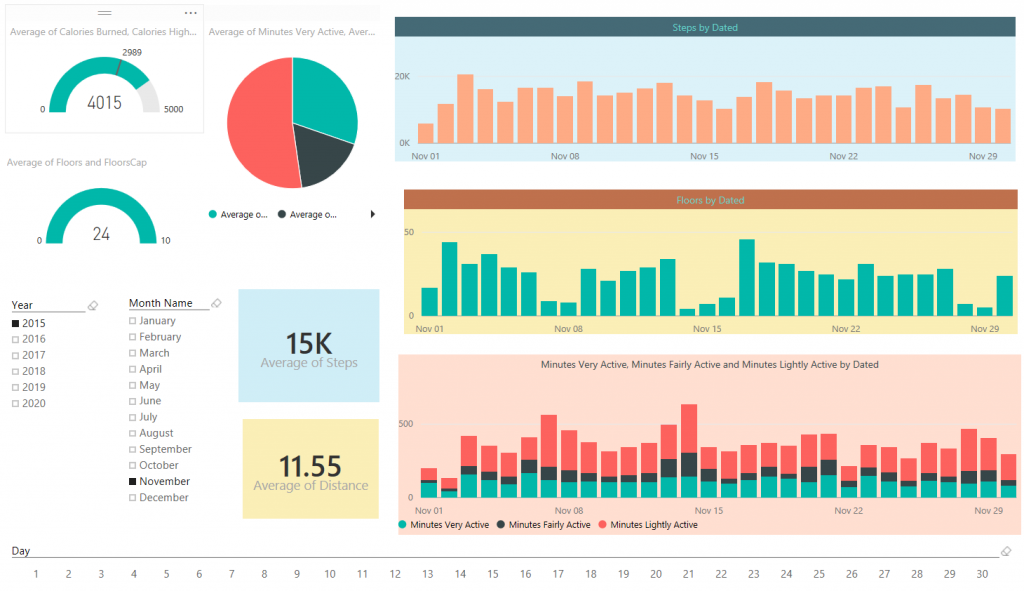
# Be Fitbit BI Developer in Few Steps: Step 3 Visualization

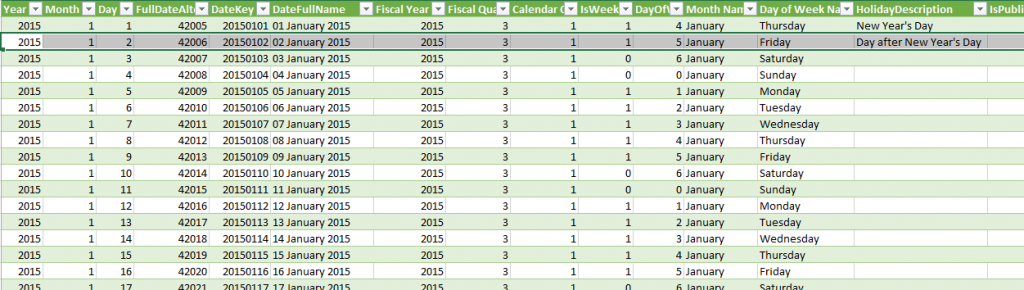
Published Date : December 3, 2015

[](http://www.radacad.com/wp-content/uploads/2015/12/131.png)

In previous steps you’ve learned [how to extract and mash up the data from CSV](http://www.radacad.com/be-fitbit-bi-developer-in-few-steps-first-step-get-data-from-csv) and [automate the process of looping through all CSV files in the source folder](http://www.radacad.com/be-fitbit-bi-developer-in-few-steps-step-2-loop-through-all-csv-files). In this step we are going to build visualizations based on the data set we’ve built so far. In this post you will see how easy is to build charts in Power BI desktop and how customize-able they are. You will also learn about using few DAX calculations to produce new measures that helps building visualization elements.

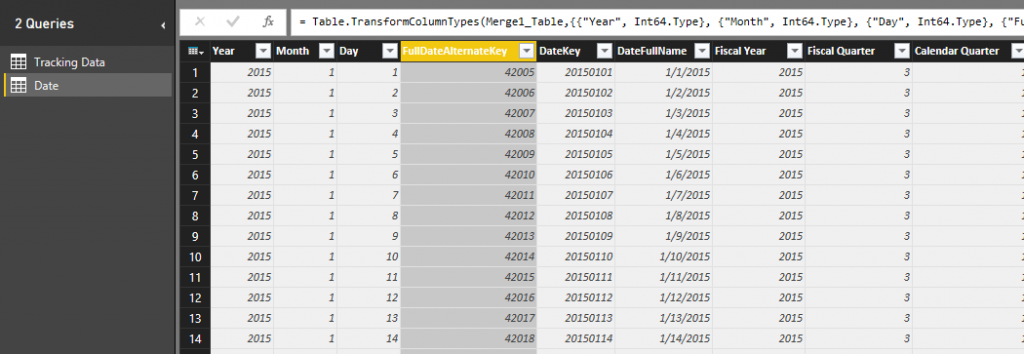
# Adding Date Dimension

Before going to visualization part let’s add a date dimension to our queries. I have previously explained an example of [date dimension with Power Query](http://www.radacad.com/script-to-generate-date-dimension-with-power-query-m-with-financial-columns), so All you need to do is to use that as the source.  The Date dimension query produces table below as result:

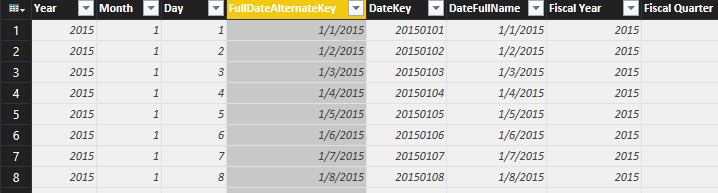
[](http://www.radacad.com/wp-content/uploads/2015/12/111.png)

It is also configurable as it mentioned in the [blog post](http://www.radacad.com/script-to-generate-date-dimension-with-power-query-m-with-financial-columns), and it includes public holidays (New Zealand holidays actually, you have to change it a bit to cover your country holidays which is easy).

I have above query in an Excel file and I use that as a source in Power BI Desktop. So here is what I have in my Query Editor; A query for Date (which comes from date dimension query example explained above), and the Tracking Data query (which we have built it in previous step of this exercise);

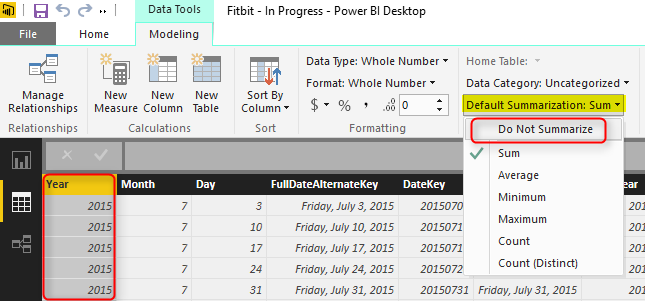
[](http://www.radacad.com/wp-content/uploads/2015/12/25.png)

As you see in the above table the FullDateAlternateKey is not showing the right date format. I just right click on it and change it to Date format so I have proper date field there;

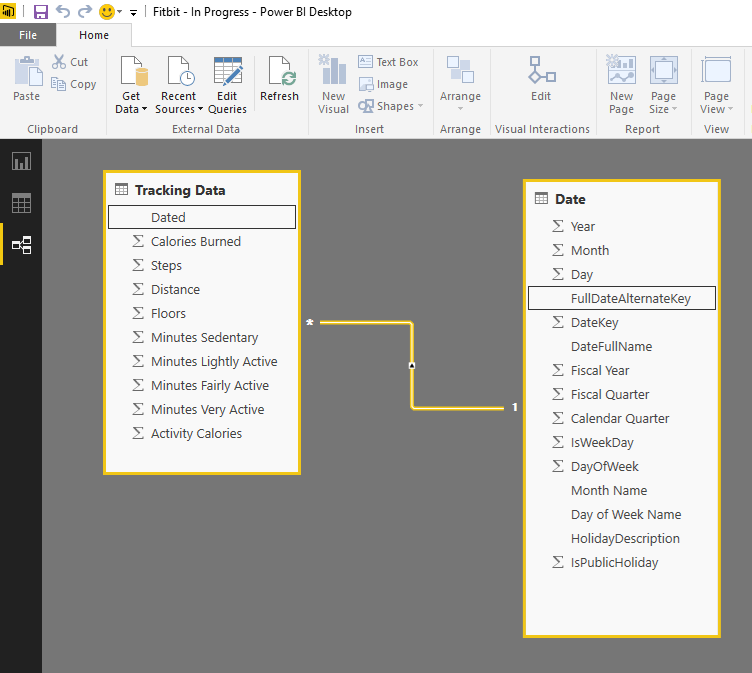
[](http://www.radacad.com/wp-content/uploads/2015/12/32.png)

# A Touch to Modelling

Now Close the query editor and apply. This will load result of both queries into the memory and model for Power BI Power Pivot component. Go to Data tab, and expand the Date under fields section. You will see that many columns have a sigma icon beside them, the reason is that Power BI desktop based on their data type (whole number or decimal) decided that these are values that can be summarized. You can change this behavior by choosing any of those columns and change default stigmatization to Do Not Summarize.

