reportworq

Reporting and Data Modeling

Enterprise reporting, distribution, and governed AI - product documentation

Version 6.0.0.0

Contents

Reporting and Data Modeling

How reporting works 4
Data models 9
The web report studio 32
The viewer 40
A web report as a job source 45
Early release features and Send feedback 49
Troubleshooting and limits 54
Bring several sources into one model 71
The Excel add-in and the studio 76
Download, burst and keep a view 82
Convert a web report to Excel 87
Build a view with the designer 91
List reports 100
Key-cell formats and what a job keeps 110
Join views across sources 113
Crosstabs 123
Table relationships 130
Charts and maps 133
Hierarchies 140
Professional reports 147
Field filtering 152
AI insights 157
Direct queries 162
Combine report types on one sheet 167
View visibility 171
Cell-driven variables 177
Data model variables 180
Global parameters 188
Discover your data 191
Ask AI on a data model 197
Build a data model from a workspace file 208

Reporting and Data Modeling

How reporting works

explanation author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

Reportworq reporting is a chain of four stages. Each stage has one job, and each hands something finished to the next. If you keep the chain in mind, every other page in this guide has an obvious place.

The chain on one page

```
1. DATA SOURCES        a planning cube, a SQL database, a workbook
        |
        v
2. DATA MODEL          one or more sources shaped into named VIEWS
        |
        v
3. REPORTS             built on views, placed on a web report
        |              or in an Excel workbook
        |
        +----------------------------+
        |                            |
        v                            v
4a. VIEWED LIVE               4b. SENT BY A DISTRIBUTION JOB
    by an author or               refreshed, personalized and
    administrator                 delivered as static output
                                  (PDF, Excel, email, and so on)
                                        |
                                        v
                                  READERS receive it
```

Stage	What it is	Who works here	Where to learn more
1. Data sources	The systems that hold your numbers. You connect them once.	Administrators and authors	Connect a data source
2. Data model	A governed layer that reads one or more sources and shapes them into curated **views** with clean, business-friendly field names.	Authors	Data models
3. Reports	The tables, crosstabs, charts and maps that read a view. They sit on a **web report** or in an Excel workbook.	Authors	The web report studio
4a. Viewing live	Opening a web report to see current data, filter it and download it.	Authors and administrators	The viewer
4b. Sending	A **distribution job** refreshes the workbook, personalizes it and delivers it.	Authors and administrators build it; readers receive it	Web reports in distribution jobs

Stage by stage

1. Data sources

A source is a connection to a system of record, such as a planning cube or a SQL database, or a file that lives in your workspace. You connect each source once and every model can then use it. Nothing about a source changes when you build a model on it.

2. The data model

A **data model** reads one or more sources and turns them into **views**. A view is one query plus a curated set of fields, and each field has a stable name that survives changes in the source. Several sources can feed one model, and views can be joined across sources. The model is also where you decide which views report authors can see.

Start with Data models, then go to the page for what you need:

- Bring several sources into one model
- Build a view, Joined views and Table relationships
- Hierarchies and Field filtering
- Direct queries and View visibility
- Data model variables and Global parameters
- Discover your data and Ask AI on a data model
- Build a data model from a workspace file

3. Building reports

A report reads a view and shows it as a list, a crosstab, a chart or a map. Reports are placed on a **web report**, an Excel-like page you edit in the browser, or in an Excel workbook that you build with the Excel add-in. The two meet in the same studio. **New > New Web Report** starts a web report.

Start with The web report studio and The Excel add-in and the studio, then:

- List reports, Crosstabs and Charts and maps
- Professional reports
- AI insights
- Combine report types
- Cell-driven variables

4a. Viewing live reports

Opening a web report in **Viewing** mode shows it with current data. A person viewing can change what the report shows (filters, expanding and collapsing, drilling to source rows) but not how it is built, and nothing they change is saved. From the viewer you can download the report or split it into parts by a filter. See The viewer and Download, burst and save as.

4b. Sending with a distribution job

A **distribution job** is how a report reaches the people who need it. The job refreshes the report with current data, personalizes it for each recipient if you ask it to, and delivers the result by email, a shared folder or another destination. What recipients get is **static**: a file that shows the numbers as of the run.

A web report reaches a job in one of two ways. The route to prefer is to convert it to an Excel workbook, which the job then refreshes in place on every run. See A web report as a job source, Convert to Excel and Key cell formats and what a job keeps. For the job itself, see the distribution job lifecycle and the distribution overview.

Live and static

	Viewed live	Sent by a job
What you get	A web report drawn with current data	A file (PDF, Excel and so on) as of the run
Who sees it	Authors and administrators	Anyone you choose, including people with no Reportworq sign-in
Can the person change it?	They can filter and drill, but nothing is saved	No, it is a finished file
When the data is read	Each time the report opens or refreshes	Each time the job runs

> **Readers never see web reports.** Web reports are working tools for the people who build and analyze. A reader receives the static output of a job. This is by design, so a reader never has to learn the studio and never sees a half-finished report.

Two kinds of splitting exist, and they do not mix. The viewer's **Burst** pane is a lightweight way for an analyst to cut one live report into parts for themselves. A distribution job is the author's full bursting, and it produces static content for the people on a burst set. A job does not carry the live report's filters. To learn the job side, see What is bursting.

Who can do what

Role	What they do in this chain
Administrator	Connects data sources, manages the license, and can do everything an author can
Author	Builds data models, builds web reports and workbooks, views them live, and builds and runs distribution jobs
Reader	Receives job output. Does not open the studio or web reports

What needs a license

Capability	License
Data models, web reports, viewing, downloading, and using web reports in jobs	No separate license for web reports
Professional reports	Not released yet; this documentation is in preparation for the next release. They need a license that includes them. Without it the controls stay on screen but are disabled and read **Professional reports are coming soon.**
A job that uses a professional report on an unlicensed server	The job fails and keeps no file, with the message "Nothing was distributed: the web report '' uses a professional report, which this license does not include."

An administrator can see which features the license grants on **Settings > Configuration > License**. See Licensing.

> **Going deeper.** Some of these features are in early release. See Early release features and Send feedback. If something does not behave as described, see Troubleshooting and limits.

Data models

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **data model** is a reusable, governed layer between a source system and the people who build reports. It reads one or more sources and shapes them into curated **views**, and those views are what report authors use when they **build reports in the Excel add-in** or in the web report studio. A model lets you rename and reshape source columns into clean business fields once, so every report reads from a stable vocabulary that survives changes to the underlying source schema.

A model feeds two downstream workflows, not just reporting: **Distribution Jobs** (template-based output) and **Contribution Campaigns** (input forms). A contribution form binds to a model view the same way a report does, so curate a model as a shared vocabulary, not just a report source.

A model can read **several sources** at once: any number of Integrations connections and workspace files, side by side. A planning cube, a SQL database and a workbook can all feed one model, and each view draws from one of them.

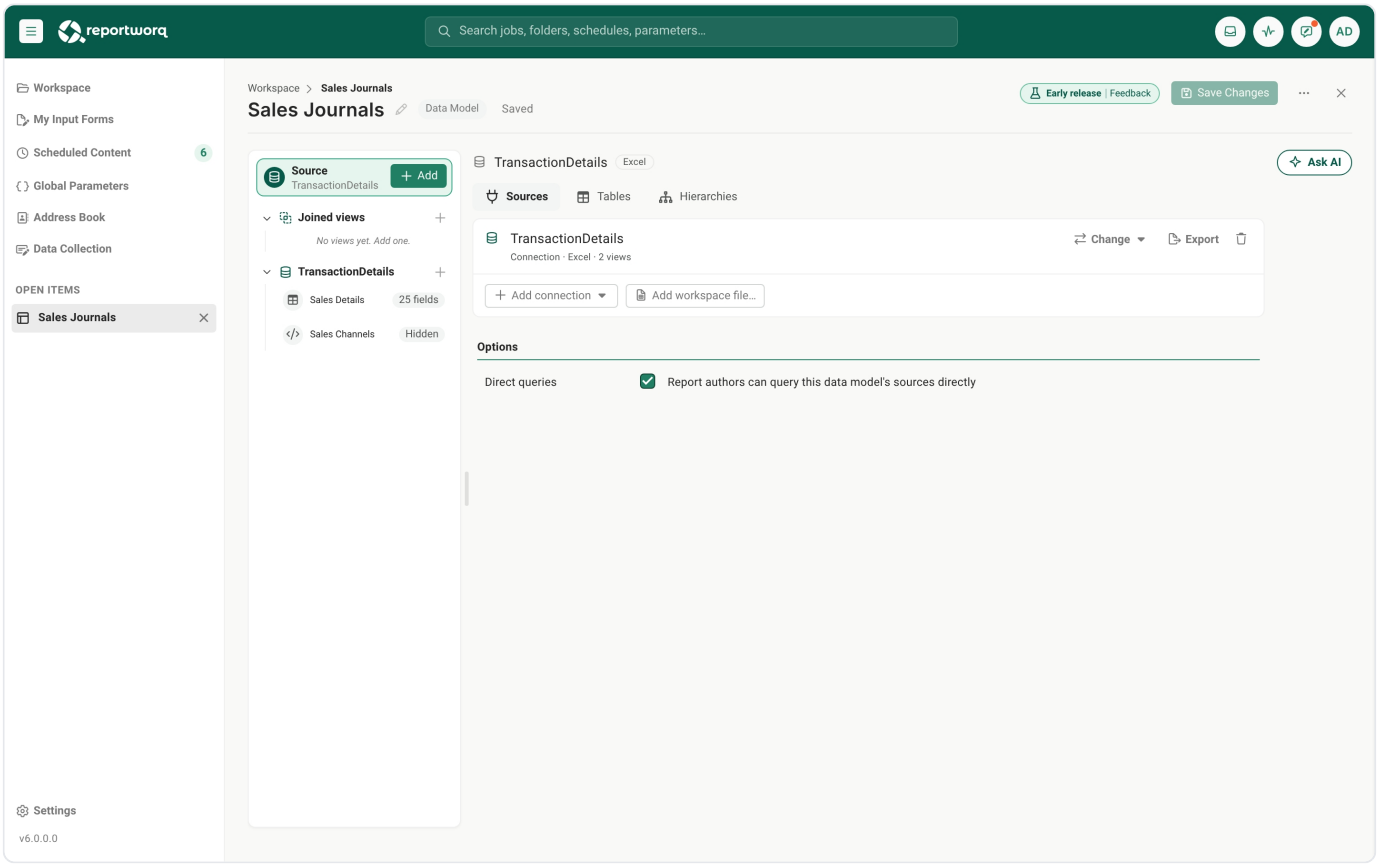
> **Modeling a single workbook or database that lives in this workspace?** You do not need a datasource connection at all. See Build a data model from a workspace file - it is the quickest way to get from a file in your workspace to a working model, with nothing added to Integrations.

Before you begin

- At least one source to read: either a datasource connection that tests successfully (see Connect a data source), or an Excel workbook or SQLite database that already lives in this workspace (see Build a data model from a workspace file).
- You have permission to author data models in the workspace.

The model editor at a glance

A model is edited in the workspace. The rail on the left lists the model's **Source**, then its joined views in a **Joined views** folder, then its other views in a folder per source. The **Source** entry's second line names the source the model reads (a connection or a workspace file), or how many when it reads several (for example **3 sources**), or **No source** until you add one; its tooltip lists every source. Beside **Source**, a **+ Add** button starts a new view (below). Select an entry to open it; your place on each screen is kept while you move between them, so you can check a table on **Source** and return to a half-built view without losing it.



- **Source (model level).** The Source screen opens on its **Sources** tab, which lists every source the model reads, with **Change** (point it at another connection or file) and **Remove** beside each, plus **Add connection** and **Add workspace file…** to add more (see Build a data model from a workspace file). One **Direct queries** checkbox on this tab applies to the whole model (see Direct queries). The **Tables** tab lets you look at the rows and links of any SQL source, including a SQLite or Excel workspace file, before you build a view. See Explore the source. **Export** on each source's row copies all its tables to one workbook (see Export a whole source to Excel). **Ask AI**, at the top right of the Source screen, opens the data model assistant (see Get views suggested by the AI assistant).

- **A source that is missing or disabled is flagged when the model opens.** If a source's connection has been deleted from Integrations or disabled, or its workspace file has been moved, renamed or deleted, an exclamation mark appears beside the source's folder in the rail (point to it to read why), and the source's row on the **Sources** tab explains how to fix it. This check only confirms that the connection or file is still there. It does not sign in to the source or test that it can be reached, so a problem of that kind shows when a view runs.
- **Add a view.** The **+ Add** button on the rail's Source node opens a short list, **Add a view of…**, with one entry per source, then **Joined view**. A model with no source yet offers **Add a source first…**, which opens the **Sources** tab. Each source's folder in the rail also has its own plus sign. A view whose source was removed moves to a **No source** folder; open it to point it at a source again. Choose a source and you get a new view of that source, opened in the editor that suits it: the table designer for a SQL database, the cube view editor for Planning Analytics, and so on. Choose **Joined view** to combine views of the model, usually across sources (see Join views across sources).
- **Views.** A view is the model's unit of output: one source query plus a curated set of fields. A model can hold several, and each reads from one of its sources. Each view has four tabs, and every tab is always available:
    - **Design** (on SQL datasources) or **Connection** (on every other connector) defines what the view queries, whether the view's source is a configured connection or a workspace file. On a SQL datasource you build the view on a canvas of tables and their relationships, or write it as SQL. See Build a view with the designer. On other connectors the tab holds that connector's own query settings, for example a cube data view for Planning Analytics. There is no import step on either: the view's fields follow its query, as described under the list. A joined view has a canvas of its own (see Join views across sources).
    - **Fields** is the curated reporting field list, generated from the view's query. Each field decouples its reporting-facing **Id** and **Name** from the source column it maps to (the **Mapped Field**), so renaming or re-mapping a source column does not break downstream reports. Fields can be hidden, reordered, renamed, re-typed, filtered, added, or deleted.
    - **Variables** are the `{{tokens}}` that parameterize the view's query and field filters, and flow downstream to reporting and bursting. Set each one's default, whether it is required, and how a report prompts for it. See Data model variables.
    - **Preview** runs the view and shows the resolved rows after variables and field projection are applied.

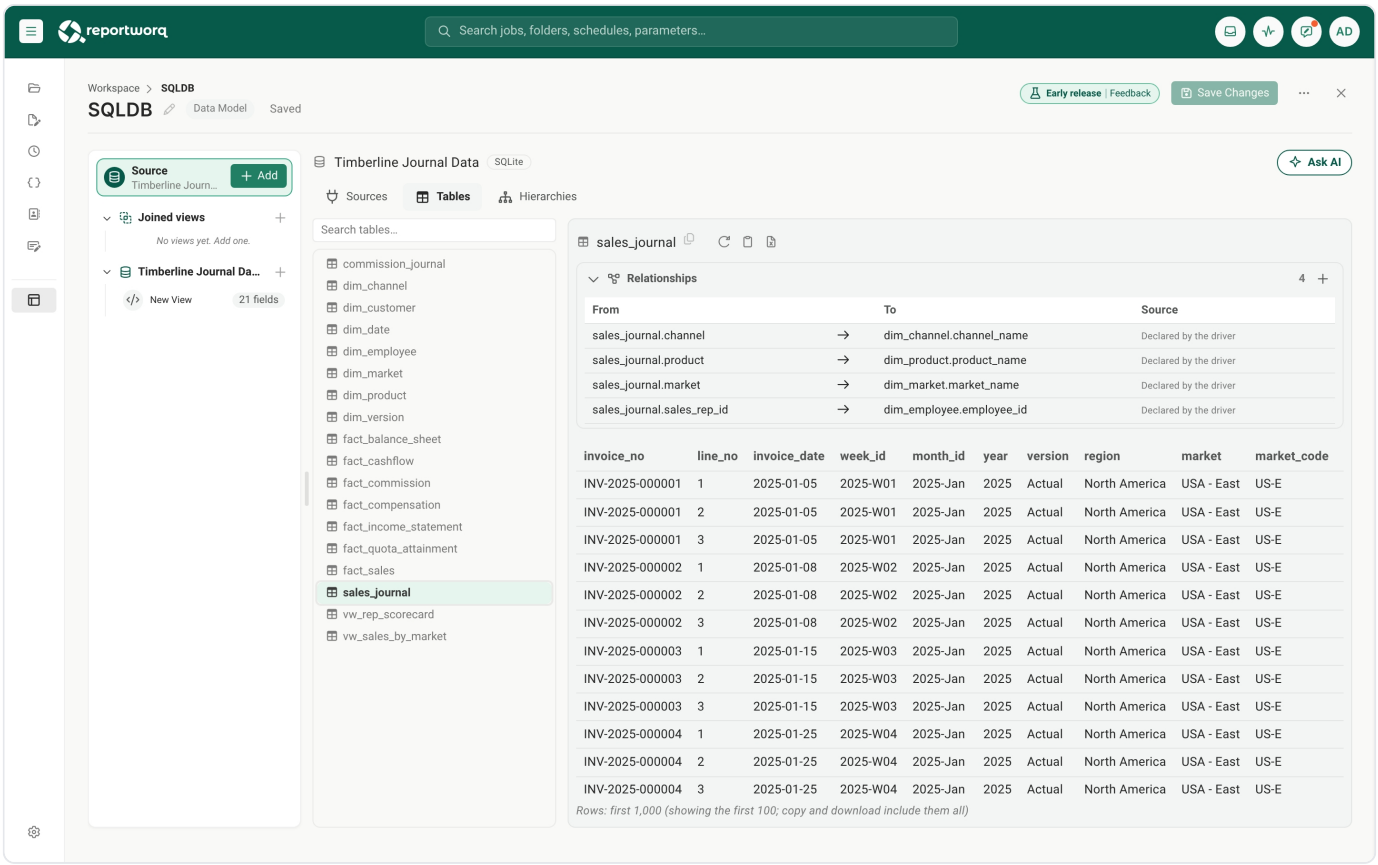
Fields follow the query. There is no button to import fields. When you leave the **Design** or **Connection** tab, and when you select **Save Changes**, Reportworq runs any query that changed and brings the view's fields up to date. While it runs, the view reads

"Updating fields from the query". If the query cannot run yet, for example because a `{{variable}}` has no sample value (a joined view never asks for one: its fields come from its inputs' samples or defaults), a warning explains why and the view reads "Fields may not match the query yet" until the next tab change or save. On the designer canvas, fields are added as you tick columns, so a designed view is always up to date.

Explore the source

The **Source** screen has a **Sources** tab and, once the model has a SQL source to browse, a **Tables** tab. A model with nothing to browse yet opens on **Sources**; a connected SQL model opens on **Tables**. When the model has several SQL sources, the table list has one folder per source, so you always know which source a table belongs to. The table you select stays selected while you visit **Sources** and come back.

Use **Tables** to find the table a view should start from, and to check how it links to other tables before you build.



1. Select **Source** in the rail, then **Tables**.
2. Select a table in the list on the left. Use **Search tables** to narrow a long list.
3. Review the table's links in the **Relationships** section above its rows: those the source declares, those Reportworq infers, and any you defined on the model. Select the plus sign beside **Relationships** to add one, or select **Relationships** to collapse the section and give the rows more room. See Table relationships.

4. Review the table's rows. The grid reads at most 1,000 rows, and the row count under it says how many were read.
5. Optional: select the clipboard button to copy every row of the table with a header row, ready to paste into Excel, or the Excel button to download a workbook of the table (what it is, its relationships, and every row). The small copy icon beside the table name copies the name. To download every table of the source at once, see Export a whole source to Excel. For all the ways to look at a model's data, see Discover your data.

The refresh button reads the table and its relationships again. Table relationships are kept per source, because a relationship becomes SQL and can only relate tables of one source. To relate views of different sources, use a joined view's relationships between views instead (see Join views across sources).

Get views suggested by the AI assistant

The data model assistant reads the model's sources and suggests a set of views, joined views included, with business names for their fields. Where the data rolls up naturally (state to region, product to category), it also suggests hierarchies and makes them for you. It is a quick way to start a model over an unfamiliar source.

The assistant needs an AI provider. An administrator adds one in **Settings** > **Integrations** (see Connect an AI provider). The assistant uses the AI connection selected for the in-app chat assistant, when one was selected and it is enabled; otherwise it uses the first enabled AI connection. With no enabled AI connection, the panel says "No AI model is set up" and offers nothing to run.

> **What the assistant reads, and what it sends to your AI provider.** When you select **Analyze my data**, the assistant reads each source: it lists the tables (up to 80 per source) with their columns and links, and it **runs a small sample read of two rows on each of up to 40 tables per source** (12 for a cloud connector, where each sample read is a call to the service's API). It then sends the provider each source's name and type, those table and column names and types, the links between tables, the names and fields of the model's existing views, the names and levels of its hierarchies, and the **sample rows**, each value cut to 40 characters. Those sample rows are **real values from your data**, and they leave Reportworq for the provider. The sample reads run against every source of the model, including a connection an administrator has disabled in Integrations. The assistant never changes the model until you add what it proposes, and it never writes SQL. Decide whether that suits the source before you select **Analyze my data**.

1. Open the data model and select **Source** in the rail.
2. Select **Ask AI** at the top right. The assistant panel opens over the editor; **Save Changes** stays within reach. To move the panel out of the way, drag it by its header. Double-click the header, or select the dock button, to put it back; the panel remembers where you left it in this browser.
3. Select **Analyze my data**. The assistant reads the sources, explains what the data is and which questions it can answer, and offers options.
4. Reply in the box to steer it, for example "views for a monthly sales pack by region". When it proposes views and hierarchies, each comes with a check box.
5. Clear the check box of any view or hierarchy you do not want, then select the add button (for example **Add these views**, or **Add 1 view and 1 hierarchy**). They are added **unsaved**: views to the rail, hierarchies to the **Hierarchies** tab of the Source screen.
6. Review each view and hierarchy, then select **Save Changes** to keep them, or **Discard** to drop them.

Reportworq checks every proposed view against what the source really has, and leaves out (with a note) any table, column, or link that does not exist. A filter the assistant proposes becomes an **optional** filter variable, so a blank value returns every row. Closing the panel keeps the conversation; select **Start over** to begin a new one. An administrator can tune the assistant's instructions in **Settings** > **AI Prompts** (see The AI prompt store). **The assistant never writes SQL**: every view it suggests, a hierarchy's own included, is a table and columns that must exist in the source, and Reportworq writes the query. A suggestion that comes back with SQL in it is left out, with a note, and nothing in it runs.

What you can ask for. Once the assistant has analyzed the data, three chips under the chat offer the most common requests: **Suggest views for my reports**, **Build a joined view** and **Build hierarchies**. **More things I can do** opens the whole list, by group:

Group	Starter
Views	Suggest views for my reports
Joins	Build a joined view; Find where my sources could join
Hierarchies	Build hierarchies
Understand the data	Explain what is in my sources; Tell it our naming standards

Picking one sends nothing: it starts the request for you to finish (see How the starters work). Ask for something it cannot do here, such as changing a view you already have, writing SQL or changing a source, and the assistant says so plainly and proposes nothing; to change a view, open it and use **Ask AI** inside it.

