Checklist for Finalizing a Data Model in Power BI Desktop

Last updated: Dec. 27, 2017 (V2)

PBIX File

To Do	Why
Use a Consistent, Unchanging PBIX File Name with Business Relevance	The PBIX name you assign is what the user will see after it's been published. The name for the PBIX file should describe the set of reports, and it should be a consistent name which does not change over time. While a version number (like "SalesAnalysis-V1-December) is ok to keep straight interim copies of the PBIX, the file name needs to remain consistent once it is published. This is to ensure favorites, links, and apps all continue to work whenever changes are released.
Consider Reusing an Existing Dataset	Before starting a new PBIX, give some consideration to whether the data already exists in another Power BI data model. If new reports are the goal, yet the data already exists, a "hub and spoke" approach may work in which an existing published data model is reused. This can be done by creating new reports in the Power BI Service that refer to an existing dataset, by using a Power BI Service Live Connection, or via Analyze in Excel functionality. A hub and spoke approach (a) decreases the amount of redundant data stored in various PBIX files, (b) reduces the number of data refresh operations, (c) reduces future maintenance needs, (d) improves consistency because there are fewer files with data models, and (e) facilitates the separation of dataset development from report development. More details: Reusing Datasets Imported to the Power BI Service.

To Do	Why
Use the Correct Release of Power BI Desktop Based on Your Target Deployment Destination	There are now two 'flavors' of Power BI Desktop. The standard one is updated monthly, and it correlates with functionality releases in the Power BI Service. The newer one is Power BI Desktop Optimized for Power BI Report Server – that one is updated every 3-4 months. If you develop Power BI solutions which are deployed to *both* the Power BI Service and to Power BI Report Server, you will need to carefully manage which release you are using for your data model development (they can run side by side on the same machine). Keep in mind that if collaboration will occur on a Power BI Desktop file, all users need to have the same version of Power BI Desktop installed on their machine (this is easier if you install the standard version of Power BI Desktop via the Windows Store so that it will update itself in the background).
Manage the Dataset Size	File size limits impact the amount of data which can be imported into the data model. At the time of this writing, the file size limits (after compression is applied) are as follows: Power BI Desktop – based on individual PC system capability Power BI Service – 1GB, or 10GB if PBIX is deployed to a Premium workspace Power BI Report Server – 2GB if PBIX is deployed to PBIRS
Utilize Version History for PBIX File	Where to store the original PBIX file is an important decision. One common location is OneDrive for Business, which has built-in versioning available, so you can go back to a prior version if something goes wrong. A source control repository is also ideal. The original file should reside in a team site, not an individual user's laptop. If report edits are made directly in the Power BI Service, you may also need to download from the Service and synchronize with the version in your version history.
Consider Using a Power BI Template	A Power BI template (PBIT) can contain anything from a PBIX, except the data itself. (Note this is different from solution templates available on AppSource.) Templates can contain common queries and data tables (such as a Date table in every model). The original template file should reside in the same original location with version history.

To Do	Why
Consider Using a Power BI Theme	A Power BI report theme allows for a consistent use of a color palette in a report. Though theming affects reports and not the data model (the data model is the focus of this checklist), it is possible a data modeler will handle this task since custom report themes are based on JSON. The original JSON file should reside in the same original location with version history.

PBIX File Properties

To Do	Where	Why
Decide if Using Auto Time Intelligence	File > Options and Settings > Options > Current File: Auto Date Time	Auto time intelligence is enabled by default, and it applies to each individual PBIX file (there's not a global option). For most datetime columns that exist in the dataset, a hidden date table is created in the model to support time-oriented DAX calculations. This is great functionality for newer users, or if you have a very simple data model. However, if you typically utilize a standard Date table for the analytical dates, then you may want to disable the hidden date tables to reduce the file size – particularly if you have some dates which are just attributes (most dates can be analytical wherein you do analysis like YoY, though some dates can just be simple attributes which don't need to be connected to a Date dimension). Using the built-in functionality along with a user-generated Date table can work as well. (Tip: You can view the hidden date tables if you connect to the PBIX via DAX Studio.)
Control Options for Relationships	File > Options and Settings > Options > Current File: Relationships	Once the data model and relationships are complete, you may wish to ensure the relationships do not change without your knowledge. To do this, disable two items in each PBIX file (there's not a global option): "Update relationships when refreshing queries" and "Autodetect new relationships after data is loaded."

