Contents

[Part 1 – Introducing Advanced Performance Analyser by SmartPowerBI (from the SmartPowerBI.co.uk blog post). 2](#_Toc84768760)

[What is Power BI Advanced Performance Analyzer by SmartPowerBI? 2](#_Toc84768761)

[Use cases are for this Advanced Performance Analyzer. 4](#_Toc84768762)

[What is native Power BI Desktop Performance Analyzer? 4](#_Toc84768763)

[What do I get from Advanced Performance Analyzer by SmartPowerBI? 5](#_Toc84768764)

[Analysis by Action. 5](#_Toc84768765)

[All Actions Summary. 8](#_Toc84768766)

[Action Gantt. 10](#_Toc84768767)

[Credits. 12](#_Toc84768768)

[Part 2 - Starting with Advanced Performance Analyzer. 13](#_Toc84768769)

[Initial notes. 13](#_Toc84768770)

[How to read the results of the tool. 14](#_Toc84768771)

[Analysis by Action Page. 14](#_Toc84768772)

[All Actions Summary Page. 18](#_Toc84768773)

[Kicker bonus! Action Gannt and parallelism. 19](#_Toc84768774)

[Part 3 - Guide to download and use “Advanced Performance Analyzer by SmartPowerBI”. 21](#_Toc84768775)

[Initial considerations. 21](#_Toc84768776)

[Step by step guide for capturing performance analysis. 23](#_Toc84768777)

[Download Advanced Power BI Performance Analyser. 26](#_Toc84768778)

[How To’s. 26](#_Toc84768779)

[How to configure different thresholds. 26](#_Toc84768780)

[How to configure different rationale, recommendations and links to best practices and guidelines. 27](#_Toc84768781)

[Credits. 28](#_Toc84768782)

Power BI Advanced Performance Analyzer by SmartPowerBI.

# Part 1 – Introducing Advanced Performance Analyser by SmartPowerBI (from the SmartPowerBI.co.uk blog post).

Improving Power BI reports’ performance is not always easy and straight. Although Power BI Desktop provides native “Performance Analyzer”, the details given are not as rich as they could or should.  
“Power BI Advanced Performance Analyzer by SmartPowerBI” is designed to take Performance Analyzer to the next level and provide clear insights and it is easy to use, for PBI Developers and BI Managers.  
The use of “Advanced Performance Analyzer by SmartPowerBI” returns a full and clear report showing all performance affecting factors, recommendations specific to the magnitude of those factors and links to best practices and guidelines associated to such recommendations.

Power BI is a platform that **can perform** very well, and deliver reports that render fast, displaying complex insights over datasets of hundreds of millions of rows, if not more. However, there will be a few considerations and best practices to be applied so you can get the best performance that the combination of dataset, business requirements and technology can achieve.

This article is meant for Power BI Developers and BI Managers alike, so they can use a simple tool to gain immediate insights on what could be improved in their reports for better performance. In addition to this, it is also meant for Power BI Centres of Excellence or other IT teams that need to help their business to build faster solutions.

## What is Power BI Advanced Performance Analyzer by SmartPowerBI?

It is a Power BI template that takes .json file generated by Power BI Desktop native “Performance Analyzer”, and uses it to deliver clearer insights about different factors affecting reports’ performance – yes, that simple!

In other words, it is a Power BI Report that ingests, transforms, models, enriches and presents the .json file exported from a PBI Desktop native performance analyzer session.

But that’s not all, this tool also has a place to assign best practices and guidelines associated to specific recommendations, therefore this tool can:

* Produce immediately actionable insights.
* Ensure that best practices and guidelines are communicated so they can be applied…

**There is a more important agenda to this tool** **though**. As Head of Power BI Centre of Excellence, I face the challenge of supporting a business with **thousands of PBI authors**, and I cannot support them all directly. This tool has been conceived and created so I can help these authors to help themselves, through custom insights and best practices – this tool will make clear what factors are affecting their reports’ performance.

I built this tool to help a very large community of Power BI Practitioners to build faster reports, and with this article and associated documentation, I am trying to make that community even larger.

For those eager to see the product in action, here is a [**link**](https://app.powerbi.com/view?r=eyJrIjoiNGRhYzJhNTItMjY1Ny00M2Y2LTk1ODgtNTY1Yjc0YWI1ZmEyIiwidCI6IjI5ZmUyMzE1LTNmNzEtNDczYi05ZGMyLThkMjY4MmQxM2NjNSJ9) to a working live report (that I have used for most of the relevant screenshots in this article).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Analysis By Action | All Actions Summary | Action Gannt |

You can download the solution files and document from the [GitHub repository](https://github.com/alexjgarcia/Advanced-PowerBI-Performance-Analyzer-by-SmartPowerBI). This product comes “As Is” under [MIT license](https://choosealicense.com/licenses/mit/). Please read the conditions for this type of license.

In this repository, you will find the following files:

* .Pbix file named “Advanced PBI Performance Analyser by SmartPowerBI”, that contains the connection to the .json file (and others), loads, transforms, and models the data within it and presents the results in several pages.
* Excel file named “Advanced Performance Analyser by SmartPowerBI Support File”, which contains the rationale, recommendations, and links to other resources.
* Excel file named “Advanced Performance Analyser by SmartPowerBI Actions Script” which you will use to capture all the actions taken on the report subject to analysis that help enriching the results report, as will contain information that is not present in the original json file.

**A word of caution** (disclaimer): this solution is provided “as is” and there is no warrantee that will work in untested scenarios or if Microsoft decides to change the format or data quality of the .json file. There is also a known bug in the .json file affecting visuals internal ids, which this solution solves by removing duplicates, at the cost of potential data loss.

## Use cases are for this Advanced Performance Analyzer.

At the end, it is all the same: understanding reports’ performance affecting factors and knowing what to do to improve such performance. The big deal is that it can be beneficial in **different cases** and for **different personas**.

The personas that I have mainly in mind are 3:

* **Power BI Developers**, who work on the creation of data models, DAX calculations and report pages, all of it influencing end users’ experience.
* **BI Managers or BI Leads,**that can request their teams to run this tool to gain immediate and easy-to-understand insights on performance for those solutions that they are responsible for.
* **IT Organisations,**that need either, to enforce certain performance levels or to support large communities of Power BI authors.

The main use cases that I have in mind would be the following:

* **Initial performance assessment** when developing new solutions or introducing significant changes to existing ones (for developers).
* **Continuous performance assessment** and comparison between versions of solution via performance tracking (for developers and BI managers).
* **BI managers’ review and guide**: as not all BI managers know all the intricacies of PBI performance, this tool gives them a simplified and accurate view of what is going on in their reports.
* **Analysis prior to go live and/or onboarding in Premium capacity** shared with other solutions.
* **Continuous assessment** of solutions hosted in **Premium**.
* **Supporting community of PBI practitioners**, by IT organisations (i.e. CoEs) with a stake on performance.
* **Enforcing certain thresholds regarding performance parameters**, by IT organisations that want or need to do so.
* **Communicating Best Practices and Guidelines**, again by IT orgs with a bested interest.

## What is native Power BI Desktop Performance Analyzer?

It is an out-of-the-box functionality existing in Power BI Desktop where you can view visuals’ loading times for one or more pages.

|  |  |
| --- | --- |
|  | The Performance Analyzer pane in Power BI Desktop will show different timings for all visual elements rendered in your PBI report, triggered by either “Refresh visuals” button or actions taken on the report (filter change, data point selection,…).  DAX query and Visual display measure time taken for query resolution and visual render, respectively. |

There is an “Other” element in the visual timings, which is very important as does affect the complete report render time and depends, amongst other things, on the parallelism when loading content in the report canvas. We will talk more about this in this article as this is very important.

If you wish to know more about this PBI Desktop feature, please refer to:

* Microsoft’s [Performance Analyzer official documentation](https://docs.microsoft.com/en-us/power-bi/create-reports/desktop-performance-analyzer).
* GitHub’s [documentation about the JSON file format](https://github.com/microsoft/powerbi-desktop-samples/blob/main/Performance%20Analyzer/Power%20BI%20Performance%20Analyzer%20Export%20File%20Format.docx) (published in GH by [Will Thompson](https://github.com/MI77)).
* Not only [one](https://www.youtube.com/watch?v=1lfeW9283vA), but [two](https://www.youtube.com/watch?v=aYXRx0pZZcs) videos from [Marco Russo (SQLBI)](https://www.sqlbi.com/author/marco-russo/), alongside a myriad of [related articles](https://www.sqlbi.com/?s=%22performance+analyzer%22&type=).
* Chris Webb’s [related articles](https://blog.crossjoin.co.uk/?s=performance+analyzer&submit=Search).

## What do I get from Advanced Performance Analyzer by SmartPowerBI?

During a Power BI Desktop Performance Analyzer session, we can record **several actions** and the time taken for all the visual elements rendered within each action. The most common action recorded will be “**page refresh**”, which we will do by changing tabs, however we can also record actions such as drill down, change filter selection, or highlight a data point.