Hierarchies. A proposed hierarchy lists its levels from the most detailed up (for example State, then Region), the view it reads, and where it is attached. It usually comes with a view of its own: a table and only its level columns (a parent level can come from a table it links to), for example "Reads table Lookups: State, Region". That view is built like any view the assistant suggests, so you can open and edit it like one. A suggestion that sends SQL for a hierarchy's view is refused: the hierarchy is left out with a note, and nothing runs.

A hierarchy is attached to a **field name on a source**, not to one view: attaching it to State means every view of that source with a field named State gets it, the views you already have included. The card lists those views, and after you add it a note names the existing views it reached, and says so when it is the first hierarchy on their field (professional reports use a field's first hierarchy). Reportworq never replaces a hierarchy you already have: if the name is taken, the new one gets a number ("Sales Territories 2"), and a field that already has a hierarchy keeps it as its first, with the new one added after it. A hierarchy is attached only to fields of views of its own source. A level, field or source that does not exist leaves it out, with a note, and so does anything the card warns about. At most 10 hierarchies come in one reply. Reportworq does not compare the hierarchy's values with the field's values, so open it on the **Hierarchies** tab and select **Preview tree** before you save.

Ask AI inside a view

Inside a view or a joined view, **Ask AI** helps you shape the view you have open. Ask it to rename fields, set a field's type and format, show, hide or reorder fields, add, change or remove filters, set a variable's default and how it is asked for, join another view (in a joined view), relate this view to another one, or build a hierarchy and attach it to a field. It proposes the changes as a list you review; nothing is applied until you choose, and nothing is saved until you save.

The button sits in the view's header, beside the **Public**, **Hidden** and **Disabled** choice. It only appears when an AI provider is set up.

> **What Ask AI inside a view sends to your AI provider.** With every message, it sends:
>
> - the data model's name, and the open view's name, kind and source;
> - each field's name, source column, type (and any type and format you set), whether it is shown, and **the sample value the field already holds** (a real value from your data, cut to 40 characters);
> - each filter condition, **with its value** (a fixed value you typed, or the `{{Variable}}` it uses);
> - each variable's name, kind, whether it is required, **its default value** and its caption;
> - the view's relationships with other views, and, for a joined view, its starting view, its joins (keys, kind and matching options) and the views it may join;
> - the names and fields of the model's other views, and the names and levels of its hierarchies;
> - for a view of one source, that source's tables (up to 80) and their column names and types;
> - the conversation so far, including the list of changes you applied or skipped and the reasons.
>
> It reads **no rows** from your data: the only values from the data are the sample values the fields already hold, and the values you typed into filters and defaults. It never writes SQL and never changes the view's query.

1. Open the view (or joined view) and select **Ask AI** in its header. The panel opens, titled "Ask about" and the view's name.

2. Say what you want, for example "Tidy the field names and hide the id columns", or "Join Teams on TeamId". Or pick one of the chips under the chat, or a request from **More things I can do**, and finish it (see How the starters work).

3. The assistant lists its changes, each with a check box. Each line says exactly what the change does, written by Reportworq from the change itself (for example "Rename cust_nm to Customer" or "Filter Status <> Open"); the assistant's own description, when it says something else, is shown under it. Clear the check box of any change you do not want.

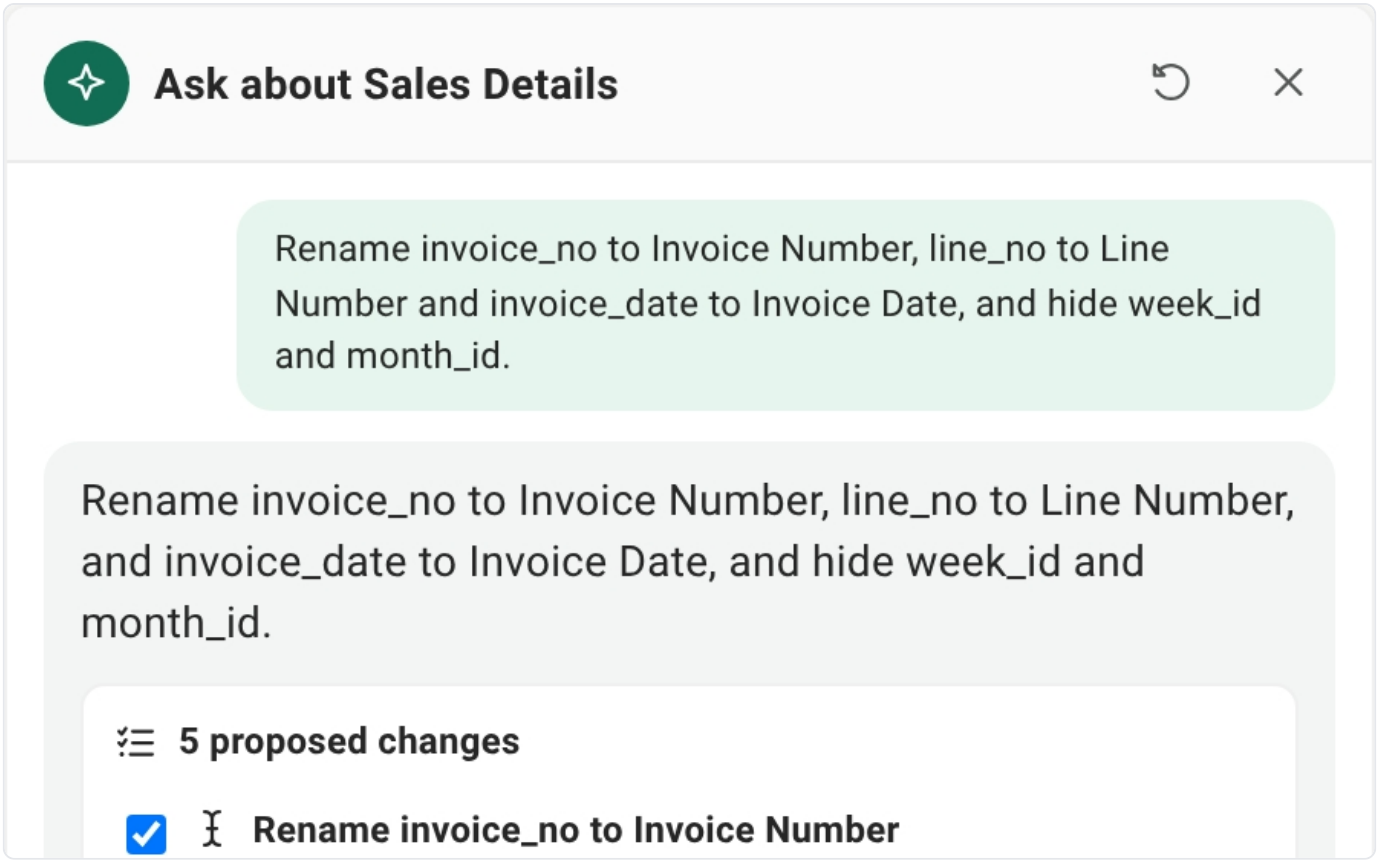
   A change Ask AI already knows it can't make is grayed out, with the reason on its line, and can't be ticked, so you never apply it by mistake. For example, if Ask AI writes a `{{Token}}` into a filter's fixed value, the change says the filter holds a token and isn't a fixed value, and that to let each report pick the value you ask for

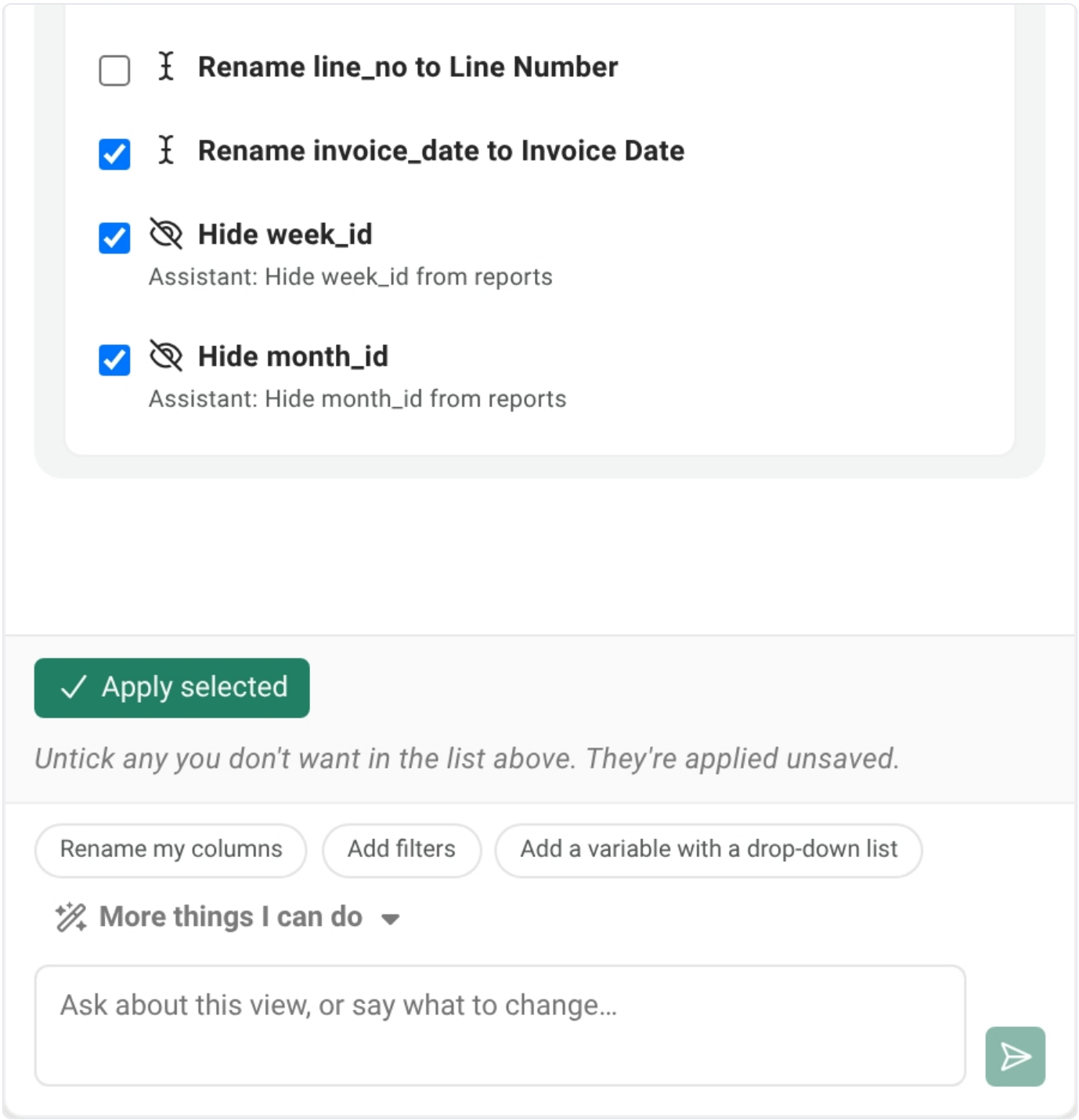
a filter on a variable. The change that would set up how that variable is asked for, or its default, is grayed out with it, because the variable would only come from the filter that can't be added. A change to a variable is grayed out too when the view doesn't have that variable, when it is pasted into the query as raw text (only you set those), or when it uses the settings from one of the joined view's inputs (change it on that view). The other changes in the reply can still be ticked and applied, and once you apply them, Ask AI is told which changes weren't made, so it doesn't offer them again.

When none of the changes in a reply can be made, and at least one can't be made for a reason like these, Ask AI answers in a plain sentence that says why instead. When every change was refused this way, it lists no changes at all; a change it does not support yet still shows on the list, grayed out. A reply made only of changes Ask AI does not support yet keeps the assistant's own answer, with those changes grayed out.

4. Select **Apply selected** (it stays off until at least one change is ticked). The changes are applied in order, **unsaved**, exactly as if you had made them yourself on the view's tabs. Each line then says what happened: applied, or skipped with the reason (for example "no field TeamId in Players"), and a summary such as "Applied 5 of 6" heads the list.

5. Review the view, then select **Save Changes** to keep the changes, or **Discard** to drop all of them.





A change that cannot be made is skipped and the rest still apply: a field the view does not have, a name another field already uses, a join that would make a view read itself (a cycle), a second copy of a view that is already in the joined view (add that copy in the joined view editor), a join from a view an **Only in left** or **Only in right** join leaves out, a relationship that already exists, or, in a joined view, a change to a variable listed after a join, when the variable still can't be set once that join is made. If you save or discard while changes are being applied, the ones not yet applied are skipped and say so. A change Ask AI does not support yet is shown as "Not supported yet" and is never applied; so is any change that comes with SQL. Ask AI never sets a variable that is pasted into the query as raw text (the **Raw query text, pasted as is** option on the Variables tab): a change to one is grayed out before you apply anything, as above. Any

default it sets is filled into the query by the same safe rules as a value a reader types (see Data model variables). Joining or relating views follows the same rules as the joined view's canvas: a join to a view this one is already related to follows that relationship, and changing how rows match changes the relationship for every joined view that uses it. Rewriting a SQL view's query is not something Ask AI does: it says what to change, and you change it on the view's **Design** tab.

Each view keeps its own conversation while the editor is open, and moving to another view moves the panel with you.

What you can ask for

Three chips under the chat offer the most common requests: **Rename my columns**, **Add filters**, and **Add a variable with a drop-down list** (in a joined view, **Join another view**). **More things I can do** opens everything Ask AI can do in the view you have open, by group:

Group	Starter	Offered on
Columns	Rename my columns; Reorder my columns; Show or hide columns; Change a column's type (text to a date, a number, or yes and no)	Every view
Filters	Add filters; Change or remove a filter	Every view
Variables	Add a variable the database filters on	A view of one source
Variables	Add a variable with a drop-down list (its choices come from a field of another view)	Every view
Variables	Set up how a variable is asked for (its default, caption, help text, and whether it is required)	Every view
Joins	Join another view; Change how rows match (all from the left, only matches, everything, only in left, only in right; one match or many; ignore case and spaces); Remove a join	A joined view
Relationships	Relate this view to another	Every view
Hierarchies	Build a hierarchy; Attach a hierarchy to a field	A view of one source

A variable that filters a view of a database source is applied by the database, so a report reads only the rows it needs. A joined view's own filters can take a variable too; one with the same name as a variable of a view it joins is that same variable, so one value fills both. Ask AI never sets up a variable that is pasted into the query as raw text: that one stays yours. Ask for something outside the list, such as rewriting the view's SQL query or

adding a calculated column, and Ask AI says plainly that it cannot do it here and proposes no change.

How the starters work

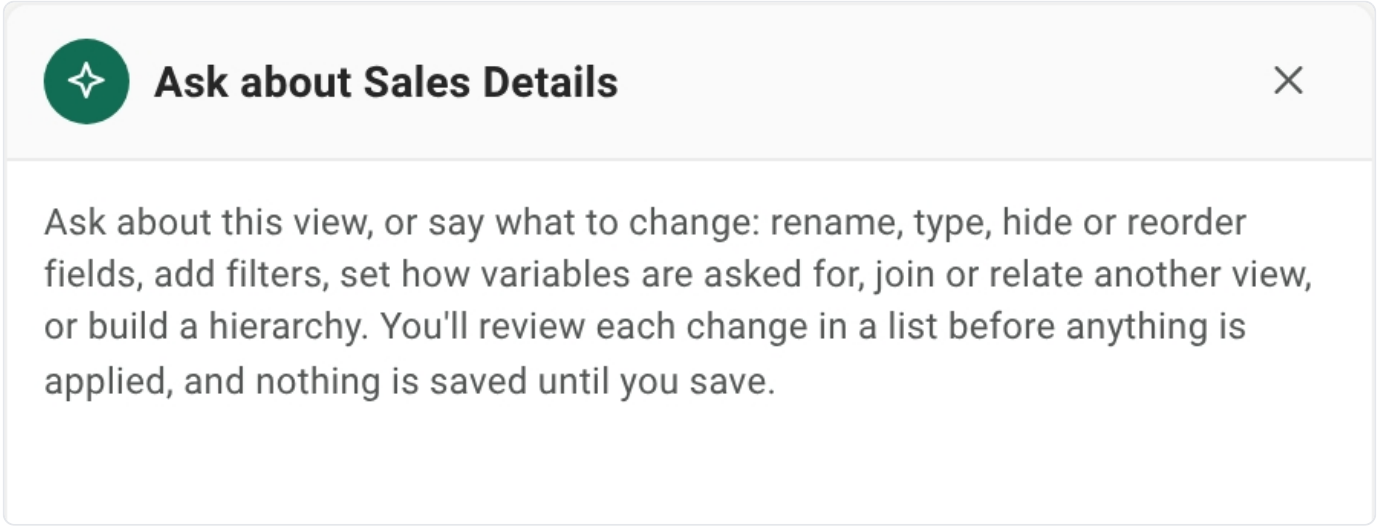
Picking a chip or a request from **More things I can do** never sends anything. It starts the request in the box for you to finish:

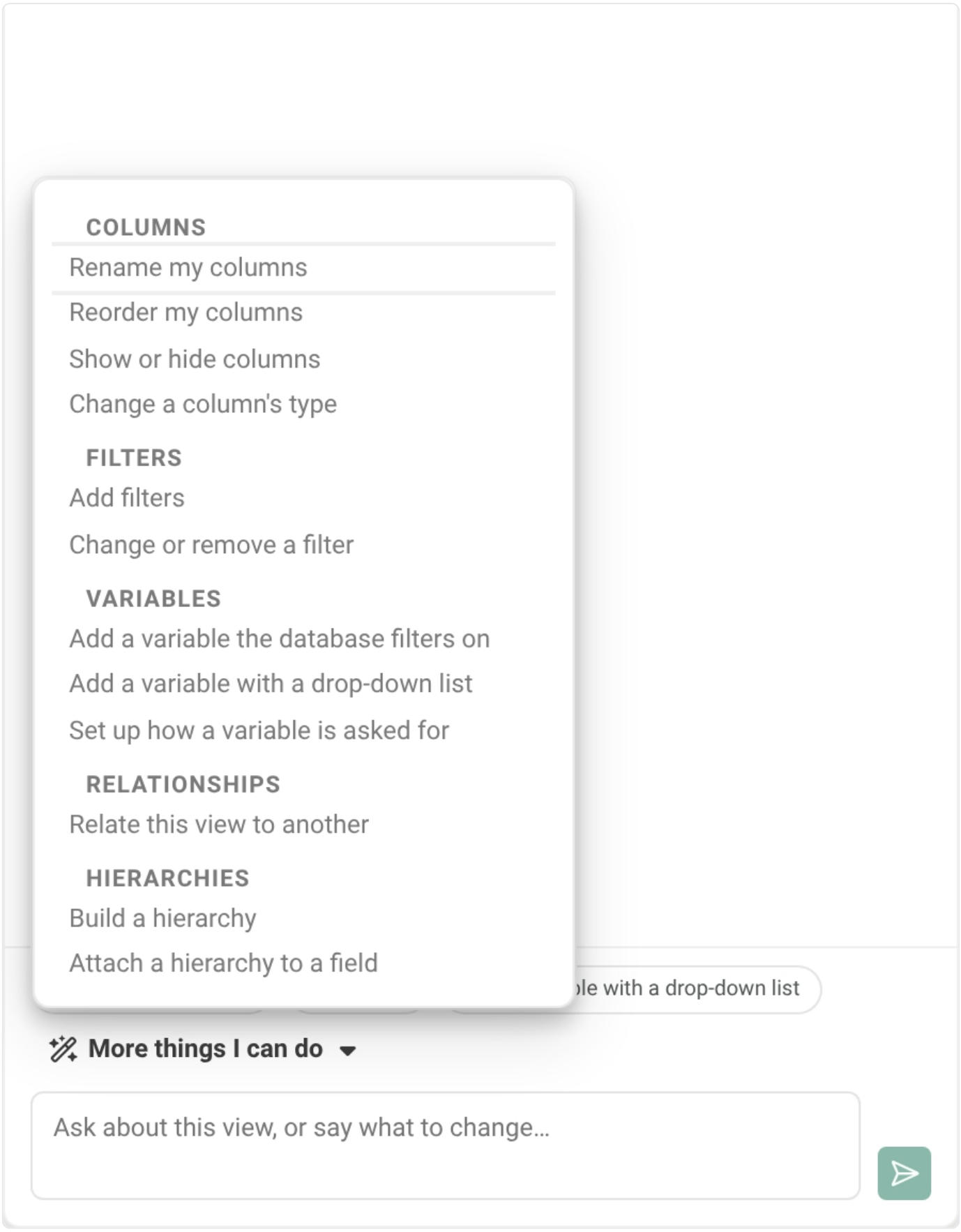
1. The box holds the start of the request with a part in brackets selected, for example "Add a variable with a drop-down list: [which field, and which view lists the choices]". Type to replace the bracketed part. If you had already typed something, it stays, and the request starts on a new line after it. Pick another request before you finish this one and only the unfinished request is replaced: what you typed is kept, however many times you pick.
2. Above the box, a question says what to tell it ("Which field should it filter, and which view lists the values to pick from?"), with a few example answers. Select an example to put it in place of the bracketed part; the rest of the box is left as it is.
3. Edit the request as you like, then send it. Pressing Enter while the bracketed part is still there selects it again rather than sending half a request. Clearing the box or sending puts the question away.

More things I can do works from the keyboard too: open it with Enter or Space, move through the list with the arrow keys, press Enter to pick a request, or Escape or Tab to close the list.

The Source screen's assistant works the same way, with its own list (see What you can ask for above).

The assistant is told the same list, so it knows what it can do in each place. An administrator who saved their own version of an assistant's instructions in **Settings** > **AI Prompts** keeps that text: it does not gain the list. Reset it to the default to get it (see The AI prompt store).




COLUMNS
Rename my columns
Reorder my columns
Show or hide columns
Change a column's type
FILTERS
Add filters
Change or remove a filter
VARIABLES
Add a variable the database filters on
Add a variable with a drop-down list
Set up how a variable is asked for
RELATIONSHIPS
Relate this view to another
HIERARCHIES
Build a hierarchy
Attach a hierarchy to a field
ble with a drop-down list
More things I can do
Ask about this view, or say what to change...
Ask about Sales Details

Ask about this view, or say what to change: rename, type, hide or reorder fields, add filters, set how variables are asked for, join or relate another view, or build a hierarchy. You'll review each change in a list before anything is applied, and nothing is saved until you save.