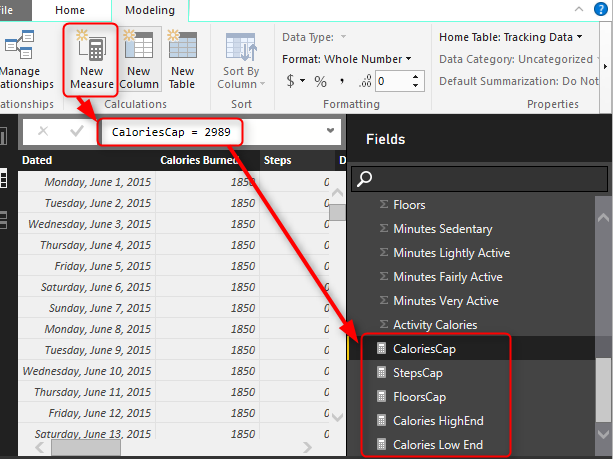
[](http://www.radacad.com/wp-content/uploads/2015/12/42.png)

Now go to Relationship tab and create relationship between two tables based on their full date columns which are; FullDateAlternateKey Column from Date, and Dated Column in Tracking Data table.

[](http://www.radacad.com/wp-content/uploads/2015/12/52.png)

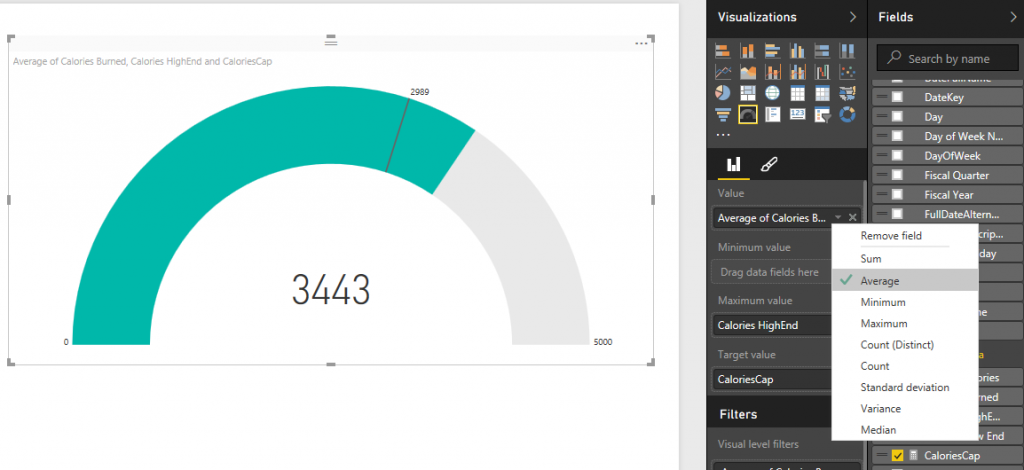
## Adding Few Static Measures

Fitbit calculates based on my current weight and age (I assume) how much calories I have to spend each day. I don’t know that calculation, So I create a static measure with the value of 2989 for the amount of calories I have to spend each day. I also create StepsCap measure with 12000 value showing that I have to walk 12000 steps a day, and another one for FloorCap with the value of 10. I created a Calories HighEnd measure with 5000 calories as value (I will die if I burn more than that!). You can create all these measures easily in Data tab.

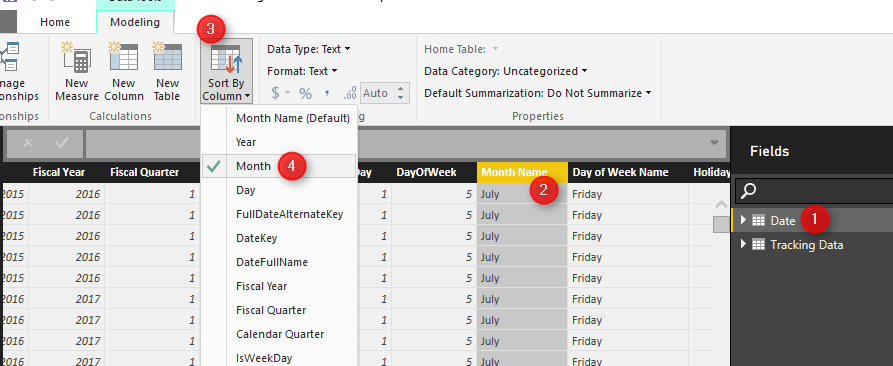
[](http://www.radacad.com/wp-content/uploads/2015/12/61.png)

# Let’s Visualize

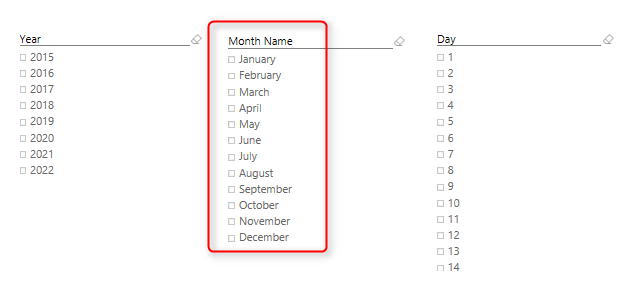
It’s time to visualize what we have so far in the data set. Go to Report tab, and start with building a Gauge. Then drag and drop CaloriesCap as Target Value. You can also put Calories HighEnd measure as Maximum. and then drag and drop Calories measure into the value. change the aggregation from sum to Average as it showed in below image.

[](http://www.radacad.com/wp-content/uploads/2015/12/71.png)

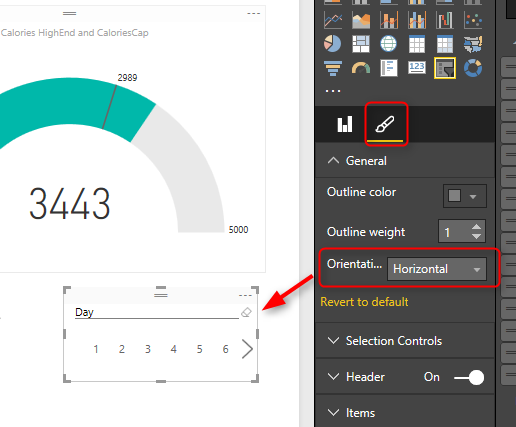
Easy! and lucky me who spent (in average) more than the calories I have to spend each day. Do the same for Floor with FloorCap and Floors measure. Now Add Three Slicers for; Year, Month Name, and Day. When you create slicer for Month Name you will see month names are ordered by their alphabetic order of names which is not right. Go to data tab, and click on Month Name Column of Date table. Then from the menu options of Modeling choose Sort By Column and then choose Month (which is the number of month in year);

[](http://www.radacad.com/wp-content/uploads/2015/12/81.png)

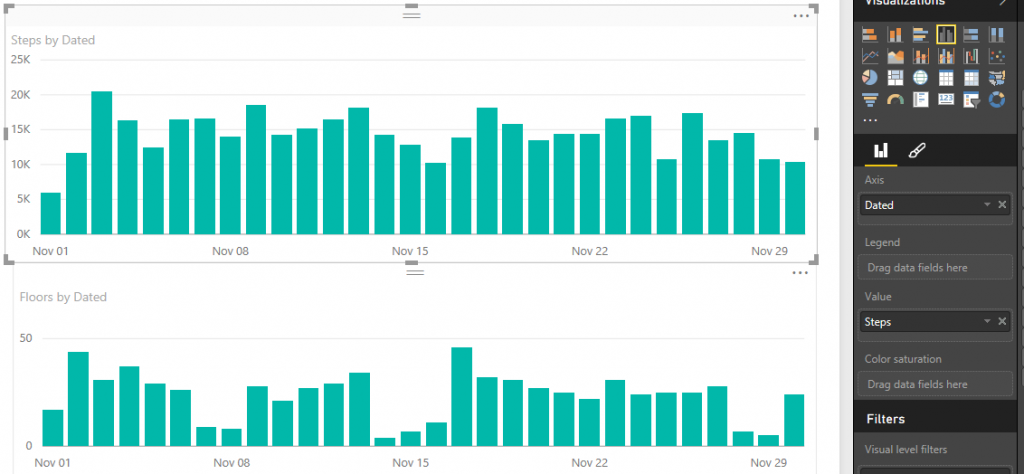
Now that you’ve set ordering you will be able to see Month names in the correct order in the slicer as below;

[](http://www.radacad.com/wp-content/uploads/2015/12/91.png)

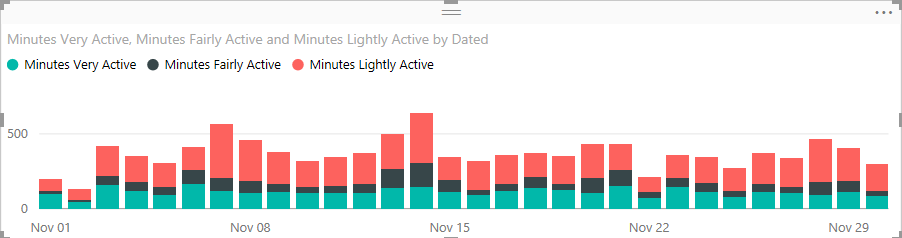
Change the orientation of Day slicer to be horizontal rather than vertical in the formatting option of the slicer