Each action will force refresh of different visual elements placed in the report canvas, and whilst full page refresh will trigger the update for all visuals in such page, changing filters or selecting data points, depending on how the interactions have been defined, will trigger actions in all or only some of them. Drilling up/down and expanding/collapsing hierarchies in a visual will only affect that visual.

This tool allows you to capture all those actions at once, and export in a single .json file and then analyse each action individually.

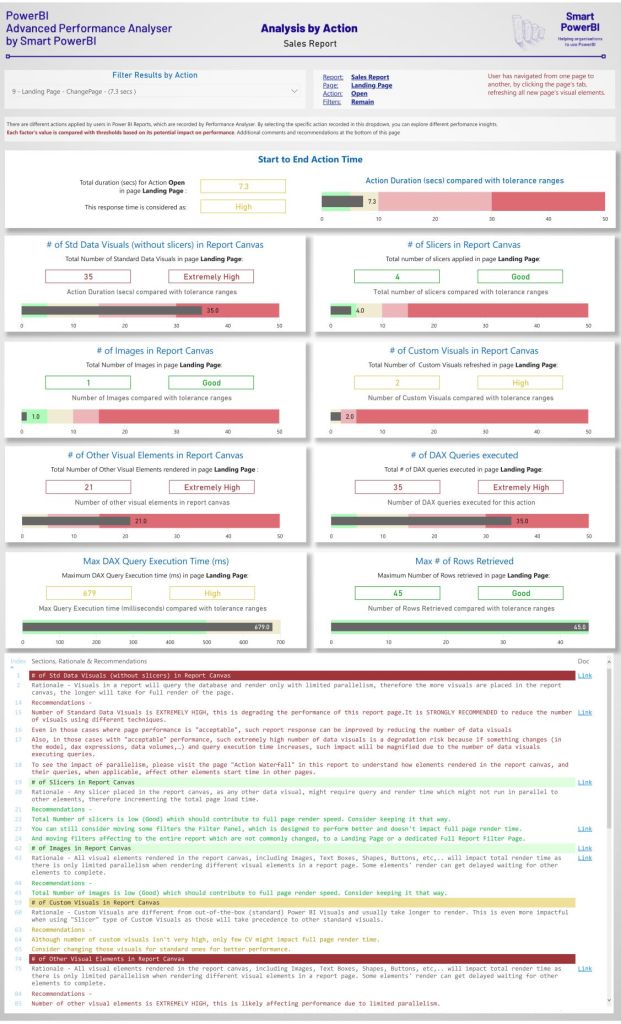
### Analysis by Action.

The first page in the report is one that produces insights about whatever action we have recorded and selected in the only filter existing in this page.

Once a certain action has been selected, the page will display all affecting factors (that can be extracted from the .json file) and:

* **Benchmarks**all the results.
* Provide a **rationale**of how that factor can affect performance.
* Provide **recommendations**depending on the benchmarking result (Good, High,…)
* Provide **links**that will refer to documentation relevant to the factor and the recommendation.

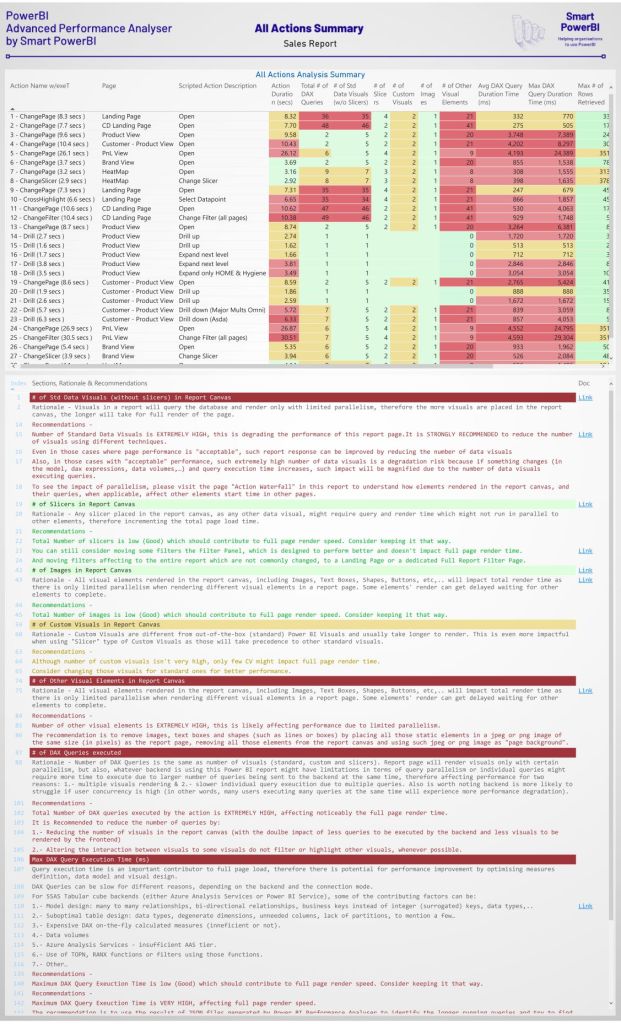
More details about all sections of the report explained in the full document in [GitHub repository](https://github.com/alexjgarcia/Advanced-PowerBI-Performance-Analyzer-by-SmartPowerBI).

[[](https://smartpowerbi.files.wordpress.com/2021/10/0001-m.jpg)](https://smartpowerbi.files.wordpress.com/2021/10/0001-m.jpg)Page: Analysis by Action

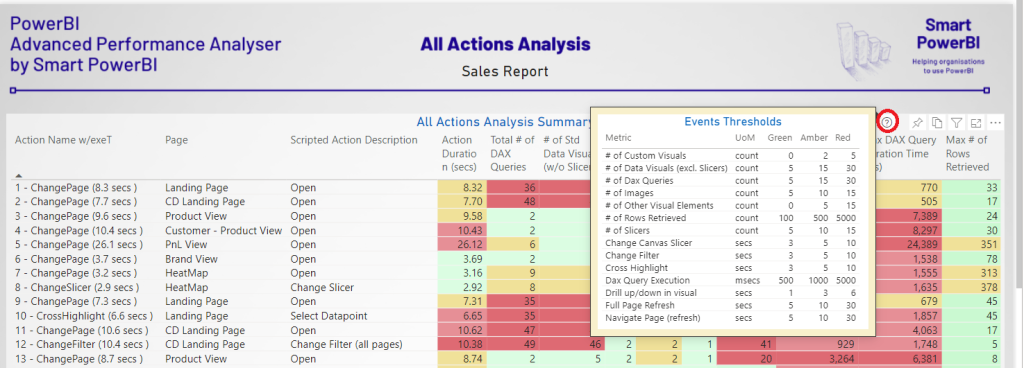
One important detail is that such **benchmarks**not only **drive the severity of the warnings**and the recommendations, but also **can be configured** to meet each organisation’s need.

### All Actions Summary.

Provides the same insights from the “Analysis by Action” page but summarises all actions recorded in performance analyzer and provides recommendations for the worst-case scenario for each factor, or for an individual action if it has been selected in the main table.

[[](https://smartpowerbi.files.wordpress.com/2021/10/0002-m.jpg)](https://smartpowerbi.files.wordpress.com/2021/10/0002-m.jpg)Page: All Actions Summary

In this page we can also see what the different thresholds are, as shown below.

Configurable Thresholds

### Action Gantt.

Not all Power BI practitioners are aware of the **influence of having a large number of visual elements in the report**.

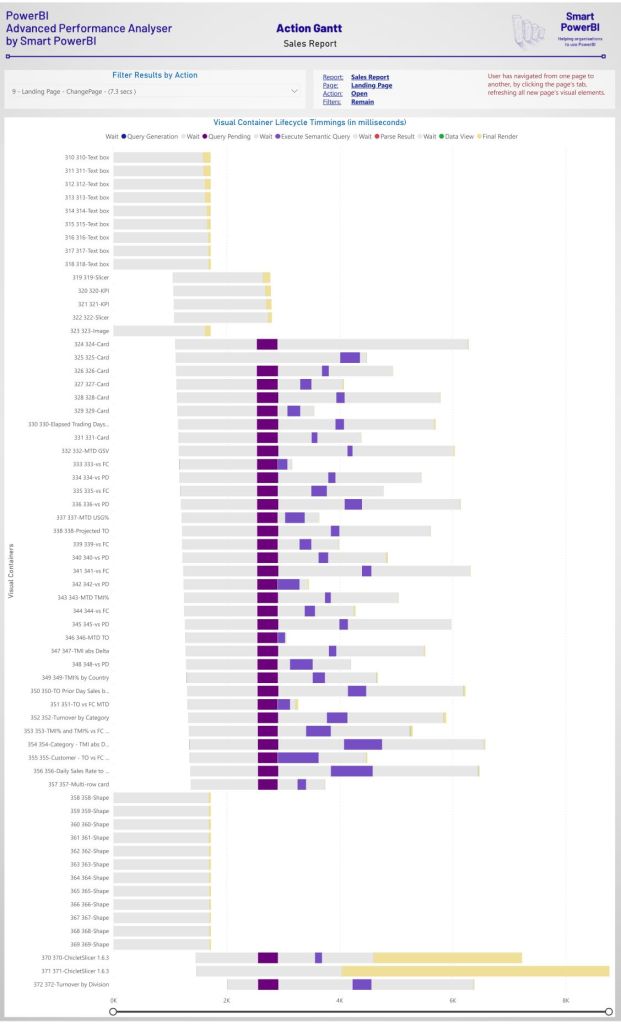
More specifically, visual elements, including those with static content (images, shapes, buttons,..) will render in the report canvas with certain parallelism, potentially pushing other more important visuals to be rendered at a later time.