Rename my columns | Add filters | Add a variable with a drop-down list

More things I can do

Which field should it filter, and which view lists the values to pick from?

region, with the choices from a view of regions

customer, picked from the customer names of another view

product, from the view that lists products

Add a variable with a drop-down list: [which field, and which view lists



Join views across sources

A **joined view** combines the rows of other views of the same model, usually views of different sources: actuals from the ERP beside a forecast from the planning cube, or CRM sales reps beside HR employees. Each input runs on its own, then Reportworq matches the rows by the fields you choose. You relate two views once, check the match on real rows with **Test match** or let **Discover relationship** find the key, and line up values the two systems write differently with a value mapping. **Explain** shows how a joined view runs and what helps or hurts it.

See Join views across sources.

Build hierarchies

A **hierarchy** rolls a field's values up into groups: sales reps into teams, states into regions, players into teams, divisions and sports. Attach one to a field, and the crosstabs of web reports and Explore can show subtotals for each group. A hierarchy belongs to one source of the model, and you build it on the **Hierarchies** tab of the **Source** screen.

There are two kinds:

- **Manual**: you type the members and how they roll up, as an indented outline.
- **Levels**: you pick a view of the model and the fields it rolls up through, from the most detailed up, for example **Player**, **Team**, **Division**, **Sport**. The members are read from the view's data each time a report uses the hierarchy, so they stay current as the data changes.

Create or edit a hierarchy

1. Open the data model, select **Source** in the rail, and select the **Hierarchies** tab. The tab lists every hierarchy of the model, with its **Type**, what it is **Built from**, its **Size**, and the fields it is **Attached to**.
2. Select **New hierarchy**, or **Edit** on a hierarchy's row. The hierarchy opens on its own screen; the **Hierarchies** link at the top goes back to the list.
3. Enter a **Name**, and choose the **Type**: **Manual** or **Levels**.
4. For a **Manual** hierarchy, choose the source it is **Built from** (when the model has several), then type the outline under **Members**: one member per line, indented two spaces or a tab per level. A line with lines under it is a group; a line without is a value of the field you will attach the hierarchy to.

5. For a **Levels** hierarchy, choose the **View** to read. The list offers every view of every source of the model (not joined views), with the source in brackets when the model has several. Then, under **Levels, from the leaf up**, choose a field for each level: the first is the most detailed (the values the report's field holds), the last is the top. Use **Add a level above**, the arrows and the remove button to arrange them. A hierarchy needs at least two levels.
6. Select **Preview tree** to see the members the hierarchy gives, with anything the data says about them under **What the data says**, for example a team that rolls up to two divisions. It reads the view as you have it in the editor, unsaved changes included.
7. Select **Save hierarchy**, then **Save Changes** to save the data model. The hierarchy is saved with the model, so it is kept when the server restarts, and it travels with the model when you copy, export or import it.

How a **Levels** hierarchy reads its view:

- **Each value is one member**, however many rows hold it, matched without regard to case. The order is the order the view returns its rows, so sort the view to sort the hierarchy.
- **A member with two parents is reported, not merged.** It is shown under the parent of its first row only, so its totals are counted once, and the preview names every parent it has.
- **Blank levels are skipped.** A member whose parent is blank sits under the next level up that has a value. A row with a blank leaf gives nothing.
- **Variables follow the report.** When the view has variables, a report reads the hierarchy with its own values, so a report on one season shows that season's teams.
- **At most 1,000,000 rows are read.** If the view has more, or can't be read when a report runs, the field shows its values without the hierarchy, and the report says so. A hierarchy is never built from only some of the rows.

Attach a hierarchy to a field

1. Open a view of the hierarchy's source and select the **Fields** tab.
2. In the **Hierarchies** column of the field, select the hierarchy. For a **Levels** hierarchy, attach it to its most detailed field, the one its leaves are values of. **Detach all** removes every hierarchy from the field.
3. Select **Save Changes**.

A hierarchy is attached to a field **name** on its source, so every view of that source with a field of that name gets it. A field can carry several hierarchies: a crosstab lets you choose one.

Move a hierarchy to another source

A **Levels** hierarchy belongs to the source of its view. To move it to another source of the model, edit it and choose a view of that source. Reportworq refuses the move, and says why, while the hierarchy is attached to fields (detach it on the **Fields** tab first), when the other source already has a hierarchy of that name, or when that source has a problem. A **Manual** hierarchy stays on the source it was created on.

Export a whole source to Excel

Use **Export** to copy every table of a source into one Excel workbook, for example to hand a snapshot to someone without access, or to read a source's tables side by side. On **Source**, open the **Sources** tab and select **Export** on the source's row. **Export** appears on every source whose tables can be browsed (SQL databases, cloud connectors that expose tables, and SQLite or Excel workspace files).

Nothing is exported until you say so. The dialog looks at the source first:

1. **It inspects.** It lists the source's tables, reads each table's columns and counts its rows. Counting rows is skipped for cloud connectors, because counting a table there would mean reading all of it; their tables show columns only.
2. **It gives a verdict.** Quick, Medium or Large, with the number of tables, rows and cells (rows times columns) and a rough time. A table that could not be counted makes the figures a lower bound, so the verdict is never better than Medium. A very large export is refused, with a pointer to download the tables you need one at a time from the **Tables** tab: past 10,000,000 cells there is no **Export** button.
3. **It warns about load.** Exporting reads every row of every table. On a database that reads more than 2,000,000 rows, the dialog warns that it can slow the database for other people and suggests running it outside working hours. For a cloud connector it warns that rows are read through the service's API, one table at a time, and count against its rate limits.
4. **You select Export.** A bar shows each table's progress and one bar shows the whole export. Up to two tables are read at a time (one at a time for a cloud connector).

Stop ends the export and downloads nothing. Once every table has been read the workbook is built and sent to your browser, and there is no Stop from that point.

What the workbook contains

Sheet	Contents
About	The data model, the source and its connector, how many tables and rows were exported, and when. Below that, a contents list of every table with its row and column counts and any note. Each exported table's name links to its sheet.
Relationships	Every link between the source's tables: those the source declares, those Reportworq infers, and any you defined on the data model.
One sheet per table	A header row and the table's rows, with the header row frozen. Sheets follow the order of the table list.

The file is named after the source, for example `Timberline ERP.xlsx`.

When a table is too big for a sheet

A table is never refused for being large. It is cut to fit, and its row on the **About** sheet and in the dialog says what was cut:

- **Rows.** A sheet holds 1,048,575 rows of data, so a longer table keeps its first 1,048,575.
- **Columns.** A sheet holds 16,384 columns; the last columns beyond that are left out.
- **Text length.** A cell holds 32,767 characters; a longer value is cut to that length.
- **Workbook size.** The whole workbook is held to 10,000,000 cells. The export tracks the cells it has used as it goes, so a table that could not be counted, which may be much larger than it looks, is cut once the budget runs out, and any table after it is left out and listed as such.

A table that cannot be read, for example because of a permission, does not stop the export. It is listed on the **About** sheet with the reason.

> **Large workbooks open slowly.** A Long export, in the millions of cells, can take a minute or more to build and is slow to open in Excel. For one table, download it from the **Tables** tab instead.

Curate the fields

The Fields tab is where a raw source becomes a governed model. Each field shows a green **mapped** chip or an amber **unmapped** chip, recomputed automatically as you edit.

- **Unmapped fields are excluded until you remap them.** A field that is unmapped, whether you added it without a mapping or it lost its mapping after a source change, is left out of the model, the preview, and every report until you point it back at a source column. Unmapped does not mean "author fills it in later", there is no calculated or literal field type; unmapped simply means excluded.
- **Field names must be unique within a view.** A duplicate display name is rejected and the edit reverts.
- **The Fields list is not paged.** Field order is set by dragging rows, and a drag cannot cross a page boundary, so every field is listed on one scrolling list however many there are. Other long lists in Reportworq page at 100 rows, this one deliberately does not.
- **Hidden fields stay queryable.** Hiding a field removes it from the report author's field list but keeps it usable as a filter reference, so you can filter on a key column without exposing it. Use **Hide all** or **Show all** to toggle visibility in bulk.
- **Rename many fields at once.** The **Rename** menu offers **Tidy names**, which turns source names such as `Account_FullyQualifiedName` into `Account Fully Qualified Name` and `Transaction_Type_Id` into `Transaction Type ID`, and **Edit names as text**, which lists every name one per line in field order so you can edit them in place or paste them back from a spreadsheet. Line 1 renames the first field, line 2 the second, and so on, so **Apply names** stays unavailable until there is exactly one non-empty, unique name per field. Both change display names only; reports bind to field ids and keep working, and **Discard** undoes a rename you do not want.
- **Filter on a variable.** A field filter's value can be a `{{variable}}`, so a report decides the value at run time. Reportworq writes the value into the source's query by the field's type, so it needs no quotes. To set its default and prompt, see Data model variables.

Where filters run

A model's field filters run **in the source's own query** wherever the source supports it, so the database returns only matching rows and less data crosses the wire. There is nothing to switch on; large sources and selective filters are fast by default.

Sources that cannot express a filter in their own query language, such as Planning Analytics, the planning-tool APIs and other non-relational connectors, filter the rows in memory after retrieving them. You get the same result either way.

> **Text comparison can differ slightly.** Because the database does the filtering, a text filter follows the database's own rules for comparing text (its collation), for example whether `maine` matches `Maine`. If a filter returns different rows than you expect, check how the source compares text.

Direct queries: modeled view or direct query

Direct queries is a checkbox on the Sources tab, labeled "Report authors can query this data model's sources directly", and it **defaults off**. It applies to the whole model, not to one source. Leaving it off keeps the model view-only: authors consume the curated, Public views. Turning it on lets a report author query the model's sources directly and bind to the query result's own columns, skipping the model's curated fields entirely. In the Excel add-in, the author gets one **New query** per source.

When to use each:

- **Use a Public view** for governed, reusable, change-resilient reporting. The field-mapping abstraction survives source schema changes, and the view is reusable across many reports and forms.
- **Use a direct query** for a one-off, well-understood, or exploratory pull, for example running a known SQL extract or inspecting a source before deciding whether to invest in a curated view. Direct-query results do not carry the field-mapping abstraction, so they do not survive source changes and are not centrally reusable.

Choose who sees a view: Public, Hidden or Disabled

Each view has a visibility, set in the view's header with the **Public**, **Hidden** and **Disabled** buttons. Hover a button for a short reminder of what it does. Each view has its own setting, so one model can hold a mix.

Visibility	Listed in the Excel add-in and web report pickers	Other views of the model can join it	Runs
Public	Yes, for everyone	Yes	Yes
Hidden	No, for anyone, authors included	Yes	Yes
Disabled	No	No	No. You can still edit it and run **Preview** in the data model editor

- **A new view starts Hidden.** Build and preview it, then set it to **Public** when report authors should find it in the Excel add-in or the web report studio.
- **Use Hidden for a building block.** A view that only feeds a joined view (a lookup, a mapping table) can stay Hidden: a Public joined view reads a Hidden view like any other, and nobody picks it by mistake.
- **Hiding a view changes the picker, not existing reports.** A report or job already bound to a view keeps running after you set it to **Hidden**, so you can take a view out of the add-in without breaking anything.
- **Use Disabled to stop a view from running.** A report, job, joined view or variable choice list that uses a Disabled view fails with a message that names it, for example "“CRM Accounts” is disabled". It never runs without the view's columns.
- **Disabling a view that something uses asks first.** When joined views or relationships in the model use the view, Reportworq lists them (the same list as **Used by** in the view's header) and asks before it disables the view. You can still go ahead.

> **Hidden is not a permission.** Hiding a view only leaves it out of the pickers in the Excel add-in and the web report studio. Anyone who can see the data model, through its folder permissions, can still run a Hidden view, readers included. For example, a web report shared with readers shows them the rows of the Hidden view it is built on. To keep data private, use folder permissions on the data model, or set the view to **Disabled**.

Upgrading from an earlier version. Views that were **Published** are now **Public**, and views that were **Draft** are now **Hidden**. Nothing that ran before stops running. Two things change:

- Authors no longer see draft (now Hidden) views in the Excel add-in. The rule is the same for everybody.
- Readers can now run views that were Draft. Before, a reader-only user could not run a report built on a draft view. Now a reader with access to the data model's folder can, for example through a shared web report. Before you upgrade, check that any draft view holding data readers should not see is protected by folder permissions, or set it to **Disabled** after the upgrade.

Other changes in this release. Keep on a relationship between views now belongs to each joined view, so changing it no longer changes other joined views; existing joined views keep the rows they kept. A key whose field is typed **Date** now matches as a date even when the source stores it as text, so a joined view can find more matches than before. Hierarchies are stored in the data model and can be built from a view's levels. See Join views across sources and Build hierarchies.

Rolling back. An earlier version does not know **Disabled**: it treats a Disabled view like a draft one, and runs it. Do not roll back past this version while views are Disabled, or first set those views to **Hidden** and expect them to run.

Limits

Know these before you rely on a data model:

- **A source is checked only for being there.** When the model opens, Reportworq checks that each connection is still in Integrations and enabled, and that each workspace file is still in the workspace. It does not sign in to the source or test that it can be reached, so a wrong password or an unreachable server shows only when a view runs.
- **SQL Server, OLE DB and ODBC sources supply no table relationships.** Add the links you need as manual relationships. See Table relationships.
- **Hidden is not a permission.** Anyone who can see the data model can run a Hidden view. Use folder permissions, or **Disabled**. See Choose who sees a view.
- **Older versions of Reportworq get some new settings wrong.** An older version runs a **Disabled** view, runs **Only in left** and **Only in right** joins as **Matches only**, and drops a **Levels** hierarchy if it saves the model. Upgrade every server that shares a data model, and don't roll back while you use them.

- **Planning Analytics, Analysis Services, Adaptive and Pigment are read as tables.** A view of a cube is flattened into rows and columns; a data model has no cube model of its own.
- **Joined views run in the server's memory**, within limits an administrator sets, and a Pigment or Planning Analytics input is always read in full. See Join views across sources.
- **Relationships between views are found from a joined view.** To see, edit or delete one, open a joined view that uses it.
- **Hierarchies work in web reports and Explore**, in crosstabs. They are not used by the Excel add-in's own reports, and a joined view can't carry one.
- **Checks read samples. Test match**, **Discover relationship**, **Explain** and **Preview tree** read the first rows of each view, not all of them.
- **The AI assistant reads your data.** It reads two sample rows from up to 40 tables of every source, a disabled connection included, and sends them to your AI provider. If an administrator has changed the assistant's instructions in **Settings** > **AI Prompts**, they keep that text: select **Reset to default** there to get the newest guidance, such as how to suggest hierarchies.
- **Variable values in Adaptive names are not checked.** A `{{variable}}` in an Adaptive Planning sheet, version, level or dimension name is filled in as given. Give such a variable a list of **Choices** with typing turned off.
- **No calculated fields.** A field must map to a source column; an unmapped field is left out until you remap it.

Notes

- The Preview tab runs the view fresh each time, with no cache. Report display and formatting (grouping, aggregates, sort priority, format modes) are authored in the report builder, in the Excel add-in or the web report studio, not in the model editor.
- To work on a model, open it with **Edit** from its row in the workspace. Someone without edit rights can open its **Properties**.

> **Going deeper.** To build a view on a SQL datasource, see Build a view with the designer and Table relationships. To combine views of different sources, see Join views across sources. For defaults, prompts, and how values are filled into a query, see Data model variables. For the limits of data models, see Limits. For the field labels and query types of a specific source, see Connect IBM Planning Analytics, Connect SQL Server, OLE DB, and ODBC, and the Connector catalog. To bind a model to an Excel workbook or SQLite database in this workspace with no Integrations connection, see Build a data model from a workspace file.

The web report studio

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A web report is a page of live Reportworq reports that you build in the browser. It looks and behaves like an Excel workbook: sheets, cells, formats and pictures. The numbers do not come from formulas in the cells. They come from the lists, crosstabs, charts and other reports you place on the page, and each one reads its data model when you open or refresh the report. The **web report studio** is the editor where you build it.

Web reports are for authors and administrators. People who only read reports never see them, and a web report is not something you send to a reader. Readers receive the static output of a distribution job instead. To turn a web report into something a job can run, see Convert to Excel.

When to use it. You want a management pack that shows today's numbers without opening Excel or waiting for a run. For example, a title, a crosstab of net sales by product line and year from the Sales Journals data model, and a chart, all on one sheet that fellow authors open from the workspace.

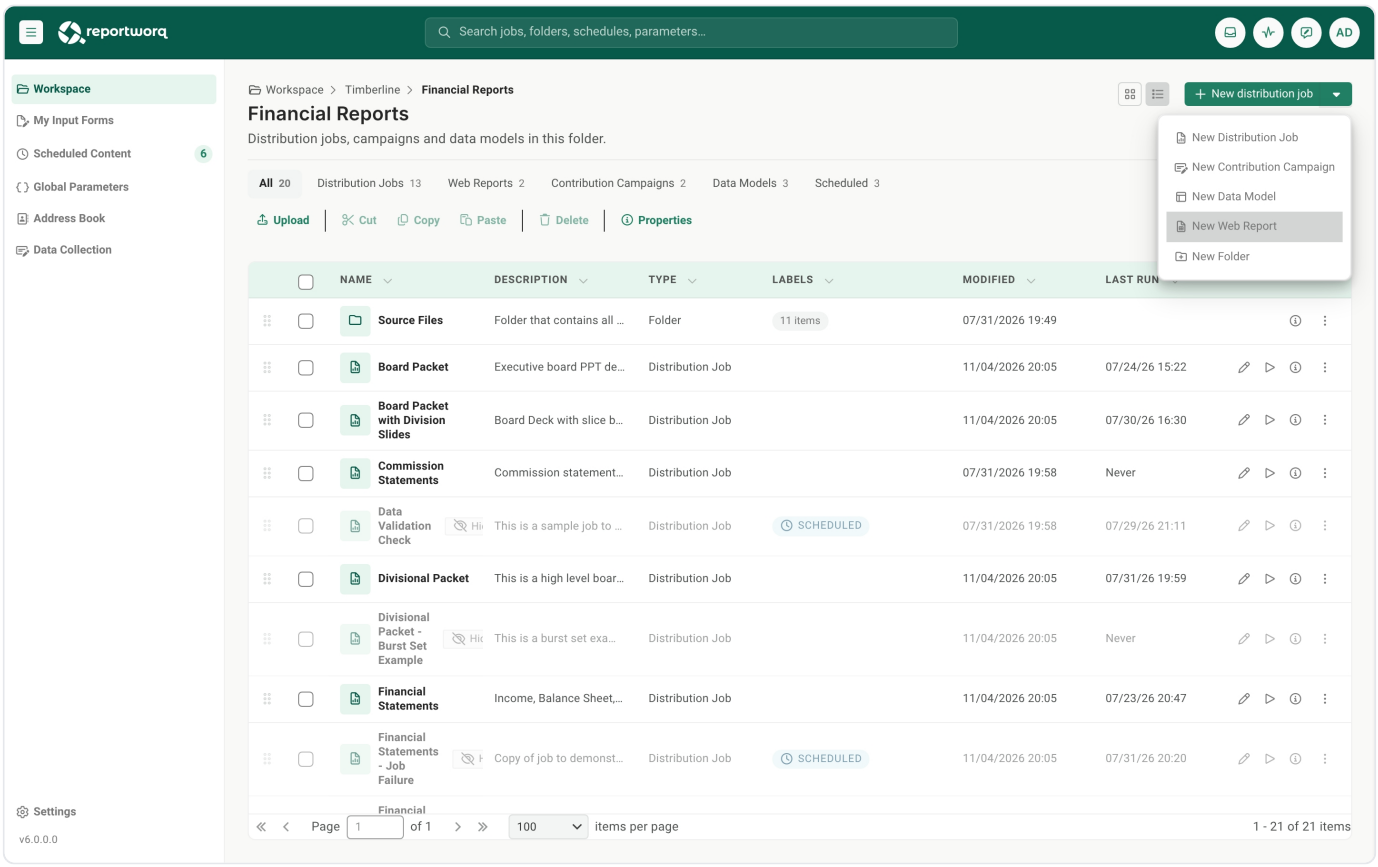
A web report needs no license of its own. Professional reports, which you can place on a web report, are license-gated and not released yet. See Professional reports.

Before you start

- You are signed in as an author or a system administrator, with permission to create content in the folder you want to use.
- A data model with at least one **Public** view you can see. A report can only be started from a Public view. See Data models and View visibility.

Create a web report

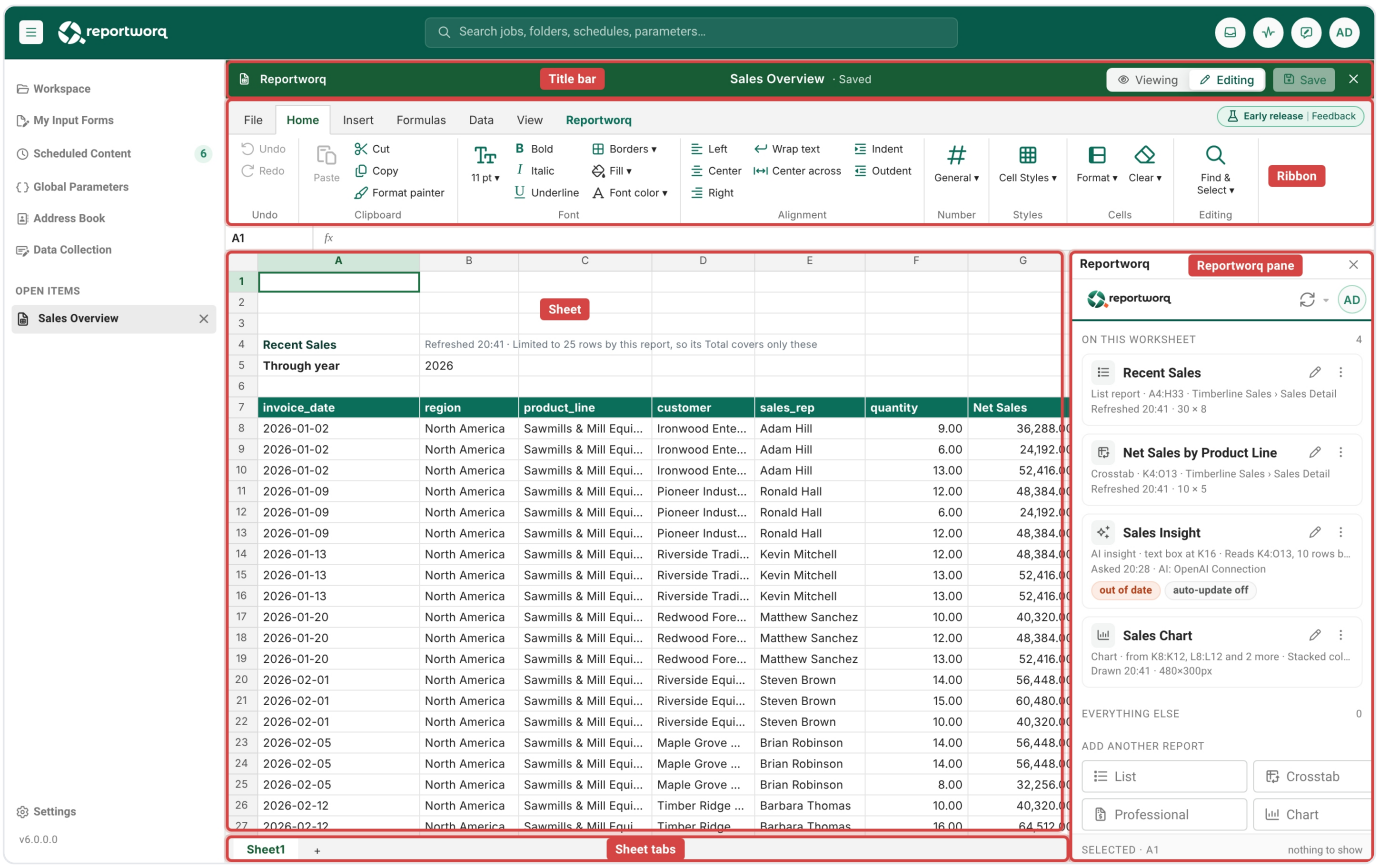
1. In the workspace, open the folder where the report should live.
2. Open the menu on the **New distribution job** button (its arrow) and select **New Web Report**.
3. In the **New Web Report** box, type a **Name**. The box starts with "New web report".
4. Confirm. The new report opens in a tab of its own, in **Editing** mode.



