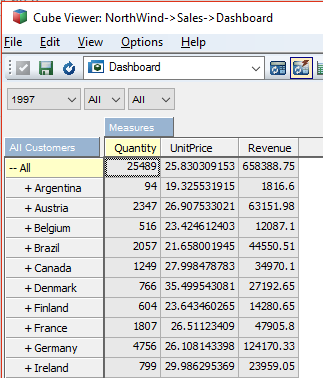
Now let's populate the 3 “Value” fields with values coming from TM1:



## Get the right cell intersection in TM1

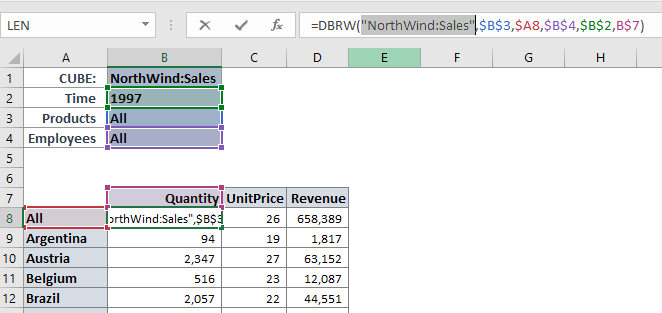
Similar to TM1 and Excel, Canvas has adopted a cell based approach to application design.  This means each cell in Canvas has its own DBR formula to retrieve and to update TM1. 

In perspectives open the `view we created:

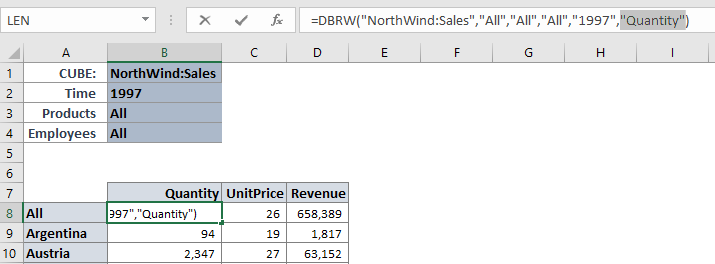


 In this view we have 3 measures Quantity, UnitPrice and Revenue which will ultimately populate the "Value" fields in our dashboard.

Let's slice the view and then use the F9 key on the first parameter to get the exact element value:



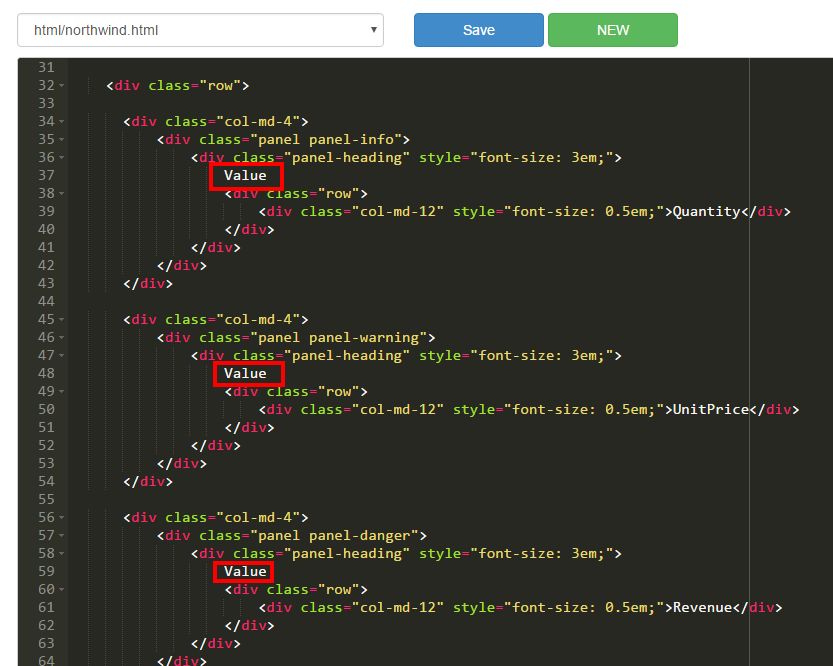
And do the same for all other parameters the DBRW:



Now we have the exact TM1 reference for this specific cell: "All","All","All","1997","Quantity"

## Replace the first "VALUE" panel

Let's jump back into the Source Editor and look for the HTML code where the "Value" is hard coded. We can see in red below the three bloc of code which create the 3 “Value” fields:



Let's replace the first "Value" with a DBR.

Similar to the DBR function in Excel, Canvas uses DBR function to retrieve and update the TM1 server. With Canvas you can customize the contents of cells by adding cell validation and in cell instruction.

The HTML code to create a DBR is:

<tm1-ui-dbr tm1-instance=" "

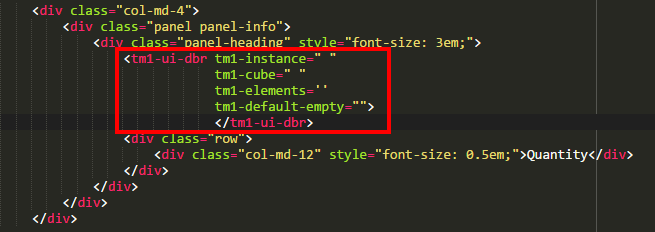
tm1-cube=" "

tm1-elements=''

tm1-default-empty="">

</tm1-ui-dbr>

Let's copy and paste this code over the first "Value":



Now let's replace the tm1-cube value with "Sales" and the tm1-elements with the intersection we just copied ("All","All","All","1997","Quantity").

Be aware that because there is double quote in the list you copied, you have to enclose the complete elements string with single quote:

<tm1-ui-dbr tm1-instance="northwind"

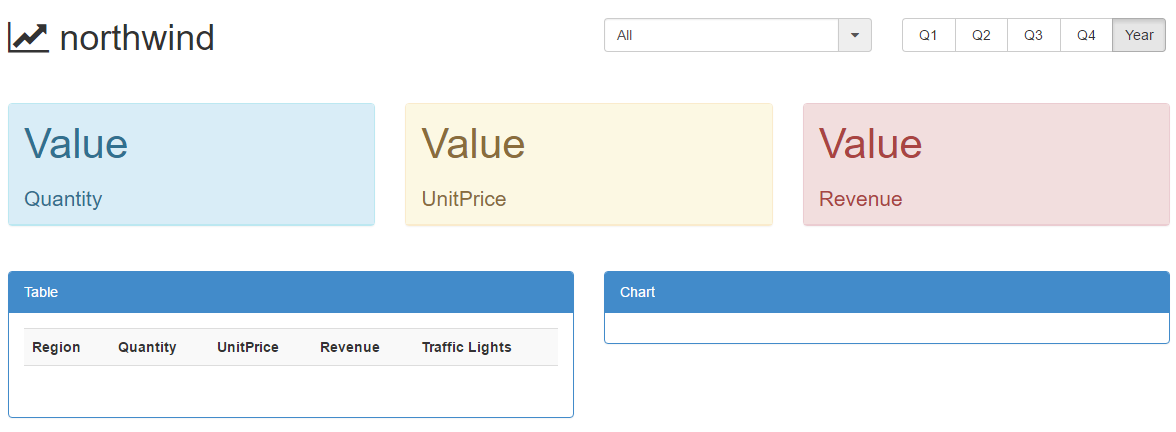
tm1-cube="Sales"

tm1-elements='"All","All","All","1997","Quantity"'

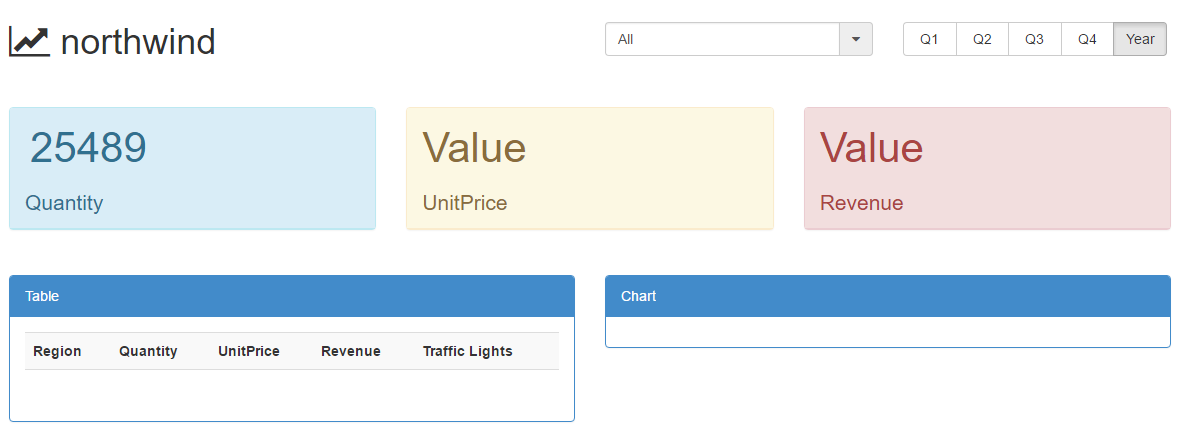
tm1-default-empty="">

</tm1-ui-dbr>

Let's save and go back to the dashboard:

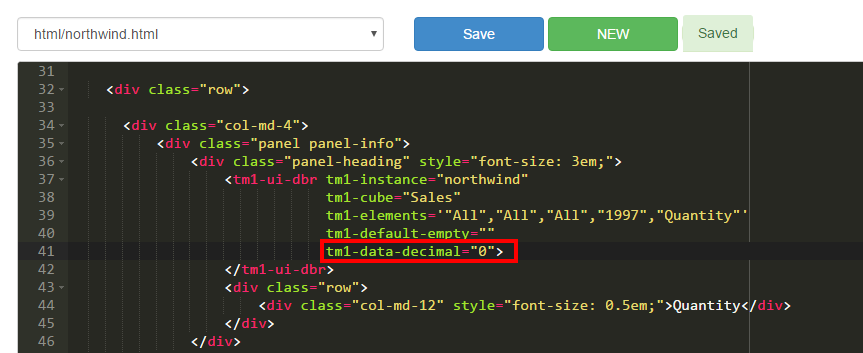


Refresh the dashboard:

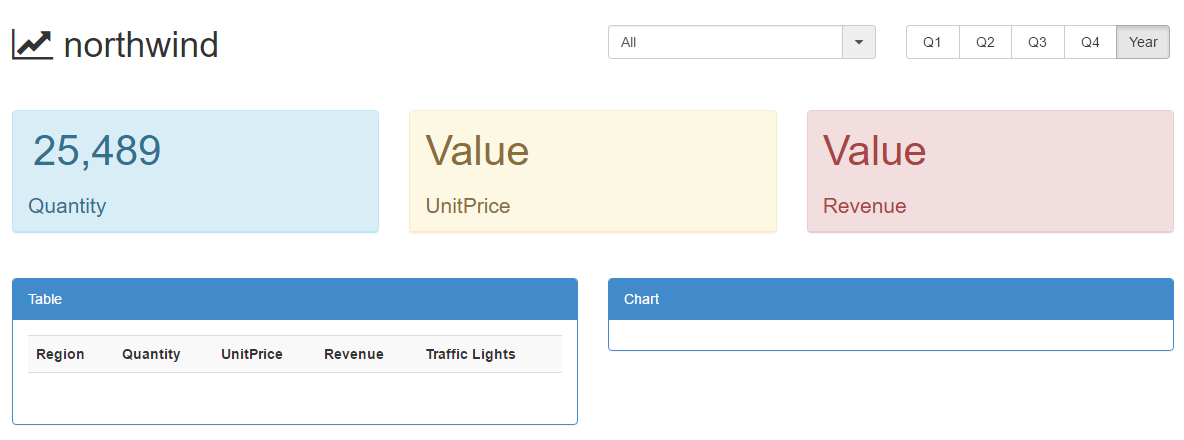


We can see now a real value instead of the first "Value". In order to add some formatting we can use the attribute tm1-data-decimal="0".

Go back to the Source Editor and add tm1-data-decimal="0"after the tm1-default-empty="":

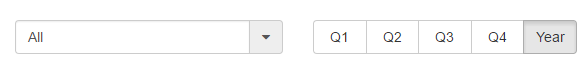


Let's save and refresh the page:

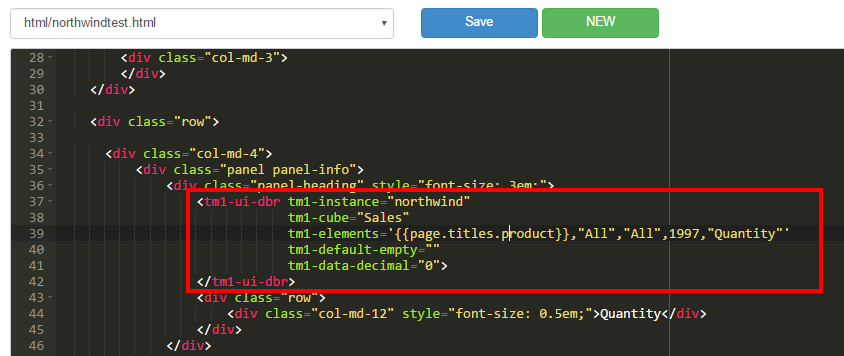


## Make the DBR dynamic

The elements in the DBR just created are static, what we want to do is to link our DBR to our product drop-down list and our quarters:



The way we do that is by replacing our hard coded product "All" by our variable we defined previously "page.titles.product" and we need to enclose the variable name by"{{" and "}}"**,**{{page.titles.product}}**:**



tm1-ui-dbr code:

<tm1-ui-dbr tm1-instance="northwind"

tm1-cube="Sales"

tm1-elements='{{page.titles.product}},"All","All","1997","Quantity"'

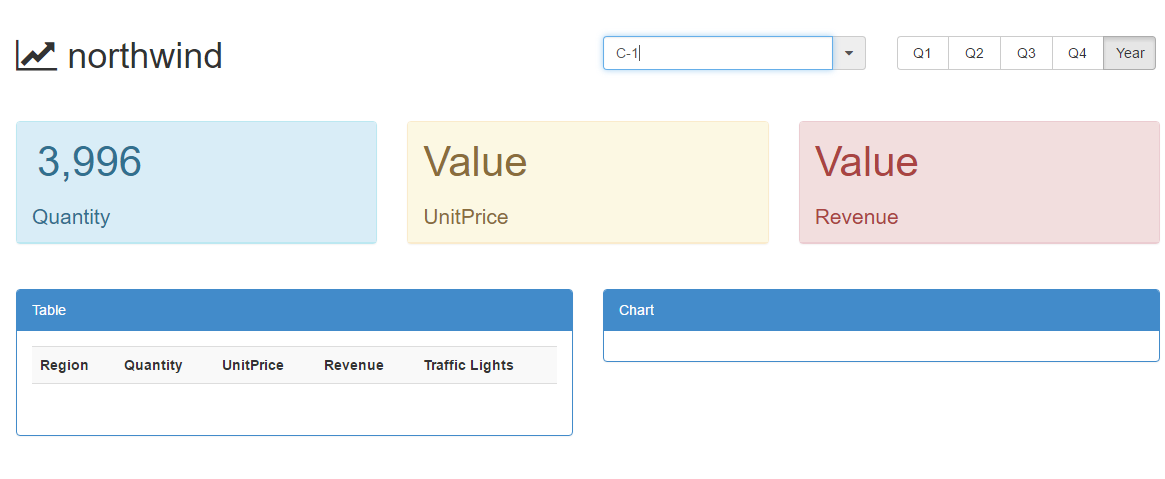
tm1-default-empty=""

tm1-data-decimal="0">

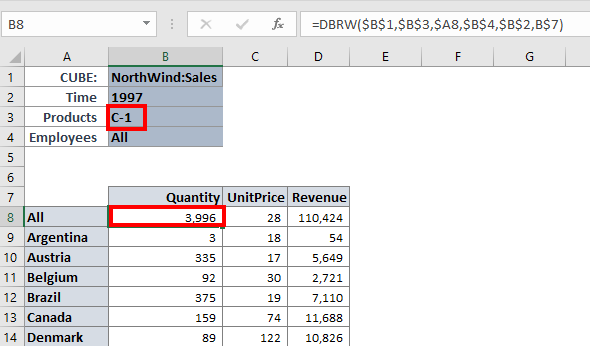
</tm1-ui-dbr>

The way this works is that every time the SUBNM at the top of the page changes, this value is then passed dynamically to theDBR.

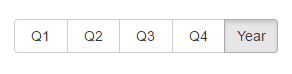
Now if we change the product we can see the value being updated:



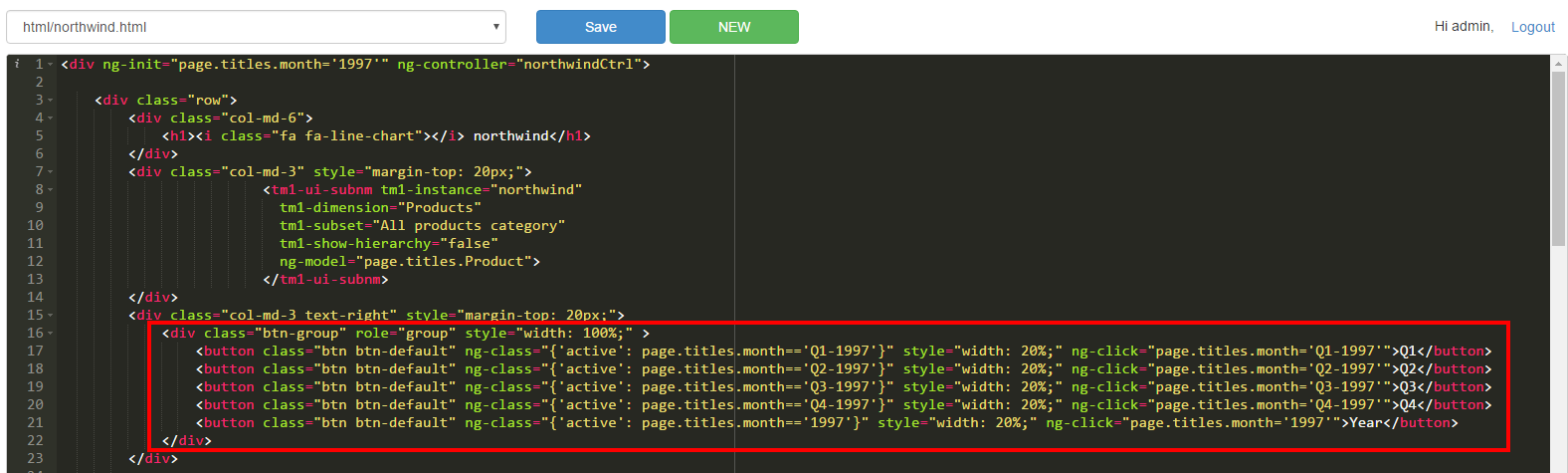
We can double check the value in TM1 match our value in Canvas:



The second step is to link our DBR to the quarter buttons defined at the top right:



The code which creates these buttons is:



In the code we can see that the variable used to store the value is page.titles.month**:**

 If it's Q1 the variable will be equal to Q1-1997**.**

 If Q2, it will beQ2-1997,

ForQ3 it will be Q3-1997,

Q4 it will be Q4-1997

and for Year it will be 1997.