[](http://www.radacad.com/wp-content/uploads/2015/12/101.png)

Now let’s add two column charts; one for Steps as Value, and Dated column (from Tracking Data table) as axis. The other one with same axis, and Floors as value

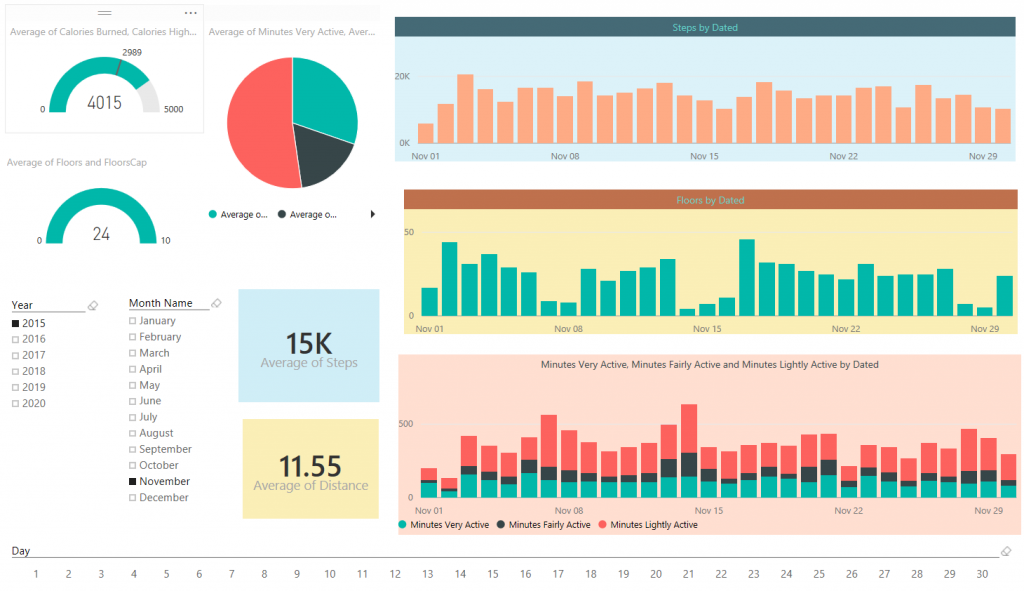
[](http://www.radacad.com/wp-content/uploads/2015/12/112.png)

Add a stacked column chart with Dated as axis, and three measures as value: Minutes Very Active, Minutes Fairly Active, and Minutes Lightly Active.

[](http://www.radacad.com/wp-content/uploads/2015/12/121.png)

Add Also an average of all of three measure above in a pie chart to show the total average. and Add two Card visualization one for Average of steps, and another for average of distance.

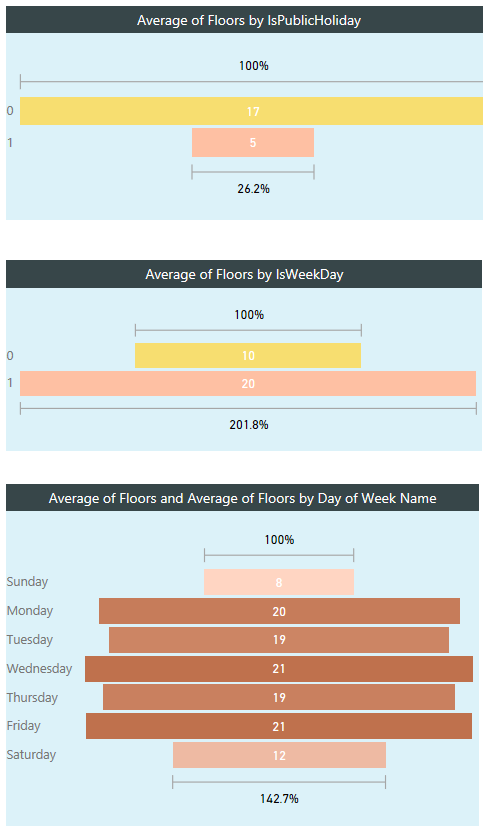
Well, you can customize the visualization as you want in the formatting option of each chart or element. Have a play with it and create something remarkable. I’m sure you design it better than me, here is what I built in few minutes;

[](http://www.radacad.com/wp-content/uploads/2015/12/131.png)

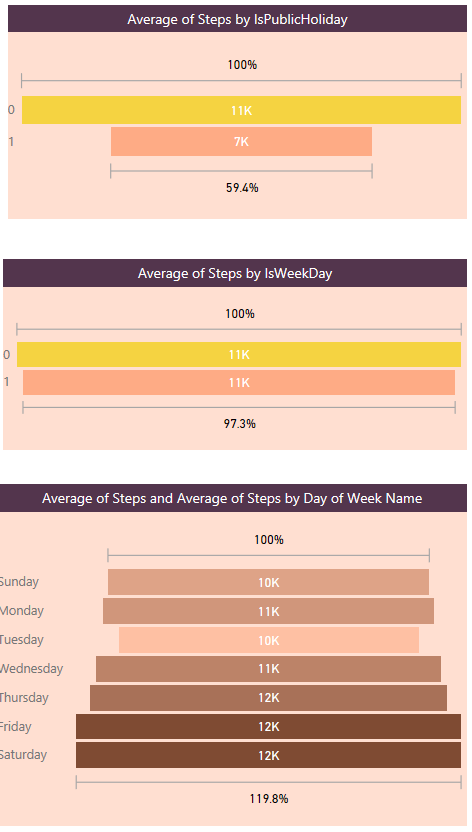
We are using the date dimension here, and one of the benefits of having date dimension is that I can analyze the data based on weekdays, and public holidays. and see which day of week usually I perform best in walking! Here we go;

[](http://www.radacad.com/wp-content/uploads/2015/12/141.png)

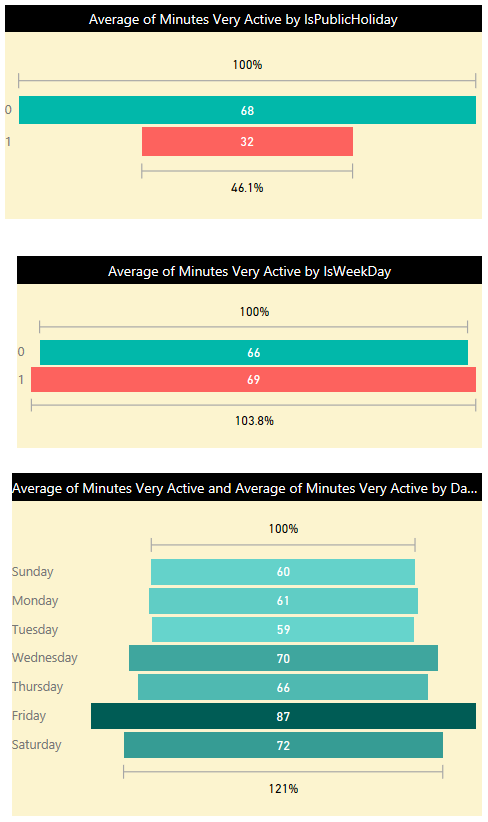
Interesting! Isn’t it? I can get more insights out of this visualization than what I see in Fitbit dashboard. I can see that I’m terribly performing bad in taking floors in public holidays! Well our home is flat single floor. However I’m doing a little bit better in weekends. My best days for Floors is always weekdays and there isn’t really so much difference between them.

[](http://www.radacad.com/wp-content/uploads/2015/12/151.png)

Now if I check my steps; I’m not doing that bad in public holidays, 7K in public holidays in comparison with 11K is other days is fine for me 😉 And I’m doing even very close in weekends and weekdays. If I look at each day’s average I can see that I’m a bit tired in Sundays and prefer to rest more, however Saturdays are good days for me! so my overall weekend average goes up because of that. and for some unknown reasons Tuesdays I walk less! That’s something I have to consider really why it happens.

[](http://www.radacad.com/wp-content/uploads/2015/12/161.png)

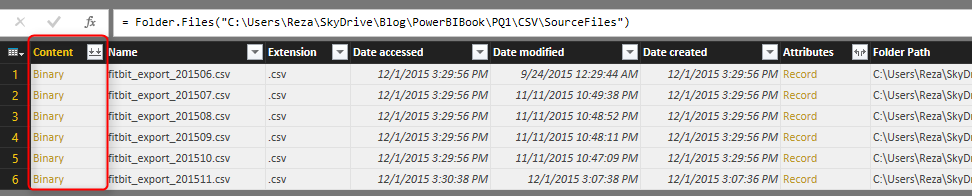
In the spit of what I see in Floors and Steps, I’m still doing a bit closer to my steps results in my very active minutes average. and my best active minutes comes Fridays. There is an obvious reason for it. Fridays I play indoor soccer with some friends, and I really feel my heart is coming out of my chest at some minutes in the game! Here is the evidence; I got most active minutes in Fridays!