If a report with that name is already in the folder, Reportworq adds a number, for example "Sales pack (1)". A web report is an item like a distribution job: the folder's permissions decide who can see and change it, and you can move, copy, rename, delete, export and import it from the workspace the same way.

Find your way around

The studio opens with a title bar across the top, then a ribbon, a name box and formula bar, the sheet, a row of sheet tabs, and the **Reportworq pane** on the right.



Part	What it does
Title bar	Shows the report name. While you have changes that are not saved it also shows an orange dot, the words **Unsaved changes** and a **Discard** link. At the right are the **Viewing** and **Editing** switch, **Save**, and **Close** (the **x**).
Ribbon	Tabs named **File**, **Home**, **Insert**, **Formulas**, **Data**, **View** and **Reportworq**. When a report on the page is selected, a tab for that kind of report joins them.
Name box and formula bar	Type a cell reference and press Enter to go there; edit the active cell.
Sheet	The page you design. See "Sheets" below for its size.
Sheet tabs	One tab per sheet, with a **+** button to add a sheet.
Reportworq pane	The same pane that runs in Excel. It lists the reports on the page and is where you connect a report to a view, pick fields and set filters. Show or hide it with **Show task pane** or **Hide task pane** on the Reportworq tab.

The **Viewing** side of the switch shows the report as a reader of live reports would see it: current data, filters and drilling, but no way to change the page. See The viewer. Going back to **Editing** puts the report back as it was when Viewing began, so filters or expansions you tried while viewing are never saved by accident.

Put reports on the page

Click the cell where the report should start, then add it from the **Reportworq** tab (the **New** group) or the **Insert** tab (the **Reportworq** group). Both offer the same buttons:

Button	Builds	Read more
Build with AI	A report from a plain-language request	AI insights
List report	Rows from a view	List reports
Crosstab	Row and column groupings with a measure	Crosstabs
Professional report	A formatted, formula-driven report	Professional reports
Chart and **Map**	A picture drawn from a range on the sheet	Charts and maps
AI insight	A short written explanation of a report	AI insights

A new report appears at the active cell. If that cell is on a report already, the new one goes two rows below the lowest report on the sheet, so it never lands on top of another report. A new list, crosstab or professional report opens the pane on its **Connection** page so you can choose a view.

A web report can hold many reports of different kinds, on one sheet or several. See Combine report types.

Sheets

Each sheet is **1,000 rows by 52 columns** (columns A to AZ). The studio's page is smaller than a real Excel sheet on purpose. A list that would run past the last row stops at the sheet's end and says so. If you need a report that reaches further, build it in Excel; see The Excel add-in and the studio.

- **Add a sheet** with the **+** button at the right of the tabs.
- **Rename** a sheet by double-clicking its tab, or right-click the tab and choose **Rename**. A sheet name cannot contain `[ ] : * ? / \`, cannot be longer than 31 characters, and cannot start or end with an apostrophe.
- **Reorder** sheets by dragging their tabs.
- **Delete** a sheet with the **x** on its tab, or right-click the tab and choose **Delete**. A report always keeps at least one sheet that Viewing can show.
- **Hide** a sheet with **Hide sheet** on the **View** tab, or right-click the tab and choose **Hide**. A hidden sheet is left out of Viewing but still goes, hidden, into Excel downloads and job output. Choose **Unhide** to bring it back.

Formats

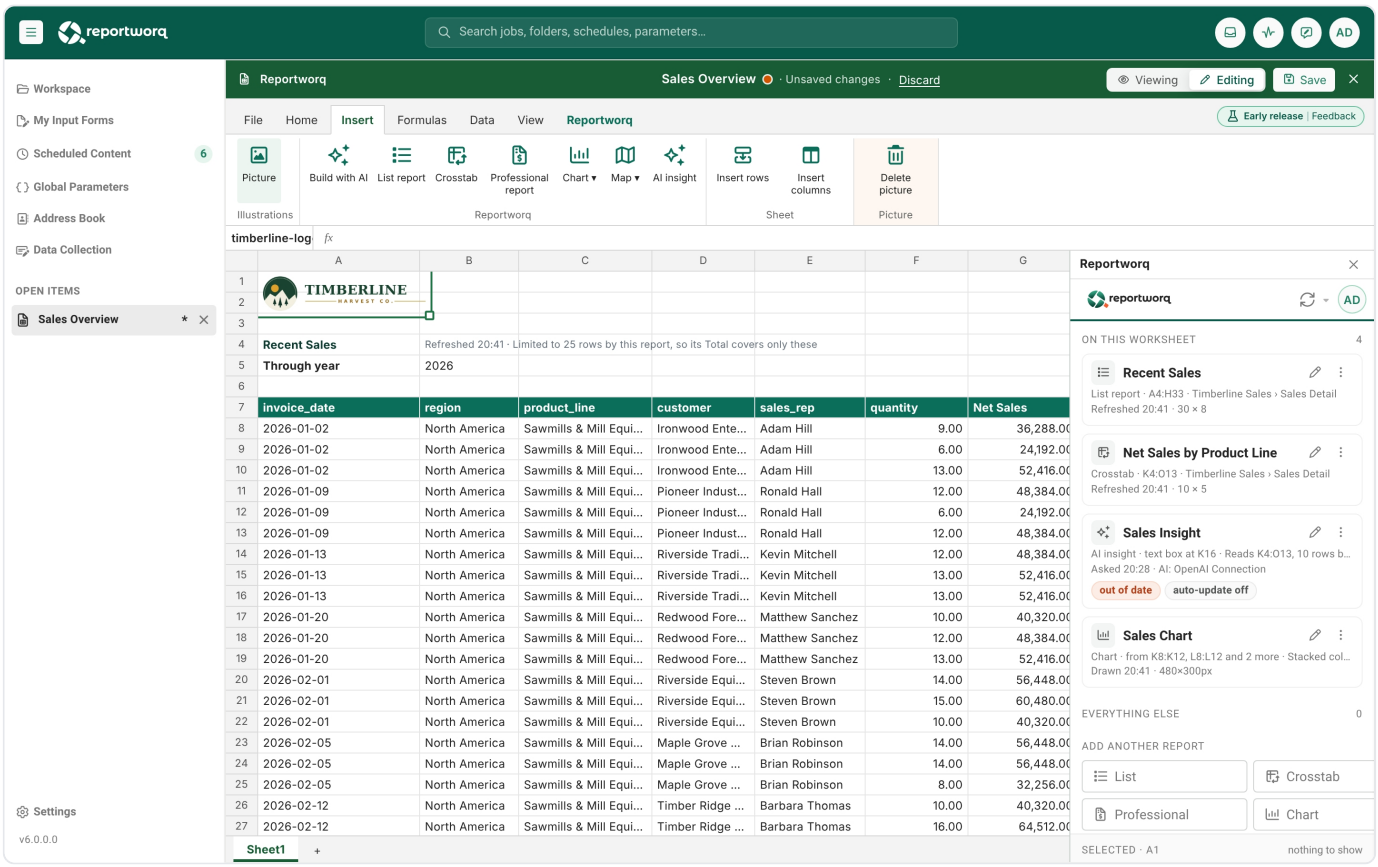
The **Home** tab works like Excel's. It has groups for **Clipboard** (including the format painter), **Font** (size, borders, fill and font color), **Alignment**, **Number**, **Styles** (**Cell Styles**), **Cells** and **Editing** (**Find & Select**).

- **Number** includes the usual formats plus **Millions (0.00M)** and **Thousands (0.0K)**.
- The **View** tab has **Gridlines** and **Headings** switches and **Freeze Panes** with **Freeze Top Row**, **Freeze First Column** and **Unfreeze**. Frozen panes are saved with the report and kept in the Excel download.
- **Format a report's cells the way you format any cells.** Select a cell in a list or crosstab and apply a format. Reportworq remembers it for that kind of cell and applies it to every cell like it, including after the report refreshes or grows. This is what keeps a column's number format when the data changes. To see where Reportworq keeps these formats, open the report's menu in the pane and choose **Show format range**; choose **Hide format range** to hide it again.

Add a picture

1. Select the cell where the picture's top-left corner should go.
2. On the **Insert** tab, in the **Illustrations** group, select **Picture** and choose a file.

The file can be PNG, JPEG, GIF, WebP or SVG, up to 2 MB. Reportworq checks what the file really is, not just its name. The picture opens at its own shape, scaled down to fit a 180 by 72 pixel box when it is larger. Every Excel, PDF and job output stretches a picture to its box, which is why the studio sets the size for you. Drag the picture to move it, and drag its corner to resize it.



To remove a picture, select it and choose **Delete picture** in the **Picture** group that appears on the **Insert** tab.

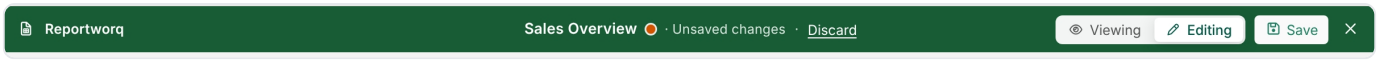
If a file is refused, a message under the ribbon says why, for example `"logo.png" wasn't inserted. The file is 3 MB; the limit is 2 MB.`

Save, Save a copy and Discard

A web report is saved only when you save it. There is no autosave.

- **Save** (or **Ctrl+S**, or **File** then **Save**) writes the report. The button is dim and reads "Saved: nothing to save" when there is nothing new.
- **Save a copy** (on the **File** tab) saves the report as a new item named like "Sales pack (copy)" in the same folder. If you have unsaved changes, they go into the copy, and this tab then edits the copy; the original stays as you last saved it. With no unsaved changes, the copy is made and you stay on the original.
- **Discard** (in the title bar, next to **Unsaved changes**) reopens the report as last saved, on the sheet you are on. Undoing every change by hand also clears the marker.
- **Close** asks first when there are unsaved changes: "Save them before closing? Discard closes the report as last saved." The choices are **Cancel**, **Discard** and **Save**. Switching to **Viewing** asks the same question. Leaving or reloading the page also gets the browser's own warning.

- **Undo** and **Redo** (**Ctrl+Z** and **Ctrl+Y**, or **Undo** on the **Home** tab) step back through cell, format, size, sheet and report-definition changes. They are not available while a report is refreshing.



Opening or refreshing a report never counts as a change. Only edits to the page or to a report's definition do.

A save counts only when the server reads back exactly what you wrote. If it cannot, the studio keeps your edits as unsaved and shows "The report wasn't saved" with the reason, so you never see "Saved" for a report that reopens empty. If someone else saved the same report between your write and the check, the message says it was saved again, perhaps in another tab, and your changes are still on screen: save again to replace that version, or reopen the report to see theirs. Two authors editing the same web report at once is not guarded beyond this, and the last save wins.

Refresh

The studio reads each report's data when you open it. **Refresh all** on the **File** tab re-queries every report; it is dim while a refresh is already running. While a refresh runs, the sheet shows "Refreshing. The sheet takes typing again when the reports are back." and does not take typing.

Move a workbook in or out

On the **File** tab, the **Excel** group has three buttons:

Button	What it does
Import workbook	Replaces what is on the page with an `.xlsx` file. A workbook that was downloaded as **Excel, connected** brings its reports back live; any other workbook comes in as a plain page. It asks first ("replaces everything in ... on screen") and nothing is saved until you choose **Save**.
Open workbook	Makes a new web report from an `.xlsx` in the same folder, named after the file, in a tab of its own. Your current report is not changed.
Export to Excel	Downloads this report as an Excel workbook.

A workbook for either import must be 25 MB or smaller. It may unpack to at most 200 MB, hold at most 10,000 files inside, and contain at most 100 pictures (5 MB each, 20 MB together). Hidden sheets and chart sheets are skipped. A sheet name Excel itself would refuse is renamed and the **File** tab tells you. If a workbook holds more than

100,000 cells past column AZ or row 1,000, it is refused with "This workbook holds more than the 100,000 cells outside A1:AZ1000 that the studio can keep: move or remove that content in Excel, then open it again."

In the workspace, each web report also has the row actions **Export as Excel** (a connected workbook saved beside the report, which stays as it is) and **Convert to Excel** (the same workbook, with the web report replaced). Convert cannot be undone; read Convert to Excel before you use it.

Notes and limits

- **What the report uses.** The report's **Properties** list its **Data models**, **Views**, **Connections** and **Reports on it**. These are worked out each time you save and cannot be edited. A distribution job that uses the report shows under **Used by**.
- **Finding web reports.** The workspace has a **Web Reports** filter, shown when the folder holds at least one. Web reports have their own filter, apart from **Distribution Jobs**.
- **Open as a full page.** On the **View** tab, **Launch in embed mode** opens the saved report in a new browser tab, showing only the report in Viewing. It is dim until you save, because it shows the report as saved.
- **Formulas.** Your own formulas in cells are calculated, but only for a common list of Excel functions. A function off the list shows `#NAME?` rather than a number that looks right. A formula that points past row 1,000 or column AZ shows `#REF!`.
- **Older files.** A report saved by a newer version of Reportworq will not open in an older one.

> **Going deeper.** To see how a web report and an Excel workbook relate, and which one to build in, see The Excel add-in and the studio. To send a web report's numbers to readers, see A web report as a job source.

The viewer

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

The viewer is how you read a web report. It draws the report as it would print, with live data, and lets you change what the report **shows** (pick a filter value, expand a row, look at the rows behind a number) without changing how the report is **built**. Nothing you do in the viewer is saved.

When to use it. A finance analyst opens a web report from the **Financial Reports** folder, picks her region in a filter, drills a variance to its source rows and copies them into Excel, all without opening Excel or asking the report's author for a new version.

Who can open a web report

Only authors and administrators can open a web report. This is by design: a web report is a working document for the people who build and analyze, and readers get the finished result another way.

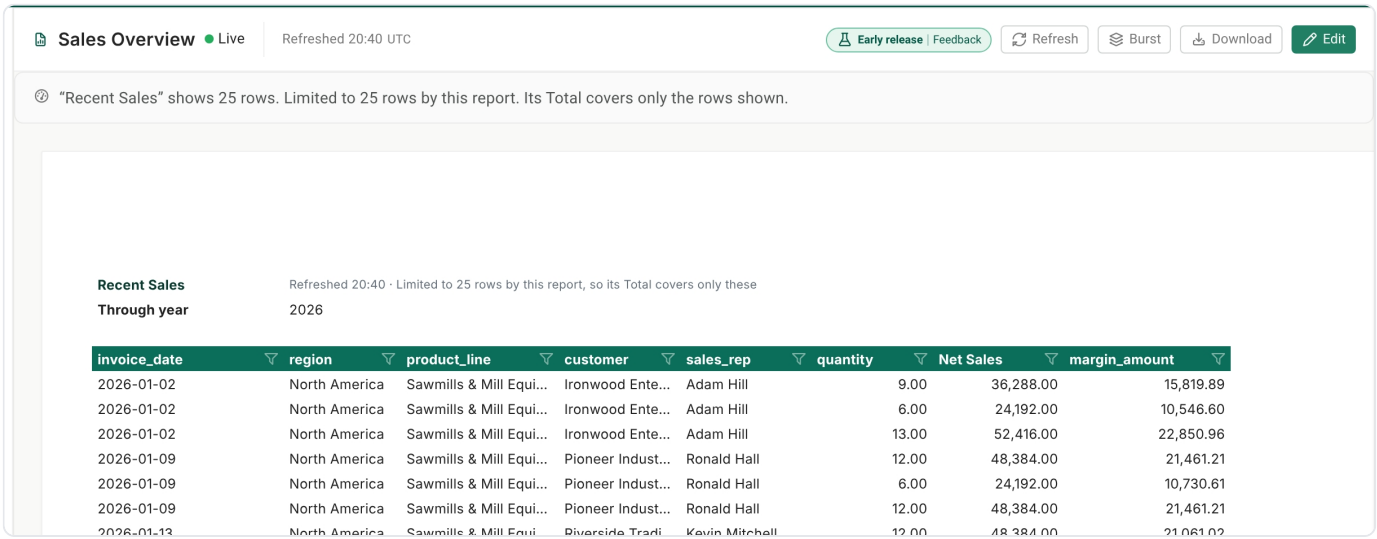
You are	What you see
Author or **administrator**	The web report in the workspace, in the viewer. The **Edit** button appears if you may change the report.
Reader	No web reports at all. They are left out of the folder tree, search and listings. A link to one lands the reader on the workspace root, and the report does not open. What a reader receives is the static output of the distribution jobs that distribute the report, such as the PDF or workbook in an email or the job's outputs.

To give readers a web report's content, put the report in a distribution job. See A web report as a job source.

Open the viewer

1. In the workspace, select the web report's name, or open its ⋯ menu and select **View**.

2. A report that already has content opens in the viewer. A report with nothing in it opens in the editor instead.
3. Wait for the status next to the report name to read **Live**. While it reads **Refreshing**, the report is reading its data again.



Opening a report always refreshes every report on it from its live source, so you see today's numbers.

The command bar

Control	What it does
Report name, **Live** or **Refreshing**	The report and its state.
Refreshed 15:42	When the report last read its data, on your own clock (24-hour), with your time zone beside it.
Unsaved edits	Shown to editors who came to the viewer from the editor with layout changes not yet saved.
View adjusted	You changed what the report shows. Nothing is saved.
Transactions	Shown when one number is selected. Lists the source rows behind it.
Copy	Shown when cells are selected.
Reset view	Shown after you adjust the view. Puts the report back as it was when you opened it.
Early release · Feedback	The early release badge, to the left of Refresh. See Early release features and Send feedback.
Refresh	Reads every report on the page again from its live source.
Burst	Runs the report once per value of a filter. See Download, burst and keep a view.
Download	Takes the report away as Excel or PDF. See Download, burst and keep a view.
Edit	Opens the editor. Shown only to people who may change the report.

A **Sheets** panel on the left lists the sheets when the report has more than one visible sheet. A sheet the author hid is not shown in the viewer, but it still goes out, hidden, in Excel downloads and job output.

Change what the report shows

Pick a filter value. A filter or variable on the page has a drop-down. Select its cell and pick a value. A variable that allows only its list of choices refuses a value that is not on the list.

Filter a column. A column heading of a list report, or of a filter the author set on plain cells, shows a funnel.

1. Select the heading's funnel.
2. Search the values, tick or clear them, or use **(Select all)**.
3. Select **Apply**, or **Clear** to put the author's own choices back.

On a list report the menu lists the distinct values of the list's source, not only the rows drawn on the page. It tells you when the list may be partial, for example "More than 500 values: only the first 500 are listed. A value past them stays in when you untick others." The viewer has no sorting.

Expand and collapse. Select the plus or minus glyph next to a row label. If there is no room to expand because another report sits right below, the viewer says so, for example: "There isn't room to expand “West” here: “Largest accounts” sits right below it. The report's author can make it insert rows as it grows."

Break a row down. On a crosstab row label, open the nest menu and choose **Break down by** a field. On a row that already has one, choose **Remove it**.

Look at the rows behind a number. Double-click a number, or select it and select **Transactions**. A dialog lists the source rows, up to 200, and says when there are more (for example "1,234 source rows · showing the first 200 · Copy rows copies all 1,234"). **Copy rows** copies every row with its headings. A cube crosstab number cannot be drilled, because a cube holds the number and not the rows behind it. A text cell or a heading says "This number has no source rows to show."

Copy cells. Drag over cells, then select **Copy** or press Ctrl+C. The cells paste into Excel as cells.

Load all rows. A list report that shows only some of its source's rows has a note above the page saying how many it shows of how many its source has. When the author set the report's **Rows read from the source** to the same number as its row limit, the note also carries a **Load all rows** button. A list limited only by **Limit rows** shows the note without the button. Select it to read up to the report's ceiling of 1,000,000 rows. If the source has more than that and the report needs every row, the report keeps its limit and tells you why.

Screenshot coming soon

A note above the page saying a list shows only the first rows of its source, with the Load all rows button

Refresh. Select **Refresh** to read every report again. An AI insight whose data changed because of your filter shows "Out of date: the report has changed since this answer was asked. Refresh to ask again." at the foot of its box. Refresh asks again.

Reset view. Select **Reset view** to undo every adjustment and go back to the report as you opened it.

Notes and limits

- **Nothing is saved.** The viewer never writes the report. Select **Edit** and the editor opens on the report as the author last saved it, not on what you adjusted. To keep a view you built, see Download, burst and keep a view.
- **Row limits.** A report reads at most 1,000,000 rows from its source. The sheet shows 1,000 rows and columns A to AZ; a list taller than that draws the rows that fit and says so. See Limits of a web report.
- **A report cannot be drawn.** The viewer shows the reason in place of numbers. Fix the source or the report's filters in the editor, or ask its author.
- **No Find button.** Use your browser's find (Ctrl+F) on the page.

> **Going deeper.** To build a web report, start with The web report studio. To understand where the viewer fits, see How reporting works.

A web report as a job source

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A distribution job can take a web report as its source report, the same way it takes an Excel workbook. Each run reads the report's live data, applies the job's parameters, and delivers static output, such as a PDF or workbook, to the recipients. This is how a web report reaches readers: readers never open the report itself, they receive what the job produces.