Let's add this variable to our DBR, we need to replace the "1997" by {{page.titles.month}}:



The final code of our first tm1-ui-dbr is:

<tm1-ui-dbr tm1-instance="northwind"

tm1-cube="Sales"

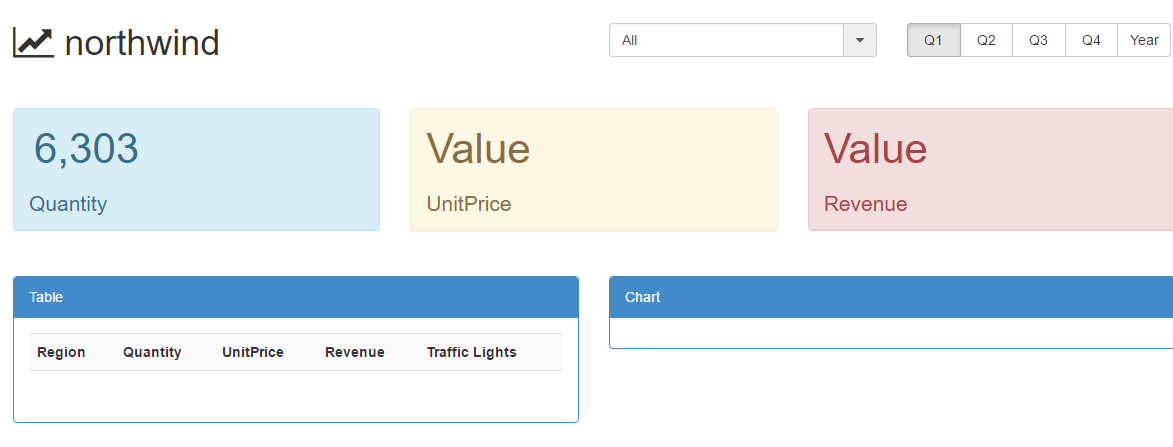
tm1-elements=' {{page.titles.product}},"All","All",{{page.titles.month}},"Quantity"'

tm1-default-empty=""

tm1-data-decimal="0">

</tm1-ui-dbr>

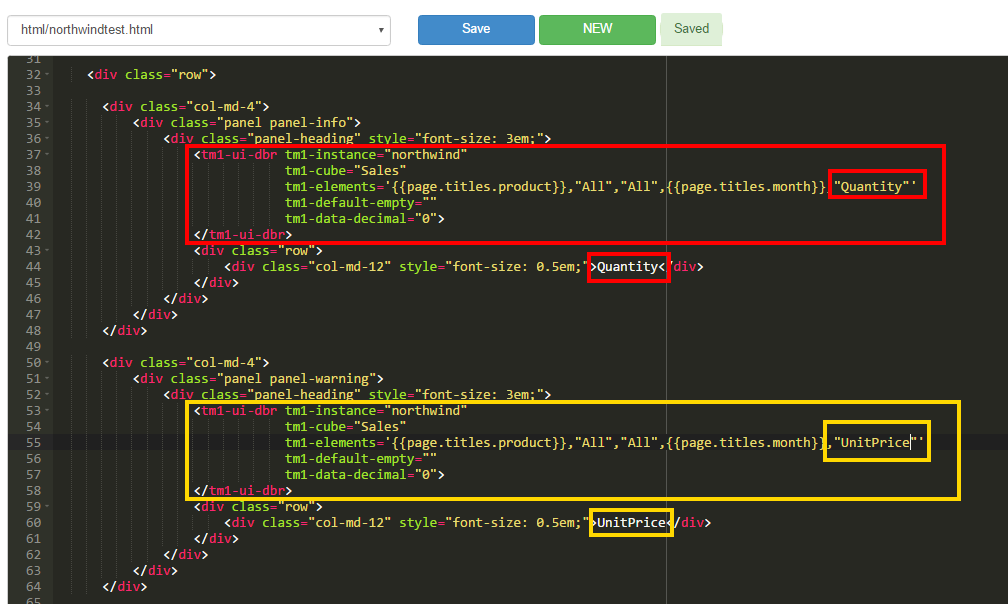
Let's save and refresh the dashboard. We can now see that if we click on Q2, the value will be updated:



## Update the 2 others "VALUE" fields

Now we need to update the others "Value" field, instead of generating from scratch the DBR we're going to replace the other "Value" with the DBR just created:

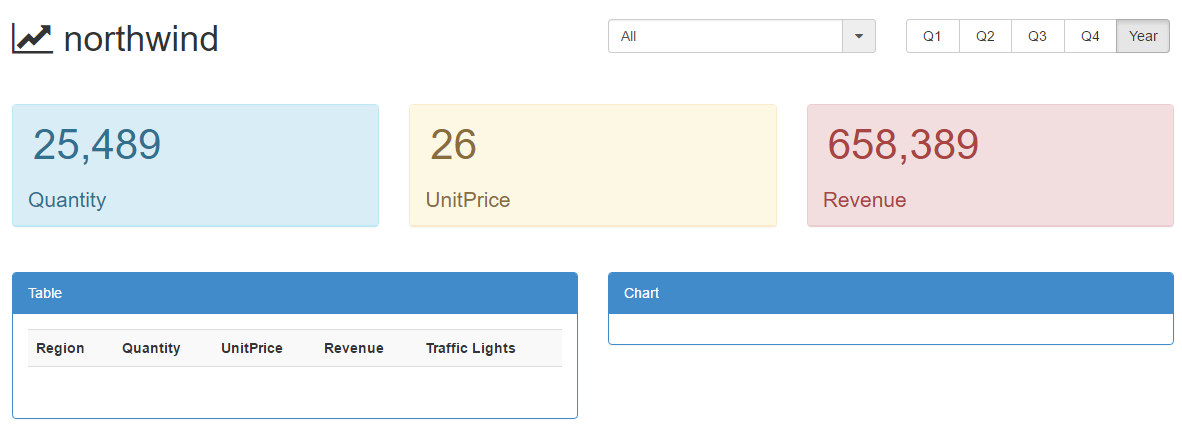


So just copy the first DBR and paste it on the second Value. Do not forget to replace the "UnitPrice" by the corresponding Measure: 

For the third Value we want to see the "Revenue" value:

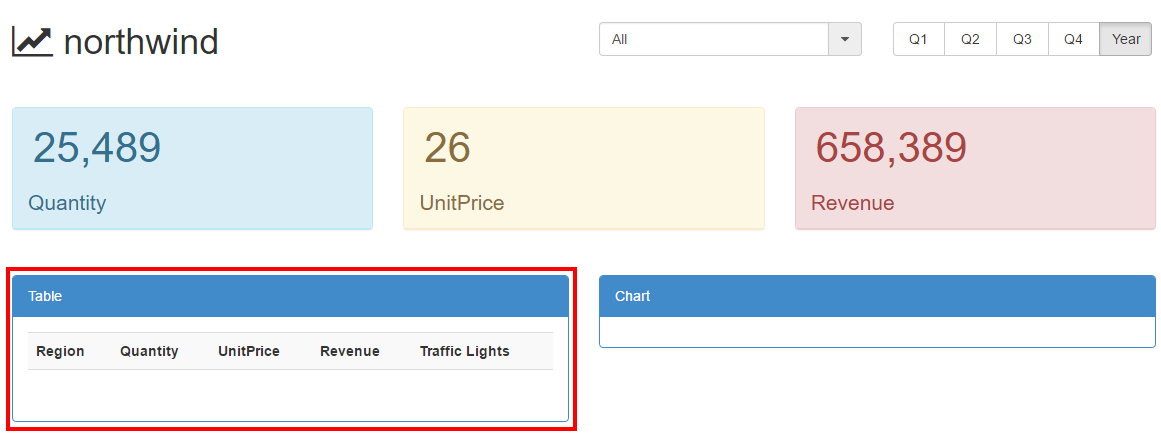


Save and refresh the dashboard, and now you can see that all 3 “Value” fields have been replaced with TM1 values:

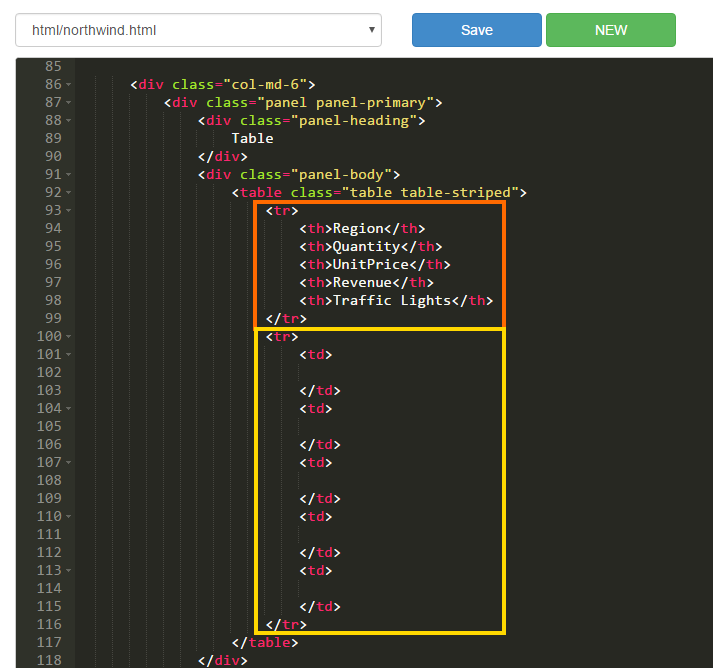


# Table

Now let's move to the table section:



Similar as an Active Form, we're going to define a list which will then drive our row set, the column headings will be Quantity, UnitPrice and Revenue.

We can see the Table section in the Source Editor:

HTML table basics:

* <tr></tr> define a row.
* <th></th> define table header
* <td></td> define a column

We can see above in Orange, the first row is our headers, and the second row in Yellow have empty columns.

