**Optimization Guide for Power BI**

To produce and maintain optimized Power BI solutions. Optimising solution at different architectural layers include

* Data source
* Data model
* Visualizations Including dashboards, Power BI reports, and Power BI paginated reports)
* Environment (Including capacities, data gateways, and the network)

**Optimising the Data Model**

Choose the appropriate semantic model type for your solution. There are three semantic model modes:

* Import
* Direct Query
* Composite.

**Data reduction techniques for Import mode**

There are eight different data reduction techniques

1) Remove unnecessary columns

I recommend that you design models with exactly the right number of columns based on the known reporting requirements. Your requirements may change over time, but bear in mind that it's easier to add columns later than it's to remove them later. Removing columns can break reports or the model structure.

2) Remove unnecessary rows

Model tables should be loaded with as few rows as possible. It can be achieved by loading filtered row sets into model tables for two different reasons

For example, instead of loading sales facts for all sales regions, only load facts for a single region. This design approach will result in many smaller models, and it can also eliminate the need to define row-level security but will require granting specific semantic model permissions in the Power BI service and creating "duplicate" reports that connect to each semantic model.

I suggest you don't automatically load all available history, unless it's a known reporting requirement. It's helpful to understand that time-based Power Query filters can be parameterized, and even set to use relative time periods.

3) Group by and summarize

The most effective technique to reduce a model size is to load pre-summarized data.

Significant data reduction could be achieved by summarizing all sales metrics, grouping by date, customer, and product. even more significant data reduction could be achieved by grouping by date at month level.

4) Optimize column data types

The VertiPaq storage engine uses separate data structures for each column. for numeric column data, it uses value encoding for optimisation. for text and other non-numeric data, it uses hash encoding. It requires the storage engine to assign a numeric identifier to each unique text value contained in the column. Then it is stored in the data structure, requiring a hash lookup during storage and querying.

In some scenarios, you can convert source text data into numeric values.

For example, a sales order number may be consistently prefixed by a text value ("SO123456"). The prefix could be removed, and the order number value converted to whole number. For large tables, it can result in significant data reduction, especially when the column contains unique or high cardinality values.

In this example, Set the column Default Summarization property to "Do Not Summarize". It helps to minimize the inappropriate summarization of the order number values.

5) Preference for custom columns

The VertiPaq storage engine stores model calculated columns defined in DAX+. They're built once all Power Query tables are loaded, which can result in extended data refresh times.

It's less efficient to add table columns as calculated columns than Power Query custom columns.

6) Disable Power Query load

Queries in power query that are intended to support data integration with other queries shouldn't be loaded to the model. To avoid loading the query to the model, take care to ensure that you disable query load in these instances.

A screenshot of a computer

Description automatically generated

Make sure that to uncheck the **Enable load** option for the queries those are not required to load into the desktop which makes it slow to load.

7) Disable auto date/time

Power BI Desktop includes an option called Auto date/time. When enabled, it creates a hidden auto date/time table for date columns to support the report when configuring filters, grouping, and drill-down actions for calendar time periods. The hidden tables are in fact calculated tables that will increase the size of the model.

8) Switch to Mixed mode

In Power BI Desktop, a mixed mode design produces a Composite model. Essentially, it allows you to determine storage mode for each table.

An effective technique to reduce the model size is to set the Storage Mode property for larger fact-type tables to Direct Query. Consider that the design approach could work well with the Group by and summarize technique mentioned earlier.

**Data reduction techniques in Direct Query**

Complex PQ Queries means the tables which are generated by using Append as new or Merge as new options.

1) Avoid complex Power Query queries

An efficient model design can be achieved by removing the need for the queries in Power Query to apply any transformations. It means that each query maps to a single relational database source table or view.

2) Examine the use of calculated columns and data type changes

Direct Query models support adding calculations and Power Query steps to convert data types. However, better performance is often achieved by doing the transformations in the database source whenever possible.