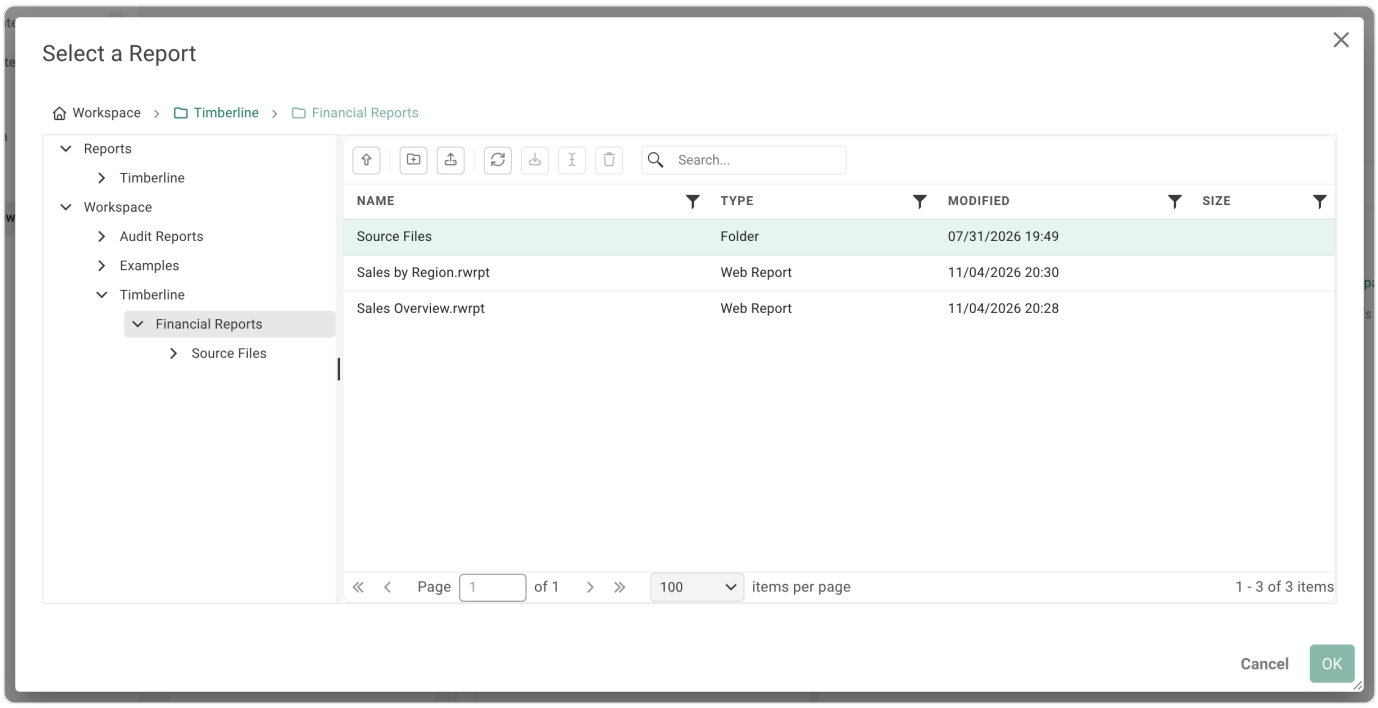
When to use it. The finance team builds a regional P&L as a web report. A controller adds it to a job, maps a Region parameter to the report's Region filter, and the job sends each regional manager a PDF with only their region.

Before you start

- You are an author or administrator, and the web report exists and is saved.
- You can create or change the job. If you have not built a job before, read Choose the source workbook and worksheets first. The steps here are the same as for a workbook.

Add the web report to a job

1. Open the job in the Distribution Job Editor and select the **Reports** step.
2. Select **Add a Report**.
3. In the **Select a Report** dialog, The dialog opens at the **Reports** root, so expand **Workspace**, then your folders (for example **Timberline** and **Financial Reports**), to reach the web report. A web report appears as a source named `<report name>.rwrpt` with the type **Web Report**.
4. Select it and select **OK**.
5. The report's sheets appear as a checklist. Tick the sheets the job should output and drag them into order, as for a workbook.



The sheet list shows the report as last saved, with no data read. The job reads live data when it runs.

Aim a parameter at the report's filters

On the job's parameters, a worksheet parameter of a web report offers the **report's own filters** instead of cells.

1. Open a sheet's parameter on the **Reports** step.
2. In the location picker, choose a filter from the **Report filters** column, listed first. The picker's note reads "This is a live report: pick one of its filters and each value runs the report filtered to it, or leave it blank if this value doesn't drive the report."
3. The field shows the filter by name, such as the filter's caption, and a second line says what it narrows and where (for example the sheet and report it sits on). A variable of the data model view reads "(variable)" after its name.

If a filter no longer exists in the report (it was renamed or removed), the editor says "The filter 'x' is no longer in this report (it was renamed or removed)..." and the run refuses it. Choose the filter again on the **Reports** step.

When several filters share a name, the picker says "Several filters are named Region: pick one from the list." Pick one from the list.

For burst sets and replicating a sheet per value, see Parameterize a distribution job and Burst with parameters.

How a job's value treats the report's own filters

A web report is a template for distribution, like an Excel workbook with a filter cell the job overwrites. A job replaces the report's filter on that field with the value, whatever the author ticked or left out. Filters on other fields stay as built. A blank value runs the report as built.

The value is not checked against the data. A value the source does not have, such as a typo, gives what the source returns for it, often a report with no rows. This differs from the viewer's burst, which only narrows within the author's own choices.

What a run does

- The job reads the report's live data as the job (an administrator-level read), not as a reader, then delivers static output in the formats you chose on the job. See Output formats and Destinations.
- Lists and crosstabs are refreshed. AI insights are asked again; if the ask fails, the saved answer is withdrawn and the run warns.
- The sheet is Excel's, not the studio's 1,000-row sheet. A list or crosstab that would pass Excel's last row fails its sheet with a message ending "Nothing was distributed."
- A report reads at most 1,000,000 rows, and 2,000,000 cells per report by default. A list that ranks or totals, and any data model crosstab, fails above that ceiling. The step log carries the reason.
- A run fails closed: if anything cannot be refreshed, the run says why and sends nothing, rather than sending stale data.

Professional reports

A job over a web report that holds a professional report is refused unless the license includes professional reports. The run ends with this message (the spelling is the product's own):

```
Nothing was distributed: the web report '<name>' uses a professional report, which this license does not include.
```

See the license note on Professional reports.

Notes and limits

- A web report converted to Excel runs as a workbook instead. See Convert to Excel.
- **Hidden sheets.** When a job sets a filter and a hidden sheet holds a report that the filter does not narrow, the run fails with "Sheet '' is hidden but its report “” isn't filtered by “”, so every recipient would get all of its data. Nothing was distributed."

Map the filter to that sheet too, or give the report that filter.

> **Going deeper.** The viewer's Burst panel is a lightweight way to produce packets by hand. A job is the way to deliver them on a timetable. See Download, burst and keep a view and Schedule a job.

REPORTING AND DATA MODELING · EARLY RELEASE

Early release features and Send feedback

reference author administrator

An **early release** feature is new and still being shaped. It works and you can use it, but its screens and behavior may change as the team learns from how you use it. Every note you send is read by the team building the feature.

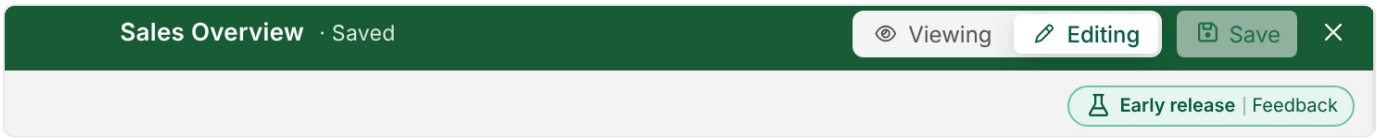
What is in early release

Feature	Where you see it
Data Models	The data model editor: a model's source connection, its views, and their fields and joins
Web reports	The web report editor and the viewer: a live workbook built on a data model
Excel add-in	The Excel task pane, including the analyst pane that is shared with the web report studio

These are the areas that carry the Early release pill today. Other screens in Reportworq are not marked.

The Early release · Feedback pill

On an early release screen you see a small pill that reads **Early release · Feedback**. It sits inside a row the screen already has, such as a toolbar or title bar, so it does not take up extra space. Select it to open the **Send feedback** dialog about that screen. Point to it to read: " is in early release. Found a bug or have an idea? Tell us."



You do not need the pill to send feedback. The **Send feedback** button sits next to your avatar at the top right of every screen. On an early release screen its tooltip reads "Send feedback: is in early release". In the Excel task pane the pill appears in the pane itself.

Send feedback

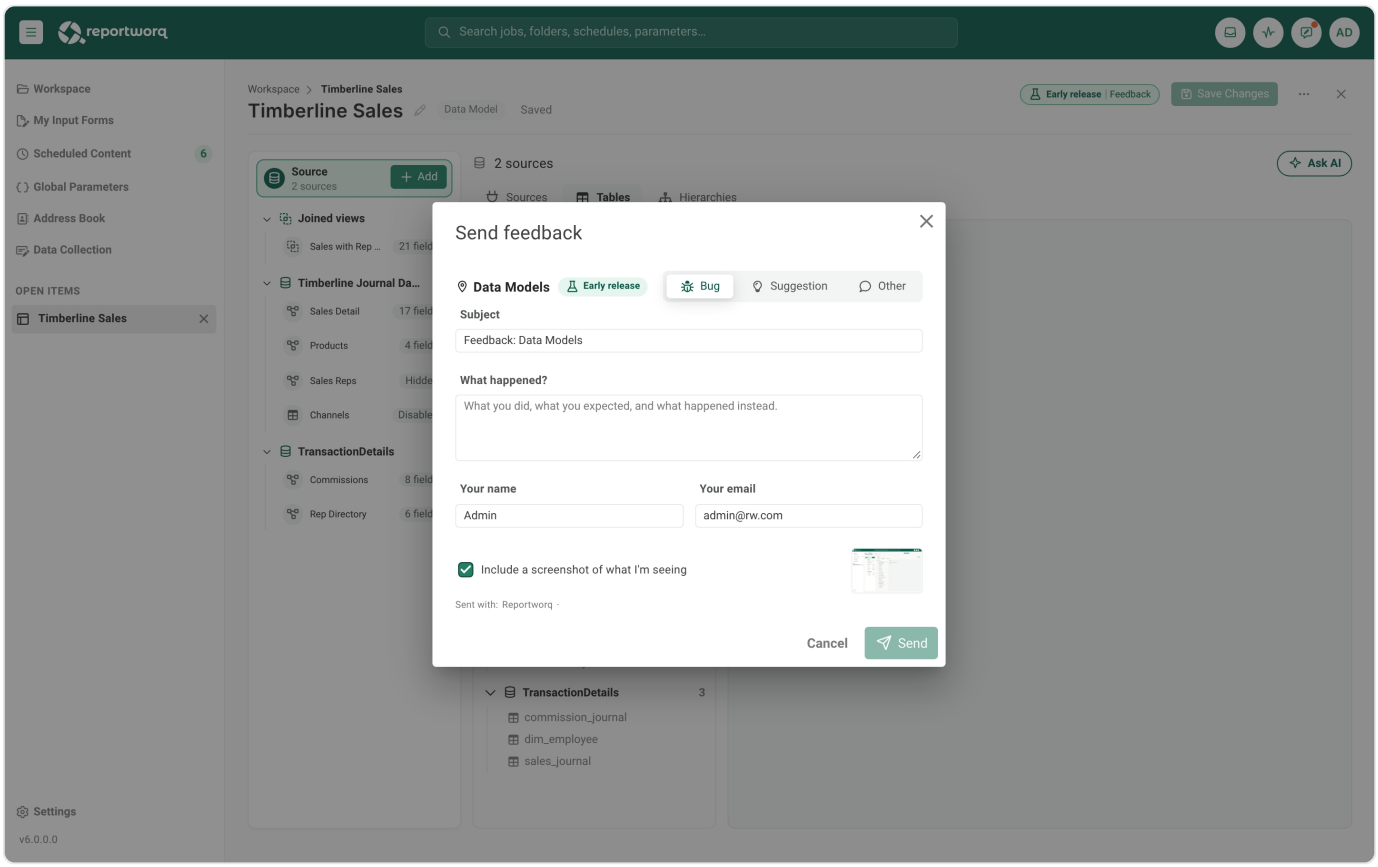
When to use it. You hit something that looks wrong, or you were trying to do a thing and it was harder than it should be. For example, a view in the Timberline Sales Journals model shows a field you did not expect. Send feedback from that screen and the team sees the screen, a picture of it and who you are, with no copying and pasting.

1. Open the screen the feedback is about, or select the **Early release · Feedback** pill on it.
2. Select **Send feedback** next to your avatar. The dialog opens at once. The heading shows the screen's name, with an **Early release** tag when the screen is in early release. In the web report studio the heading currently reads **Reportworq** with no tag, and the Subject starts "Feedback: Reportworq".
3. Choose the kind: **Bug**, **Suggestion** or **Other**.
4. Review the **Subject**. It starts as "Feedback: " followed by the screen's name. You can edit it, up to 200 characters.
5. Write your message. The label changes with the kind:

Kind	Label	Hint in the box
Bug	**What happened?**	What you did, what you expected, and what happened instead.
Suggestion	**What would make this better?**	What you were trying to do, and what would have made it easier.
Other	**What would you like to tell us?**	Anything at all.

   The message can be up to 8,000 characters.
6. Check **Your name** and **Your email**. They are filled in from your sign-in and you can edit them.
7. Decide about the screenshot (below), then select **Send**.

When it works, the dialog says "Thank you. Your feedback is on its way to the Reportworq team." Select **Close**.



The screenshot

Reportworq takes a picture of the screen you were on, leaving out the dialog itself and the Send feedback button. It is taken after the dialog appears, so the dialog never makes you wait. While it works you see "Capturing screenshot..." and a **Cancel** button; the form stays usable.

- Clear **Include a screenshot of what I'm seeing** to send without it. The preview dims.
- If you select **Send** before the screenshot is ready, you are asked "The screenshot isn't ready yet." Choose **Wait for the screenshot** or **Send without it**.
- If you canceled it, select **Take it again** to capture again.
- If none could be taken, the dialog says "No screenshot could be taken on this screen." and you can still send.

What goes along

Beneath the form a gray line reads **Sent with:** followed by your company and a masked version of your license key, or "no company on the license" and "no license key" when there are none. This is how the support team knows which customer and license the note is from. It cannot be edited. Along with your message and screenshot, the note carries the screen, the page address, your browser, and the Reportworq version.

Where it goes

Send uploads your feedback to Reportworq support as a support upload: a zip with your note, the details above and the screenshot. The support team reads it.

If the upload cannot be sent, the dialog stays open, keeps what you typed, and shows "Couldn't send to Reportworq. Nothing is lost; send it another way:" with three buttons:

Button	What it does
Zip	Downloads your feedback, its details and the screenshot as a zip
Text	Downloads your feedback as a text file
Email	Opens a new email to support@reportworq.com, filled in. Attach the zip to include the screenshot

The **Send** button changes to **Try again** after a failure.

The early release notice on documentation pages

A documentation page about an early release feature starts with this notice:

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

The last sentence links to this page. The link target is relative to the page that carries the notice. From a page in another section of this guide it is `../50-early-release/early-release-features-and-send-feedback.md`, and from a page in another guide it is `../../reporting-guide/50-early-release/early-release-features-and-send-feedback.md`.

A page without the notice describes a feature that is not marked early release, such as this one.

Notes and limits

- Feedback is about the screen you are on. If you open **Send feedback** from a screen with no pill, it is about the page you are on.
- The screenshot is a picture of your screen, so it can show your data. Clear **Include a screenshot of what I'm seeing** if the screen shows something you do not want to share.

- For help with a problem you need solved, contact support. Feedback is read to improve the product and may not get an individual reply.

See also: How reporting works.

Troubleshooting and limits

reference author administrator

This page collects the numbers and the messages that the rest of the Reporting and Data Modeling guide mentions one at a time. Find the area you are in, match the message you see to the text in quotes or a code span, and read across to what it means and what to do.

Where a message is quoted, it is the product's own wording. Names in angle brackets, such as `<name>`, stand for your own report, sheet or filter.

Find your problem

You are working in	Go to
Any size, row or cell ceiling	Limits
A data model variable that is "refused"	Refused variable values
A control that says "coming soon", or a job that names the license	License messages
A view, a joined view or a global parameter	Data models
The web report studio	The web report studio
The viewer, a download or a burst	The viewer, downloads and burst
Convert to Excel	Convert to Excel
A distribution job whose source is a web report or a converted workbook	Jobs over web reports

Limits

Every number below is a limit of the product as it ships. An administrator can change the ones marked as settings. Nothing is cut silently: a report that passes a limit is refused with a message that says so.

The sheet and the report

Limit	Value	Notes
Rows on a web report sheet	1,000	A list taller than this draws the rows that fit and says so.
Columns on a web report sheet	A to AZ (52)	A formula that points past row 1,000 or column AZ shows `#REF!`.
Where a report may sit in the Excel add-in	Up to XFD1048576	Current Excel for Windows, Mac and the web. An older or unrecognized client works in A1:AZ1000 as the studio does.
Where a report may end when a job refreshes it	Row 1,048,576 and column XFD	A job uses Excel's sheet, not the studio's.
Sheet name	31 characters	It cannot contain `[ ] : * ? / \` and cannot start or end with an apostrophe.
Largest report	2,000,000 cells by default	A setting: **Settings** > **Configuration** > **Excel add-in** > **Largest report (cells)**, from 1 to 5,000,000. A report over it is refused, never cut.
A render held for confirmation	Above 100,000 cells	A setting, **Ask above (cells)**, in the same place. A list or crosstab over it asks before it writes.
Chart kinds and map kinds	13 and 2	Chart pictures from a job are drawn at three times the on-sheet size.
A map place	Latitude -90 to 90, longitude -180 to 180	A place outside is left off the map and the Places page counts them. A place with a blank latitude or longitude is kept but not drawn.

Reading data

Limit	Value	Notes
Rows a report reads from its source	1,000,000	A list that ranks, filters or totals, and any data model crosstab, is refused above it. A plain list shows its first rows and says so. **Load all rows** in the viewer reads up to this ceiling.
Rows behind a number in the viewer	200 shown	**Copy rows** copies every row.
Distinct values listed in a column menu or a filter	500	The menu says "More than 500 values: only the first 500 are listed. A value past them stays in when you untick others."
Rows a hierarchy reads	1,000,000	Past it the field is shown flat and the crosstab says so. A hierarchy is never built from only some of the rows.
Members in a hierarchy preview tree	500	
Rows in the table browser on the Source screen	1,000	
Rows the view Preview grid draws	5,000	Copy and download take every row that was fetched.
Rows to check how views match	10,000 per view	**Discover relationship** uses 1,000 per view.

Saving, importing and pictures

Limit	Value	Notes
A saved web report	25 MB	Counted in bytes, as stored. A larger report is not saved, and your edits stay on screen.
A workbook to Import or Open	25 MB	It may unpack to at most 200 MB and hold at most 10,000 files.
Pictures in an imported workbook	100, at most 5 MB each and 20 MB together	
Cells outside A1:AZ1000 in an imported workbook	100,000	More than that and the workbook is refused.
A picture you insert	2 MB	PNG, JPEG, GIF, WebP or SVG. A tile map background can be PNG, JPEG, GIF, WebP or BMP, and not SVG.

Viewer, download and burst

Limit	Value	Notes
Parts in one download burst	60	Past it, make it a job.
Combinations listed to page through	1,000	Past it, the panel shows only the count.
Value chips shown	12	The panel says "and N more".
Values a burst filter offers	500	

AI

Limit	Value	Notes
Rows an insight sends to your AI provider	200 rows or 24,000 characters	If the data is cut, the request says so and the total row count is still sent.
Rows **Fill in from** sends	12	All of their columns.
Tables the data model assistant lists per source	80	It reads two sample rows from up to 40 tables per source, or 12 for a cloud connector.

Data models

Limit	Value	Notes
Rows read per joined input	1,000,000	A setting: **Settings** > **Configuration** > **Performance** > **Data Models** > **Rows Read per Joined Input**. 0 turns it off.
Joined view memory budget	20,000,000 cells	**Joined View Memory Budget**, same place. About 1 GB.
Rows a joined view returns	2,000,000	**Rows Returned by a Joined View**, same place.
Join keys sent to a source	1,000	**Join Keys Sent to a Source**, same place. Above it, that input is read whole.
Tables in one designer view	20	A path from the starting table is at most four tables long.
An export of a source	Refused past 10,000,000 cells	A table too big for a sheet is cut to fit, never refused.

For the details behind a row, follow the page that owns it: The web report studio, The viewer, Download, burst and keep a view, Join views across sources, Hierarchies and AI insights.

Refused variable values

A data model variable's value is placed into a query by the position of its `{{Token}}`. A value that does not fit its position is refused: the run fails, nothing is sent to the source, and the message names the variable. The full rules are in Data model variables.

If you see this message for a bare token in SQL:

> The value for “Region” can't be used: it is pasted into the query without quotes, so it only takes a number (or, inside IN ( ), a list such as 1, 2, 3 or 'East', 'West'). For a text value, the data model's author should put quotes around it in the query: '{{Region}}'. Only as a last resort, for query text such as a column name, can the data model's author mark the variable as raw query text, which turns this protection off and pastes readers' values unchecked.

then the token sits bare in the query (`WHERE region = {{Region}}`), so it only accepts a number. The data model's author puts quotes around it: `WHERE region = '{{Region}}'`. A bare date such as `2024-01-01` fails the same way and is quoted the same way.

If you see the same message with a list, such as `East, West`, then write `IN ({{Regions}})` and supply `'East', 'West'`. A list is accepted only directly inside parentheses, and each text item needs its own quotes.

If the message asks for brackets or double quotes instead, then the view is MDX or DAX:

Language	The message says the value "is pasted into the"	The author's fix
MDX	`MDX without quotes or brackets, so it only takes a number or member names such as [Region].[East]`	Put brackets around it: `[{{Region}}]`, or `[Region].[{{Region}}]`.
DAX	`DAX without quotes or brackets, so it only takes a number`	Put double quotes around it: `"{{Region}}"`.

If the message ends "so on this connection a variable can only take a plain number unless the author marks it as raw query text. Rewrite that part of the query, or connect with a SQL Server or SQLite connection", then the query uses syntax that different databases read differently (a backslash, a `#` or `//` comment, `$` quoting, brackets, a lone carriage return, or `EXEC` of a string) on an ODBC, OLE DB or CData connection, or anywhere it runs a string as SQL. Rewrite that part of the query, or use a SQL Server or SQLite connection. Pass values to dynamic SQL as `sp_executesql` parameters.