## Create the rowset

The first thing we need to do is to create the set of Regions which will drive the rows. In this example we're going to use the tm1-ui-element-list:

<tm1-ui-element-list

tm1-instance=" "

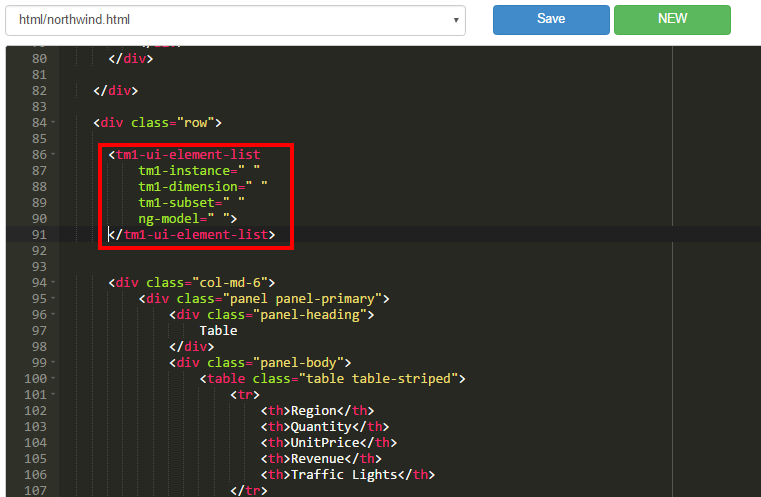
tm1-dimension=" "

tm1-subset=" "

ng-model=" ">

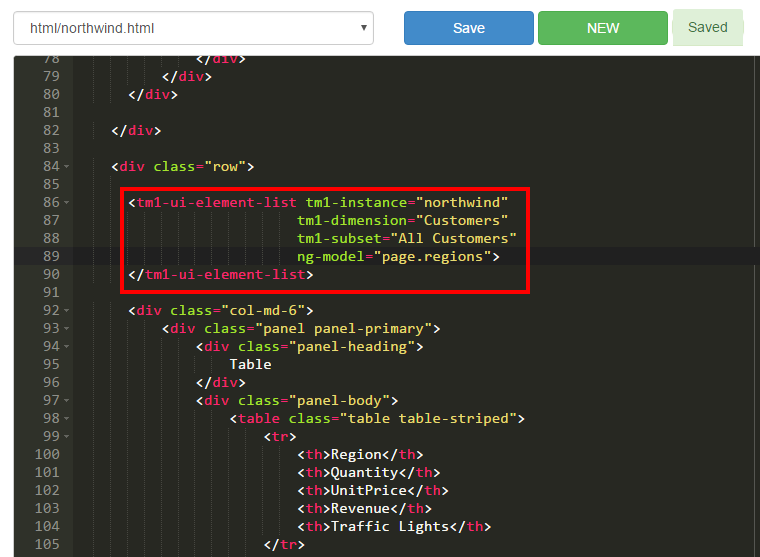
</tm1-ui-element-list>

Copy the code and paste it just below **<**div class="row">:



Let's update parameters:

* tm1-instance: northwind
* tm1-dimension: Customers
* tm1-subset: All Customers
* ng-model: page.regions



<tm1-ui-element-list

tm1-instance="northwind"

tm1-dimension="Customers"

tm1-subset="All Customers"

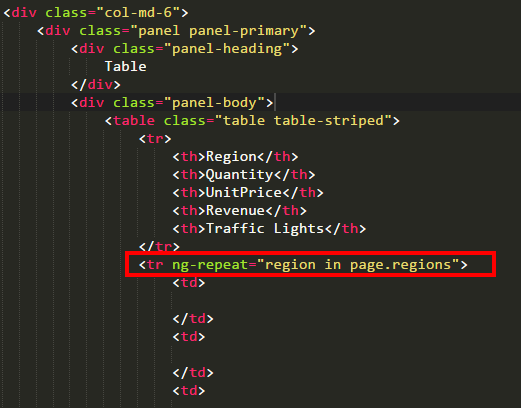
ng-model="page.regions">

</tm1-ui-element-list>

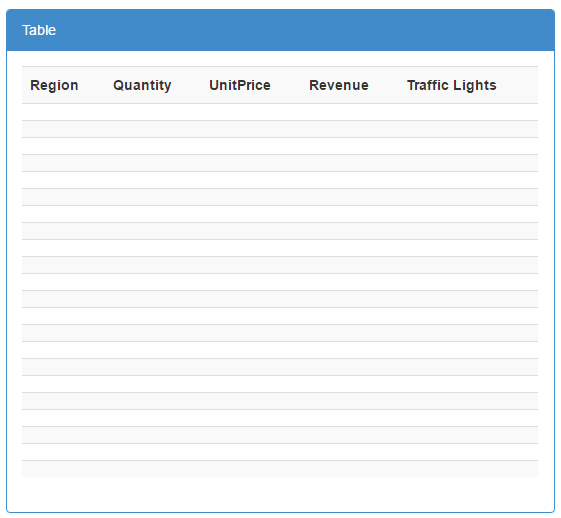
The tm1-ui-element-list will create a list of regions based of the All Customers subset and store the element list in the variable "page.regions".

We're going to use the Angular directives [ng-repeat](http://www.canvasfortm1.com/help-content/2016/7/8/how-to-use-ng-repeat-with-tm1) in order to repeat the row for every item stored in our list page.regions**.** ng-repeat is going to create one item "region" for every item in "page.regions".

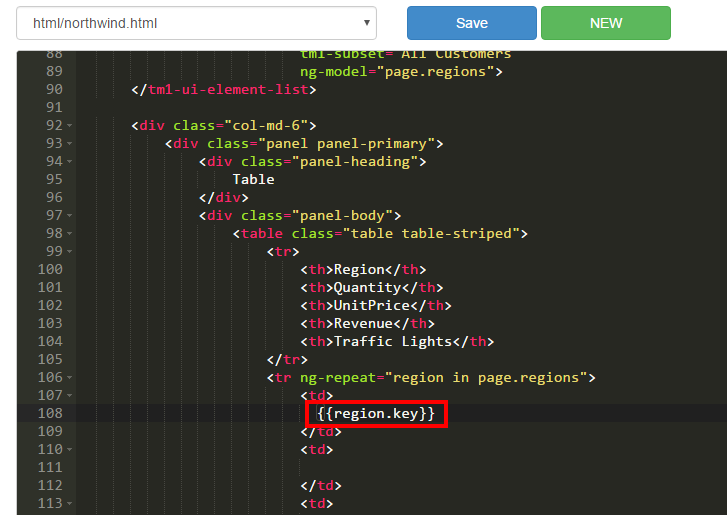
To do that just add the following code <tr ng-repeat="region in page.regions"> in the second <tr>:



Now if we refresh the dashboard, we can see that lots of rows have been created but they are all empty:



Now we need to populate the first column by the region alias, to do that we're using {{region.key}} in the first <td> section (which means first column), just add the following code:



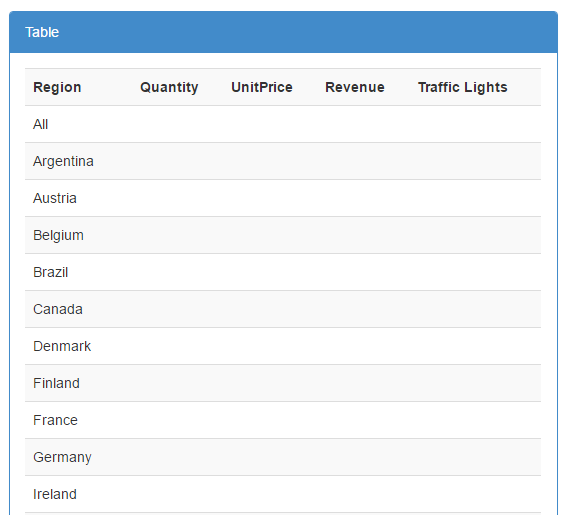
<tr ng-repeat="region in page.regions">

<td>

{{region.key}}

</td>

Just save and refresh the dashboard:



We can see the Regions have populated in the first column, now we need to populate the table with values. ng-repeat is similar as an Active Form, you just need to populate the first row and then it will loop down the page and duplicate the rows.

## Populate the table with values

Let's grab our first DBR which looks at Quantity**:**

<tm1-ui-dbr tm1-instance="northwind"

tm1-cube="Sales"

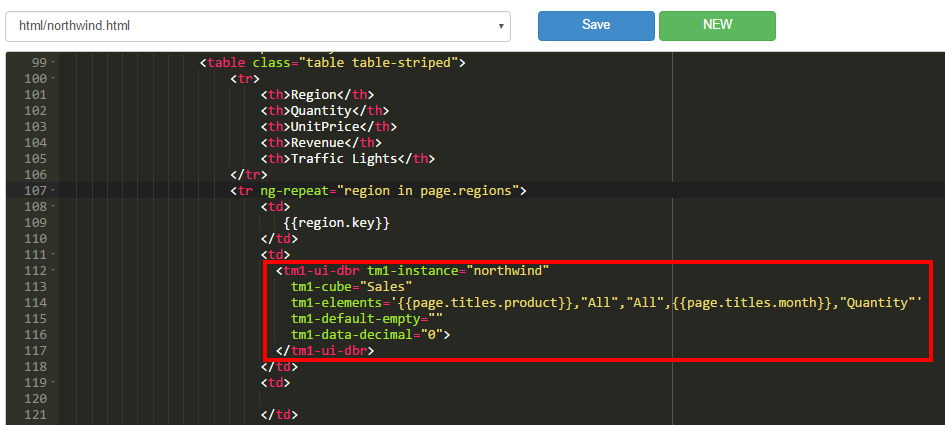
tm1-elements=' {{page.titles.product}},"All","All",{{page.titles.month}},"Quantity"'

tm1-default-empty=""