To Do	Where	Why
Use Preview Features Carefully	File > Options and Settings > Options > Global: Preview Features	Enabling public preview features should be used cautiously since preview features are not typically supported in the Power BI Service nor Power BI Report Server. It's easy to forget you used a preview feature which won't work properly after the report has been published.

Query Editor

To Do	Where	Why
Decide the Data Connectivity Mode	Home > Edit Queries	A very important decision is whether to utilize import mode or DirectQuery mode. Guidance about this decision can be found in the Planning a Power BI Enterprise Deployment whitepaper.
Import the Minimum # of Columns	Home > Edit Queries	Importing *only* the columns that are needed simplifies the model and reduces the size. Model size is very important since the Power BI model resides in-memory (when in import mode). Since hidden columns still consume size and memory, the rule of thumb is: if a column doesn't contribute to a report, calculation, or a relationship, then it shouldn't be imported to the model.
Disable Load for Intermediary Queries	Home > Edit Queries > All Properties > Enable Load	Intermediary queries in the Query Editor are very useful in certain circumstances, such as when two queries are then merged together into one. Any intermediary queries should be disabled from being loaded to the data model.
Parameterize Data Source Connections or Other Changing Data	Home > Edit Queries > Manage Parameters	Parameters can be useful to minimize hard-coding and reduce maintenance when changes need to occur. For data source connections, by default the connection is stored in the first step of every query. For dev>test>prod scenarios, changing the connection only once in a parameter is far less tedious than for every query. It also reduces the chance of error or omission.

To Do	Where	Why
Set Data Source Privacy Level	File > Options and Settings > Data Source Settings > Edit Permissions > Privacy Level	When importing data from various types of sources, the types of optimizations available vary per data source (such as query folding which passes filters down to an underlying data source). If data is sensitive and you do not want it to be passed in-memory to another server for consolidation or aggregations, and/or you want to ensure a DBA monitoring the server cannot see values, you can control isolation by setting the privacy level. This may result in a slower consolidation or comparison of data, but is more secure for sensitive data.
Set Encryption for Data Source Connections	File > Options and Settings > Data Source Settings > Edit Permissions > Encryption	Connections should be encrypted whenever possible as a standard security measure. This does need to be supported by the data source which is being queried.
Use Consistent Source File Locations	Home > Edit Queries > Data Source Settings	When accessing sources such as flat files (ex: CSV or Excel), make sure the files are accessible from a centralized location which is not based on an individual user path. This will ensure a data refresh operation won't fail if another person executes it.
Retrieve Source Data from a Database View	Home > Edit Queries	When possible, when retrieving data from a relational database, select data from a database view rather than a table. Different sets of views allow for different calculations and customization for different purposes: DW views can feed a semantic layer for corporate BI delivery (ex: Analysis Services). Alternatively, USR or RPT (user / report) views can be customized for power users who are allowed to access relational data (ex: from Power BI Desktop).