3) Keep measures simple

At least initially, it's recommended to limit measures to simple aggregates. If the measures are sufficiently responsive but paying attention to the performance for each. The CALCULATE DAX function can be used to produce measures that manipulate filter context, they generate complex native queries that don't perform well.

4) Avoid relationships on calculated columns

Model relationships can only relate a single column in one table to a single column in a different table. Sometimes, it's necessary to relate tables by using multiple columns.

For example, the Sales and Geography tables are related by two columns: CountryRegion and City. To create a relationship between the tables, a single column is required, and in the geography table, the column must contain unique values. Concatenating the country/region and city with a hyphen separator could achieve this result.

5) Avoid relationships on "Unique Identifier" columns

Power BI doesn't natively support the unique identifier (GUID) data type. When defining a relationship between columns of this type, Power BI generates a source query with a join involving a cast. This query-time data conversion commonly results in poor performance. the only way is to changing columns to an alternative data type in the underlying database.

6) Hide the one-side column of relationships

The one-side column of a relationship should be hidden. It's usually the primary key column of dimension tables. When hidden, it isn't available in the Fields pane and so can't be used to configure a visual. The many-side column can remain visible if it is useful to group or filter reports by the column values.

For example, A model where a relationship exists between Sales and Product tables. The relationship columns contain product SKU (Stock-Keeping Unit) values. If product SKU must be added to visuals, it should be visible only in the Sales table. When this column is used to filter or group in a visual, Power BI generates a query that doesn't need to join the Sales and Product tables.

7) Avoid use of bi-directional relationship filtering

Use of bi-directional relationship filtering can lead to query statements that don't perform well. Only use this relationship feature when necessary, and it should be when implementing a many-to-many relationship across a bridging table.

**Optimize Report Designs**

1) Enable query reduction techniques

Power BI Desktop Options and Settings includes a Query Reduction page and it has three helpful options.

It's possible to disable cross-highlighting and cross-filtering by default, though it can be overridden by editing interactions.

It's also possible to show an Apply button on slicers and filters. The slicer or filter options won't be applied until the report user clicks the button.

2) Apply filters first

When first designing reports, apply any applicable filters—at report, page, or visual level—before mapping fields to the visual fields.

For example, rather than dragging in the CountryRegion and Sales measures, and then filtering by a particular year, apply the filter on the Year field first. It's because each step of building a visual will send a query, and while it's possible to then make another change before the first query has completed, it still places unnecessary load on the underlying data source. By applying filters early, it generally makes those intermediate queries less costly and faster. Also, failing to apply filters early can result in exceeding the 1 million-row limit that is allowed in Direct Query Mode.

3) Limit the number of visuals on a page

When a report page is opened all the visuals on a page are refreshed. However, there's a limit on the number of queries that can be sent in parallel. So, as the number of page visuals increases, there's higher chance that they'll be refreshed in a serial manner. It increases the time taken to refresh the entire page. Replacing multiple card visuals with a single multi-row card visual can achieve a similar page layout.

4) Switch off interaction between visuals

Cross-highlighting and cross-filtering interactions require queries be submitted to the underlying source. Unless these interactions are necessary, it's recommended they be switched off if the time taken to respond. These interactions can be switched off, either for the entire report, or on a case-by-case basis.

5) Measure filters

Visuals containing measures can have filters applied to those measures.

For example, Sales by Category, but only for categories with more than

$15 million of sales.

It may result in two queries being sent to the underlying source,

The first query will retrieve the categories meeting the condition

(Sales > $15 million)

The second query will then retrieve the necessary data for the visual,

Adding the categories that met the condition to the WHERE clause,

It generally performs fine if there are hundreds or thousands of categories.

Performance can degrade, however, if the number of categories is much larger

and indeed, the query will fail if there are more than 1 million row limit.

6) TopN filters

Advanced filters can be defined to filter on only the top (or bottom) N values ranked by a measure.