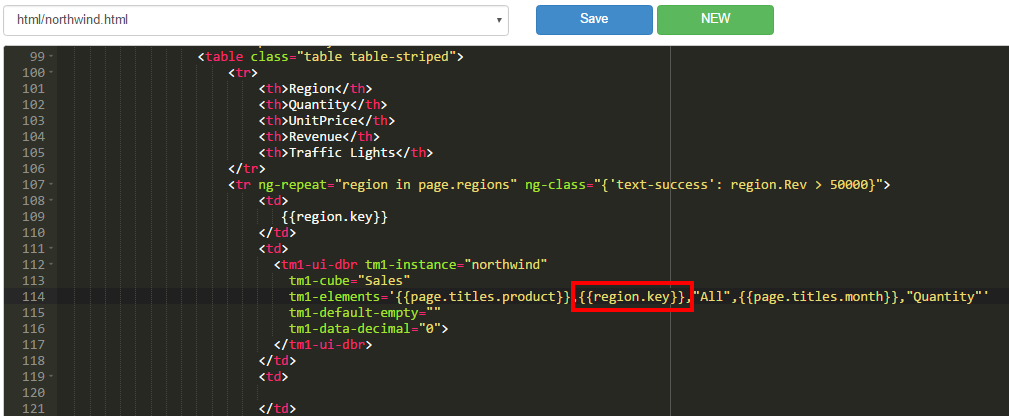
tm1-data-decimal="0">

</tm1-ui-dbr>

and paste it in the second column of our table (second <td>):



Let's replace the"All" Customer with our region item. {{region.key}}will retrieve the principal name of our item:



<tm1-ui-dbr tm1-instance="northwind"

tm1-cube="Sales"

tm1-elements='{{page.titles.product}},{{region.key}},"All",{{page.titles.month}},"Quantity"'

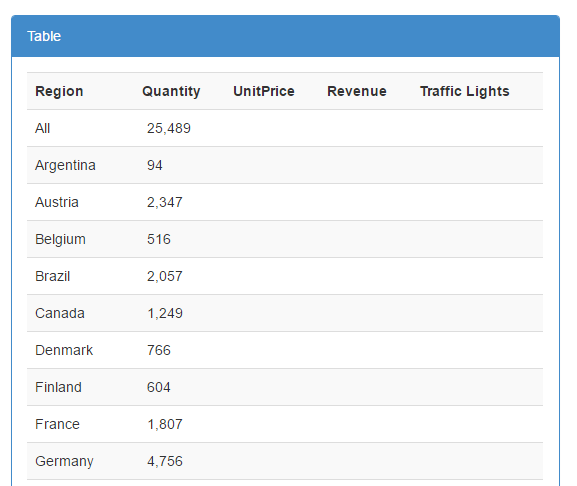
tm1-default-empty=""

tm1-data-decimal="0">

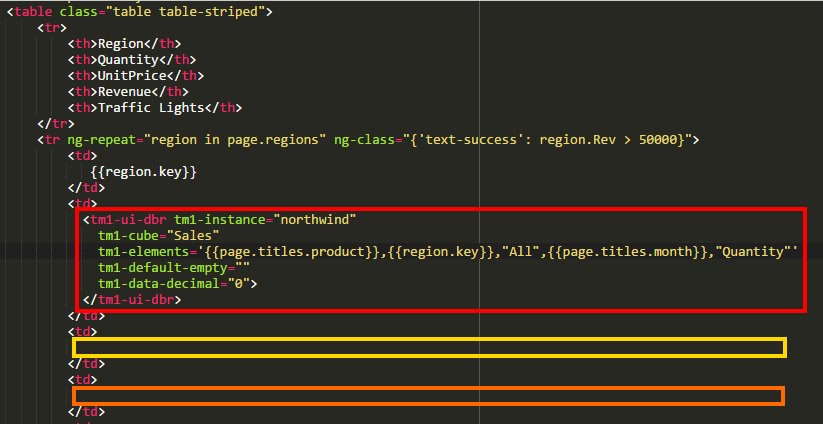
</tm1-ui-dbr>

For every region in our list, it's going to populate a row with in the first column the region and then the DBR for this region.

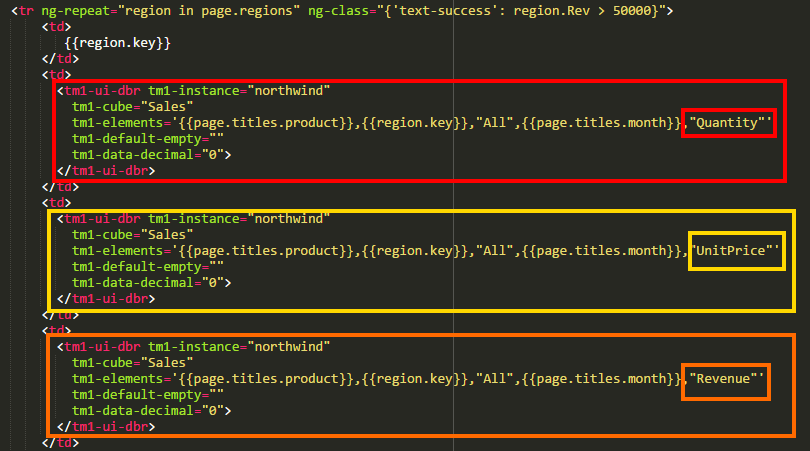
Save and refresh the dashboard:



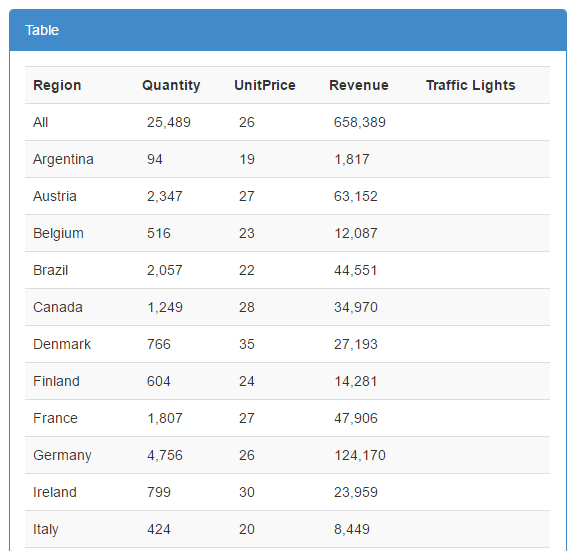
We can now do the same for all the other columns:



Just copy the DBR in the next 2 columns sections (<td>) and do not forget to update the measures, the 2nd column is UnitPrice and the 3rd is Revenue:



Let’s save and refresh the dashboard to see that we have now 4 columns populated with values.

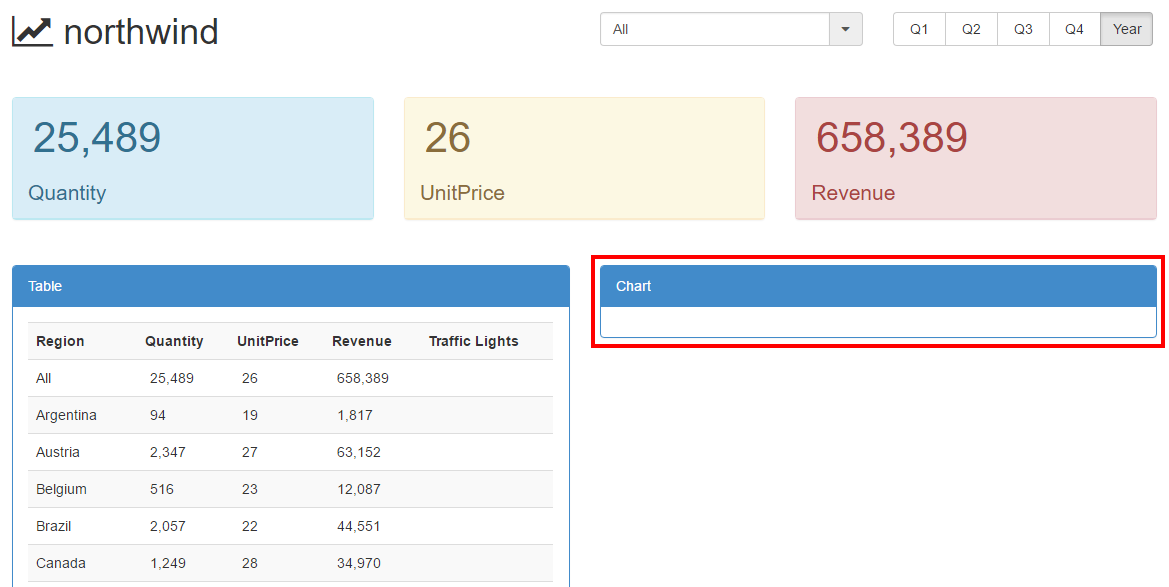


Let’s keep theTraffic Lights column for later and move on to the chart.

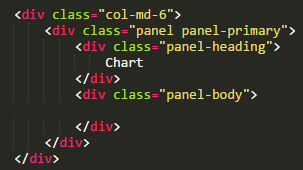
# Chart

Canvas comes with a library of commonly used charts and graphs, which are ready to use including Pie-chart, bar-chart, Donut, Waterfall... All charts and graphs in Canvas are interactive and resize dynamically based on your web browser.

During this training we’re going to build a bar chart.