**Action Gantt** page shows clearly what degree of parallelism is achieved in any given action and the dependencies when refreshing such large number of visual elements, including query preparation, query execution and wait time.

All visual elements rendered in the report page need to complete some of the following activities:

* Query Generation
* Query Pending
* Execute Semantic Query
* Parse Result
* Data View
* Final Render

And what is very important: **waiting before any of these events**, which will be affected, amongst other reasons, by limited parallelism of query and render actions.

[[](https://smartpowerbi.files.wordpress.com/2021/10/0003-m.jpg)](https://smartpowerbi.files.wordpress.com/2021/10/0003-m.jpg)Page: Action Gantt

As mentioned before, the complete documentation to understand how to use and interpret this tool can be downloaded from the [GitHub repository](https://github.com/alexjgarcia/Advanced-PowerBI-Performance-Analyzer-by-SmartPowerBI).

You are ready now to take native Performance Analyzer to the next level and leverage its details for supporting large communities of Power BI authors creating faster solutions, whilst educating such community in best practices and enforcing performance limits. Do not hesitate to contribute with your questions, comments, and recommendations.

## Credits.

In some parts of this article or the downloadable files in Github repository, I have used and referenced articles or other resources authored by:

* Patrick Leblanc ([Guy in a Cube (Power BI Training and More)](https://guyinacube.com/))
* Melissa Coates ([Coates Data Strategies](https://www.coatesdatastrategies.com/))
* Marco Russo ([Home – SQLBI](https://www.sqlbi.com/))
* Reid Havens ([Havens Consulting](https://www.havensconsulting.net/))
* Will Thompson ([Github repository](https://github.com/microsoft/powerbi-desktop-samples/tree/main/Performance%20Analyzer))
* Sam McKay ([Enterprise DNA | Empowering Power BI Users to Change Their World](https://enterprisedna.co/))
* James Serra ([James Serra’s Blog](https://www.jamesserra.com/))
* Michael Kovalsky ([ElegantBI](https://www.elegantbi.com/)).

And special thanks to [Steve Verschaeve](https://www.linkedin.com/in/steveverschaeve/), Senior Customer Engineer from Microsoft, for his thorough review of the documentation and great advice on the functionality.

xxxxxxxxxx

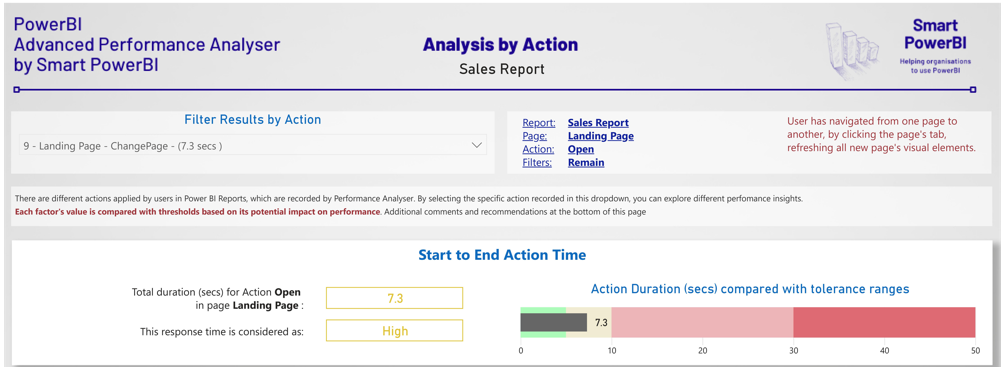
# Part 2 - Starting with Advanced Performance Analyzer.

## Initial notes.

If in the first part of this document, you could read a preamble of the tool, use cases and some considerations, and a glimpse of the tool itself. Now we will review:

* Each part of the tool (report pages) and what they mean and how to interpret them.
* Important notes about tolerance ranges.
* Consideration about Power BI Desktop and Service.
* View of parallelism when loading Power BI Reports.

Using this tool, anyone can get an immediate view of factors affecting performance, alongside an enhanced view of the content of the json file exported from Power BI Desktop Performance Analyzer.



Initially, there is a description of the action taken (we will see more about actions later) and the affected report and page.

Immediately, you will be able to see the **total rendering time in Power BI Desktop** for the page and action. This is important as this information is implied in the exported json file but is not shown out-of-the-box in Power BI Desktop performance analyzer.

One of the key aspects of this tool is that it benchmarks total render time and measurable affecting factors against certain thresholds. These thresholds (or “tolerance ranges”) can be configured (we will see how to do this) and can be interpreted using criteria defined by any organisation using this tool.

Important note about Tolerance Ranges (benchmarking thresholds).

One of the most important elements of this tool is the ability to benchmark any measurable event or factor, providing the organisation’s view of what “good” looks like and what doesn’t.

We need to be careful with these **tolerance ranges** though.

A certain amount of time to fully render a report, let’s say for example 8.3 seconds might be good or might be bad. It would be bad if it only shows a couple of visuals with basic information but might be good if it is showing the result of complex calculations, run over millions of records, that provide relevant information for decision taking. And the problem is that we cannot model that in a tool such this, therefore, what are these tolerance ranges for?

They are there so organisations can draw a line in the sand and show what is the expectation for report render time (and other factors) that can be applied to most common cases. Also, they can be defined based on final user experience: waiting 45 seconds for a report page to render will generally provide a bad experience with a Power BI Report, despite the user “can understand and accept” that there is complexity behind that report, that results in long computations and user waiting time.

The tolerance ranges provided alongside this tool (which can be configured) are a fist attempt to set boundaries. Based on your organisation’s requirements, you might want to alter them.

Differences between Power BI Service and Power BI Desktop.

Now is the time to talk about Power BI Desktop and Power BI Service. As far as I understand, Power BI Desktop doesn’t behave in exactly the same way as Power BI Service. However, query execution time will not differ (much) if the data source is hosted somewhere else (different PBI Dataset, AAS cube, etc) and data is not imported in the same Power BI Report being analysed.

We could consider Power BI Desktop as best-case scenario, and parallelism (or lack of thereof) will be similar in both (Service and Desktop).

Also, other factors such as number of visuals and queries (and their precedence) will not change.

My point is that rendering a report in Power BI Service might not be identical as doing so in Desktop, however this tool can be used to improve both.