For example, to display only the top five categories in the above visual.it will also result in two queries being sent to the underlying data source. The first query will return all categories from the underlying source, and then the top N are determined based on the returned results. Depending on the cardinality of the column involved, it can lead to performance issues (or query failures due to the 1 million-row limit).

7) Median

Generally, any aggregation is pushed to the underlying source. it's not true for Median, as this aggregate isn't supported by the underlying source. In such cases, detail data is retrieved from the underlying source, and Power BI evaluates the median from the returned results. It's fine when the median is to be calculated over a relatively small number of results, but performance issues (or query failures due to the 1 million-row limit) will occur if the cardinality is large.

8) Multi-select slicers

Allowing multi-selection in slicers and filters can cause performance issues. It's because as the user selects additional slicer items. This situation can be avoided by showing the Apply button.

9) Visual totals

By default, tables and matrices display totals and subtotals. In many cases, additional queries must be sent to the underlying source to obtain the values for the totals. It applies whenever using Count Distinct or Median aggregates, and in all cases when using Direct Query over SAP HANA or SAP Business Warehouse. Such totals should be switched off (by using the Format pane) if not necessary.

**Convert to a Composite Model**

A composite model combines multiple source groups. A source group can represent imported data or a connection to a Direct Query source. A Direct Query source can be either a relational database or another tabular model.

1) Table storage mode

The benefits of Import and Direct Query models can be combined into a single model by configuring the storage mode of the model tables. The table storage mode can be Import or Direct Query, or both, known as Dual. When a model contains tables with different storage modes, it's known as a Composite model.

Set the storage mode to Direct Query when a table is a fact-type table storing large data volumes, or when it needs to deliver near real-time results.

Set the storage mode to Dual when a table is a dimension-type table, and it will be queried together with Direct Query or hybrid fact-type tables that are in the same source group.

Set appropriate refresh frequencies to keep the model cache for dual and hybrid tables (and any dependent calculated tables) in sync with the source databases.

Ensure data integrity across source groups and model cache because limited relationships will eliminate rows in query results when related column values don't match.

Whenever possible, optimize Direct Query data sources with appropriate indexes for efficient joins, filtering, and grouping.

2) User-defined aggregations

There are many functional and performance enhancements that can be achieved by converting a Direct Query model to a Composite model. A Composite model can integrate more than one Direct Query source, and it can also include aggregations. Aggregation tables can be added to Direct Query tables to import a summarized representation of the table. They can achieve dramatic performance enhancements when visuals query higher-level aggregates.

Its row count should be at least a factor of 10 smaller than the underlying table. For example, if the underlying table stores 1 billion rows, then the aggregation table shouldn't exceed 100 million rows. This rule ensures that there's an adequate performance gain relative to the cost of creating and maintaining the aggregation.

3) Cross source group relationships

Consider carefully that a composite model is the right solution while it allows model-level integration of different data sources, it also introduces design complexities with possible consequences. Set the storage mode to Direct Query when a table is a fact-type table storing large data volumes, or when it needs to deliver near real-time results.

For best performance, it is recommended to keep the relationship columns be low cardinality, meaning they should store less than 50,000 unique values. This recommendation is especially true when combining tabular models, and for non-text columns.

4) Calculated columns

Consider specific limitations when adding calculated columns and calculation groups to a composite model.

Calculated columns added to a Direct Query table that source their data from a relational database, like Microsoft SQL Server, are limited to expressions that operate on a single row at a time. These expressions can't use DAX iterator functions, like SUMX, or filter context modification functions, like CALCULATE.

5) Model design

You should always optimize a Power BI model by adopting a star schema design.

Composite models allow a report to have two or more data connections from different source groups. These source groups can be one or more Direct Query connections and an import connection, two or more Direct Query connections, or any combination.