If the value is meant to be query text, such as `ORDER BY {{SortColumn}}`, then the author selects **Raw query text, pasted as is** on the variable's card, and limits the values to a list. The value is then pasted unchecked, so treat it as a power tool. See Use a value as query text.

If you see no error but no rows, and the query reads `IN ('{{List}}')` with a value like `East','West`, then the value is now one text string. Write `IN ({{List}})` with the value `'East', 'West'`.

If Preview shows "No value for required variable(s)" and names the variable, then enter a value in the **Variable values** pane, or set the variable's default on the **Variables** tab. At run time a required variable with no value is refused with a warning that names it.

If Preview says "East2 isn't one of the choices for Region. Pick a value from its list.", then the variable allows only its list. Pick from the list, or have the author turn on **Also allow typing a value that isn't on the list**.

License messages

Web reports themselves need no license. Professional reports inside them do, and they are not released yet. Without the license, the controls stay on screen, disabled, and say why.

If you see	Then
`Professional reports are coming soon.` on **Professional report**, **Convert to professional**, **Add dynamic row**, **Add row member** or **Add column member**	The server's license does not include professional reports. Nothing is created. A web report that already holds one opens with its value cells blank, only in that session, and cannot be saved, saved as, exported or downloaded. The stored report is unchanged. Ask your Reportworq contact for the license. See Professional reports.
`Nothing was distributed: the web report '<name>' uses a professional report, which this license does not include.` as a job's failure	A distribution job over a web report that holds a professional report is refused on an unlicensed server. The job keeps no file and sends nothing. A workbook source reads "the workbook" in place of "the web report". Get the license, or take the professional report out of the report the job uses. See A web report as a job source.
`Professional reports are coming soon.` when you choose **Convert to Excel**	The web report holds a professional report and the license does not include them. Nothing is made and nothing is deleted.
`Professional reports aren't in Excel yet.` on the professional report buttons in the Excel add-in	This is not a license message. Professional reports are built in the web report studio only. A toast adds "Build them in the web report studio."

Professional reports do not show in the **Capabilities** block of the License tab, so these messages are the signal. A license change applies without a restart. See Licensing.

Common errors

Data models

If you see	Then
`“<name>” is disabled. Set it to Public or Hidden in its data model to run it.` from a report, job, joined view, variable choice list or Preview	The view's visibility is **Disabled**, which stops it from running everywhere. Open the data model and set the view to **Public** or **Hidden**. A joined view over a Disabled input fails the same way even in Preview. See Choose who sees a view.
A view missing from the list in the Excel add-in or the studio	It is **Hidden** or **Disabled**. Neither shows in the pickers. Hidden is not a permission: anyone who can see the data model can still run a Hidden view.
`A view named “<name>” already exists. View names must be unique.`	View names are shared across the whole model, because every report binding finds a view by name. Pick another name.
The view reads "Fields may not match the query yet"	The query could not run when you left the tab or saved. Read the warning, fix the query or enter the missing sample value under **Connection values**, then change tab or save again.
An older view opens as **SQL only** with an empty editor	It was built in a way the designer cannot show. Select **Use the designer** to see why. The view still runs as saved.
A joined view stops with a message naming an input and a limit	A joined view went past one of its four limits (see Limits). Filter the input view, give it a variable, remove fields you do not need, or check **Each row finds** and run **Test match**: a key that is not unique repeats rows.
A joined view fails when one of its sources is down	A joined view needs every source its inputs read. Restore the source, or point the input at another.
A source shows an exclamation mark in the rail	The connection was deleted or disabled, or the workspace file was moved, renamed or deleted. The row on the **Sources** tab says how to fix it. Use **Change...** on the row to pick the file again.
A hierarchy preview reads **Reports can't use this yet**	The outline has a problem listed with its line number: a top-level line with nothing under it, a value listed twice, or an indent that is not two spaces or one tab per level. Fix the listed lines.
`hierarchy “<name>” (its view has more than 1,000,000 rows) unavailable, shown flat` on a crosstab	The hierarchy's view has more rows than a report reads. Narrow the view or give it a variable.
`Global Parameter Value '<name>' refers to another global`	A global parameter cannot point to another global parameter. This is an older entry that already did. Open

If you see	Then
`parameter ('<other>'). A global parameter must hold its own value: edit it in Global Parameters and choose a value type other than 'Global Parameter'.` on a job or prompt that uses it	it on the **Global Parameters** screen and choose a value type other than **Global Parameter**. The list flags such an entry in its **Value** column, and the edit panel says "This global parameter refers to another global parameter, '', which is not supported. Choose a value type for it." A global parameter that points at itself reads "refers to itself." See Global parameters.
A `.db` upload is refused with "These file types are not permitted..."	An upgraded install keeps its upload allow-list. An administrator adds the **Databases** file-type category. See Workspace file data models.
"No AI model is set up. An administrator can add one in Settings, Integrations, AI."	The data model assistant needs an enabled AI connection. See Ask AI on a data model.

The web report studio

If you see	Then
"The report wasn't saved" with a reason	The server could not read back exactly what you wrote. Your edits stay as unsaved. Save again. If the message says it was saved again, perhaps in another tab, someone else saved between your write and the check: save again to replace that version, or reopen to see theirs. The last save wins.
`the report is <n> MB, over the 25 MB a saved web report can be. Nothing was saved, and your changes are still on screen. Take a sheet, a picture or a large list out of it, then save again.`	The saved report is over 25 MB. Remove a sheet, a picture or a large list. A full sheet list of 52 columns by 1,000 rows is about 10 MB.
`"logo.png" wasn't inserted. The file is 3 MB; the limit is 2 MB.`	A picture is over 2 MB. Use a smaller file. A file that is not really a picture is refused too, because Reportworq checks what the file is, not its name.
"This workbook holds more than the 100,000 cells outside A1:AZ1000 that the studio can keep: move or remove that content in Excel, then open it again."	An imported workbook has too much past column AZ or row 1,000. A workbook over 25 MB is refused too, with "The file is MB; the limit is 25 MB."
`#NAME?` in a cell	Your formula uses a function the studio does not calculate. The studio shows an error rather than a number that looks right.
`#REF!` in a cell	The formula points past row 1,000 or column AZ.
"Edit it in Excel." on a report in an imported workbook	The report reaches past row 1,000 or column AZ. The studio shows the first 1,000 rows and 52 columns and will not change the report. Saving keeps its stored definition as it was.
A report is refused with `“<view>” has more than 1,000,000 rows, which is more than a report reads at once ... Nothing was cut.`	The report needs every row and the source has more than the ceiling. Use a data model variable or a filter on the view in the data model (a filter on the report applies after the rows are read), or set **Rows read from the source** on the report to 1,000,000 or fewer, which makes it work from only the first rows.
A report is refused for passing the largest report size	The report would write more cells than the **Largest report (cells)** setting allows. Limit the rows, or ask an administrator to raise the setting (to at most 5,000,000).

If you see	Then
A report opens with no numbers and a reason in their place	The report cannot be drawn. Fix the source or the report's filters, or ask its author.
The sheet shows "Refreshing. The sheet takes typing again when the reports are back."	This is normal while **Refresh all** runs.
A report saved by a newer version will not open	A report saved by a newer version of Reportworq cannot be read by an older server. Upgrade the server.
"The Excel Addin requires the Author entitlement."	The sign-in has no author rights. Readers cannot open a web report or use the pane. Sign in as an author or administrator.
A variable bound to a cell refuses with a "Try again" message naming the variable	Excel cannot be asked right now (a failed read, edit mode, writes still queued). Try again. It never filters on an old value.
"No row of the source has this value" beside a variable	No row has that value. Pick another.
A report filter on another sheet's cell comes out empty	A filter reads a cell on its own sheet only. Have the job write the value into a cell on the report's own sheet.
"Unverified figures: ... Check them before you rely on this answer." on an AI insight	Figures in the answer could not be found in the data. Check them before you send. The mark means nobody checked, not that the figure is wrong.
`AI insight not refreshed in this run.` or `AI insight unavailable for this filtered report.` in a delivered file	The insight could not be asked again (no AI connection, a disabled connection, a timeout or a refused answer), so the saved answer was withdrawn. The run completes with a warning that names the insight and the reason.

The viewer, downloads and burst

If you see	Then
"There isn't room to expand “West” here: “Largest accounts” sits right below it. The report's author can make it insert rows as it grows."	Another report is directly below. Ask the author to set the report to insert rows as it grows, or to leave room.
"Out of date: the report has changed since this answer was asked. Refresh to ask again." under an AI insight	Your filter changed the data under the answer. Select **Refresh**.
A note above a list with a **Load all rows** button	The author limited the rows the list reads. Select **Load all rows** to read up to 1,000,000. If the source has more and the report needs every row, it keeps its limit and says why.
A variable refuses a value	The variable allows only its list of choices. Pick a value from the list.
`A download bursts at most 60 parts, and this is <n>. Narrow the values, or make it a job.`	The burst has too many combinations. Narrow the values, or run it as a distribution job.
" has more than 500 values. The list shows the first 500."	A burst filter offers only the first 500 values.
`The burst didn't work:` followed by a reason	The burst failed. A hidden sheet whose report the burst filter does not narrow fails the burst, so no recipient is handed the whole data. Narrow that report by the same filter, or unhide the sheet.
Cannot find a web report as a reader	By design, readers never see web reports. They receive the output of distribution jobs. See The viewer.

Convert to Excel

Converting deletes the web report and cannot be undone. Reportworq builds the workbook first, and if it cannot carry the report whole it stops and nothing changes. See Convert a web report to Excel.

If you see	Then
"This web report is used by jobs you can't see, so it can't be converted. Ask an administrator to convert it."	A job or campaign you cannot change uses the web report. Every user's job is repointed or none is. Ask an administrator.
A message that begins ""“” bursts by setting this web report's filters"	A job maps its parameters to the web report's own filters, which a workbook does not have. Map the parameters to workbook cells, or remove them, then convert.
"The job “” sets this web report's filter “”, and the hidden sheet “” holds the report “”..."	A hidden sheet is refreshed too, so each of its reports must be narrowed by every filter the job sets. Map the filter to that sheet and give the report that filter, or unhide or remove the sheet. A map or professional report on a hidden sheet cannot be narrowed at all: move it to a visible sheet or remove it.
`Professional reports are coming soon.`	See License messages.
A message naming what would be lost or moved	The workbook is checked before anything is written. Fix what the message names, then convert.
`Converted to “<file>”.` and "One job now runs it." or " jobs now run it."	Success. The workbook is in the same folder and each job now refreshes it.

If you only want a copy, use **Export as Excel** on the same menu. It leaves the web report and the jobs alone.

Jobs over web reports

A job over a web report or a converted workbook fails closed: if anything cannot be refreshed, it says why and sends nothing, rather than sending stale data. Run messages from a web report source begin "Live report '':". Each ends "Nothing was distributed."

If you see	Then
"The filter 'x' is no longer in this report (it was renamed or removed)..." in the job editor	The job maps a value to a filter that is gone. The run refuses it. Choose the filter again on the **Reports** step.
"Several filters are named Region: pick one from the list."	Two filters share a name. Pick one in the location picker.
"Sheet '' is hidden but its report “” isn't filtered by “”, so every recipient would get all of its data. Nothing was distributed."	A hidden sheet holds a report the job's filter does not narrow. Map the filter to that sheet too, or give the report that filter.
A message ending "Nothing was distributed." about a list or crosstab passing Excel's last row	The report would pass row 1,048,576. Reduce what it reads.
"`<cell>` is in rows `<first>`-`<last>`, which “`<report>`” keeps for its formats (hidden), so a refresh would overwrite it. Nothing was distributed. Move it below row `<last>`, then run the job again."	Something typed into the hidden rows under a report. Move it below the report.
A message naming a cell a growing report "would write over"	The report would grow into your cells. Move your content, or set **When the report grows** to **Overwrite**, which lets the report take the cells.
`“<report>” has grown larger than a job can refresh in place (it must end by row 1,048,576 and column XFD). Nothing was distributed.`	The report would pass Excel's last row or column. Reduce what it reads.
An insert or delete "would cut through a merged area"	Unmerge the area or move it.
A report definition that "can't be read (damaged, edited by hand, or saved by a newer version of Reportworq), so it can't be refreshed. Convert the web report to Excel again, or remove the report from the sheet."	Do not edit a report's hidden definitions by hand. Convert the web report again, or remove the report from the sheet.
A step log reason that the report reads more than 1,000,000 rows, or 2,000,000 cells	A list that ranks or totals, and any data model crosstab, fails above the ceiling. Narrow the view with a variable or a filter in the data model.
`Nothing was distributed: the web report '<name>' uses a professional report, which this license does not include.`	See License messages.

A job does not check a value against the data: a value the source does not have, such as a typo, gives what the source returns for it, often a report with no rows. Maps and

professional reports go out as last saved, so a job that narrows a web report refuses a map it would send whole to every recipient.

For the rest of how a job treats a web report, see A web report as a job source and Key-cell formats and what a job keeps.

Still stuck

Early release features can behave differently from the pages that describe them. Use **Send feedback** from the early release pill on the screen you are on, which attaches a screenshot and the details support needs. See Early release features and Send feedback. For problems outside reporting, see the Troubleshooting index.

Bring several sources into one model

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A data model can read **any number of sources** at once. A source is either a **connection** from Integrations (a SQL database, IBM Planning Analytics, a SaaS system) or a **workspace file** (an Excel workbook or SQLite database that lives in the same workspace as the model). You can mix both kinds, so actuals from a database, a forecast from a planning cube and a lookup workbook can feed one model, and report authors see them as one list of views.

When to use it. Put sources in one model when people will report on them together or pick from them in the same place. At Timberline Harvest Co., the **Sales Journals** model reads the **TransactionDetails** workbook. Add the **Timberline Journal Data** database and the **TM1** connection to the same model, and a **joined view** can set invoiced sales beside a planning forecast. If two sources have nothing to do with each other, two models are tidier: each model has its own folder permissions, its own variables and its own list of views.

Before you start

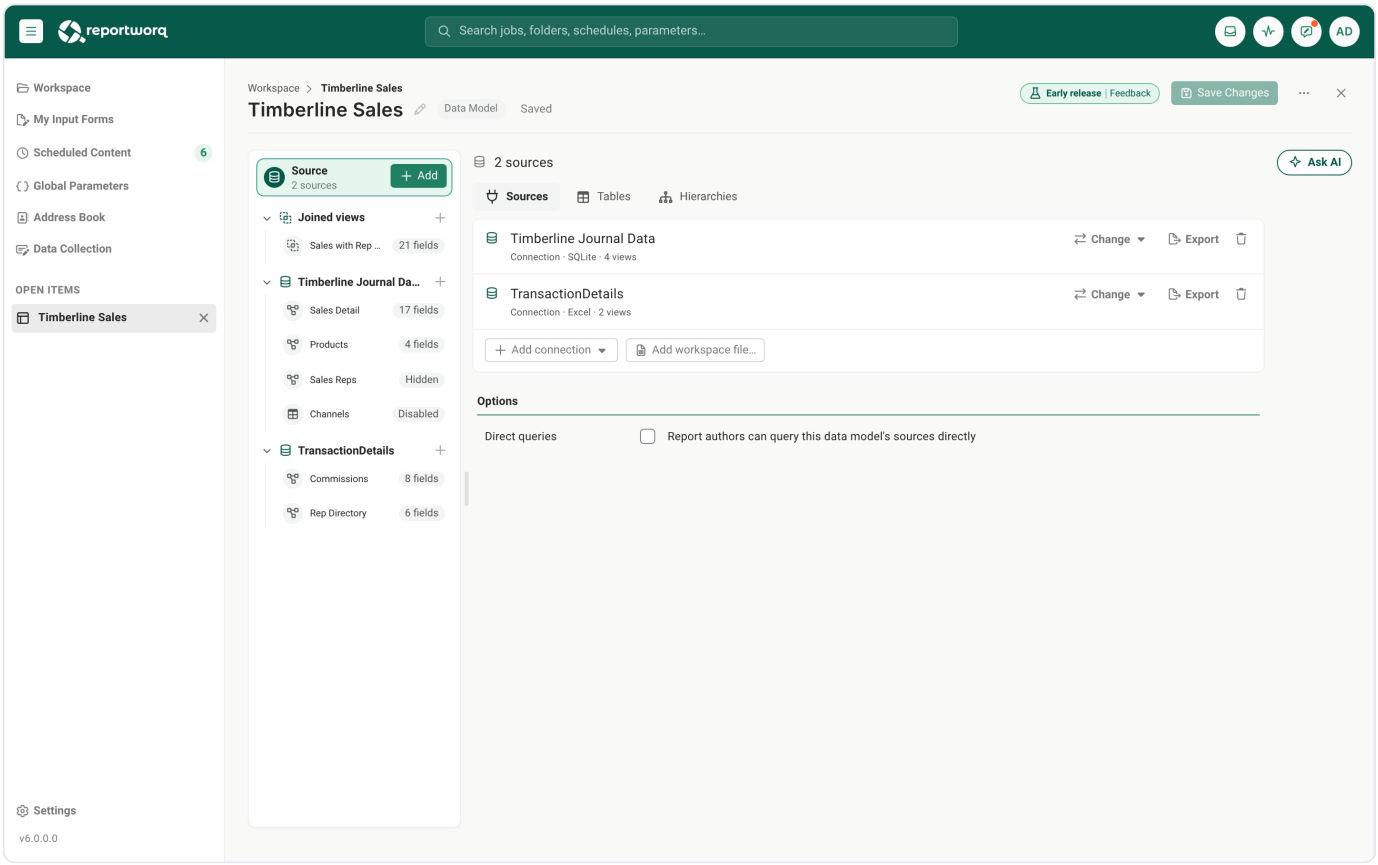
- You can author data models in the workspace. Open a model with **Edit** from its row in the workspace, or create one with **New** > **New Data Model**.
- For a connection: an administrator has added it in Integrations, it is enabled, and it is available to this workspace. See Connect a data source.
- For a workspace file: an Excel workbook (`.xlsx` , `.xlsm`) or SQLite database (`.db` , `.sqlite` , `.sqlite3`) that you can see in the workspace. See Build a data model from a workspace file.

Add a second source

1. Open the data model and select **Source** at the top of the rail on the left. The **Source** screen opens.
2. Select the **Sources** tab. It lists every source the model reads, with the kind of source and how many views read it, for example `Connection · SQLite · 1 view` .
3. Add the source:

- **Connection.** Select **Add connection** and choose a connection from the list. The list shows only enabled connections that the model does not already read. If every connection is already a source, the list reads **No more connections configured**.
- **Workspace file.** Select **Add workspace file…**. In the **Select a workspace file** dialog, pick a workbook or SQLite database and confirm. If you pick a file the model already reads, Reportworq says `'<path>' is already a source of this data model.`

4. Select **Save Changes**.



The model's first source also takes you straight to the **Tables** tab so you can look at what it holds. A second source does not: select the **Tables** tab yourself to browse it. When a model has several SQL sources, the **Tables** tab groups the tables under a folder for each source.

How each source is named

You do not name a source in the model. Reportworq shows the name it already has:

Kind	Shown as	To rename it
Connection	The connection's name from Integrations	Rename the connection in Integrations. Every model that reads it shows the new name.
Workspace file	The file's path in the workspace, for example `Data/Sales.xlsx`	Rename or move the file, then use **Change…** on the source's row to point the model at it again.

Where the names appear:

- **Rail.** Beside **Source**, the second line names the source, or reads **3 sources** (for example) when the model reads several. Point to it to see every source in the tooltip. Under it, each source has a folder named for the source, holding the views that read it, and a plus sign on the folder adds a view of that source.
- **Source screen.** The head row names the source, or reads **N sources**.
- **Pickers.** In the joined-view editor and the relationship editor, views are grouped under the name of the source they read. The hierarchy editor shows the source in brackets after a view's name, for example `Sales Details (TransactionDetails)`.

View names are shared across the whole model. Two views can read different sources, but they cannot have the same name: the Excel add-in and every report binding find a view by its name within the model. If you rename a view to a name already taken, the rename is refused with `A view named “<name>” already exists. View names must be unique.` Use the source in the name when it helps, such as `Actuals` and `Forecast`.

Add a view of a source

Each view reads **exactly one** source, and that source decides the editor you get: the table designer for a SQL database or a workspace file, the cube view editor for Planning Analytics, and each connector's own query settings otherwise.

1. Select **+ Add** beside **Source** in the rail. The **Add a view of…** list has one entry per source, then **Joined view**.
2. Choose a source. The new view opens on its **Design** (or **Connection**) tab, named **New View** (or **New View 2**, and so on).
3. Build the view and set its visibility. A new view starts **Hidden**. See Build a view with the designer and View visibility.

If the model has no source yet, **+ Add** offers **Add a source first…**, which opens the **Sources** tab.

When each source is used

Reportworq reads a source only when something needs one of its views.

- **A report, job or preview of one view** reads that view's source and no other.
- **A joined view** runs each input view on its own source, then matches the rows in Reportworq's memory. It needs every source its inputs read, so if one is down the joined view fails. See Join views across sources.
- **A hierarchy** belongs to one source. Its **Levels** version reads a view of that source each time a report uses it. See Hierarchies.
- **A table relationship** links tables of one source only. To relate data across sources, join views instead. See Table relationships.
- **Direct queries** are a model-level option, but a report author starts a direct query against one source at a time. See Direct queries.

From several sources to a joined view

Several sources become one result in a **joined view**, a view whose inputs are other views of the model. The usual path:

1. Add each source and build a view on each (a `Sales Details` view of the workbook and a `Forecast` view of the planning cube, for example).
2. Select **+ Add** > **Joined view**, pick one view to start from and add the related view.
3. Relate them on a field they share, check the match with **Test match** and save.

The joined view is the one you set to **Public** for report authors. The input views can stay **Hidden**. The whole walkthrough is in Join views across sources.

Change or remove a source

- **Change** (a connection) or **Change…** (a workspace file) points a source at another connection or file. Every view of the source moves with it and its query is kept. Use it to swap a test database for the production one, or to repoint a file after you moved it. Joined views are not tied to one source, so they are unaffected.
- **Remove** (the trash button on the row) asks first, naming up to five views that read the source. Those views move to a **No source** folder in the rail and keep their design. Open one and pick a source again to make it work. Table relationships on the removed source are dropped.
- **A source that went missing is flagged when you open the model.** If a connection was deleted from Integrations or disabled, or a workspace file was moved, renamed or deleted, an exclamation mark appears on the source's folder in the rail and the row on the **Sources** tab explains how to fix it. This check only confirms the connection or file is still there. It does not sign in, so a wrong

password shows when a view runs.