## How to read the results of the tool.

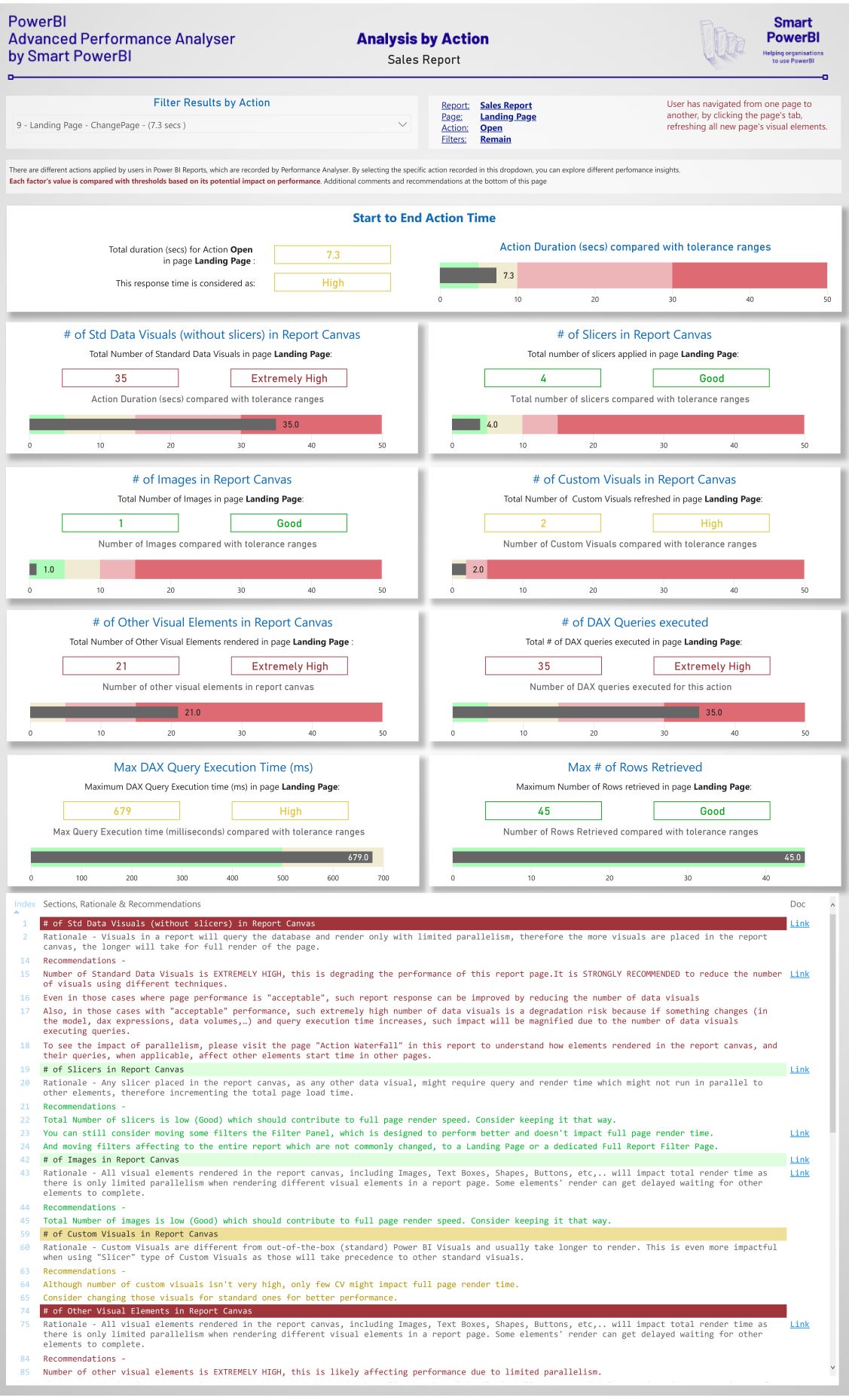
Let’s get into interesting details, by reviewing each page of the report in some depth.

### Analysis by Action Page.

Rendering a Power BI report page implies rendering all the visual elements (data visuals or otherwise), and neither, Desktop or Service can render everything in parallel – this is a **key concept**.

Furthermore, all queries required and run by a page refresh, cannot run in parallel (limited by both, Power BI and the data source).

With this page we can see what the total render time for the action is taken (as explained before) and also includes all the different factors that can be analysed (from the JSON file contents) and can affect performance. All those factors are benchmarked to easily see what could be improved to provide better performance.



**# of Std Data Visuals (without slicers) in Report Canvas**.

Visuals in a report will query the database and render only with **limited parallelism**, therefore the more visuals are placed in the report canvas, the longer will take for full render of the page.

**# of Slicers in Report Canvas**.

Any slicer placed in the report canvas, as any other data visual, might require query and render time which might not run in parallel to other elements, therefore incrementing the total page load time.

It is also worth mentioning that too many slicers in the report canvas leave less “real state” for actual insights.

**# of Images in Report Canvas**.

All visual elements rendered in the report canvas, including Images, Text Boxes, Shapes, Buttons, etc, will impact total render time as there is only limited parallelism when rendering different visual elements in a report page. Some elements' render can get delayed waiting for other elements to complete.

**# of Custom Visuals in Report Canvas**.

Custom Visuals are different from out-of-the-box (standard) Power BI Visuals and usually take longer to render. This is even more impactful when using "Slicer" type of Custom Visuals as those will take precedence to other standard visuals.

**# of Other Visual Elements in Report Canvas**.

All visual elements rendered in the report canvas, including Images, Text Boxes, Shapes, Buttons, etc,.. will impact total render time as there is only limited parallelism when rendering different visual elements in a report page. Some elements' render can get delayed waiting for other elements to complete.

**# of DAX Queries executed**.

Number of DAX Queries tends to the same as number of visuals (standard, custom and slicers). Report page will render visuals only with certain parallelism, but also, whatever backend is using this Power BI report might have limitations in terms of query parallelism or individual queries might require more time to execute due to larger number of queries being sent to the backend at the same time, therefore affecting performance for two reasons: 1.- multiple visuals rendering & 2.- slower individual query execution due to multiple queries. Also is worth noting backend is more likely to struggle if user concurrency is high (in other words, many users executing many queries at the same time will experience more performance degradation).

**Max DAX Query Execution Time (ms)**.

Query execution time is an important contributor to full page load, therefore there is potential for performance improvement by optimising measures definition, data model and visual design.

DAX Queries can be slow for different reasons, depending on the backend and the connection mode.

For SSAS Tabular cube backends (either Azure Analysis Services or Power BI Service), some of the contributing factors can be:

* Model design: many to many relationships, bi-directional relationships, business keys instead of integer (surrogated) keys, data types,..
* Suboptimal table design: data types, degenerate dimensions, unneeded columns, lack of partitions, to mention a few…
* Expensive DAX on-the-fly calculated measures (inefficient or not).
* Data volumes
* Azure Analysis Services - insufficient AAS tier.
* Use of TOPN, RANX functions or filters using those functions.
* Other…

Obviously, this tool doesn’t provide insights on **how** to improve query execution time. This is mainly driven by data volumes and source system. For those reports using a Tabular model (either Azure Analysis Services cube or Power BI dataset – in fact, the same thing but in different places), you can analyse improvements in your model using Tabular Editor best practices analyser by Michael Kovalsky (Senior Programme Manager at Microsoft). If you want to do so, please refer to these resources.

* [Article](https://powerbi.microsoft.com/en-ca/blog/best-practice-rules-to-improve-your-models-performance/) by Michael Kovalsky himself, although you might want to visit his blog at [ElegantBI](https://www.elegantbi.com/).
* [Video](https://www.youtube.com/watch?v=1Qan0_VmZRw) by Patrick Leblanc about Tabular Editor best practices analyser.

**Max # of Rows Retrieved**.

Queries fetching high number of rows (and rendering high number of data points) might also impact performance.

Recommendations (with links to best practices and guidelines).

For each action taken in the Power BI Report, the total render time and affecting factors are captured as explained the earlier sections of this article.

Now that the magnitude of those affecting factors has been measured and benchmarked, the tool is able to explain in more detail what is going on and what to do with this information.

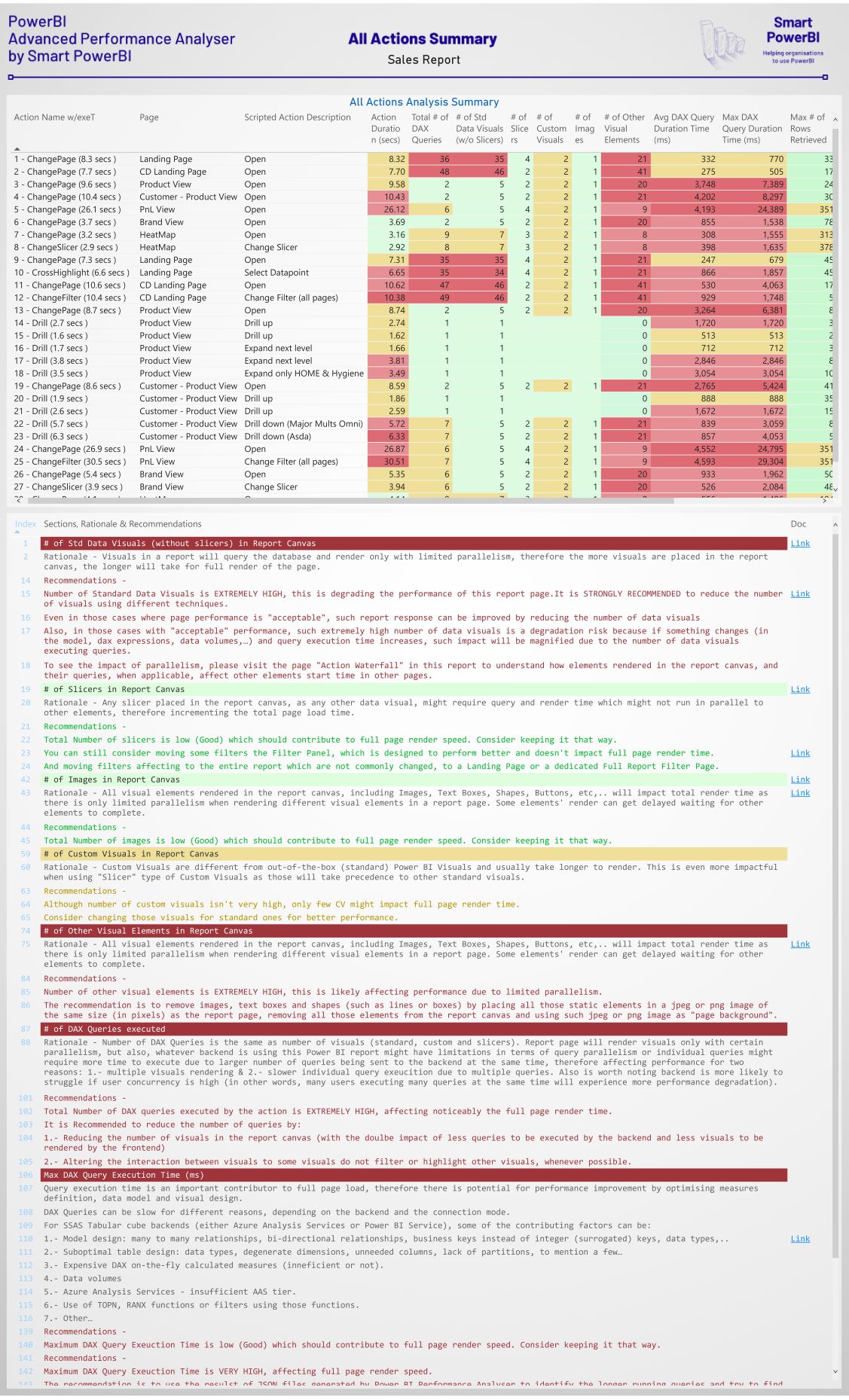


The resulting report includes a formatted table that covers in more detail each section:

* Section Title, with background colour depending on the benchmark result (tolerance ranges)
* A rationale (as explained earlier) of how this factor can affect Power BI performance. This is not coloured as rationale is always the same, regardless the tolerance range.
* Set of recommendations, depending on the tolerance range that this factor’s magnitude falls in, coloured as per such range.
  + The text for these recommendations is configurable.
* Configurable link, associated to any section title, rationale or recommendation line, that can link to either, internal or public resources. This link is meant to provide further information about the recommendation (best practices and guidelines, examples, etc,).

### All Actions Summary Page.

Once you get used to the information displayed in the page that shows results for one single action taken on the analysed report, you can see all results for all actions in one single page.



This page shows recommendations for worst case scenario (tolerance ranges) for all actions, however **if any particular action is selected in the first table** with all actions, the Recommendations section will display recommendations, based on tolerance ranges for that particular action.

### Kicker bonus! Action Gannt and parallelism.

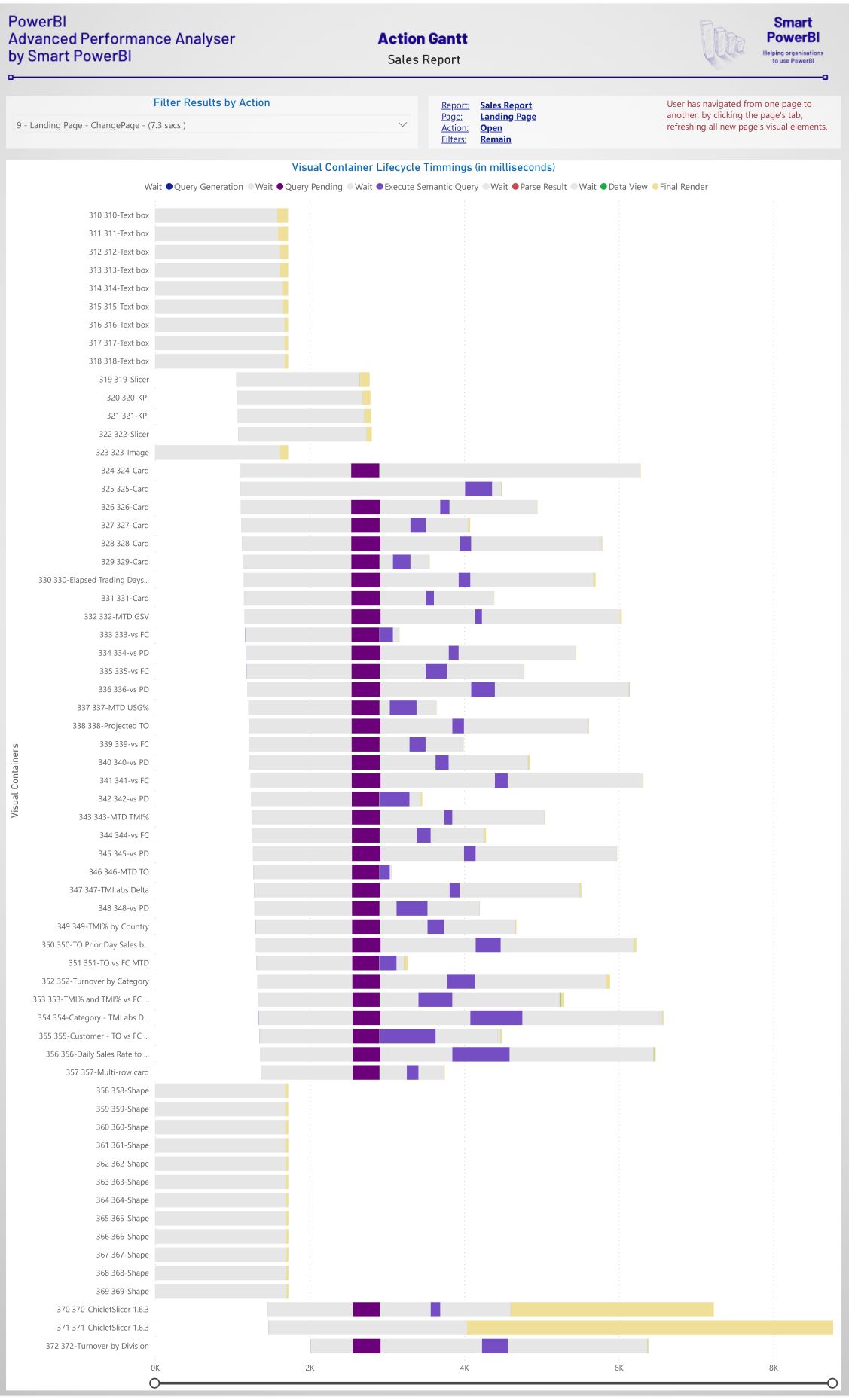
I have been rumbling about limited parallelism in various sections of this article. Let’s bring it to life with a Gantt chart that shows when different visual components are running different activities and how they impact each other.

There are two points worth mentioning:

* The timings used for this chart all come from the json file exported from PBI Desktop performance analysis session, but out-of-the-box functionality fails to show clear enough who and when different elements are being rendered.
* This is, by far, the most powerful tool that I have to convince others that too many visual elements in any report is no good for business (and performance).

In the following chart, we can see all visual elements (including non-data ones, such as images and shapes) go through the process of

* Initial wait
* Query Generation (only applied to data visuals, and generally so fast that comes as a very thin line).
* Query Pending, where the query is not executed yet.
* Execute Semantic Query & Parse result, which represents the time that takes the backend to resolve the query and send the results over to power bi.
* Data View & Final Render, which roughly represents the time taken by Power BI to shape the resultset and render the appropriate data points in the visual.



Although I don’t see much value scrutinising rendering dependencies and why some visuals take more than 6 seconds to start any query or render activity (see the last 8 visuals in the chart), and why some queries seem to wait for other queries to complete, this visual makes a solid statement: **the more visual components you have in your report canvas, the more visuals and actions within visual query and render will have to wait for other things to complete**.

# Part 3 - Guide to download and use “Advanced Performance Analyzer by SmartPowerBI”.

## Initial considerations.

Advanced Performance Analyser is easy to use, although some details must be taken carefully in consideration:

Connect to a cloud data source.

Let me be careful with this consideration.

Query resolution time will affect to end-to-end report page render. If our backend is data imported in Power BI Desktop, it will be our local computer resolving the queries, which will likely be different from the final destination of the data source.

Ideally, the Power BI Report on which you are running Performance Analyzer will not have data imported in it, but instead will be connecting to a Power BI Datasource already published in app.powerbi.com. Also could be an Azure Analysis Services cube or any other database engine.

The point is that the report that you are analysing should be connected to a backend that will perform similar to production, otherwise results associated with query execution time will be skewed.

If your model and data is already contained in the pbix file that we want to analyse, watch this [video](https://www.youtube.com/watch?v=PlrtBm9YN_Q) from on to easily split the report and the model (with the data), so you have a pbix file with only the report pages connected to a cloud source.

If you need **more reasons** to split models and reports for **all** your PBI products, there is already plenty of articles and videos explaining why, some of them listed here:

* Comprehensive [article](https://www.coatesdatastrategies.com/blog/5-tips-for-separating-power-bi-datasets-and-reports) from Coates Data Strategies (much more in their [website](https://www.coatesdatastrategies.com/)).
* Video from Patrick Leblanc
* Official Microsoft [Documentation](https://docs.microsoft.com/en-us/power-bi/guidance/report-separate-from-model).

What happens when I have a direct query connection (i.e. SQL Server RDBMS)? Well, the tool should work as well. The main point here is to “not to have the data and model resolving queries using your computer’s hardware”, for more accurate query execution time.

Always clear the cache and start from a blank page.