With composite models, you can establish many-to-many relationships between tables. This approach removes requirements for unique values in tables. It also removes introducing new tables only to establish relationships.

6) Row-level security

If your model includes user-defined aggregations, calculated columns on import tables, or calculated tables, ensure that any row-level security (RLS) is set up correctly and tested.

If the composite model connects to other tabular models, RLS rules are only applied on the source group (On Premise) where they're defined. They won't be applied to other source groups (remote models). Also, you can't define RLS rules on a table from another source group nor can you define RLS rules on a local table that has a relationship to another source group.

7) Report design

In some situations, you can improve the performance of a composite model by designing an optimized report layout.

8) Single source group visuals

Whenever possible, create visuals that use fields from a single source group. That's because queries generated by visuals will perform better when the result is retrieved from a single source group. consider creating two visuals positioned side by side that retrieve data from two different source groups.

9) Use sync slicers

In some situations, you can set up sync slicers to avoid creating a cross-source group relationship in your model. It can allow you to combine source groups visually that can perform better.

**Optimizing visualizations**

Power BI visualizations can be dashboards, Power BI reports, or Power BI paginated reports. Each has different architectures.

**Dashboards**

It's important to understand that Power BI maintains a cache for your dashboard tiles except live report tiles, and streaming tiles. If your semantic model enforces dynamic row-level security (RLS), be sure to understand performance implications as tiles will cache on a per-user basis.

Retrieving the data from the cache provides better and more consistent performance than relying on the data source. One way to take advantage of this functionality is to have dashboards be the first landing page for your users.

For Direct Query and live connection semantic models, the cache is updated on a periodic basis by querying the data source. By default, it happens every hour, though you can configure a different frequency in the semantic model settings.

Each cache update will send queries to the underlying data source to update the cache. The number of queries that generate depends on the number of visuals pinned to dashboards that rely on the data source.

**Power BI reports**

There are several ways for optimizing Power BI report designs.

1) Apply the most restrictive filters

The more data that a visual needs to display, the slower that visual is to load.

Use the "Top N" filter to reduce the max number of items that the table displays. You can set the max item to larger than what users would need.

2) Limit visuals on report pages

the filtering principle applies equally to the number of visuals added to a report page. It's highly recommended you limit the number of visuals on a particular report page to only what is necessary. Drill through pages and report page tooltips are great ways to provide additional details without jamming more visuals onto the page.

3) Evaluate custom visual performance

Be sure to put each custom visual through its paces to ensure high performance. Poorly optimized Power BI visuals can negatively affect the performance of the entire report.

**Power BI paginated reports**

Paginated reports natively support both cloud-based or on-premises.

On-premises data sources whether hosted on-premises or in a virtual machine requires a data gateway for Power BI to connect. Cloud-based means that Power BI can connect directly using an Internet connection.

1) Relational data sources

Generally, Relational data sources can define stored procedures. They're also suited for reports that need to retrieve very large datasets.

Stored Procedure can benefit like

* Parameterization
* Encapsulation of programming logic, allowing for more complex data preparation
* Improved maintainability, allowing stored procedure logic to be easily updated. In some cases, it can be done without the need to modify and republish paginated reports.
* For better performance the execution plans are cached and stored procedures across multiple reports can be reused.

2) Analytic data sources.

Analytic data sources also known as data models or models or semantic models.

Use the DAX providing it entirely meets your query needs. If the model doesn't

define the measures you need, switch to query mode.

The MDX query designer requires your model to include measures. The designer has two capabilities not supported by the DAX.

* Define query-level calculated members.
* Configure data regions to request server aggregates in non-detail groups. If your report needs to present summaries of semi- or non-additive measures (like time intelligence calculations, or distinct counts), it will likely be more efficient to use server aggregates than to retrieve low-level detail rows and have the report compute summarizations.

3) Query result size

In general, it's best practice to retrieve only the data required by your report. So, don't retrieve columns or rows that aren't required by the report.