The code area where we want to build the chart is:



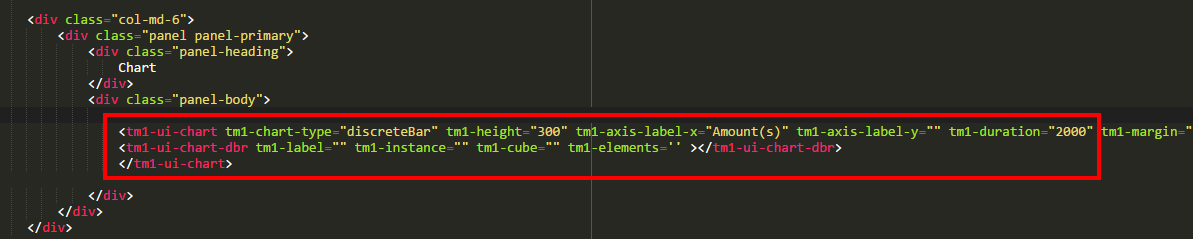
Copy the HTML code to create a discrete bar chart:

<tm1-ui-chart tm1-chart-type="discreteBar" tm1-height="300" tm1-axis-label-x="Amount(s)" tm1-axis-label-y="" tm1-duration="2000" tm1-margin="{top:10, right: 10, bottom:20, left:10}">

<tm1-ui-chart-dbr tm1-label="" tm1-instance="" tm1-cube="" tm1-elements='' ></tm1-ui-chart-dbr>

</tm1-ui-chart>

And past it in the panel-body area:



Now let’s update the tm1-ui-chart attributes:

* tm1-cube: is Sales
* tm1-label: is Quantity
* tm1-elements: '{{page.titles.product}},"All","All",{{page.titles.month}},"Quantity"':

tm1-elements value is the same list as our first tm1-ui-dbr:



The tm1-ui-chart-dbr will be:

<tm1-ui-chart-dbr

tm1-label="Quantity"

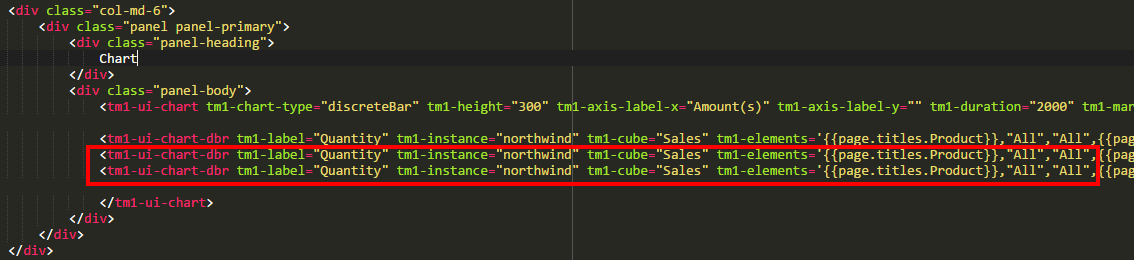
tm1-instance="northwind"

tm1-cube="Sales"

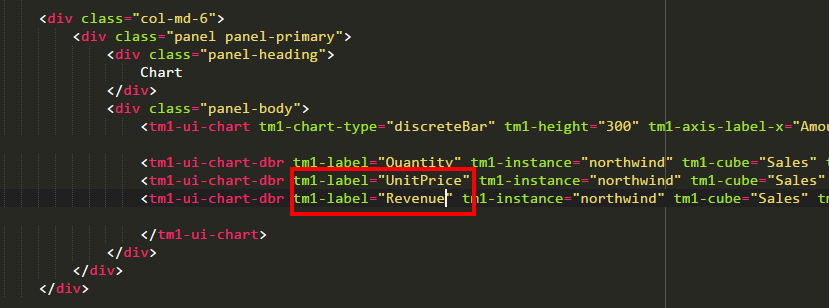
tm1-elements= '{{page.titles.product}},"All","All",{{page.titles.month}},"Quantity"' >

</tm1-ui-chart-dbr>

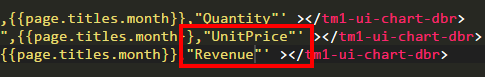
Let's copy 3 more times this tm1-ui-chart-dbr**:**



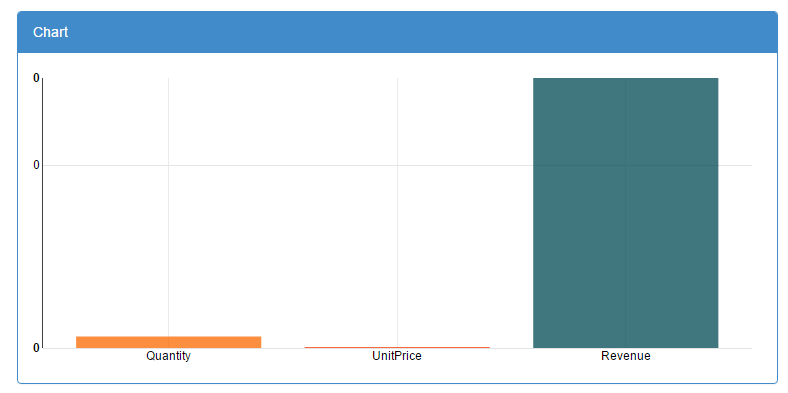
and replace the Quantity to UnitPrice for the second one and Revenue for the third one. Do not forget to replace the Quantity in tm1-label**:**



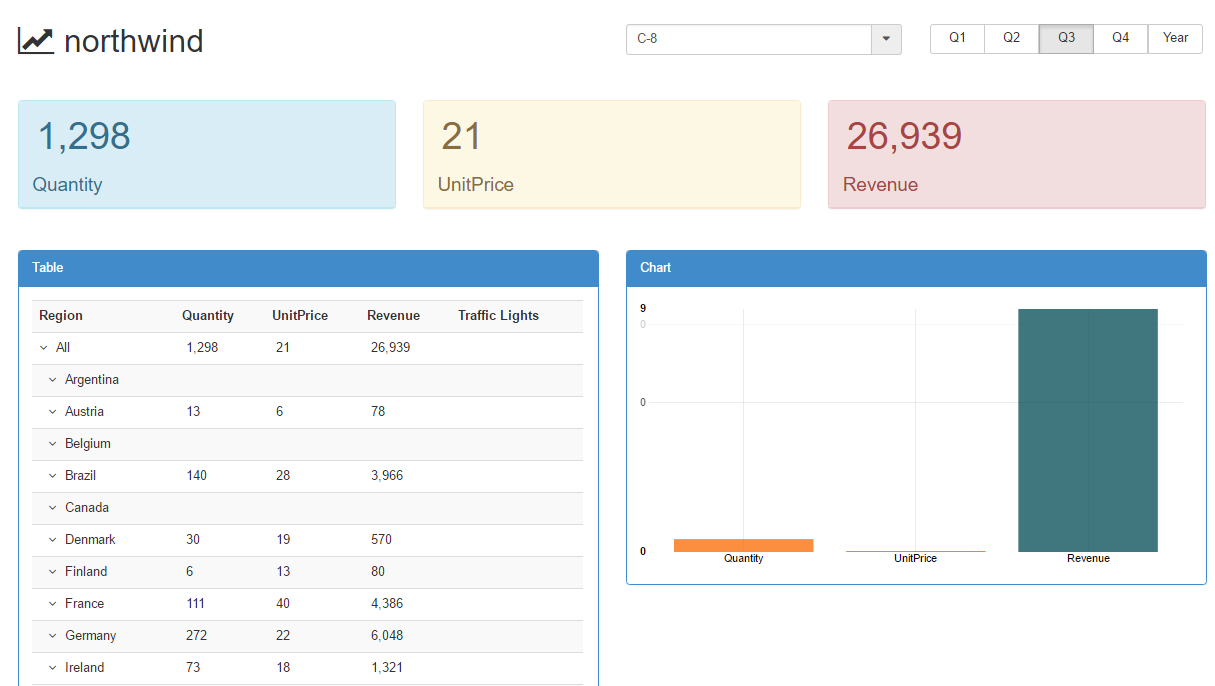
 and inside tm1-elements:



Let's save and refresh the dashboard:



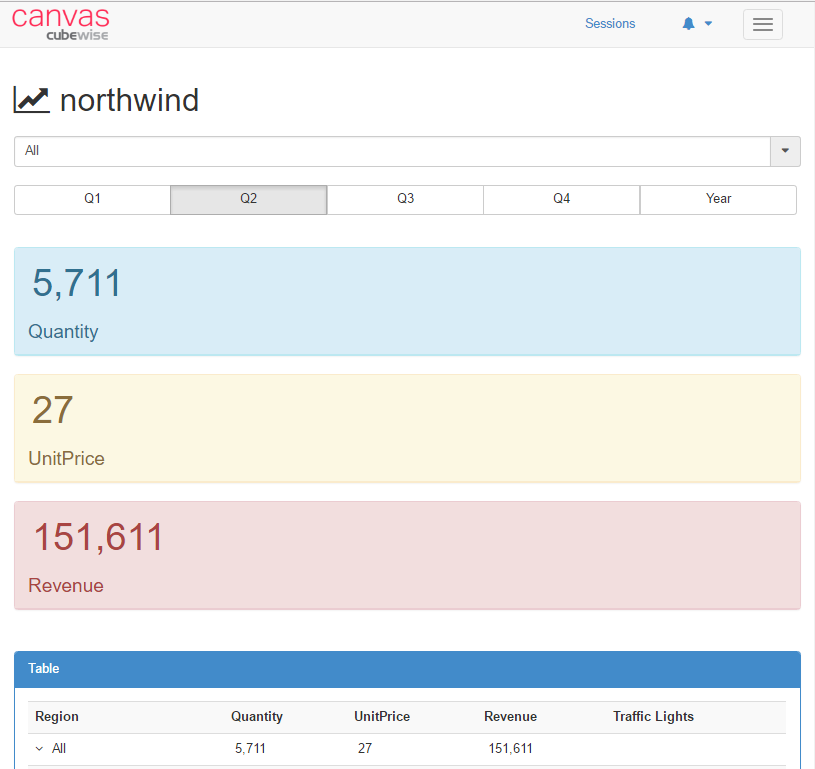
Update the parameters to see the dashboard for Q3 and Product Classification C-8:



You can notice that the dashboard is dynamic. You do not need to click refresh after updating a parameter.