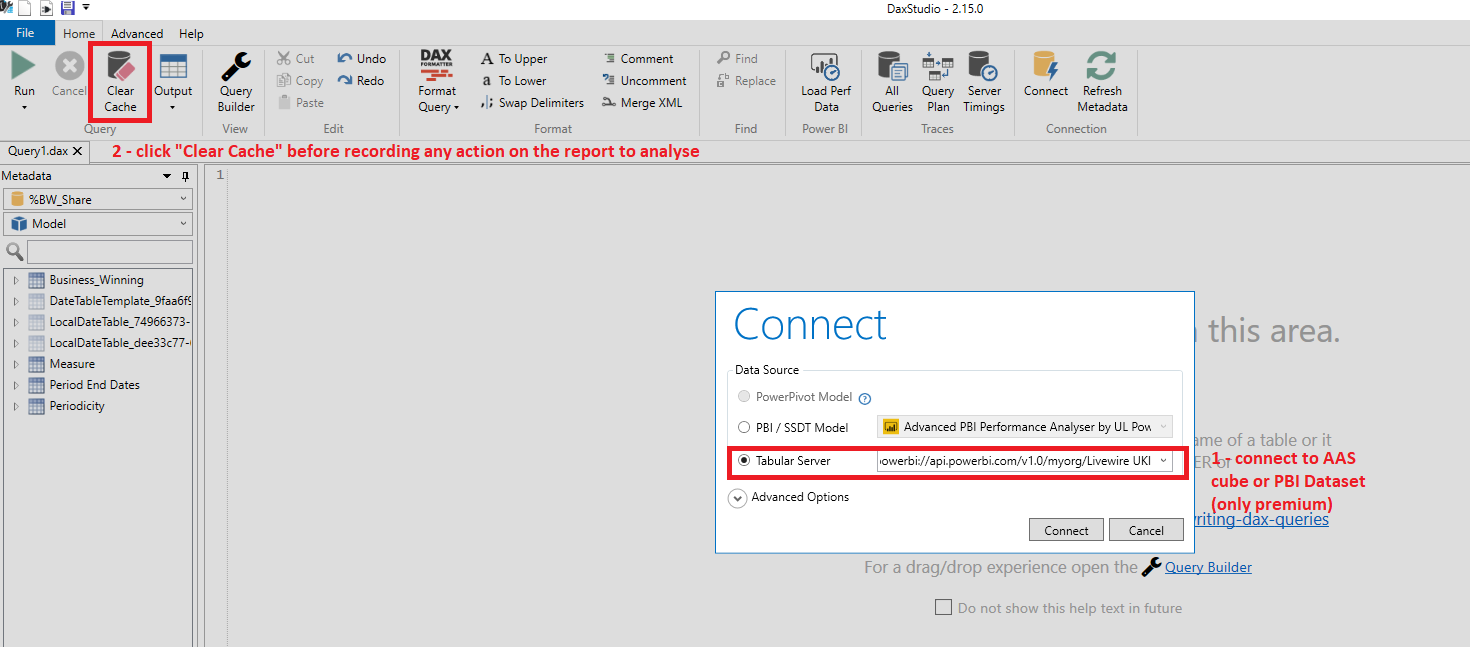
When we run performance analyser, to avoid executing queries that can be cached and therefore skew the query execution time results, we should create a blank page and save and close the report with that page active. Then the performance recording should happen after we open the pbix file again and the starting point is such blank page.

Clear cache at start and between any two consecutive interactions in a report.

Cache will likely skew results (or not), but sometimes will be hard to tell when a query resolution is helped by cache. Therefore, it is always a good idea, to get consistent results that represent the users’ experience, to clear cache before we start the performance analysis session.

If you are using DAX Studio to connect to your AAS cube or PBI Dataset (in Premium), clearing cache is simple as shown in the image below.

If your dataset is not yet (or won’t ever be) in premium, you can use Premium Per User to connect to the PBI Dataset.



Carefully click the report to execute the actions that we want to record.

Seems obvious, but when Power BI Desktop is running performance analyser, reacting to user interactions might come with a lag. It is important to click carefully (i.e. clicking on a different report page) and wait until Power BI Desktop starts recording the full render of the page.

Wait until all visuals have been rendered.

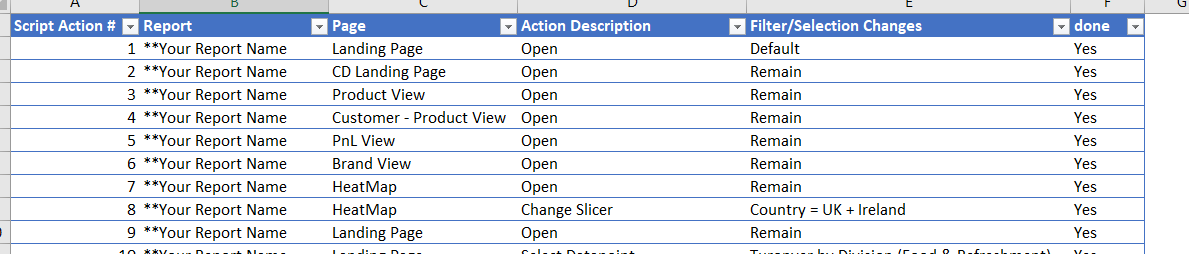
We will have to wait and carefully see, that all visuals have been rendered, and the results in the “Performance Analyzer” pane has populated all the visuals timings.

If there are still any “spinning wheels” in the “Duration (ms)” column, that means that the page is not fully rendered. We must wait for all “Duration (ms)” fields to contain a number.

|  |  |
| --- | --- |
|  |  |

Carefully record the action in the “Action Template File”.

To enrich the resulting report, we need to feed the accompanying excel template named “Action Template File”, where we will record a new row, in order, with the report name (always the same), the page name and the action description and filters applied. The “done” field is to track the actions when we repeat the recording, and it is not used by the resulting report.



## Step by step guide for capturing performance analysis.

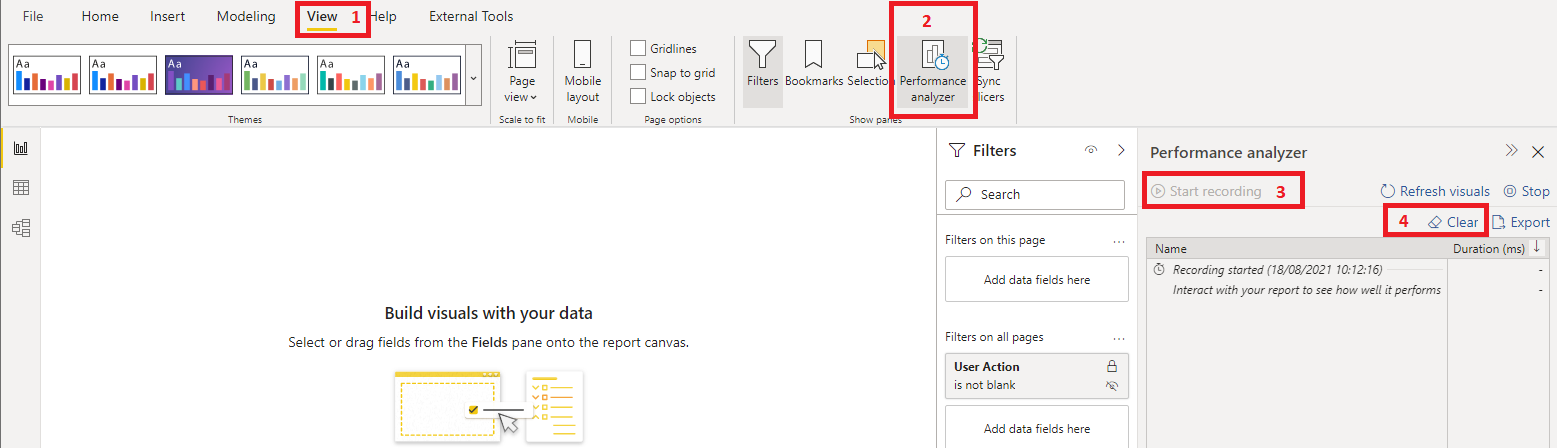
This is where all the magic happens! And takes only a few minutes…

**1 Open the report to analyse, starting from the blank page.**

As mentioned in the section above, start with a blank page so we don’t run any query before we start performance analysis recording.

**2 Start Performance Analysis Recording**.

In the “View” ribbon (1), select “Performance Analyzer” (2), then in the “Performance Analyzer pane”, click “Start recording” (3) and finally (although optional), click “Clear” to remove the first message.



**3 Clear Cache.**

To make sure that we start with a clear cache, use DAX Studio or any other suitable tool to perform a clear cache action on the dataset (AAS Cube or Power BI Dataset), as explained in the section above.

**4 Move from the Blank page to the first page of the report**.

Once we click in our report (change to another page, select a data point,..), performance analyser will populate its panel with all visuals timings, and as some visuals will take some time, instead of seeing a number in the “Duration (ms) column”, we will see a “spinning wheel”. Once all visuals contain a number, we can continue.