To limit rows, you should always apply the most restrictive filters, and define aggregate queries. Aggregate queries group and summarize source data to retrieve higher-grain results.

4) Expression-based fields

Report performance is improved because there's no need for Power BI to materialize calculated fields prior to report rendering. Calculated fields can noticeably extend report render time when datasets retrieve a large number of rows.

5) Field names

first verify that all field names are friendly, concise, yet still meaningful. If not, rename them before you commence the report layout. It's because renamed fields don't ripple changes through to the expressions used in your report layout. If you do decide to rename fields after you've commenced the report layout, you'll need to find and update all broken expressions.

6) Filter vs parameter

It's likely that your paginated report designs will have report parameters. Report parameters are commonly used to prompt your report user to filter the report. As a paginated report author, you have two ways to achieve report filtering.

* A dataset filter, in which case the report parameter values are used to filter the data already retrieved by the dataset.

Dataset filtering when you anticipate a different subset of the dataset rows will be reused many times, therefore saving rendering time because new data doesn't need to be retrieved.

* A dataset parameter, in which case the report parameter values are injected into the native query sent to the data source.

Dataset parameterization when you anticipate it's unlikely that a different subset of dataset rows will be requested or, when the number of the dataset rows to be filtered is likely to be very large.

7) Non-native data sources

If you need to develop paginated reports based on data sources that aren't natively supported by paginated reports, you should first develop a data model in Power BI Desktop. That way, you can connect to hundreds of data sources supported by Power BI. Once published to the Power BI service, you can then develop a paginated report that connects to the Power BI semantic model.

8) Data integration

If you need to combine data from multiple data sources,

* Combine report datasets, If the data sources are natively supported by paginated reports, you can consider creating calculated fields that use the Lookup or Lookup Set Report Builder functions.
* Develop a Power BI Desktop model, it’s likely more efficient, however, that you develop a data model in Power BI Desktop. You can use Power Query to combine queries based on any supported data source. Once published to the Power BI service, you can then develop a paginated report that connects to the Power BI semantic model.

9) Redundant data retrieval

It's possible your report retrieves redundant data. This can happen when you delete dataset query fields, or the report has unused datasets. Avoid these situations, as they result in an unnecessary burden on your data sources, the network, and Power BI capacity resources.

10) Deleted query fields

On the Fields page of the Dataset Properties window, it's possible to delete dataset query fields (query fields map to columns retrieved by the dataset query). However, Report Builder doesn't remove corresponding columns from the dataset query.

If you need to delete query fields from your dataset, we recommend you remove the corresponding columns from the dataset query. Report Builder will automatically remove any redundant query fields. If you do happen to delete query fields, be sure to also modify the dataset query statement to remove the columns.

11) Unused datasets

When a report is run, all datasets are evaluated even if they're not bound to report objects.

For this reason, be sure to remove any test or development datasets before you publish a report.

**Optimizing the environment**

You can optimize the Power BI environment by configuring capacity settings, sizing data gateways, and reducing network latency.

**Capacity settings**

When using capacities available with Power BI Premium (P SKUs), Premium Per User (PPU) licenses, or Power BI Embedded (A SKUs, A4-A6) you can manage capacity settings.

1) Get to the capacity settings

* To get to the capacity settings, follow these steps:
* In the Power BI service, select the gear icon, and then select Admin portal.
* In the Admin portal, select Capacity settings.

2) View your capacity

The capacity settings page shows a list of all the capacities in your tenant. At the top of the page, you can see a list of the different Fabric capacity types. Select a capacity type to view all the capacities of that type in your tenant.

**Power BI Premium**

A capacity that was bought as part of a Power BI Premium subscription. These

capacities use P SKUs.

Manage your capacity

This section lists basic capacity management tasks, such as creating a new capacity, changing a capacity's name and deleting a capacity.

Create a new capacity

To create a new Power BI Premium capacity,

* In the Capacity settings page, select Power BI Premium.
* Select Set up new capacity.