[](http://www.radacad.com/wp-content/uploads/2015/12/171.png)

We can go on and on with building more visualizations on this data set. There is no limitation on the amount of insight you can get from it, so I leave it to you to use your own creativity and build dashboards and charts that gives you more insights. You’ve seen so far how easy is to be a BI developer if you use Power BI 😉

# Be Fitbit BI Developer in Few Steps: Step 2 Loop Through All CSV Files

Published Date : December 2, 2015

[](http://www.radacad.com/wp-content/uploads/2015/12/24.png)

In the [first step](http://www.radacad.com/be-fitbit-bi-developer-in-few-steps-first-step-get-data-from-csv) you’ve learned how to manipulate and mash up the data with few changes in Power Query M script. In spite of Power BI Desktop not being able to recognize your CSV file correctly because of the header row, you managed to write script and fetch data as you required. However this process needs to be repeated for the second file, and then the third file, and god knows what happens if you have 1000s of files. Real case scenarios contains many files, even millions. So a manual process won’t be the answer. You have to create your data transformation process in a way that works automatically not matter how many files are there.

## Warning: This post contains scripting!

In this post we will write M Power Query script to loop through all CSV files in the source folder and process them one by one with the fetch table script that we produced in previous step. So be prepared to write some code, and don’t scare from it 😉 It is almost impossible to create a BI solution for real world problem solving that doesn’t include scripting (even a bit), So you have to be familiar with that. Yes, Power BI is not just a nice drag and drop dashboard building tool, and has many powerful features. You have to spend time to learn it. So here you go, let’s write some codes here.

# Looping Through CSV Files

Unfortunately Power Query or let’s say Power BI doesn’t have a loop structure, and that is because of the functional structure of this language. However there are data structures such as Table and List that can be easily used with **each** singleton function to work exactly as a loop structure does. Here in this post I will get you through the process of looping into files in a directory and processing them all, and finally combining them into a large big table. You will also learn some Power Query M functions through this process.

# Where We’ve been in Previous Step

In previous step we ended up with below script

let

Source = Table.FromColumns({Lines.FromBinary(File.Contents("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles\fitbit\_export\_201507.csv"),null,null,1252)}),

#"Removed Top Rows" = Table.Skip(Source,1),

#"Split Column by Delimiter" = Table.SplitColumn(#"Removed Top Rows","Column1",Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv),{"Column1.1", "Column1.2", "Column1.3", "Column1.4", "Column1.5", "Column1.6", "Column1.7", "Column1.8", "Column1.9", "Column1.10"}),

#"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Column1.1", type text}, {"Column1.2", type text}, {"Column1.3", type text}, {"Column1.4", type text}, {"Column1.5", type text}, {"Column1.6", type text}, {"Column1.7", type text}, {"Column1.8", type text}, {"Column1.9", type text}, {"Column1.10", type text}}),

#"Promoted Headers" = Table.PromoteHeaders(#"Changed Type"),

#"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(\_), {"", null}))),

#"Changed Type1" = Table.TransformColumnTypes(#"Removed Blank Rows",{{"Steps", Int64.Type}, {"Calories Burned", Int64.Type}, {"Floors", Int64.Type}, {"Minutes Sedentary", Int64.Type}, {"Minutes Lightly Active", Int64.Type}, {"Minutes Fairly Active", Int64.Type}, {"Minutes Very Active", Int64.Type}, {"Activity Calories", Int64.Type}, {"Distance", type number}}),

#"Added Custom" = Table.AddColumn(#"Changed Type1", "Dated", each Date.FromText([Date],"en-NZ")),

#"Changed Type2" = Table.TransformColumnTypes(#"Added Custom",{{"Dated", type date}}),

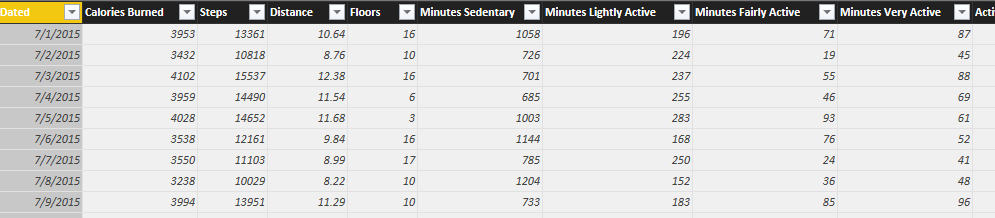
#"Removed Columns" = Table.RemoveColumns(#"Changed Type2",{"Date"}),

#"Reordered Columns" = Table.ReorderColumns(#"Removed Columns",{"Dated", "Calories Burned", "Steps", "Distance", "Floors", "Minutes Sedentary", "Minutes Lightly Active", "Minutes Fairly Active", "Minutes Very Active", "Activity Calories"})

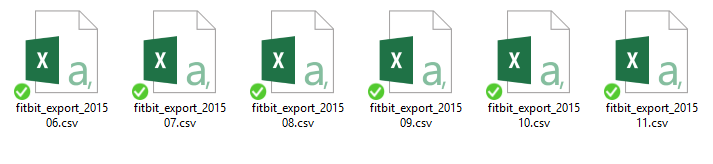
in

#"Reordered Columns"

That generates table below from an exported CSV file of Fitbit activities

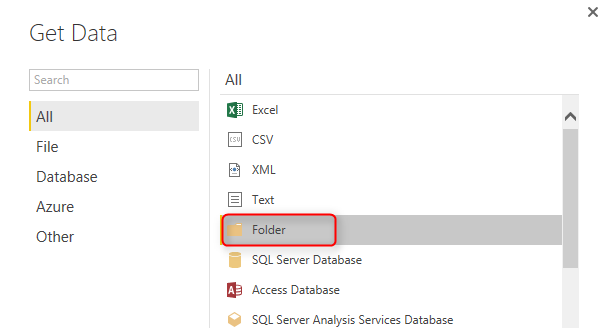
[](http://www.radacad.com/wp-content/uploads/2015/12/19.png)

Now There are multiple files in the source folder that I want to repeat the process for all of them;

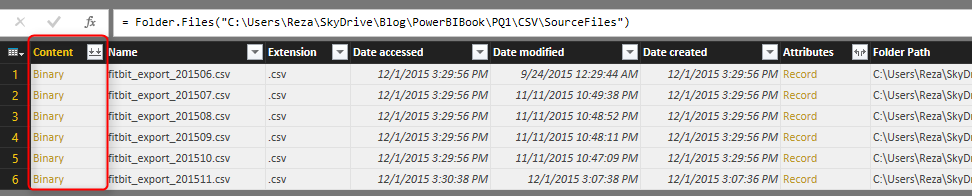
[](http://www.radacad.com/wp-content/uploads/2015/12/3.png)

# Get Data From Folder

Let’s start the process with reading all files in a folder. Power BI Desktop can do that easily. Through Get Data options choose from Folder and the select the folder containing source files as above.

[](http://www.radacad.com/wp-content/uploads/2015/12/110.png)

This will load all files and their general information into a single table. There are columns for name of file, folder path, date created, modified, and etc. There is also a column named Content. This is an important column as it contains the whole content of each file in a Binary format.

[](http://www.radacad.com/wp-content/uploads/2015/12/24.png)

# Custom Function

So far we have list of all files and the script to convert their content into a desired table structure. We can create a custom function with that script and re-use it for each file’s content. The process of creating custom functions are easy. I’m not going to explains every parts of creating custom function because that’s a whole separate topic. If you are interested to read more about [Creating Custom Functions in Power Query read this post](http://www.radacad.com/writing-custom-functions-in-power-query-m).

Go into Advanced Editor, and create the function as below highlighted section;

let

FetchTable = (table) =>

let

Source = Table.FromColumns({Lines.FromBinary(table,null,null,1252)}),

#"Removed Top Rows" = Table.Skip(Source,1),

#"Split Column by Delimiter" = Table.SplitColumn(#"Removed Top Rows","Column1",Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv),{"Column1.1", "Column1.2", "Column1.3", "Column1.4", "Column1.5", "Column1.6", "Column1.7", "Column1.8", "Column1.9", "Column1.10"}),

#"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Column1.1", type text}, {"Column1.2", type text}, {"Column1.3", type text}, {"Column1.4", type text}, {"Column1.5", type text}, {"Column1.6", type text}, {"Column1.7", type text}, {"Column1.8", type text}, {"Column1.9", type text}, {"Column1.10", type text}}),

#"Promoted Headers" = Table.PromoteHeaders(#"Changed Type"),

#"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(\_), {"", null}))),

#"Changed Type1" = Table.TransformColumnTypes(#"Removed Blank Rows",{{"Steps", Int64.Type}, {"Calories Burned", Int64.Type}, {"Floors", Int64.Type}, {"Minutes Sedentary", Int64.Type}, {"Minutes Lightly Active", Int64.Type}, {"Minutes Fairly Active", Int64.Type}, {"Minutes Very Active", Int64.Type}, {"Activity Calories", Int64.Type}, {"Distance", type number}}),

#"Added Custom" = Table.AddColumn(#"Changed Type1", "Dated", each Date.FromText([Date],"en-NZ")),

#"Changed Type2" = Table.TransformColumnTypes(#"Added Custom",{{"Dated", type date}}),

#"Removed Columns" = Table.RemoveColumns(#"Changed Type2",{"Date"}),

#"Reordered Columns" = Table.ReorderColumns(#"Removed Columns",{"Dated", "Calories Burned", "Steps", "Distance", "Floors", "Minutes Sedentary", "Minutes Lightly Active", "Minutes Fairly Active", "Minutes Very Active", "Activity Calories"})

in

#"Reordered Columns",

Files = Folder.Files("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles")

in

Files

Note that all the highlighted part from line 2 to line 16 in code above is the code for the custom function. The first line of function (line 2 in above script) contains name of the function; **FetchTable**, and the input parameter; **table**. From line 3 to line 16 is all the script we’ve copied from [first step](http://www.radacad.com/be-fitbit-bi-developer-in-few-steps-first-step-get-data-from-csv) of this exercise.