**5 Record the action in the “Action Template File”**.

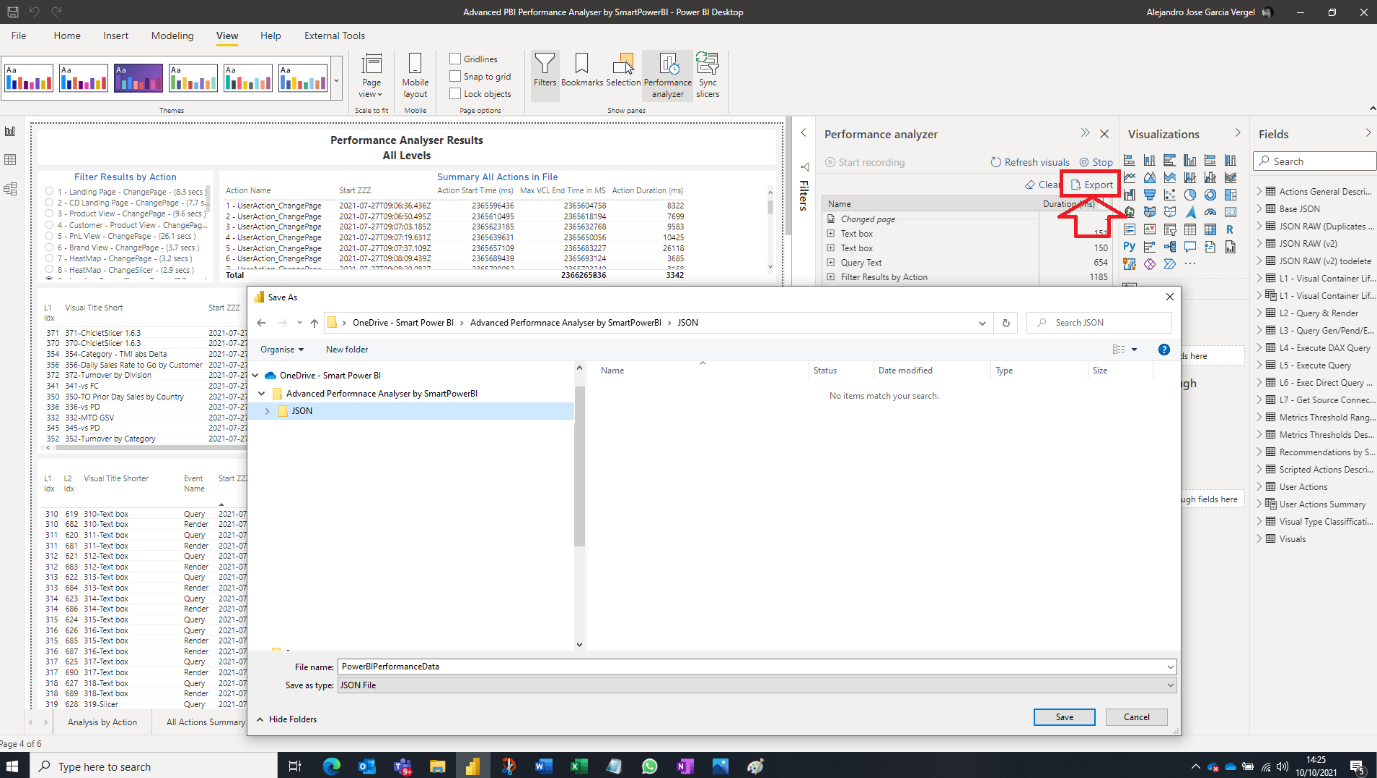
As explained before, recording in this excel file all the actions that we take on the report to analyse, will enrich the report to be more descriptive and readable content.

**6 More actions**.

Once we have completed the first action, we can repeat the process from step 3 (Clear cache), as many pages and/or actions we want to record.

**7 Export the json file**.

Last, but not least, once we have completed all the actions that we want to record, we must export the performance analyser results to a JSON file, which we will use to feed the “Advanced PBI Performance Analyser by SmartPowerBI” tool.

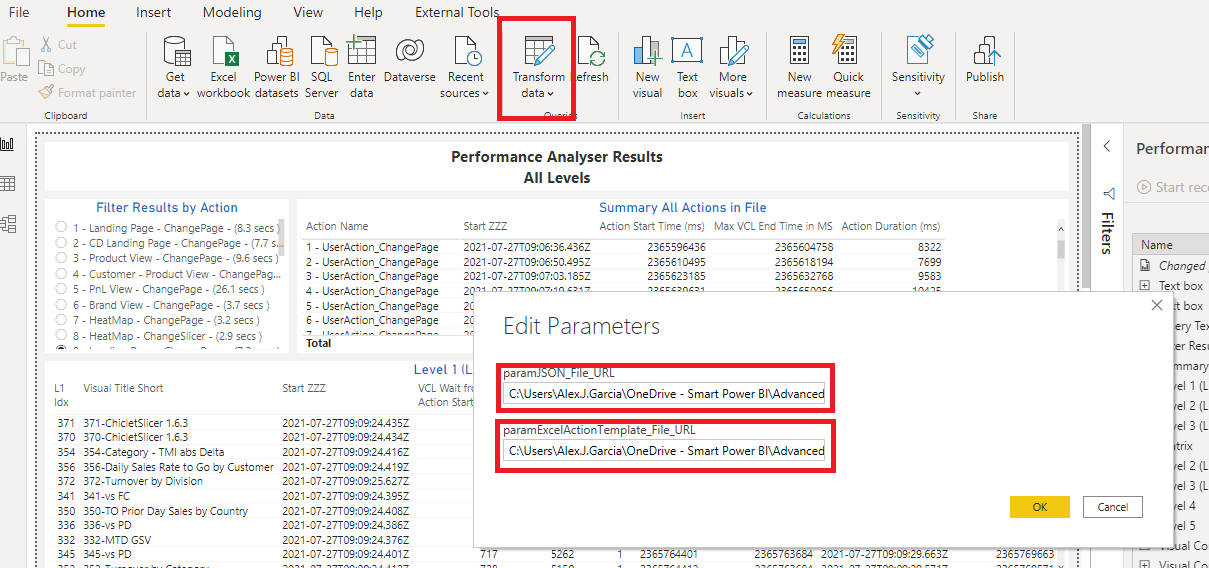


**8 Feed the “Advanced PBI Performance Analyser by SmartPowerBI” tool with the results**.

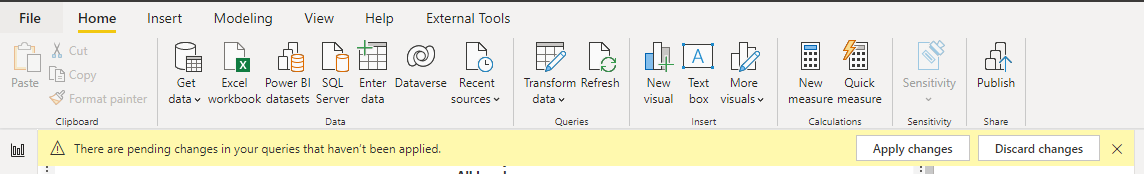
As result of this performance analysis session, we have two files:

* JSON file from PBI Desktop Performance Analyzer, with the recording of all the actions that we took on the report.
* Excel file, as “Action Template File”, with the description of such actions.

After opening the tool **“Advanced PBI Performance Analyser by SmartPowerBI”**, we must click on “Transform Data” -> “Edit Parameters” and replace the existing values with the two files location, as shown in the image below.



Once that’s done, Power BI Desktop will prompt us with “Apply Changes”, which we need to click to force the ingestion of the newly provided files.



With the Power BI Desktop version used at the time of this writing, sometimes “Apply Changes” get stuck and needs to be cancelled and clicked again.

## Download Advanced Power BI Performance Analyser.

Time has come for you to use the tool and get on with it. For that you’ll need to download and use the following files.