Furthermore … the web technology used in Canvas is so smart it will manage the look and feel of your TM1 applications as you resize your web browser or switch from a tablet to a smart phone.

You can try to resize your window as below:



This is the end of the first part, now we’re going to add advanced dashboard features:

* Table indentation
* Create Traffic Lights
* Conditional formatting

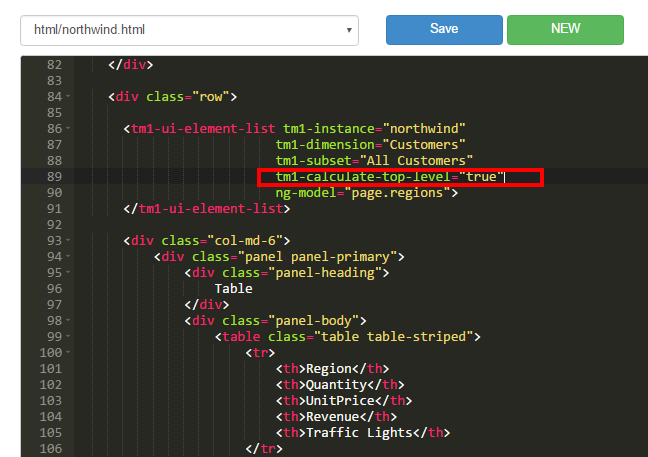
# Advance dashboard

## Indentation

Let's go back to our table:



In order to create an indentation in the Region list, the first step is to add tm1-calculate-top-level="true":



<tm1-ui-element-list

tm1-instance="northwind"

tm1-dimension="Region"

tm1-subset="All Customers"

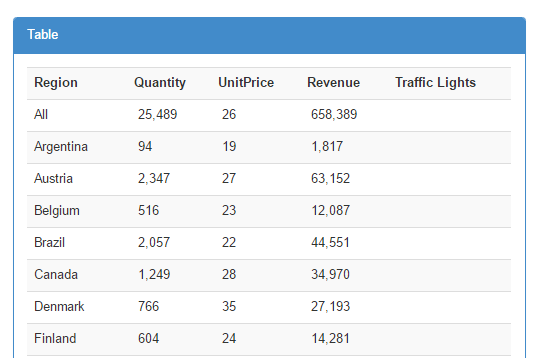
tm1-calculate-top-level="true"

ng-model="page.regions">

</tm1-ui-element-list>

The purpose of this attribute is to calculate the level of each element.

If you save and refresh the dashboard, you will not see the indentation:



tm1-calculate-top-level only calculates the levels of elements. If we want to show the indentation we have to replace{{region.key}}:

****

by using the tm1-ui-element-list-item code below:

<tm1-ui-element-list-item

tm1-item="Customers"

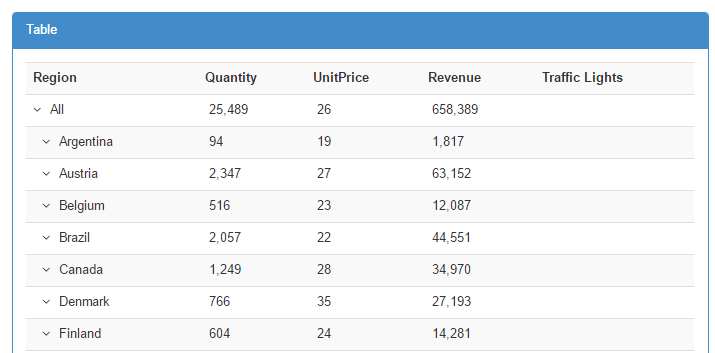
tm1-item-display="{{region.ke}}">

</tm1-ui-element-list-item>



This component is going to use the region item from the list page.regions, it will loop over the list, find out the top level, display the indentation and display the region element.

Let's save and refresh our dashboard and we can see now that we have indentation running down the page:



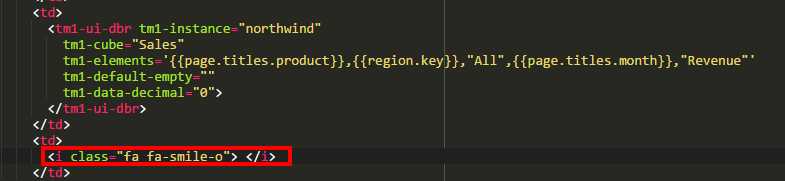
## Traffic lights

In the last column of the table we're going to display different icons.

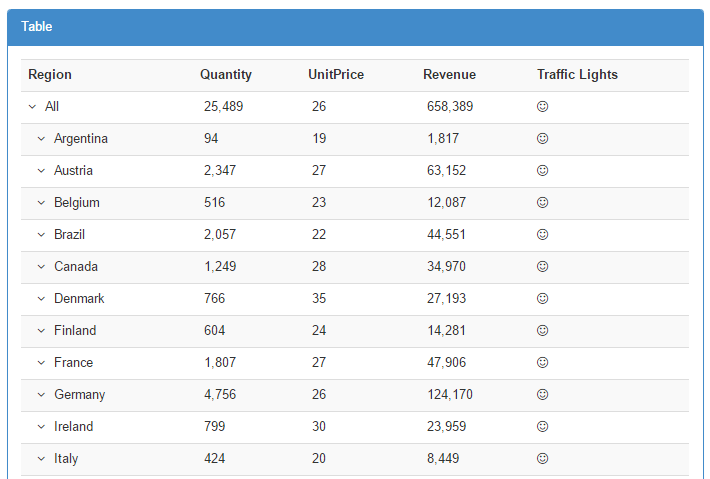
We can use a special element called <iwhich will enable us to show an icon inside the table. You can pick any icons you like from the [Font Awesome](http://fontawesome.io/icons/) library.

The good thing with this library is that all icons are font, regardless of the size attributes, they will look perfect. Let's pick for this example the smile face [fa-smile-o](http://fontawesome.io/icon/smile-o/).

To insert our icon, we need to use "class". A class is a way of formatting a particular HTML element. In the last column (<td>) of our table let's add the following code <i class="fa fa-smile-o"> </i>:



Let's save and refresh our dashboard:



Now we can see the smile faces throughout the column Traffic Lights. There is not much use of it if we showing the icon on every row. So let's display the smile face only when a specific value is reached.

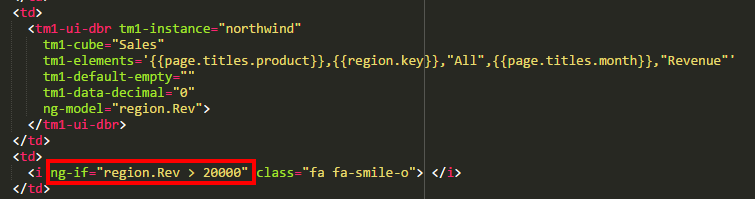
Let's say that we want to show a smile face only if Revenue is over 20,000.

The first thing we need to do is to store the Revenue value into a variable. We can use the optional parameter ng-model to store the value of a DBR we want to use in our calculation. Let's add ng-model="region.Rev" to the DBR in the table which gets the Revenue:



Now we can use the variable region.Rev as a condition of the [ng-if](http://www.w3schools.com/angular/ng_ng-if.asp).

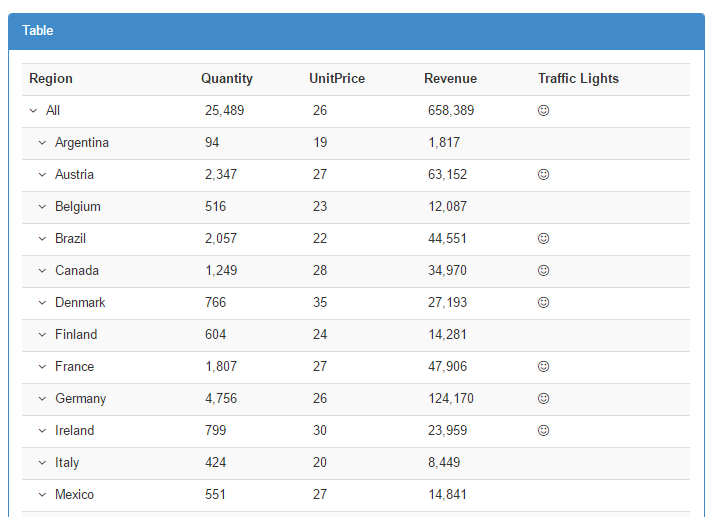
Let's add ng-if="region.Rev > 20000"just before the class="fa fa-smile-o" we just created above:



<i ng-if="region.Rev > 20000" class="fa fa-smile-o"> </i>

The ng-if directives will display the element only if the expression evaluates to true.

Let's save and refresh our dashboard, new we have a smile face only if Revenue > 20,000:



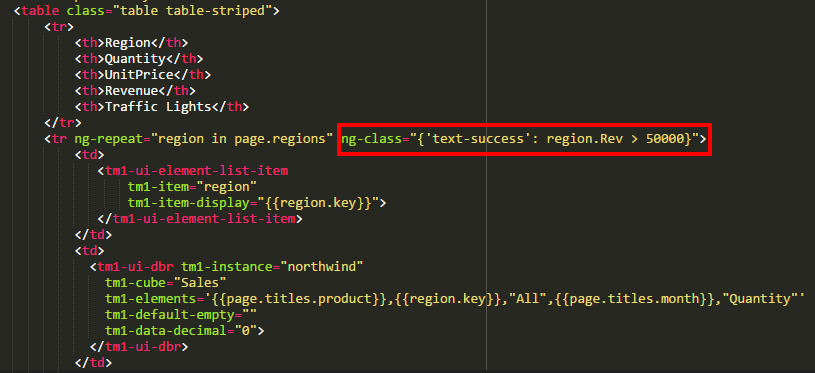
## Conditional Formatting

To take that one step further, we would like now to add some conditional formatting. We can use a [bootstrap class](http://www.w3schools.com/bootstrap/bootstrap_ref_css_helpers.asp) such as text-success to help us to quickly add some colors in your table.