Notes and limits

- **A connection can be a source only once** in a model, and so can a workspace file. To read two databases of the same kind, add two connections in Integrations.
- **A workspace file source must live in the model's own workspace.** A model exported to another workspace finds its files by their path, so the same folder layout must exist there. See Build a data model from a workspace file.
- **Everything about the model is shared by its sources:** the folder permissions on the model, the **Direct queries** checkbox, and the variables you define on views. Put sources that need different access in different models.
- **Planning Analytics, Analysis Services, Adaptive and Pigment are read as tables**, so a view of a cube is rows and columns. Details are in Data models.

> **Going deeper.** To combine views of different sources, see Join views across sources. To start from a single file, see Build a data model from a workspace file. For everything the model editor does, see Data models.

The Excel add-in and the studio

explanation author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

You can build Reportworq reports in two places: in your own Excel workbook, with the Reportworq pane beside the sheet, and in the web report studio, in the browser. They are the same reporting engine. This page explains what they share, what is different, and how a workbook and a web report move between the two. For the step-by-step builds, see List reports and Crosstabs.

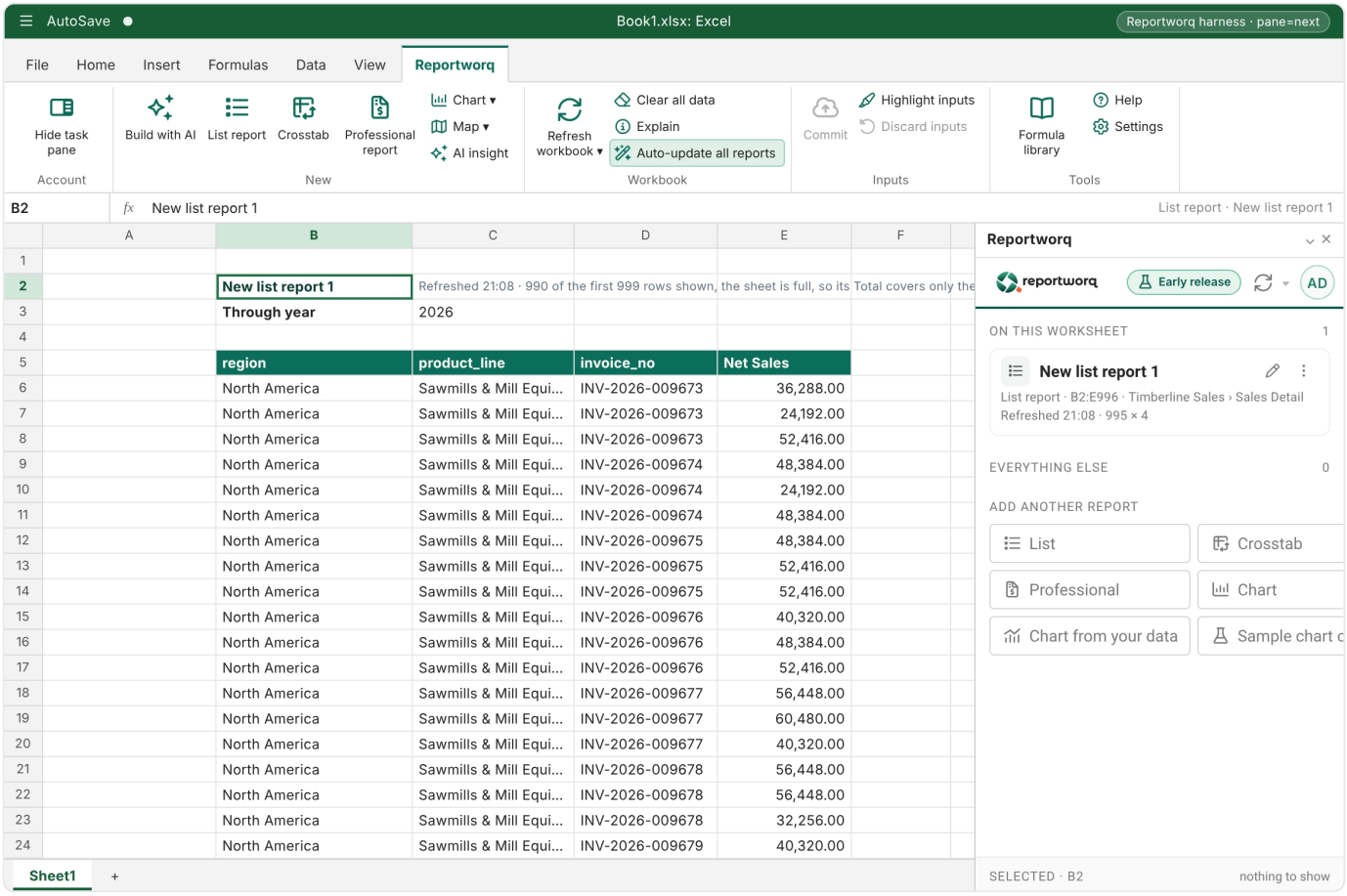
One pane, two hosts

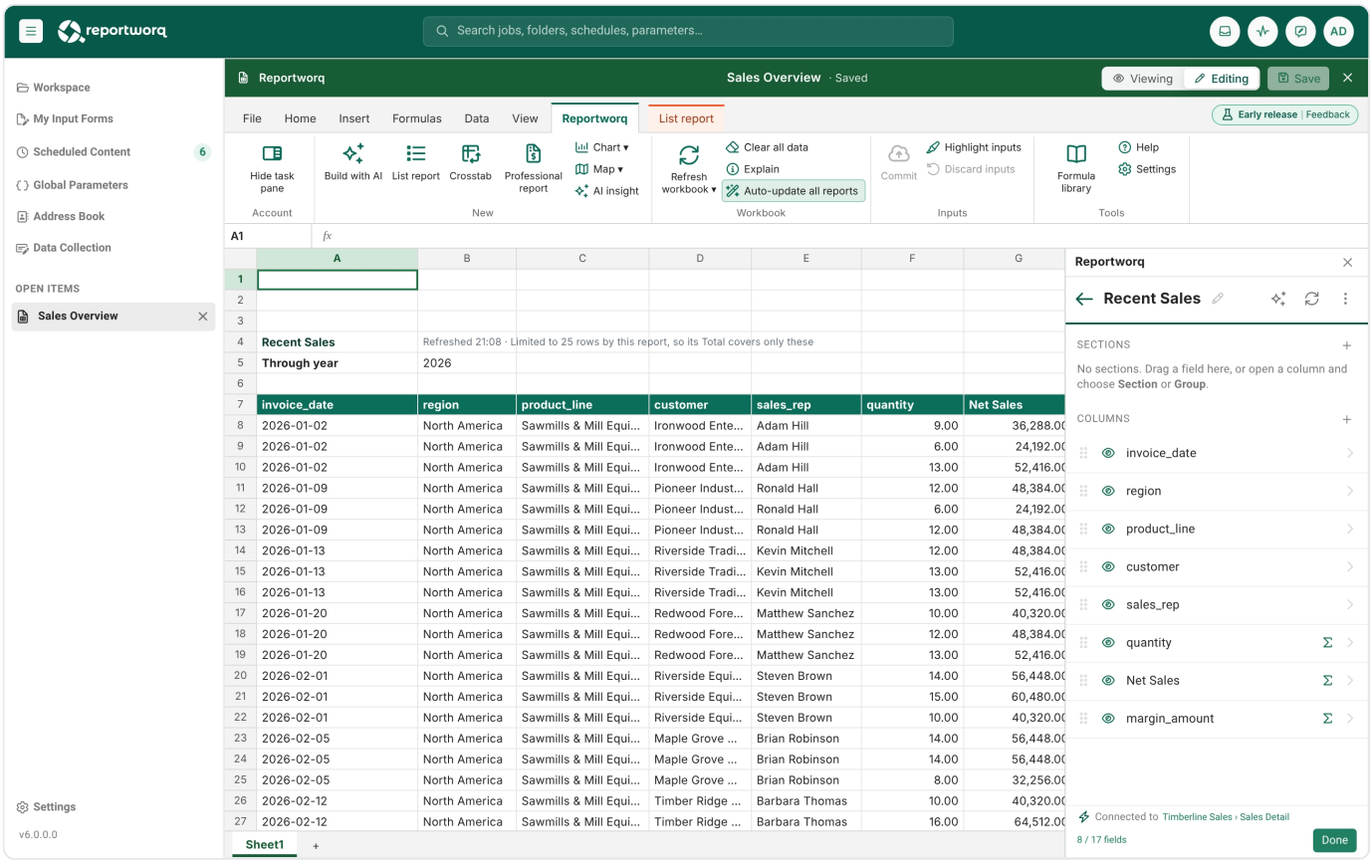
The **Reportworq pane** (also called the analyst pane) is one piece of software with two homes.

- **In Excel**, the pane sits in a task pane beside your real worksheet. Reports are written into your cells, and the workbook you save is the record.
- **In the web report studio**, the same pane sits beside an Excel-like page in the browser. The report is saved to the workspace as a web report.

Because it is the same pane, the report types, the editors, the filters, the totals and the limits for a list or a crosstab behave the same way in both. A list you build in one is a list you can open in the other.

	Excel add-in	Web report studio
Where the page lives	Your workbook	A web report in the workspace
How you open it	The **Reportworq** button on Excel's **Reportworq** tab	**New**, then **New Web Report**, or open an existing web report
Who can use it	Authors and system administrators	Authors and system administrators
Saved by	Saving the workbook in Excel	**Save** in the studio
Sheet size	Excel's whole sheet on current Excel for Windows, Mac and the web; a smaller window on an older client	1,000 rows by 52 columns
Formulas in your own cells	Excel calculates them	The studio calculates a common list of Excel functions; others show `#NAME?`
Report types	List, crosstab, chart, map, AI insight	List, crosstab, professional report, chart, map, AI insight
Large reports	Asks before writing more than 100,000 cells (the default; an administrator can change it)	Writes without asking
Live viewing and the reader's page	Not available	**Viewing** mode and the viewer
Feeding a distribution job	The workbook itself is the job's source report	Convert the web report to an Excel workbook first


AutoSave
Book1.xlsx: Excel
Reportworq harness · pane=next
File
Home
Insert
Formulas
Data
View
Reportworq
Hide task pane
Account
Build with AI
List report
Crosstab
Professional report
Chart ▾
Map ▾
AI insight
New
Refresh workbook ▾
Clear all data
Explain
Auto-update all reports
Workbook
Commit
Highlight inputs
Discard inputs
Inputs
Formula library
Help
Settings
Tools
B2
New list report 1
List report · New list report 1
New list report 1
Refreshed 21:08 · 990 of the first 999 rows shown, the sheet is full, so its Total covers only the
Through year
2026
region
product_line
invoice_no
Net Sales
North America
Sawmills & Mill Equi...
INV-2026-009673
36,288.00
INV-2026-009673
24,192.00
INV-2026-009673
52,416.00
INV-2026-009674
48,384.00
INV-2026-009674
24,192.00
INV-2026-009674
48,384.00
INV-2026-009675
48,384.00
INV-2026-009675
52,416.00
INV-2026-009675
52,416.00
INV-2026-009676
40,320.00
INV-2026-009676
48,384.00
INV-2026-009676
52,416.00
INV-2026-009677
56,448.00
INV-2026-009677
60,480.00
INV-2026-009677
40,320.00
INV-2026-009678
56,448.00
INV-2026-009678
56,448.00
INV-2026-009678
32,256.00
INV-2026-009679
40,320.00
Sheet1
Reportworq
Early release
AD
ON THIS WORKSHEET
1
New list report 1
List report · B2:E996 · Timberline Sales › Sales Detail
Refreshed 21:08 · 995 × 4
EVERYTHING ELSE
0
ADD ANOTHER REPORT
List
Crosstab
Professional
Chart
Chart from your data
Sample chart
SELECTED · B2
nothing to show


reportworq
Search jobs, folders, schedules, parameters...
AD
Workspace
My Input Forms
Scheduled Content
6
Global Parameters
Address Book
Data Collection
OPEN ITEMS
Sales Overview
Settings
v6.0.0.0
Reportworq
Sales Overview
Saved
Viewing
Editing
Save
File
Home
Insert
Formulas
Data
View
Reportworq
List report
Early release | Feedback
Hide task pane
Account
Build with AI
List report
Crosstab
Professional report
Chart ▾
Map ▾
AI insight
New
Refresh workbook ▾
Clear all data
Explain
Auto-update all reports
Workbook
Commit
Highlight inputs
Discard inputs
Inputs
Formula library
Help
Settings
Tools
A1
Recent Sales
Refreshed 21:08 · Limited to 25 rows by this report, so its Total covers only these
Through year
2026
invoice_date
region
product_line
customer
sales_rep
quantity
Net Sales
2026-01-02
North America
Sawmills & Mill Equi...
Ironwood Ente...
Adam Hill
9.00
36,288.00
6.00
24,192.00
13.00
52,416.00
2026-01-09
Pioneer Indust...
Ronald Hall
12.00
48,384.00
6.00
24,192.00
12.00
48,384.00
2026-01-13
Riverside Tradi...
Kevin Mitchell
13.00
52,416.00
2026-01-20
Redwood Fore...
Matthew Sanchez
10.00
40,320.00
12.00
48,384.00
13.00
52,416.00
2026-02-01
Riverside Equi...
Steven Brown
14.00
56,448.00
15.00
60,480.00
10.00
40,320.00
2026-02-05
Maple Grove ...
Brian Robinson
14.00
56,448.00
8.00
32,256.00
2026-02-12
Timber Ridge ...
Barbara Thomas
10.00
40,320.00
16.00
64,512.00
Sheet1
Reportworq
Recent Sales
SECTIONS
No sections. Drag a field here, or open a column and choose Section or Group.
COLUMNS
invoice_date
region
product_line
customer
sales_rep
quantity
Net Sales
margin_amount
Connected to Timberline Sales › Sales Detail
8 / 17 fields
Done

Which should I start in?

- **Start in Excel** when the finished product is an Excel workbook: a formatted package you send by a distribution job, or one you keep working on with formulas, charts and PivotTables beside the live reports. A workbook with Reportworq reports can be a job's source report as it is.
- **Start in the studio** when you want a live page people open in the browser, when you want a professional report, or when you need the **Viewing** mode with filters and drilling.
- **Switch when you need to.** The two hosts share the stored definition of each report, so you can move a report from one to the other (see below).

What is the same

- **The report types and editors.** The Connection page, the field picker, a field's page, the Report filters, Variables, Place, Totals and Limit rows pages are the same.
- **The data.** Both read your data models (the Public views you can see) and, for crosstabs, saved cube connections. Neither can show a Hidden or Disabled view in its list.
- **The limits that guard a report.** The largest report, the rows read from the source and the read timeout are the same.
- **The definition.** Each report keeps its definition (its source view, fields, filters, totals and settings) with the page that holds it, in a workbook or in a web report. A report saved by a newer version of Reportworq cannot be read by an older server.

What is different

- **No Professional report in Excel.** In Excel the professional report buttons are dim and say "Professional reports aren't in Excel yet." Build them in the studio. See Professional reports.
- **Viewing only in the studio.** The studio can show a web report as a live page, with filters, drilling and download. A workbook is just a workbook. See The viewer.
- **Where a report may sit.** In the studio, a report must lie inside rows 1 to 1,000 and columns A to AZ. In current Excel for Windows, Mac and the web, a report can sit anywhere up to XFD1048576. On an older or unrecognized client, Excel works in A1:AZ1000 as the studio does. Reportworq never moves a report for you; a report outside the area the host can hold is left alone and not treated as a report.
- **The ribbon.** In Excel, the add-in adds a **Reportworq** tab with an **Open** group (the **Reportworq** button that opens the pane) and a **New report** group with **List**, **Crosstab**, **Professional** (dim), **Chart** and **Build with AI**. In the studio, the Reportworq tab and a ribbon that looks like Excel's are part of the page.

- **An administrator can turn the pane off in Excel.** The setting is **Use the analyst pane in Excel** under **Settings**, **Configuration**, **Excel add-in**. If it is off, or Excel is too old, Excel shows the older Companion pane. The studio is not affected.

How a workbook relates to a web report

A web report is not a file. A workbook is. You can move between them in five ways.

You want to	Do this	What happens
Keep the web report and get a copy as a workbook	In the workspace, on the web report's row: **Export as Excel**	A connected `.xlsx` is saved beside the web report. The web report is untouched. No data is read; the workbook is the report as last saved.
Download the web report	In the studio: **File**, **Excel**, **Export to Excel**	The workbook is downloaded to your computer.
Put a workbook's reports onto an existing web report	In the studio: **File**, **Excel**, **Import workbook**	Replaces what is on the page. A connected workbook's reports come back live; other workbooks come in as a plain page. Nothing is saved until you choose **Save**.
Make a new web report from a workbook	In the studio: **File**, **Excel**, **Open workbook**	Creates a new web report in the same folder, in a tab of its own.
Replace a web report with a workbook that jobs will refresh	In the workspace, on the web report's row: **Convert to Excel**	Creates the workbook, points the distribution jobs at it, and deletes the web report. It cannot be undone. See Convert to Excel.

A workbook that you build in the Excel add-in comes into the studio through **Import workbook** or **Open workbook** with its lists, crosstabs and other reports intact, and they refresh from their live sources. If a report in that workbook reaches past row 1,000 or column AZ, the studio shows the first 1,000 rows and 52 columns, will not change the report, and says "Edit it in Excel." Saving keeps its stored definition as it was.

A workbook made from a web report also carries a hidden **Reportworq definitions** sheet that explains each report in plain language: where it sits, what it reads and which filters and variables it uses. It is written from the same text as the pane's **Explain this workbook**.

> **Why not just use the web report as a job source?** A distribution job produces static output and never carries a live report's viewing filters. A workbook is what a job knows how to refresh and send. That is why **Convert to Excel** exists. See A web report as a job source.

Who can use which

Readers (people who only read output) cannot open a web report, and they cannot use the pane in Excel. The Excel add-in refuses a sign-in without author rights with "The Excel Addin requires the Author entitlement." Reports in the pane read data live, and Reportworq keeps that for authors and administrators. Readers receive what distribution jobs send.

> **Going deeper.** The pane's limits, how a job handles a workbook saved from the pane, and the settings an administrator can change are in Excel: the analyst pane.

Download, burst and keep a view

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

You can take a web report away from the viewer as an Excel workbook or a PDF, **burst** it (run it once for each value of a filter and collect the results), and keep what you adjusted. This page covers all three. For one-off packets this is the quick route; for recurring delivery to recipients, use a distribution job.

When to use it. A controller opens the P&L web report, bursts it by Region, ticks five regions and downloads one PDF packet for a meeting. Nothing is scheduled and no job is built.

Before you start

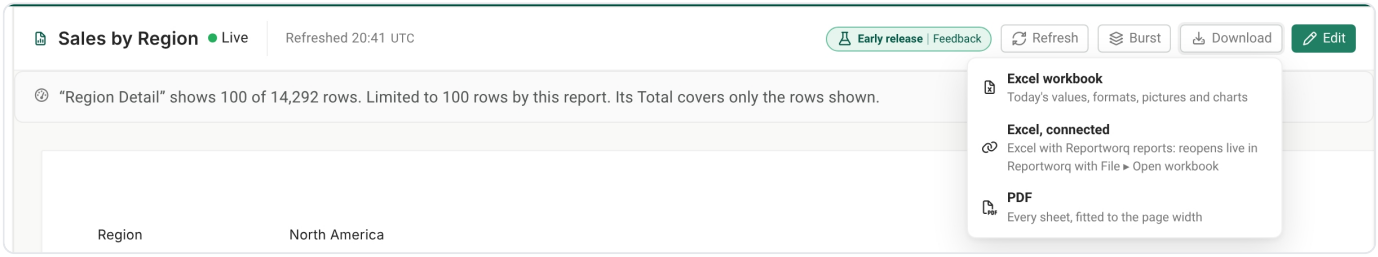
- You are an author or an administrator. Readers cannot open web reports. See The viewer.
- The report is open in the viewer. Adjust it first (filters, expands) if you want the download to carry your view. A download contains the page as the viewer shows it right now.

Download the report

1. Select **Download**.
2. Choose one of three items.

Item	What you get	Use it when
Excel workbook	Today's values, formats, pictures and charts. Standard Excel formulas carry their computed results. Nothing reconnects to Reportworq.	You want a plain workbook to send or keep.
Excel, connected	The same workbook, carrying each report's definition. The menu note reads "Excel with Reportworq reports: reopens live in Reportworq with File ▸ Open workbook".	You want to bring the reports back live in the web report studio, or use the file as a distribution job source. See Convert to Excel.
PDF	Every visible sheet, fitted to the page width.	You want to print or send a fixed copy.

3. Your browser saves the file and Reportworq says `Downloaded “<file>”.` The file is named after the report: `<name>.xlsx`, `<name> (connected).xlsx` or `<name>.pdf`.



If something goes wrong, the viewer shows `The download didn't work:` followed by the reason.

Charts and maps come out as Reportworq chart pictures in Excel. Sheet names are made safe for Excel (no `: \ / ? * [ ]`, at most 31 characters, each unique). A sheet the author hid goes out hidden.