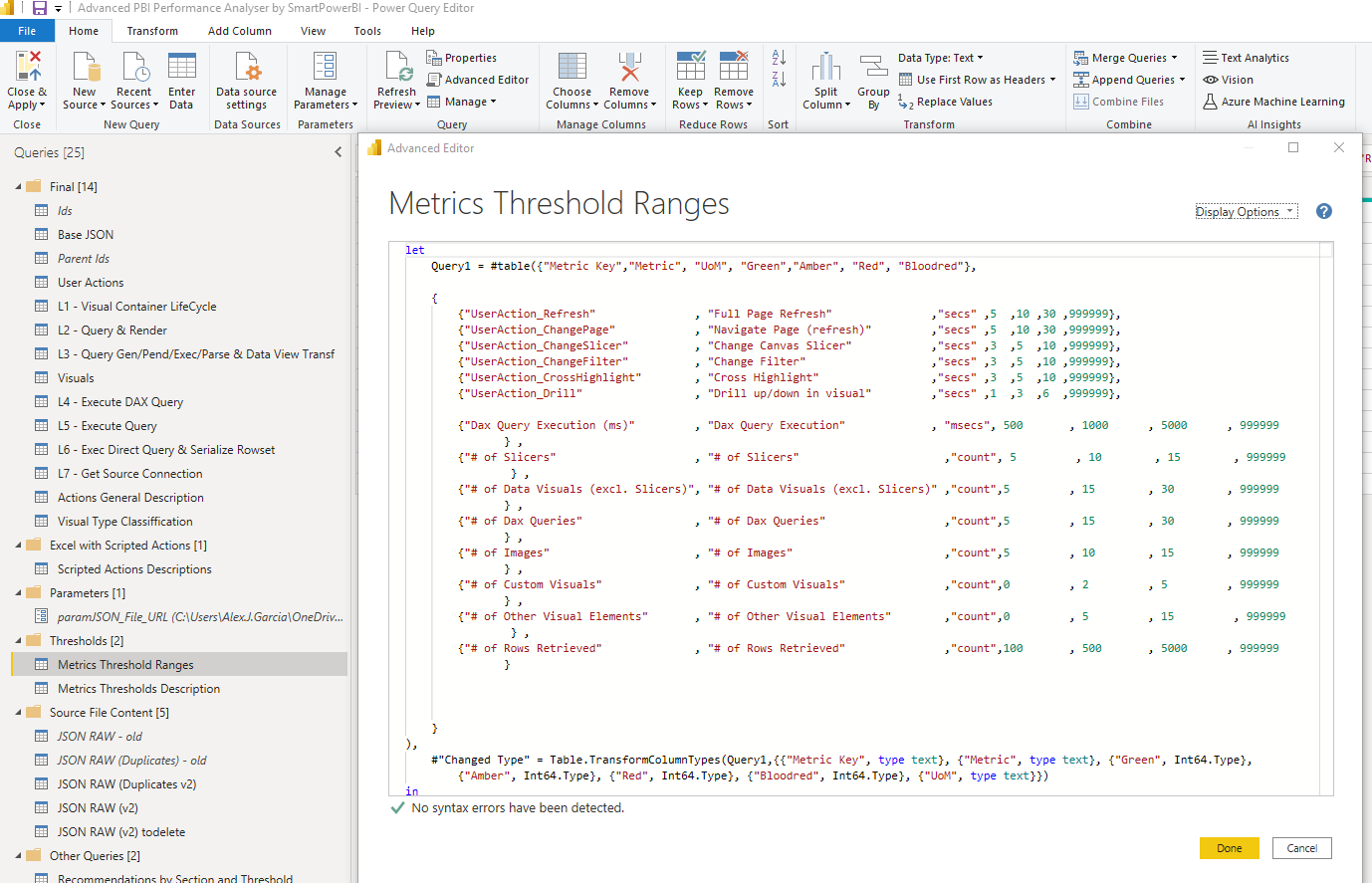
* PBIX file named “Advanced PBI Performance Analyser by SmartPowerBI”, that contains the connection to the JSON file (and others), loads, transforms, and models the data within it and presents the results in several pages.
* Excel file named “Advanced Performance Analyser by SmartPowerBI Support File”, which contains the rationale, recommendations, and links to other resources.
* Excel file named “Advanced Performance Analyser by SmartPowerBI Actions Script” which you will use to capture all the actions taken on the report subject to analysis that help enriching the results report, as will contain information that is not present in the original json file.

Disclaimer: this solution is provided “as is”, under MIT License, and it has been tested to a limited degree. There is no warrantee that solution will work if Microsoft decides to modify the json file format when exporting from PBI Desktop’s performance analyzer, and there is no warrantee that it will work in untested scenarios.

## How To’s.

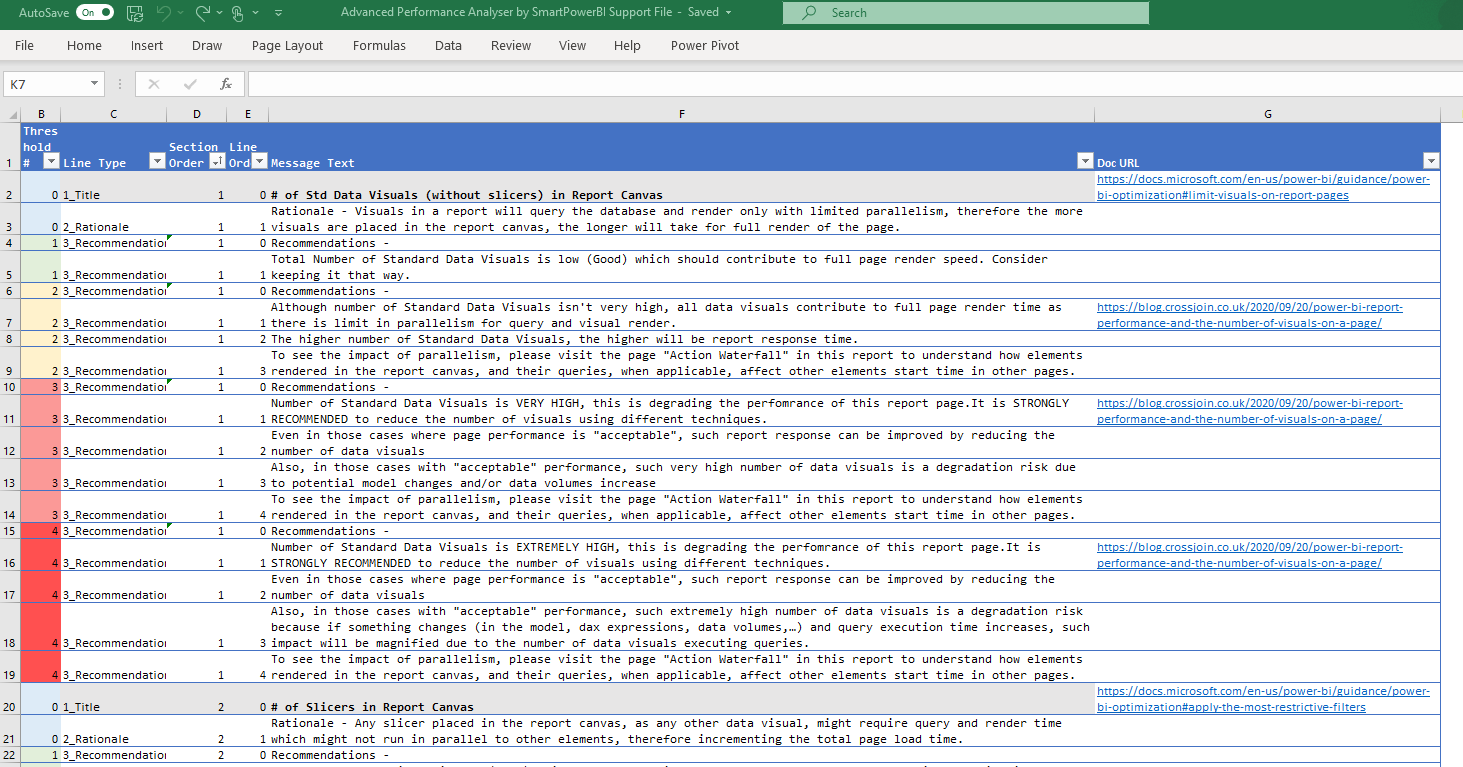
### How to configure different thresholds.

In the pbix file “Advanced PBI Performance Analyser by SmartPowerBI”, open PowerQuery and edit the query named “Metrics Threshold Ranges” with advanced editor and all the different threshold ranges used in the report are there for you to modify.



### How to configure different rationale, recommendations and links to best practices and guidelines.

In the Excel file named “Advanced Performance Analyser by SmartPowerBI Support File”, there is a table with all the affecting factors described in this article and analysed in the resulting report. In such table you can configure the text to be displayed, not only for the rationale but also for the different recommendations for each tolerance range (1-4: Good, High, Very High, Extremely High).



## Credits.

In some parts of this series of articles and in the downloadable files, I have used and referenced articles or other resources authored by:

* Patrick Leblanc ([Guy in a Cube (Power BI Training and More)](https://guyinacube.com/))
* Melissa Coates ([Coates Data Strategies](https://www.coatesdatastrategies.com/))
* Marco Russo ([Home - SQLBI](https://www.sqlbi.com/))
* Reid Havens ([Havens Consulting](https://www.havensconsulting.net/))
* Will Thompson (Github repository)
* Sam McKay ([Enterprise DNA | Empowering Power BI Users to Change Their World](https://enterprisedna.co/))
* James Serra ([James Serra's Blog](https://www.jamesserra.com/))
* Michael Kovalsky ([ElegantBI](https://www.elegantbi.com/)).

And special thanks to [Steve Verschaeve](https://www.linkedin.com/in/steveverschaeve/), Senior Customer Engineer from Microsoft, for his thorough review of the documentation and great advice on the functionality.