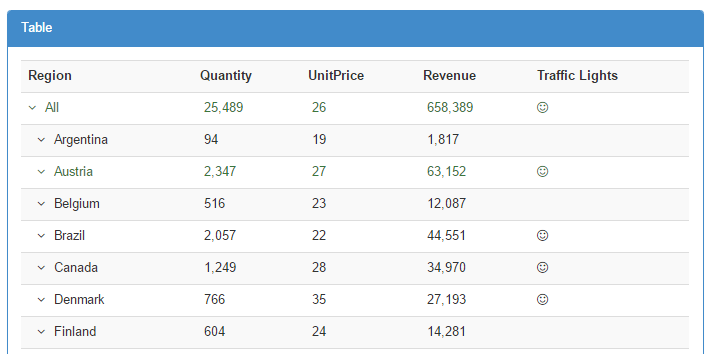
We're adding a condition to show the color only if Revenue > 50,000 (region.Rev > 50000):

<tr ng-repeat="region in page.regions" ng-class="{'text-success': region.Rev > 50000}">

We’re adding the condition inside the ng-repeat so that the conditional formatting will be applied to all rows:

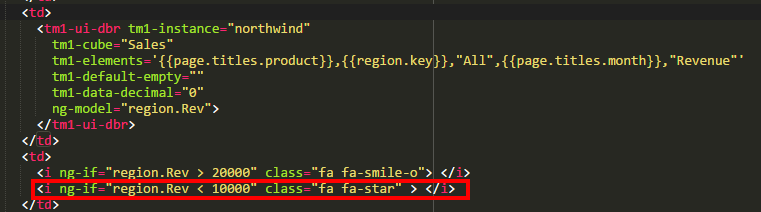


Let's save and refresh the dashboard, now we can see the row in green are the ones with Revenue > 50,000:



Let's add some more formatting. We're going add a new <i element showing the[fa-star](http://fontawesome.io/icon/star/)icon only if Revenue < 10,000.

Just add below the <i ng-if="region.Rev > 20000" class="fa fa-smile-o"> </i>  
 the new star icon <i ng-if="region.Rev < 10000" class="fa fa-star"> </i>



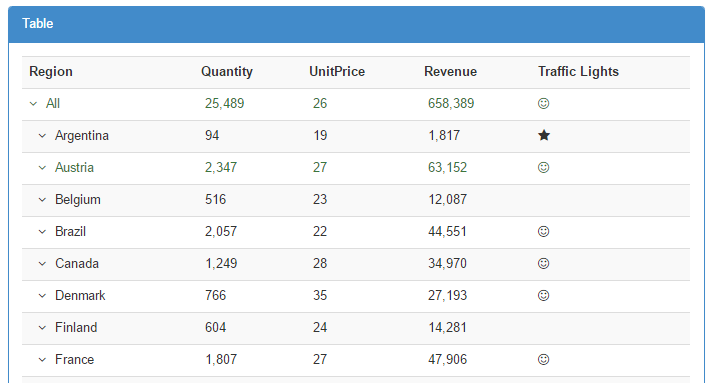
<td>

<i ng-if="region.Rev > 20000" class="fa fa-smile-o"> </i>

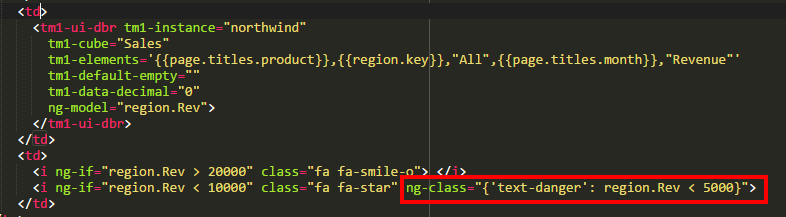
<i ng-if="region.Rev < 10000" class="fa fa-star"> </i>

</td>

Let's save and refresh the page, we can see now the star if Operating Expenses < 400,000:

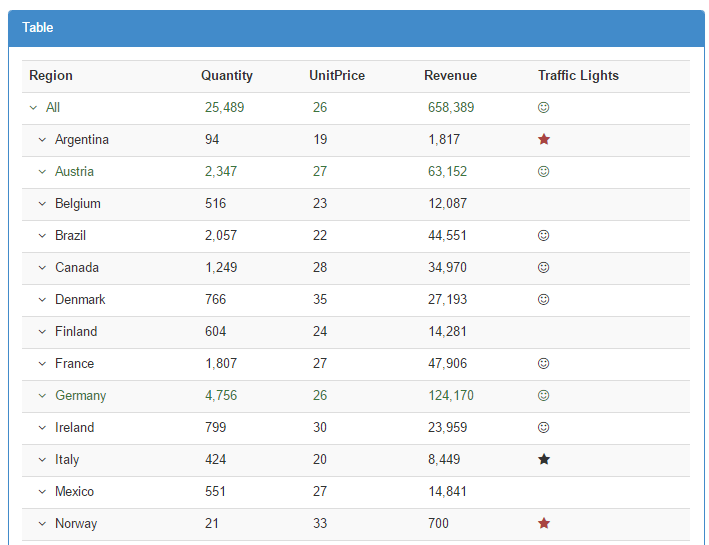


Let's put some colors on the star only if Revenue < 5,000 (ng-class="{'text-danger': region.OE < 200000}"):

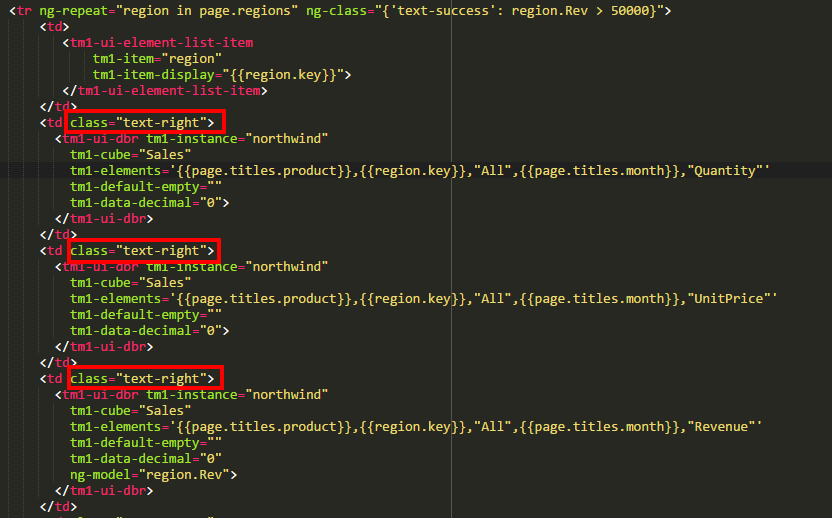


<i ng-if="region.Rev < 10000" class="fa fa-star" ng-class="{'text-danger': region.Rev < 5000}">

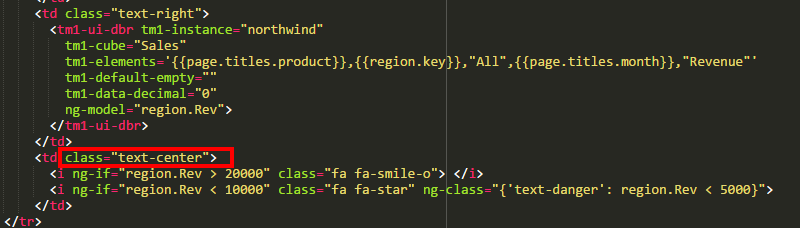
Let's save and refresh the dashboard, you should see a green star if Operating Expenses < 200,000:



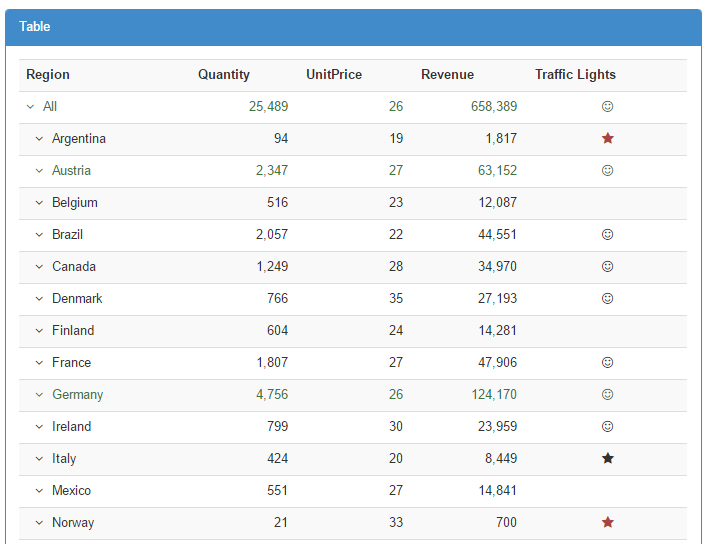
The last thing we want to do is to align all numbers to the right and the traffic lights to the middle of the column. In order to do that we're going to use another bootstrap item. In all <td>which include the DBR we're going to add class="text-right":



and for the traffic lights we're going to addclass="text-center":



Let's save and refresh the dashboard:



That's the end of this training to learn more about Canvas, you can go to <http://www.canvasfortm1.com/>