To Make sure that the function works properly you can call it as below;

FetchTable(Files{[#"Folder Path"="C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles\",Name="fitbit\_export\_201509.csv"]}[Content])

Add above single line of code in the IN section of script instead of Files. and you will see the result table as below:

[](http://www.radacad.com/wp-content/uploads/2015/12/31.png)

So this shows the function definition and function call work perfectly. Now let’s loop through files.

# EACH Singleton Function

EACH is a singleton function, which means it has one input parameter and one output. It can be applied on EACH record in a table or EACH item in a list. with using that you can apply calculations, data transformations, adding custom columns and many other operations. For our case I want to use EACH to apply FetchTable function on every Content value of the Files table. Each is easy to use as below;

Files = Folder.Files("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles"),

TableAdded= Table.AddColumn(Files,"Data",each FetchTable([Content])),

DataColumnSelected=Table.SelectColumns(TableAdded,{"Data"})

in

DataColumnSelected

As you see the single line above and specially the section ***each FetchTable([Content])*** will apply FetchTable function for every Content value in each record of the table. and as a result we will have a new column called Data that includes tables. line 3 of script above just select Data column (because we don’t need other columns). Here is the result;

[](http://www.radacad.com/wp-content/uploads/2015/12/41.png)

As you see in the result set above we have one table with one column (named Data), and it includes a Table in every item. This is the table that we want to use, so we have to combine all of these tables into one table. Fortunately there is a function for it in Power Query called Table.Combine .

# Combining, Appending, or UNION of Tables

[Table.Combine](https://msdn.microsoft.com/en-us/library/mt260748.aspx) does the UNION of multiple tables into one table. This function get a LIST of tables (they should be all with the same structure), and returns a single table which is appended version (or UNION version) of all the table. Here is the definition of this function;

Table.Combine(tables as list) as table

As you mentioned probably this function needs a LIST as an input parameter. And so far we have built a TABLE (single column table) which needs to be converted to list. There are different methods of converting a table into a list. I’ll show you one method here. I first fetch only a single column list from this table with [Table.ToColumns](https://msdn.microsoft.com/en-us/library/mt260798.aspx) function, and then I get the only first value of the list with [List.First](https://msdn.microsoft.com/en-us/library/mt253624.aspx) function. Finally I use the Table.Combine with the list parameter, here’s the code:

Files = Folder.Files("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles"),

TableAdded= Table.AddColumn(Files,"Data",each FetchTable([Content])),

DataColumnSelected=Table.SelectColumns(TableAdded,{"Data"}),

TablesColumn=Table.ToColumns(DataColumnSelected),

TablesList=List.First(TablesColumn),

AllTables=Table.Combine(TablesList)

in

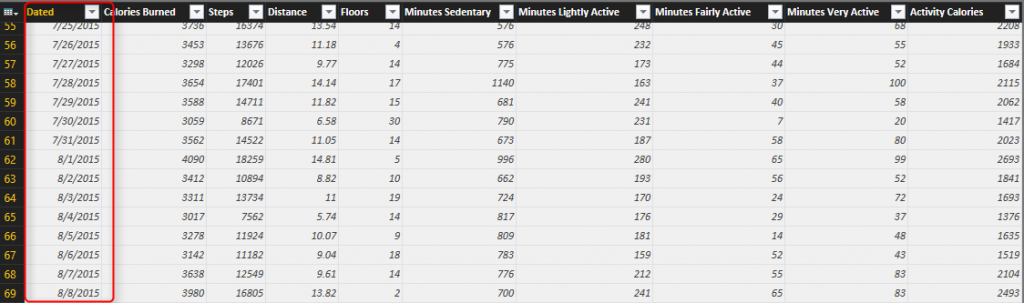
AllTables

Line 4 converts the table to a list, but a list that has another list within. Because a table might have more than one column.

Line 5 gets the first item of the list, which would be the first column of the table. For our example as we have only a single column then only that column would be fetched.

Line 6 finally combines all the tables in the list into one large table.

And the result:

[](http://www.radacad.com/wp-content/uploads/2015/12/51.png)

You can see that I have now all tables combined as I have all months together in this final table.

Here is the full script of this demo so far:

let

FetchTable = (table) =>

let

Source = Table.FromColumns({Lines.FromBinary(table,null,null,1252)}),

#"Removed Top Rows" = Table.Skip(Source,1),

#"Split Column by Delimiter" = Table.SplitColumn(#"Removed Top Rows","Column1",Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv),{"Column1.1", "Column1.2", "Column1.3", "Column1.4", "Column1.5", "Column1.6", "Column1.7", "Column1.8", "Column1.9", "Column1.10"}),

#"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Column1.1", type text}, {"Column1.2", type text}, {"Column1.3", type text}, {"Column1.4", type text}, {"Column1.5", type text}, {"Column1.6", type text}, {"Column1.7", type text}, {"Column1.8", type text}, {"Column1.9", type text}, {"Column1.10", type text}}),

#"Promoted Headers" = Table.PromoteHeaders(#"Changed Type"),

#"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(\_), {"", null}))),

#"Changed Type1" = Table.TransformColumnTypes(#"Removed Blank Rows",{{"Steps", Int64.Type}, {"Calories Burned", Int64.Type}, {"Floors", Int64.Type}, {"Minutes Sedentary", Int64.Type}, {"Minutes Lightly Active", Int64.Type}, {"Minutes Fairly Active", Int64.Type}, {"Minutes Very Active", Int64.Type}, {"Activity Calories", Int64.Type}, {"Distance", type number}}),

#"Added Custom" = Table.AddColumn(#"Changed Type1", "Dated", each Date.FromText([Date],"en-NZ")),

#"Changed Type2" = Table.TransformColumnTypes(#"Added Custom",{{"Dated", type date}}),

#"Removed Columns" = Table.RemoveColumns(#"Changed Type2",{"Date"}),

#"Reordered Columns" = Table.ReorderColumns(#"Removed Columns",{"Dated", "Calories Burned", "Steps", "Distance", "Floors", "Minutes Sedentary", "Minutes Lightly Active", "Minutes Fairly Active", "Minutes Very Active", "Activity Calories"})

in

#"Reordered Columns",

Files = Folder.Files("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles"),

TableAdded= Table.AddColumn(Files,"Data",each FetchTable([Content])),

DataColumnSelected=Table.SelectColumns(TableAdded,{"Data"}),

TablesColumn=Table.ToColumns(DataColumnSelected),

TablesList=List.First(TablesColumn),

AllTables=Table.Combine(TablesList)

in

AllTables

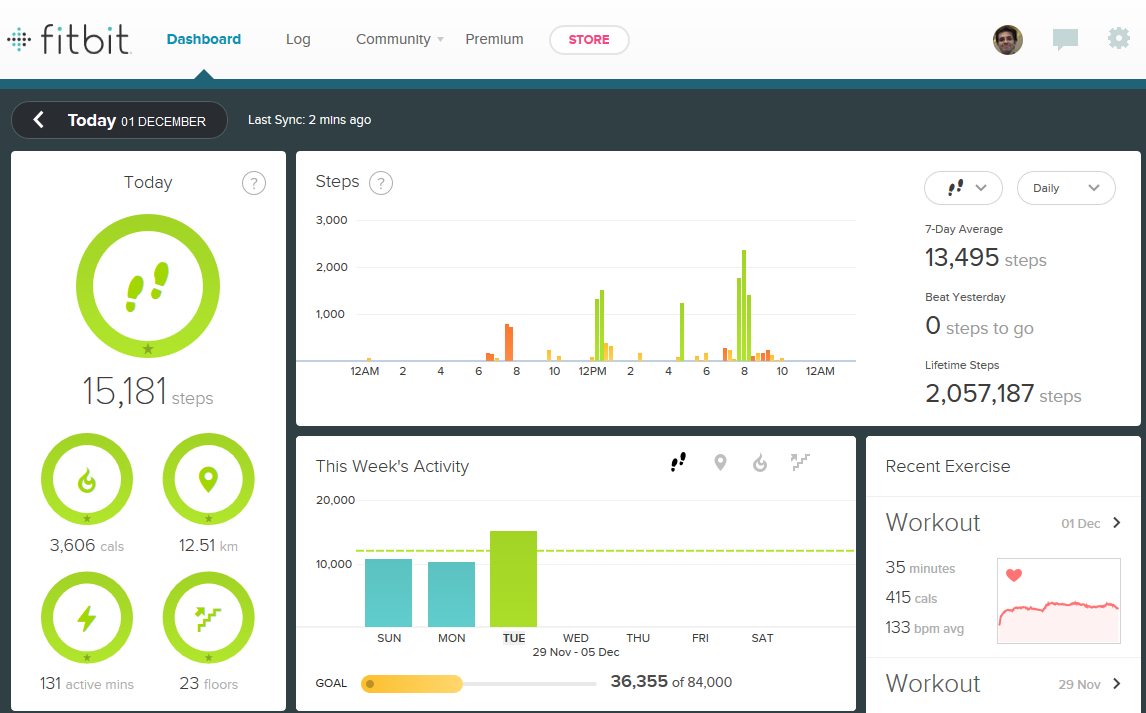
That ends this step. You’ve learned some scripting skills here today, and I hope you’ve enjoyed it. With few lines of scripting like this you can add massive power to your queries, you can automate things and get rid of manual extra work. Power Query is one of the most power full components of Power BI that gives you ability to apply many transformations and data mash up. The next step will be about Visualization of the data we’ve prepared so far, so stay tuned!