In the Set up a new capacity page,

* Capacity name

Give your capacity a name.

* Capacity admins

Add capacity admins.

* Region

Select the region you want to create the capacity in.

* Available v-cores

Select the number of v-cores you want to use for the capacity.

* Capacity size

Select the size of the capacity.

**Power BI Embedded**

A capacity that was bought as part of a Power BI Embedded subscription. These capacities use A SKUs.

You can create a new Power BI Embedded capacity with an A SKU or an EM SKU.

* To create a new Power BI Embedded capacity with an A SKU,

Log into Azure and search for Power BI Embedded.

* Select Create.

In the Basics tab,

* Subscription

Select the Azure subscription you want to use for the capacity.

* Resource group

Select the Azure resource group you want to use for the capacity.

* Resource name

Give your capacity a name.

* Location

Select the region you want to create the capacity in.

* Size

Select the size of the capacity.

* Power BI capacity administrator

Select the capacity admins.

* Select Review + create.

Review the details of your capacity, and then select Create.

To create a new **Power BI Embedded** with an **EM SKU**,

In the Capacity settings page, select Power BI Premium.

* Select Set up new capacity.

In the Set up a new capacity page,

* Capacity name

Give your capacity a name.

* Capacity admins

Add capacity admins.

* Region

Select the region you want to create the capacity in.

* Available v-cores

Select the number of v-cores you want to use for the capacity.

Capacity size - Select one of these sizes

EM1 - 1 v-core

EM2 - 2 v-cores

EM3 - 4 v-cores

* Select Create.

**Trial** capacity

A Microsoft Fabric Trial capacity. These capacities use **Trial SKUs**.

To create a Fabric Trial capacity

* Open the Fabric homepage and select the Account manager.
* In the Account manager, select Start trial. If you don't see the Start trial button, trials might be disabled for your tenant.
* If prompted, agree to the terms and then select Start trial.
* Once your trial capacity is ready, you receive a confirmation message.

Select Got it to begin working in Fabric. You're now the Capacity administrator for that trial capacity.

* Change the name of your capacity

You can't change a trial capacity's name.

* Add and remove admins

A trial capacity is assigned to the user who signed up for the trial.

You can't add or remove admins to a Trial capacity.

* Resize a capacity

You can't resize a trial capacity.

* Delete a capacity

When you delete a capacity,

the workspaces associated with the capacity are retained for seven days before they're deleted.

If you have defined a workspace retention policy your workspaces are retained for the period specified.

Ongoing operations are stopped or cancelled and scheduled operations are cancelled.

To delete a trial capacity, you need to cancel the trial.

* Auto scale

Auto scale isn't available for Trial capacities.

* Capacity settings

After selecting a capacity, you can control its settings from,

* Details

Capacity details are settings that are specific to the capacity.

* Delegated tenant settings

Tenant settings are delegated by Fabric admins to be managed by capacity admins. Changes to these settings only affect the capacity the changes are made in.

Delegated tenant settings are available for Power BI Premium and Fabric capacities.

**Fabric Capacity**

To create a new Fabric capacity,

* In the Capacity settings page, select Fabric capacity.

Select the link Set up a new capacity in Azure. The Create Fabric capacity page in Azure, opens in a new tab.

In the Azure Create Fabric capacity page,

* Subscription

Select the Azure subscription you want to use for the capacity.

* Resource group

Select the Azure resource group you want to use for the capacity.

* Capacity name

Give your capacity a name.

* Region

Select the region you want to create the capacity in.

* Size

Select the size of the capacity.

* Fabric capacity administrator

Select the capacity admins.

* Select Review + create.

Review the details of your capacity, and then select Create.

* Change the name of your capacity

You can't change a Fabric capacity's name.

* Add and remove admins

To add or remove admins in a Fabric capacity,

In the Capacity settings page, select Fabric Capacity.

select the capacity you want to make changes to.