Tables and Relationships

To Do	**Where**	**Why**
Use Friendly Business Name for Tables	Data pane > Fields	A table name like "Students" is nicer to look at within a field list than something like "TblStdt." The name should represent the data contained within the table. Other contextual information about the type of table is very helpful to include in the name, such as: Financial Monthly Snapshot, Financial Transactions, or Financial Current Balances. Though often debated, either singular or plural table names are fine - just be consistent.
Validate Relationships are Accurate	Relationships pane	Depending on the source of data, relationships may or may not be auto-generated. One of the most critical tasks is to verify all required relationships are in place and accurate with respect to the 1-to-many direction of each.
Verify Relationships Do Not Change Direction	Relationships pane	The direction of a relationship affects how filters propagate through a model. If the direction of relationships change, a 'dead-end' is created which means reports do not display correct data. This is more likely to occur for more normalized data models. Verifying this is a critical piece of model validation.
Validate and Use Bi-Directional Relationships Purposefully	Relationships pane	Bidirectional relationships should be used minimally, and with a specific purpose. They should be used as the exception – to accomplish something specific – rather than the default, particularly if you have any complexity to your data model. Sometimes CrossFilter() can be a substitute for a bidirectional relationship. Verifying that any bidirectional relationships are set up correctly is critical to model validation. (Tip: According to the BiDi whitepaper, a relationship with a Date table should always be one-to-many, never bidirectional.)
Validate and Use Inactive Relationships Purposefully	Relationships pane	Inactive relationships in the data model should exist only when they have purposefully been set up that way, in which case the relationship is invoked via a DAX calculation. Verifying that any inactive relationships are set up correctly is critical to model validation.

To Do	Where	Why
Create Synonyms to Improve Q&A Experience	Relationships pane > Modeling > Synonyms	If report consumers also use Q&A functionality for natural language querying (ex: Sales by Year by Division), the thoughtful creation of synonyms improves the user experience significantly so that users do not always have to input field names exactly. For example, perhaps the terms Division and Business Unit are interchangeable at your organization. Or, perhaps there are common acronyms users might input into Q&A that need to be translated to full field names.
Assume Referential Integrity When Possible	Home > Manage Relationships	If you are utilizing data in DirectQuery mode from a reliable and clean data source (such as a data warehouse), you can improve performance by assuming referential integrity.

Fields, Columns, and Measures

To Do	Where	Why
Create Unique Field Names Across the Entire Dataset	Data pane	Although the Power BI software permits columns to exist which are named the same across tables, that is a poor practice to allow in a data model. Let's say that there is a Sales Rep table and a Sales Manager table, and both have a Sales Region column. Both Sales Regions should be renamed to be Sales Rep Region, and Sales Manager Region for clarity (or, even better, renamed in an underlying database view where Power BI retrieves the data). The general rule to ensure report labels are easy to understand: the name assigned to each column needs to be self-explanatory *on its own* rather than relying on the context of its table.

To Do	Where	Why
Expose a Field Only Once in the Entire Dataset	Data pane	Sometimes a column exists in the data model more than once. The ideal choice is for the redundant column to not be imported at all to the dataset. If that isn't a viable choice for whatever reason, then make sure the "extra" copy of the column is hidden from report view. For instance, if a Sales Header ID exists in both the Header and the Lines tables – make sure it's hidden in the Lines table and only shown once in the data model. This way, users won't accidentally use both columns. Using both could result in possibly needing to use two separate filters which would provide a very poor user experience.
Hide Fields Not Utilized for Reporting	Data pane	IDs and surrogate keys are needed for relationships, but are not useful for reporting. Hiding them simplifies the data model because there's less fields shown in the field list. The consumers of your data model will appreciate the lack of what they'd view as clutter. (Reminder: if a column is not needed for something – a relationship, basis for a calculation, sorting, something – then don't import it at all.)
Use Friendly Names for Fields	Data pane	A field such as "Student Name" is nicer to look at for reporting than something like "Stdt_nm" which may be how it's stored in the source. Since field names impact the naming of column titles, filters, and slicers, assigning friendly names with business-relevance are well worth a bit of time investment. Just be consistent with respect to spaces, casing, and abbreviations (such as Desc or Nbr). Try to only use acronyms if you are certain *all* users know the meaning.
Set Formatting for All Numeric and Date Fields	Modeling > Formatting	It's no fun to add a field onto a report that needs to be reformatted every single time. Defining units as whole numbers, or amounts as currency, is a helpful timesaver on the reporting end. Don't forget to also set the comma to assist with readability.