# Be Fitbit BI Developer in Few Steps: First Step Get Data from CSV

Published Date : December 1, 2015

[](http://www.radacad.com/wp-content/uploads/2015/12/0.png)

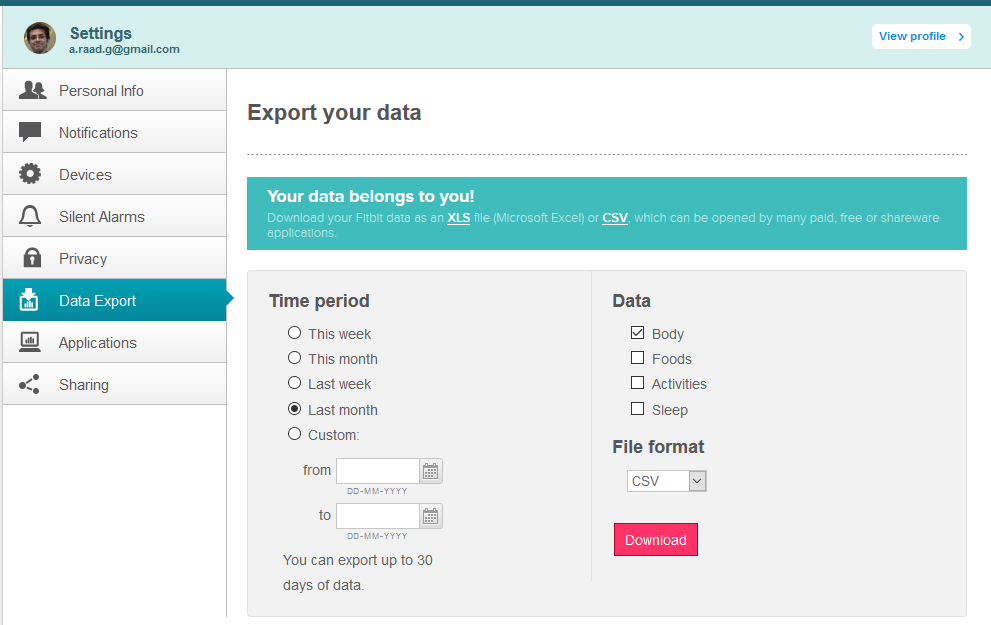
Power BI made things really easy, it isn’t hard to build dashboards and data analysis on almost any types of data sources. Previously in other sections of the [online book; Power BI from Rookie to Rockstar](http://www.radacad.com/online-book-power-bi-from-rookie-to-rockstar) I’ve mentioned few data sources and ways of connecting to them. In this post I like to show you how Power BI easily works with CSV or Text files. CSV or Text files can be in different formats, but for this example I through using Fitbit exported CSV data sounds to be great demo. The reason is that you probably have seen your Fitbit dashboard in the website as below:

[](http://www.radacad.com/wp-content/uploads/2015/12/1.png)

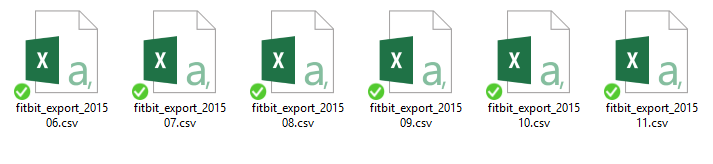
So for this post we are going to build that dashboard (not all of that obviously, because we don’t have the data required for all of that), but most part of it with Power BI. You will see how easy and powerful is Power BI in this kind of scenarios, and you will see how you can be the BI Developer of Fitbit in a few steps of building this demo.

Before start I have to apologize in advance to Fitbit developer team, and I have to say that I don’t want to say they are not doing anything. Obviously they are doing great job, and the most part of this great job is hidden from our view, they are mashing up the data to build it in the way that can be visualized best. So I admire them because of their efforts. In this post I just want to show how easy is to build dashboards with Power BI when the data is available in CSV or Text format.

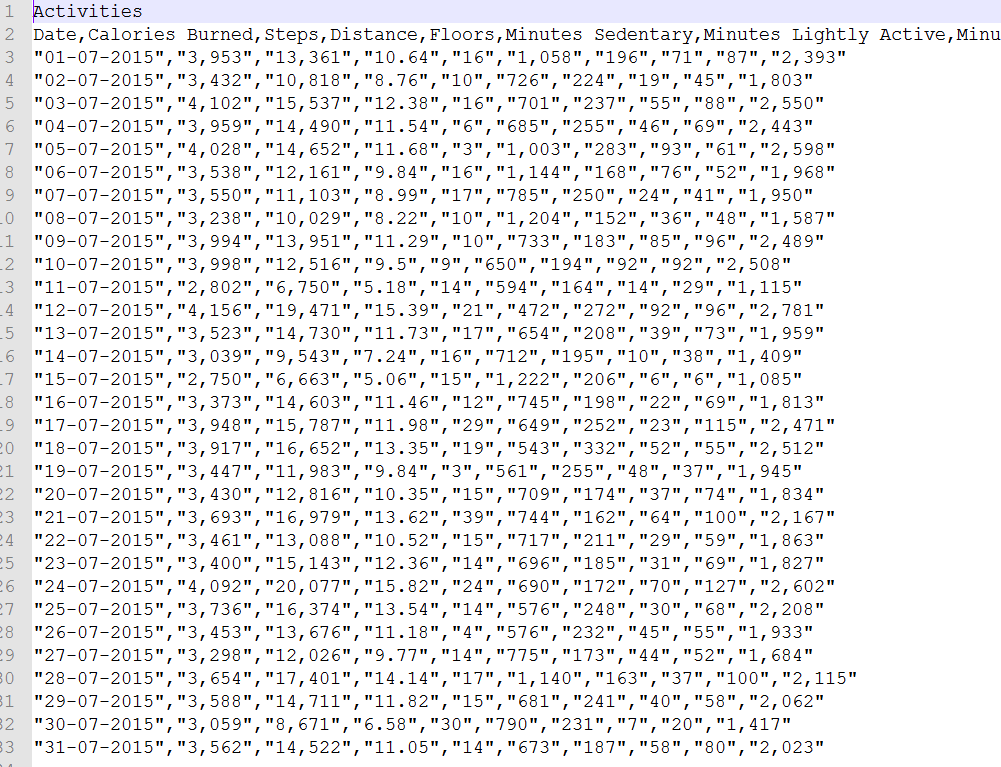
So Let’s get started. Fitbit devices are usually smart bands that measure your steps, number of floors, calories spent, heart rate and some other information. You can always download an extracted version of your activities as CSV file from your account page in Fitbit website. This always gives you a month period of data. So if you want the data to be exported for two different month, you will end up with two CSV files.

[](http://www.radacad.com/wp-content/uploads/2015/12/2.png)

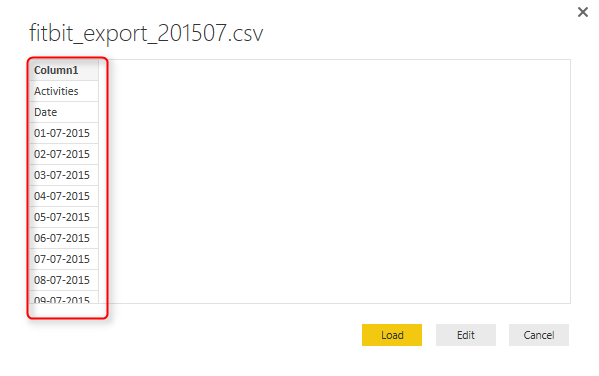
I have exported my Fitbit activities history from the time that I got it as a gift (Great gift I have to say 🙂 ) till now, which is 5 months.