In the capacity's setting page, from the Details tab,

select expand Admin permissions.

* Add or remove admins from the text box

Select Apply.

* Resize a capacity

To resize a Fabric capacity, see Scale your capacity.

* Delete a capacity

When you delete a capacity, the workspaces associated with the capacity are retained for seven days before they're deleted.

Ongoing operations are stopped or cancelled and scheduled operations are cancelled.

* To delete a Fabric Capacity,

From the list of Fabric capacities, select the gear icon next to the capacity you want to delete.

In the capacity's setting page, select the link Manage fabric capacities in Azure. A list of your Fabric capacities in Azure opens in a new tab.

From the Fabric capacities list in Azure, select the capacity you want to delete by clicking its name.

Select Delete.

In the confirmation dialog, retype the capacity name, and then select Delete.

* Auto scale

Auto scale isn't available for Fabric capacities.

* Capacity settings

After selecting a capacity, you can control its settings from these two tabs:

* Details

Capacity details are settings that are specific to the capacity.

* Delegated tenant settings

Tenant settings are delegated by Fabric admins to be managed by capacity admins. Changes to these settings only affect the capacity the changes are made in.

Delegated tenant settings are available for Power BI Premium and Fabric capacities.

* To view the settings of a specific capacity,

Go to the capacity settings page.

Select the capacity type your capacity belongs to.

From the capacity list, select the capacity you want to view.

* To view the settings of a specific capacity,

Go to the capacity settings page.

Select the capacity type your capacity belongs to.

From the capacity list, select the capacity you want to view.

**Gateway sizing**

A gateway is required whenever Power BI must access data that isn't accessible directly over the Internet. You can install the on-premises data gateway on a server on-premises, or VM-hosted Infrastructure as a Service (IaaS).

**Network latency**

Network latency can impact report performance by increasing the time required for requests to reach the Power BI service, and for responses to be delivered. Tenants in Power BI are assigned to a specific region.

**Monitoring performance**

Monitor performance is to identify bottlenecks. Slow queries or report visuals should be a focal point of continued optimization. Monitoring can be done at design time in Power BI Desktop, or on production workloads in Power BI Premium capacities.

Monitoring performance is relevant in the following situations:

* Your Import data model refresh is slow.
* Your Direct Query or Live Connection reports are slow.
* Your model calculations are slow.

1) Use Query Diagnostics

Use Query Diagnostics in Power BI Desktop to determine what Power Query is doing when previewing or applying queries.

2) Use Performance Analyzer

Use Performance Analyzer in Power BI Desktop to find out how each of your report elements such as visuals and DAX formulas are doing. It's especially useful to determine whether it's the query or visual rendering that's contributing to performance issues.

3) Use SQL Server Profiler

You can use SQL Server Profiler to identify queries that are slow.

SQL Server Profiler is available as part of SQL Server Management Studio.

Use SQL Server Profiler when your data source is either:

* SQL Server
* SQL Server Analysis Services
* Azure Analysis Services

To create a SQL Server Profiler trace

* Open your Power BI Desktop report

To determine the port being used by Power BI Desktop

* in PowerShell or at the Command Prompt, enter the command as

"netstat -b -n"

The output will be a list of applications and their open ports. Look for the port used by msmdsrv.exe and record it for later use.

To connect SQL Server Profiler to your Power BI Desktop report

* Open SQL Server Profiler.
* In SQL Server Profiler, on the File menu, select New Trace.
* For Server Type, select Analysis Services.
* For Server Name, enter “localhost:[port recorded earlier]”
* Click Run

Now the SQL Server Profiler trace is live and is actively profiling Power BI Desktop queries.

As Power BI Desktop queries are executed, you'll see their respective durations and CPU times. Depending on the data source type, you may see other events indicating how the query was executed. Using this information, you can determine which queries are the bottlenecks.