To Do	Where	Why
Specify Sorting for Columns	Modeling > Sort	Creating a default sort order is a common need for certain types of columns. For example, you may need to sort the Month Name column by the Month Number column. Oftentimes the column serving as the sort value can be hidden from report view.
Set Default Summarization for All Numeric Columns	Modeling > Properties > Default Summarization	The summarization default is sum, but some numeric columns are more suitable as count, min, or max. For example, high temperature per day would never be summed for a meaningful value; average is likely a better choice. Setting this properly allows subtotals and totals to be presented properly. Summarization for column that aren't really numeric, such as Customer Number or Year Number, should be set to "don't summarize." More details: Why the Default Summarization Property in Power BI is So Important.
Create Useful Calculated Columns	Modeling > Calculations > New Column	Creation of calculated columns (aka derived fields) is vital for enrichment of the data model. A very simple example of this is names – perhaps the underlying data source keeps First Name and Last Name in separate columns; you may wish to derive a Full Name field for reporting purposes which concatenates them together. Note: there are times when it's preferable to derive columns in the Query Editor instead of DAX: (a) to take advantage of query folding, (b) to remove intermediary columns from being loaded to the model, (c) to achieve faster processing time since DAX columns compute sequentially, or (d) to potentially achieve a better compression rate. Marco Russo discussed the last two items in this post. For complex calculations in a larger model, test the difference in performance.
Create Useful Calculated Measures	Modeling > Calculations > New Measure	Calculated measures (aka explicit measures) are extremely useful to augment reporting and analysis. Year-to-date and year-over-year are common examples of calculated measures which rely upon current context for calculation at runtime. Typically a model starts with common calculations, and continues to evolve over time as business users identify additional relevant calculations.

To Do	Where	Why
Decide on Location for Calculated Measures	Modeling > Properties > Home Table	Some Power BI practitioners prefer to locate all calculated measures in a specific table just for measures. However, this results in calculated columns and calculated measures residing in different tables which might seem confusing to users. If you do choose to use a "home table" for all measures, or certain critical measures, focus on organization to help users find them easily.
Consolidate Intermediary Calculated Columns	N/A	Frequently it is easier to understand and test a complicated calculated column in a series of steps that build upon each other. The tradeoff for this understandability is additional size and memory usage. Therefore, you may consider consolidating calculated columns once testing is complete rather than leaving the separate steps permanently in the model. If they remain, make certain that any intermediary calculated columns are hidden from report view. Alternatively, you could handle adding new columns in the Query Editor (instead of DAX), and delete the intermediary columns before the last step in the query.
Decide on Use of Implicit Measures	N/A	An implicit measure utilizes the default summarization setting such as "Sum of Sales Amount." Some Power BI practitioners prefer to use explicit calculated measures only, in which case base numeric columns are hidden. Like many other things, consistency here is what matters.
Set the Data Category	Modeling > Properties > Data Category	Address-related columns, such as city/state/zip and so forth, need to be specified in the data model to ensure geocoding can occur properly. Typically, Power BI is very good about detecting data categories from the data, but the data category for addresses, URLs and barcodes needs to be verified.
Create Columns to Support Query String Parameters	Modeling > Calculations > New Column	There are some rules for using URL query string parameters, such as no spaces allowed in the name. Therefore, there may be circumstances when you need to create some specific columns for use with query string parameters. In this case, you'll likely want to name them something similar to 'pStateName' and ensure they are hidden from report view.