[](http://www.radacad.com/wp-content/uploads/2015/12/3.png)

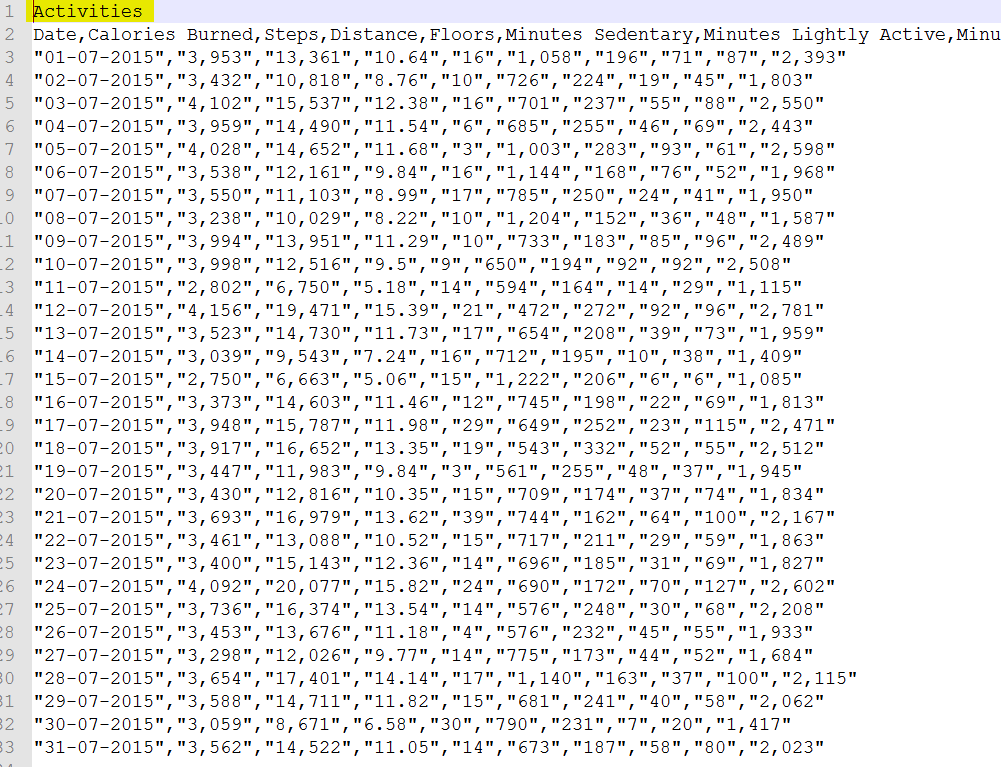
Here is sample data set in files:

[](http://www.radacad.com/wp-content/uploads/2015/12/4.png)

I start reading data from CSV in Power BI Desktop with Get Data from CSV. But this is the result I get:

[](http://www.radacad.com/wp-content/uploads/2015/12/5.png)

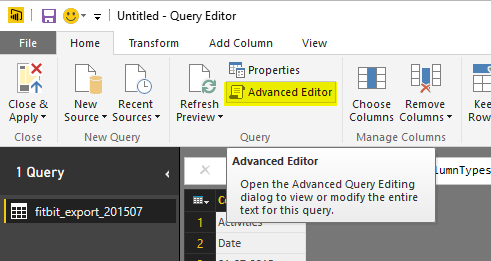
Well, this is happening because of the very first record in CSV file which is the header row, as you see below;

[](http://www.radacad.com/wp-content/uploads/2015/12/6.png)

The first row doesn’t have any delimiter, and that is why Power BI Desktop THINKS that this is a one column CSV file. To fix this issue you can use two approaches:

* Manually change each CSV file and remove the header row, and save it back.
* Write few lines in Power Query scripts and get rid of the first line through script automatically.

The first approach isn’t my favorite as it is manual, and won’t be so much of use in real world cases when you have 1000s of files. so let’s go with the second. Click on Edit and go to Query Editor window. There you can click on Advanced Editor to Change the code.

[](http://www.radacad.com/wp-content/uploads/2015/12/7.png)

In the Advanced Editor window you see the M script that generates the output.

[](http://www.radacad.com/wp-content/uploads/2015/12/8.png)

Change the Code to below;

let

Source = Table.FromColumns({Lines.FromBinary(File.Contents("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles\fitbit\_export\_201507.csv"),null,null,1252)})

in

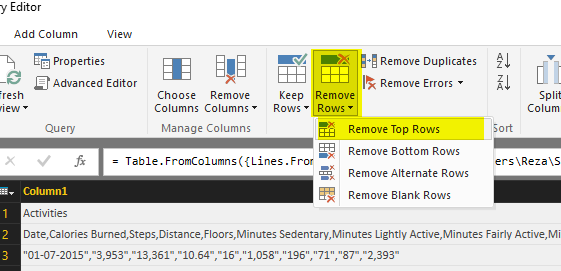
Source

Now you can see the result as full data in one column rather than only first column with date values.

[](http://www.radacad.com/wp-content/uploads/2015/12/9.png)

Follow steps below to shape the data for using the best in report;

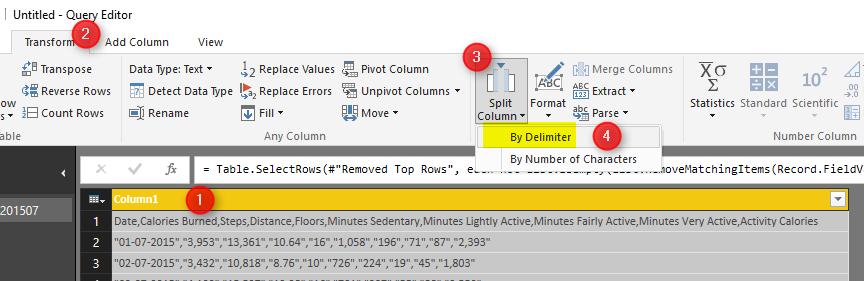
Get rid of first record with Remove Rows, and then choosing 1 top row to remove;

[](http://www.radacad.com/wp-content/uploads/2015/12/10.png)

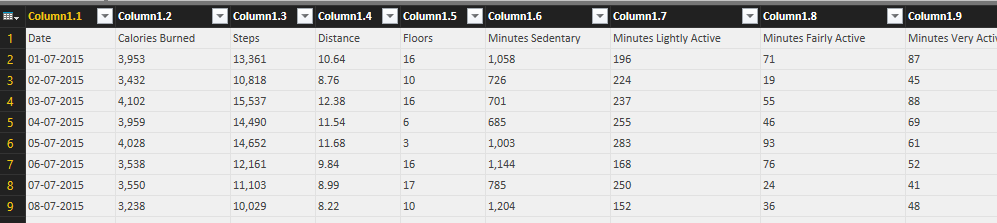
You can also remove any blank rows at the end of report

[](http://www.radacad.com/wp-content/uploads/2015/12/11.png)

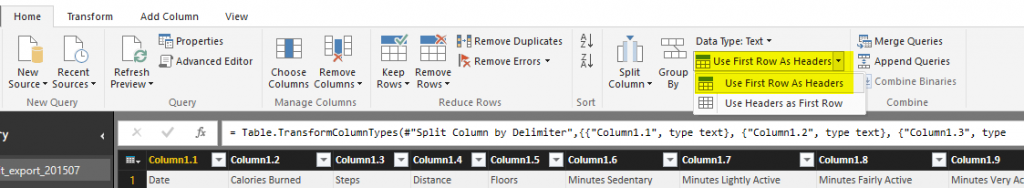
Then select Column1 and Got to Transform Tab, Choose Split Column, then By Delimiter

[](http://www.radacad.com/wp-content/uploads/2015/12/12.png)

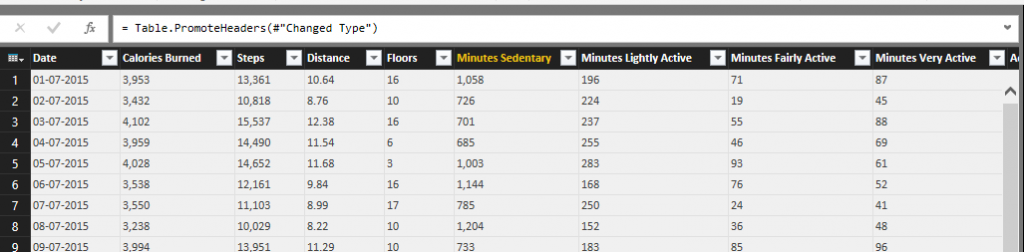
Leave the options as default (which is delimiter Comma, and Split at each occurrence of the delimiter) in Split Column by Delimiter dialog box and continue. You will see how columns all split as below;

[](http://www.radacad.com/wp-content/uploads/2015/12/13.png)

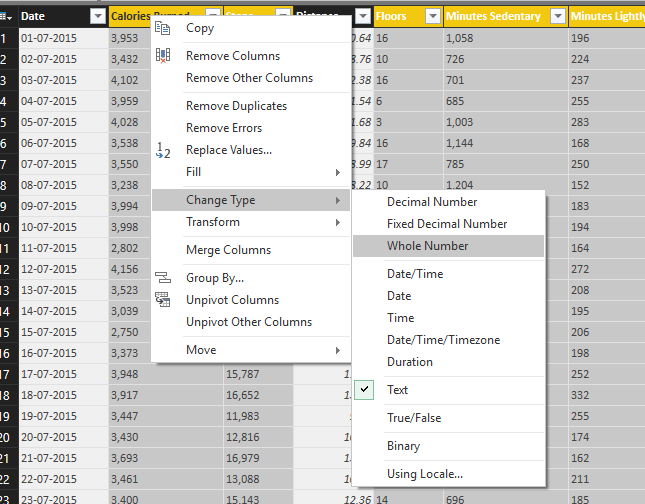
Set the first row as header, which going to Home Tab in Query Editor window and click on Use First Row As Headers option.

[](http://www.radacad.com/wp-content/uploads/2015/12/14.png)

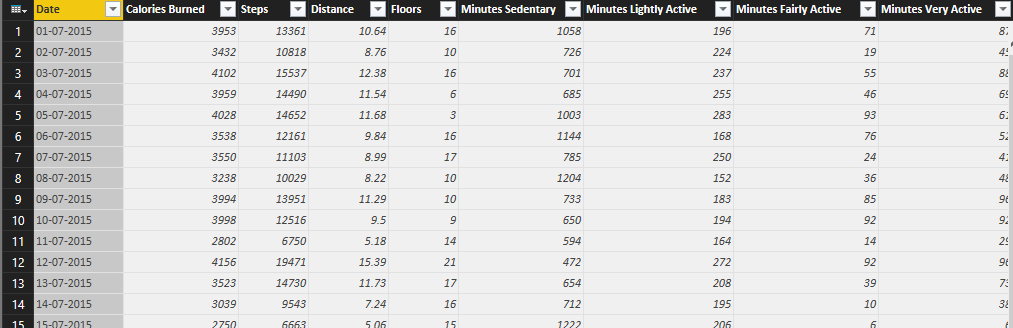
This option will rename all column headers to the values from the first row;

[](http://www.radacad.com/wp-content/uploads/2015/12/15.png)

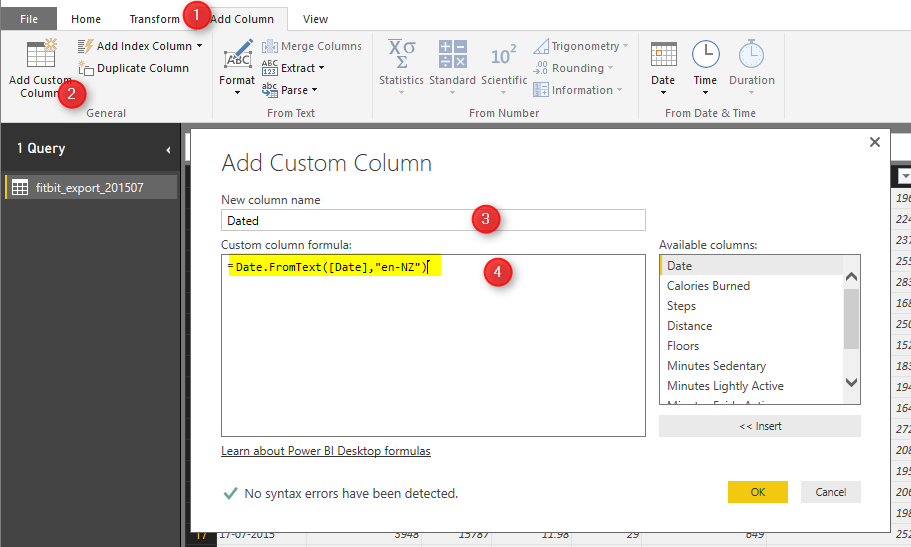
Now change data type of Distance column to Decimal, and all other columns (except Date) to Whole Number

[](http://www.radacad.com/wp-content/uploads/2015/12/16.png)

For the Date Column itself you can’t always change the type that easily, because the Date conversion works with locale of your system. As you see in screenshot below the Date format is DD-MM-YYYY which not always equal to locale of all systems.

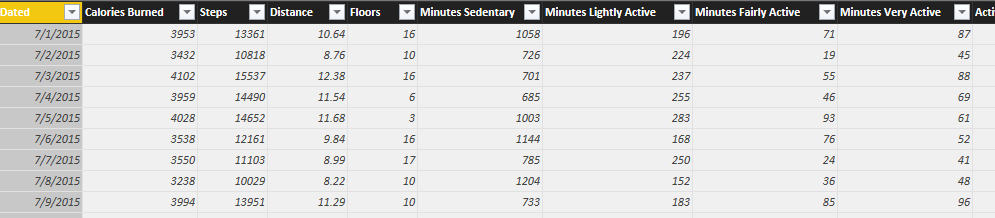
[](http://www.radacad.com/wp-content/uploads/2015/12/17.png)

If you try to convert it you might get an error. So because of that I convert it through M script. Click on Add Column, and then choose Custom Column. Here is where you can write M script to generate new column. the script is:

[](http://www.radacad.com/wp-content/uploads/2015/12/18.png)

Date.FromText([Date],"en-NZ")

The locale that you might be using might be different. [Here](https://msdn.microsoft.com/en-us/goglobal/bb896001.aspx) are list of culture and locale information. After doing that you will see Dated Column added to the end of table with Date data type. Now you can remove the original Date column and move the new Dated Column to the beginning of the table as below;

[](http://www.radacad.com/wp-content/uploads/2015/12/19.png)

So our data is ready now to work with. And this is the final M script of our work;

let

Source = Table.FromColumns({Lines.FromBinary(File.Contents("C:\Users\Reza\SkyDrive\Blog\PowerBIBook\PQ1\CSV\SourceFiles\fitbit\_export\_201507.csv"),null,null,1252)}),

#"Removed Top Rows" = Table.Skip(Source,1),

#"Split Column by Delimiter" = Table.SplitColumn(#"Removed Top Rows","Column1",Splitter.SplitTextByDelimiter(",", QuoteStyle.Csv),{"Column1.1", "Column1.2", "Column1.3", "Column1.4", "Column1.5", "Column1.6", "Column1.7", "Column1.8", "Column1.9", "Column1.10"}),

#"Changed Type" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Column1.1", type text}, {"Column1.2", type text}, {"Column1.3", type text}, {"Column1.4", type text}, {"Column1.5", type text}, {"Column1.6", type text}, {"Column1.7", type text}, {"Column1.8", type text}, {"Column1.9", type text}, {"Column1.10", type text}}),

#"Promoted Headers" = Table.PromoteHeaders(#"Changed Type"),

#"Removed Blank Rows" = Table.SelectRows(#"Promoted Headers", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(\_), {"", null}))),

#"Changed Type1" = Table.TransformColumnTypes(#"Removed Blank Rows",{{"Steps", Int64.Type}, {"Calories Burned", Int64.Type}, {"Floors", Int64.Type}, {"Minutes Sedentary", Int64.Type}, {"Minutes Lightly Active", Int64.Type}, {"Minutes Fairly Active", Int64.Type}, {"Minutes Very Active", Int64.Type}, {"Activity Calories", Int64.Type}, {"Distance", type number}}),

#"Added Custom" = Table.AddColumn(#"Changed Type1", "Dated", each Date.FromText([Date],"en-NZ")),

#"Changed Type2" = Table.TransformColumnTypes(#"Added Custom",{{"Dated", type date}}),

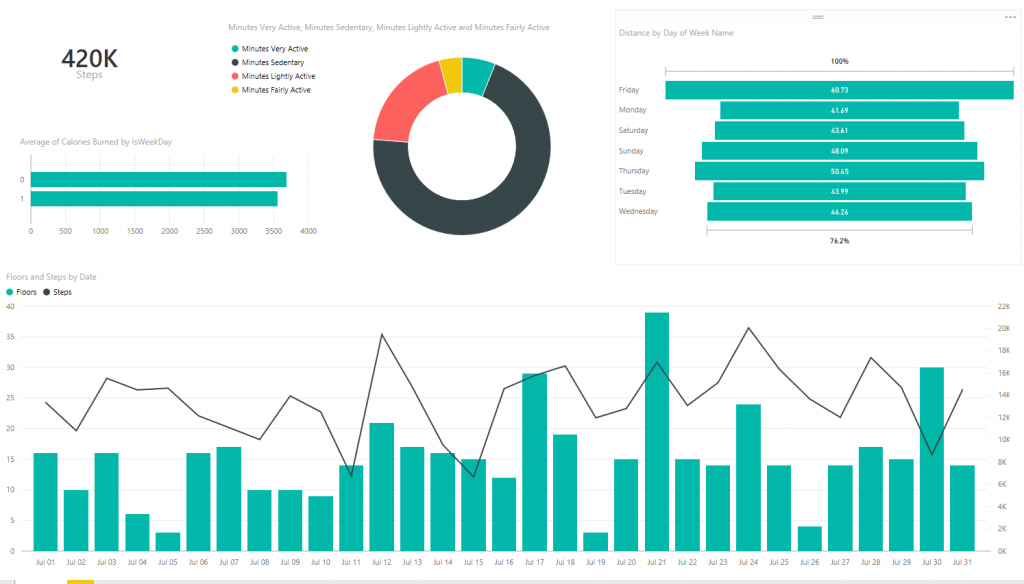
#"Removed Columns" = Table.RemoveColumns(#"Changed Type2",{"Date"}),

#"Reordered Columns" = Table.ReorderColumns(#"Removed Columns",{"Dated", "Calories Burned", "Steps", "Distance", "Floors", "Minutes Sedentary", "Minutes Lightly Active", "Minutes Fairly Active", "Minutes Very Active", "Activity Calories"})

in

#"Reordered Columns"

However we have some other files to work with as well, and they all have same structure. We can use M Script to loop through all files in the source directory and load them one by one automatically, but that would make this post a very long post, and requires also a bit of scripting which you might not like. I will explain that part in the next section. For now just for you to stay interested here is a glance of a sample report at the end of this effort (we will build that together through next steps);

[](http://www.radacad.com/wp-content/uploads/2015/12/23.png)