Other Dataset Items

To Do	Where	Why
Ensure a 'Data Of Date' Exists in the Dataset	Modeling > Calculations > New Column	It is common to display a 'Data As Of' date on reports. This should exist in the data model, based on the data itself and not reliant on the current datetime. The objective is for report consumers to understand the effective date of the data being displayed. As of dates tend to be most significant for certain types of data such as sales quotas, or during key periods such as month-end.
Create Hierarchies for Usability	Field List > New Hierarchy	Date columns (such as Year>Quarter>Month) are a common scenario for use of a hierarchy and should be present in most models. Geography columns (such as Country>State>City) are great candidates as well. After a column has been added to a hierarchy, then it's up to you whether the individual columns should still be visible and available for reporting (sometimes users find that seeing both the individual columns & the hierarchy columns to be confusing).
Set Up Row-Level Security	Modeling > Security > Manage Roles	Row-level security is useful to specify which rows of data certain users are permitted to view. There are several techniques for how to accomplish RLS. (Tip: if you find you are repeating work in numerous PBIX files to set RLS, that means it may be time to look at using Analysis Services instead.)

A Few Common DAX Best Practices

This is a very short list of good DAX habits – certainly not intended to be a complete reference.

To Do	Where	Why
Use a Naming Convention When Referencing Measures and Columns	Modeling > Calculations	When referencing a measure in a DAX calculation, do *not* include the table name. Conversely, when referencing a column in a DAX calculation, always include the table name as a prefix in the format of 'table'[Column]. This commonly used best practice in the Power BI community helps you identify what type of column is input to the calculation, especially if you are troubleshooting context and calculation results. If you need convincing, watch this 2017 PASS Summit video from Marco Russo.
Apply the Most Selective Condition on Inner Functions	Modeling > Calculations	If a DAX calculation is nested, the innermost function is evaluated first. For performance optimization purposes, the most selective condition should typically be nested to limit data as early as possible.
Use Error Messaging in DAX	Modeling > Calculations	The Error() function provides a description if a calculation result cannot be rendered, which significantly can prevent users being frustrated should something go wrong with the report display. Error messaging is particularly important when displaying data which is manually maintained.
Consider Using the Divide() Function	Modeling > Calculations	If you utilize the Divide() function, instead of Column A / Column B, you don't have to worry about nulls or coding around divide-by-zero errors. This standard practice prevents invalid numbers being displayed on reports (which in turn minimizes users being tempted to download to Excel just to 'clean up' the report output). In a typical data model, use of Divide() will be fine. However, if you have complex calculations and/or a complex data model, check out Alberto Ferrari's post about Divide() performance.

To Do	Where	Why
Use Variables as a Common Practice	Modeling > Calculations	Use of variables in DAX is becoming a best practice, as it can make the syntax shorter and easier to read. Variables can improve performance because they are "lazy" meaning they evaluate only once and won't be evaluated if not used.

Documentation Within a PBIX

To Do	Where	Why
Solution Documentation	Report > New Page	You may consider using the first or last report page as a place to include helpful tips, definitions, or an introduction to the solution. It can also link to additional information, identify who to contact with questions, and/or who the owner of the solution is.
Descriptions for Fields	Field List > Properties > Description	A description can be immensely helpful for users to understand the contents of a column or measure, when and how to use it. When a description has been defined, it is shown in the field list as a tooltip when the mouse hovers on the field name. (Tip: if you find you are repeating work in lots of different PBIX files to input descriptions, that means it may be time to look at using Analysis Services instead.)
Comments in DAX Calculations	Modeling > Calculations	Comments in DAX (using // or /* */) can be very helpful to describe what's happening in a calculation. This can be to help others understand the calculation, or to leave yourself a reminder.
Comments Within M Scripts	Home > Edit Queries > View > Advanced Editor --or-- Home > Edit Queries > Applied Steps > Properties > Description	Comments in the M script (using // or /* */) can be very helpful to describe the data transformations which are occurring. Comments can be created one of two ways, as Chris Webb describes in his blog post.

To Do	Where	Why
Description for Query	Home > Edit Queries > All Properties > Description	If a query is doing something unusual, or is serving as an intermediary query, it is very helpful to include a description of the purpose for the query.
Naming of Query Steps	Home > Edit Queries > Query Settings > Applied Steps	Steps in the query editor get auto-named, which is sufficient for most needs. If you are doing a lot of data transformations, or something complex, it can be helpful to rename the query steps to do what's happening at a particular stage.