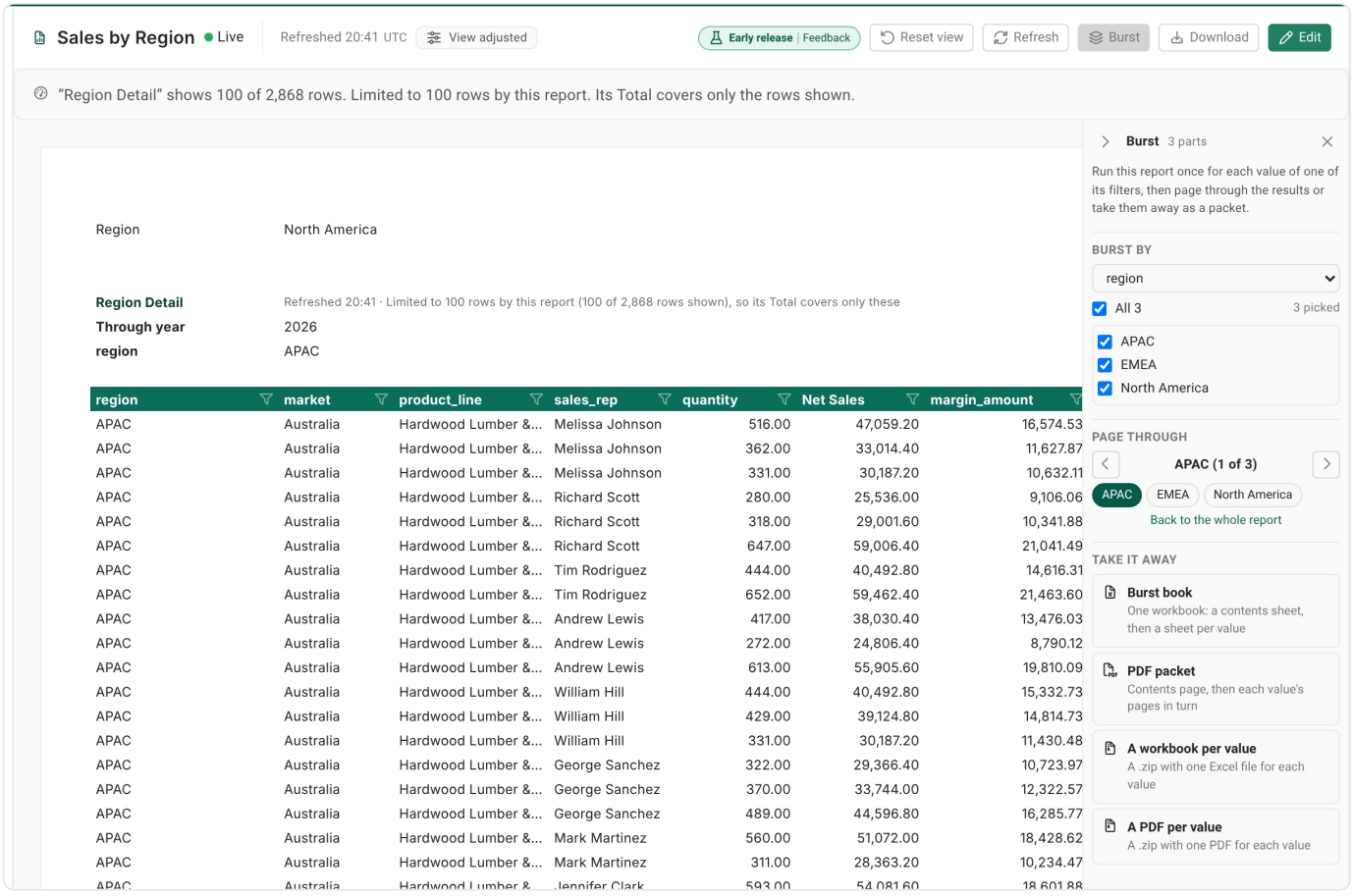
Burst the report

Bursting runs the report once per value of a filter, so you can page through the parts or take them all away.

1. Select **Burst**. The **Burst** panel opens on the right.
2. Under **Burst by**, pick one of the report's filters. These are the report's own filters: a crosstab's report filters, a list's checklists and report filters, and the variables of the views it reads. Professional report filters and free-text variables with no pick list are not offered.
3. Optionally select **Then by another filter (every combination)** to add a second filter under **Then by**. The first filter is the outer one.

4. Tick the values you want. **All** is ticked by default, and the panel shows how many parts that makes.
5. Under **Page through**, use the previous and next arrows or a value chip to view the report as it would be for each value. **Back to the whole report** returns to the full page. The panel lists up to 12 chips and says "and N more".
6. Under **Take it away**, choose an output.

Output	What you get
Burst book	One workbook: a contents sheet with a link per value, then each value's sheets named after the value (for example "EMEA", or "EMEA - Summary"), each with the value in its page header.
PDF packet	The same book as one PDF: a contents page, then each value's pages in turn.
A workbook per value	A .zip with one Excel file for each value.
A PDF per value	A .zip with one PDF for each value.



A part's value narrows the report: it never shows rows the author's own filters already left out. A value that the author's checklist leaves out gives an empty part. A fold arrow at the top of the panel folds it to a slim rail without losing your choices, and the X closes it.

Burst limits

- A download bursts at most **60** parts. Past that the panel says `A download bursts at most 60 parts, and this is N. Narrow the values, or make it a job.`
- The panel lists up to 1,000 combinations to page through. Past that it shows only the count.
- A filter offers at most 500 values and says so: " has more than 500 values. The list shows the first 500."
- A burst that fails shows `The burst didn't work:` and the reason. A hidden sheet whose report the burst filter does not narrow fails the burst, so no recipient is handed the whole data.
- The panel never creates a job. For a recurring burst with recipients and a timetable, build a distribution job. See A web report as a job source and What is bursting.

Keep a view you adjusted

Adjusting the viewer is never saved, and **Edit** opens the report as the author last saved it. That is deliberate, so one person's filter never changes a shared report. To keep what you see, pick the route that fits.

You want to keep	Do this
The numbers as they look now	**Download** an **Excel workbook** or **PDF**.
The view and the reports, so you can reopen it live	**Download**, **Excel, connected**. Open it later in the web report studio with **File**, **Open workbook**, which makes a new web report from it.
A copy of the report to change	In the editor, select **File**, then **Save a copy**. The copy is saved next to the original as " (copy)".
A new report started from Explore	In the editor, select **Save as report**, choose a name and folder and select **Save**. If that name is taken, the new one is saved as “Name (1)”.

Your filter and expand choices from the viewer are not carried into a copy made with **Save a copy** from the editor, because switching to the editor drops every adjustment. To make a choice permanent, set it in the editor and select **Save**.

Notes and limits

- A saved web report can be at most 25 MB. Save refuses a larger report and keeps your changes on the screen.

- An AI insight in a download carries its answer, with an out-of-date line if the data moved.
- The downloaded file and a job's file are made by the same engine, so a formula cell shows the same value in both.

> **Going deeper.** To turn a burst into a recurring, delivered job, see Bursting and Schedule a job.

Convert a web report to Excel

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

Convert to Excel turns a web report into an Excel workbook in the same folder and points every job that used the web report at that workbook. From then on, each job run refreshes the workbook's lists and crosstabs where they sit, then runs the workbook like any other Excel source. The web report is deleted, and this cannot be undone.

When to use it. A controller built the monthly packet pages as a web report, and the job that distributes them now needs the full set of workbook features, such as cell parameters and a sheet replicated per region. Converting gives the job an ordinary Excel source that still holds the live reports.

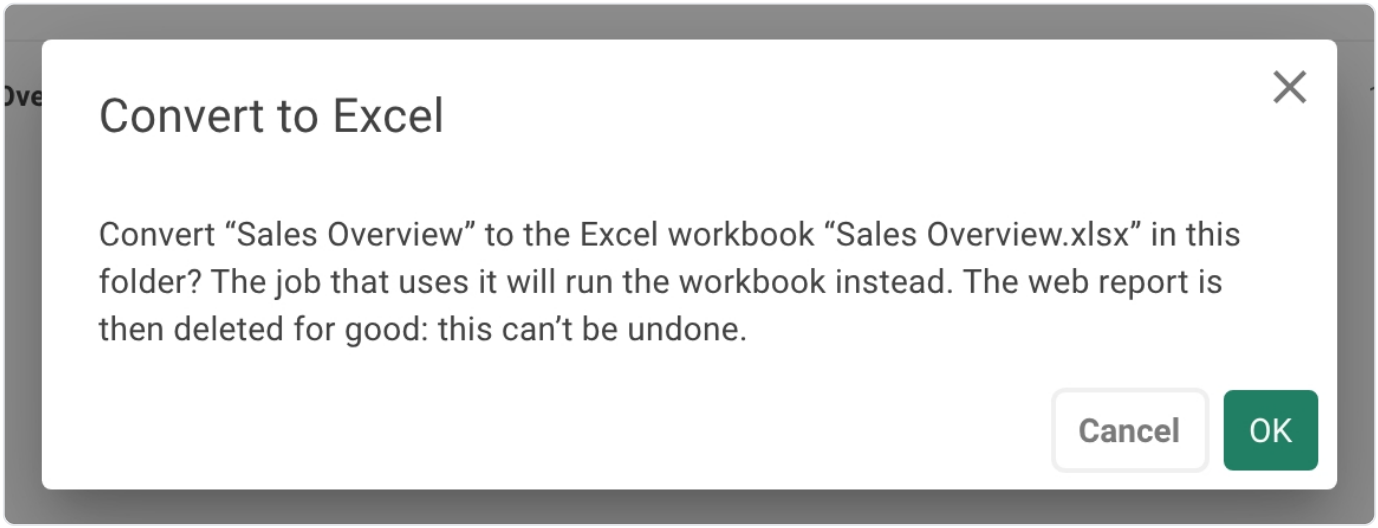
Before you start

- You are an author or administrator who may change content. Readers cannot see web reports, so they cannot convert them.
- You can see every job and campaign that uses the web report. If one is hidden from you, the conversion is refused (see the table below). Ask an administrator.
- If you only want a copy, use **Export as Excel** on the same ⋯ menu. It writes the same workbook next to the web report ("Exported “.xlsx” beside the web report.") and leaves the web report and the jobs alone. The viewer's **Download** menu is covered in Download, burst and keep a view.

Convert a web report

1. In the workspace, find the web report. Select its ⋯ menu and select **Convert to Excel**.
2. Reportworq builds the workbook first. If it cannot carry the report whole, it stops here and says why, and nothing is changed.
3. Read the confirmation. It reads: `Convert “<name>” to the Excel workbook “<name>.xlsx” in this folder? <jobs> The web report is then deleted for good: this can’t be undone.` The second sentence says how many jobs are affected: "No job uses it.", "The job that uses it will run the workbook instead." or "The N jobs that use it will run the workbook instead."
4. Confirm.

5. Reportworq says `Converted to “<file>”.` and how many jobs now run it ("One job now runs it." or "N jobs now run it.").



If the name `<name>.xlsx` is taken, the workbook gets a free name such as "name (1).xlsx", and the jobs use the name actually given. A job's report that was still named after the web report (`<name>.rwrpt`) is renamed `<name>.xlsx`; one you renamed keeps its name.

Nothing is refreshed at conversion time. The jobs refresh the workbook when they next run.

What the jobs do afterward

A job whose source is a converted workbook:

- Recognizes the workbook by the report definitions stored in it. Any other workbook is left alone.
- Refreshes each list and crosstab on each sheet where it sits. Rows a report adds or gives back move the author's formulas, names, merged areas, charts and conditional formats the way Excel would.
- Calculates the author's own formulas around a report after the report refreshes, so a total under a list follows the refreshed list.
- Runs the job's parameters, replication and bursting on the workbook as on any Excel source.
- Strips the report definitions from the output, so recipients never get the report internals (data model ids, filters).

AI insights are asked again. Maps and professional reports go out as last saved. A column the author hid stays hidden.

When the conversion is refused

Nothing is changed in any of these cases.

What you see	Why, and what to do
"This web report is used by jobs you can't see, so it can't be converted. Ask an administrator to convert it."	A job or campaign you cannot change uses it. Every user's job is repointed or none is.
“” bursts by setting this web report's filters (it was set up from the web report's Burst pane)"	A job that maps its parameters to the report's own filters has nothing to map to in a workbook, and every recipient would get everything. Map its parameters to workbook cells, or remove them, then convert.
A message naming a job, a hidden sheet and a report ("The job “” sets this web report's filter “”, and the hidden sheet “” holds the report “”...")	A hidden sheet is refreshed too, so each of its reports must be narrowed by every filter the job sets. Map the filter to that sheet and give the report that filter, or unhide or remove the sheet, then convert.
`Professional reports are coming soon.`	The web report holds a professional report and the license does not include professional reports. Nothing is made and nothing is deleted.
A message naming what would be lost or moved	The workbook is checked before anything is written, and the conversion stops if it would lose or move anything. Fix what the message names, then convert.

Notes and limits

- **Converting cannot be undone.** The web report is deleted for good, not moved to a recycle area. If the conversion or the repointing fails partway, everything is put back and the new workbook is removed.
- **Jobs that fail closed.** A run of a converted workbook fails with "Nothing was distributed." when it cannot refresh. For example a report definition that "can't be read (damaged, edited by hand, or saved by a newer version of Reportworq), so it can't be refreshed. Convert the web report to Excel again, or remove the report from the sheet."
- **Space for the report.** A report needs its area free. Anything you type into the hidden rows under a list or crosstab, which the report keeps for its formats, makes the run fail. See Key-cell formats and what a job keeps.
- **Size.** Reports in the workbook must fit Excel's grid, ending by row 1,048,576 and column XFD.

- The workbook can be opened in the web report studio again with **File**, **Open workbook**, which makes a new web report from it.

> **Going deeper.** To keep a web report as a source without converting it, see A web report as a job source.

Build a view with the designer

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

On a SQL datasource, a data model view is built on the **Design** tab. You pick the table the view is about, tick the columns you want as fields, and follow a column's link to add a related table, for example the sales rep and product behind each invoice line. Reportworq writes the joins for you and keeps the view's fields in step as you work. When a view needs something the designer cannot express, you write it as SQL instead.

This page applies to SQL datasources: SQL Server, OLE DB, ODBC, SQLite, Excel, the Reportworq content store, and the CData-based sources such as QuickBooks Online and Salesforce. Other connectors, such as Planning Analytics, keep their own **Connection** tab; see the page for your connector.

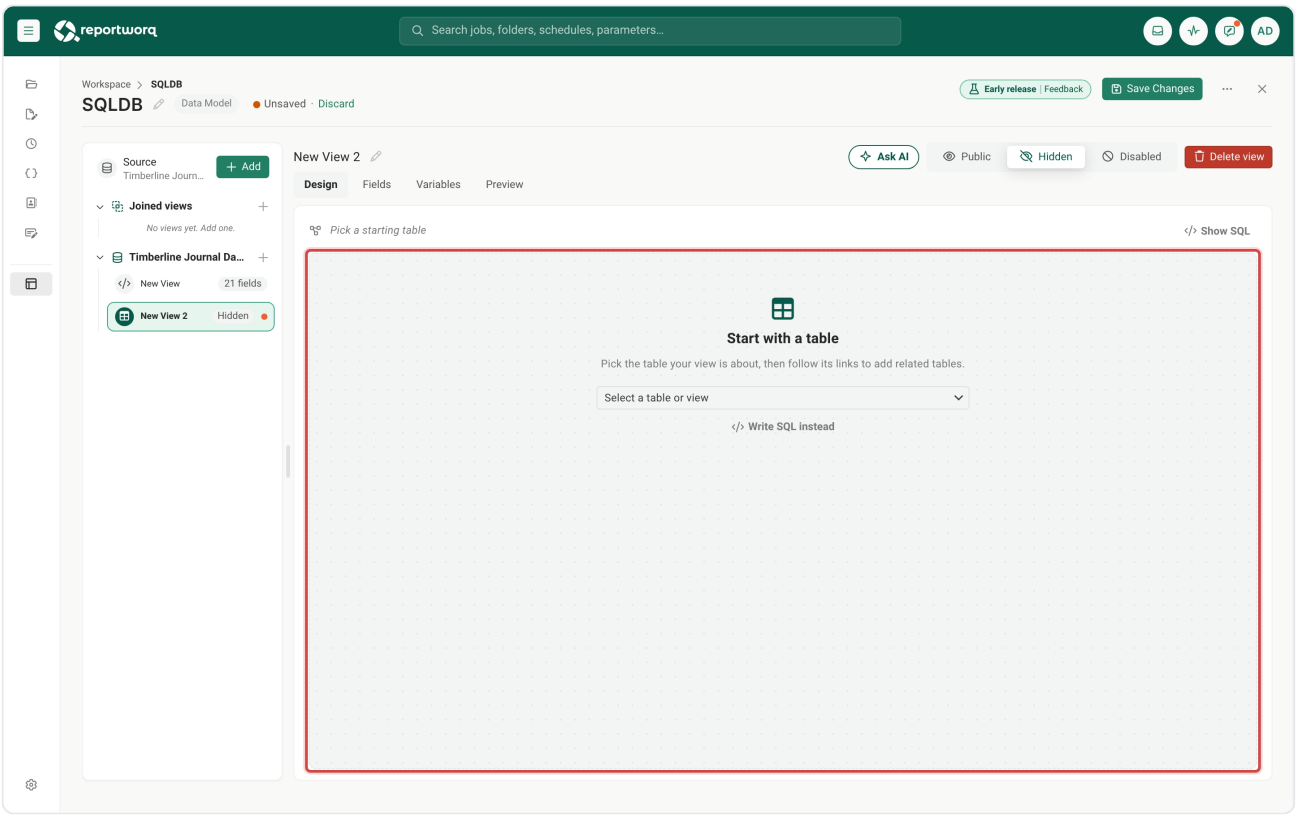
The examples use the Timberline Harvest Co. demo database, where `sales_journal` holds one row per invoice line and links to `dim_employee`, `dim_product`, `dim_market`, and `dim_channel`.

Before you begin

- A data model bound to a SQL datasource. See Data models.
- Optional: any relationships your database does not declare, added to the model. See Table relationships.

Start the view

1. Open the data model, select **+ Add** beside **Source** in the rail, and select the SQL source the view reads. (Or select the plus sign on that source's folder in the rail.) The new view opens on its **Design** tab with an empty canvas.



2. Under **Start with a table**, select the table the view is about, for example `sales_journal`. Choose the table whose rows you want one report row for: the view keeps one row per row of this table.

In the rail the new view shows **Hidden** beside its name (a **Disabled** view shows **Disabled**); a **Public** view shows its field count instead. Point to the view to see its type, **Table or View**, and, once the canvas holds more than one table, the table count, for example **Table or View · 4 tables**. (A **joined view** is something else: a view that combines other views of the model. See Join views across sources.)

The table appears on the canvas as a card labeled **Starting table**, with every column ticked. A new view takes the table's name; rename it with the pencil beside the view name.

Choose the fields

Each ticked column is a field of the view. Fields are added and removed as you tick, so the **Fields** tab and the field count in the rail are always current.

- To start from a short list, clear the checkbox in the card's header row, which clears every column, then tick the columns you want.
- To find a column in a wide table, type in the **Search columns** box at the top of the card.
- The order you tick columns in is the order of the fields.

Unticking a column that a saved view already used does not delete its field. The field stays on the **Fields** tab, marked unmapped and left out of the view, so you can point it at another column or delete it yourself.

Add a related table

A column that holds another table's key shows that table's name, with an arrow, in the **Related table** column of the card.

1. Select the related table's name, for example **dim_employee** beside `sales_rep_id`. The table opens as a new card to the right, joined to the starting table.
2. Tick the columns you want from it. Reportworq ticks the related table's name column for you when it can tell which one that is, for example the rep's name; clear it if you do not need it.
3. Optional: follow a link from the new card to go further, for example from `dim_employee`'s `territory` to `dim_market` to add the rep's region.

Fields from a related table are named after the link, so `name` from the sales rep becomes `sales_rep_name`, and `region` reached through the rep's territory becomes `sales_rep_territory_region`. Rename them as you like; see the next section.

How the designer joins tables:

- **Every row of the starting table stays in the view.** A related table is added with a left join, so an invoice line with no matching rep still appears, with the rep's fields empty.
- **Links are followed from a key to the table it identifies.** Each invoice line has one rep, so adding the rep never repeats an invoice line. The designer does not offer the other direction, such as every invoice line for a rep; use a SQL view for that.
- **A table is only joined when you use it.** A card you open but take no fields from is not part of the query.
- **Some links are blocked.** A table already on the path back to the starting table cannot be added again, a path is at most four tables long, and a view can join at most 20 tables. A blocked link shows a no-entry icon; hover over it to see why.

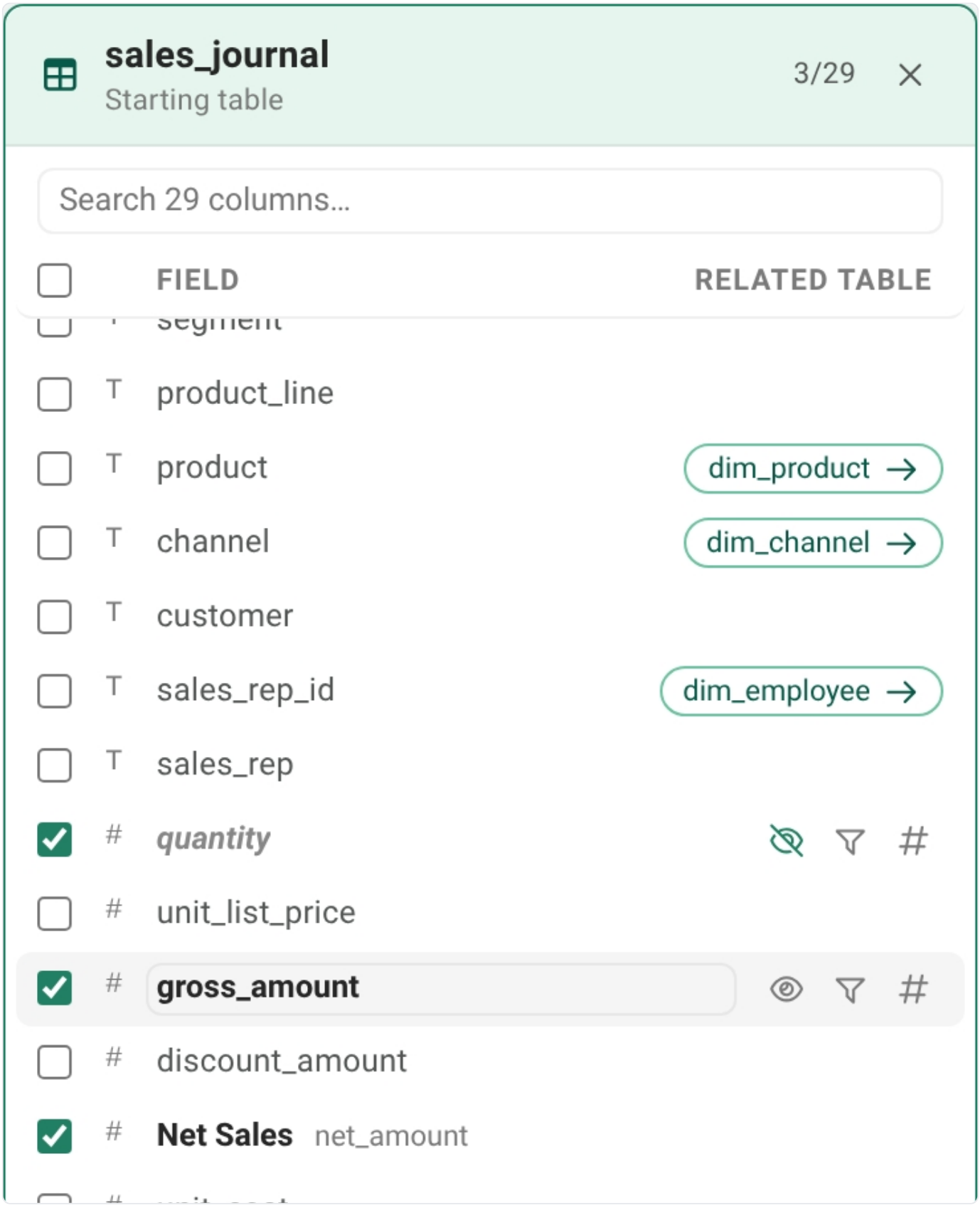
To remove a related table, and every table opened from it, select its name in the **Related table** column again or select the close button on its card. **Start over** clears the whole canvas, including the fields it created, so you can pick a different starting table.

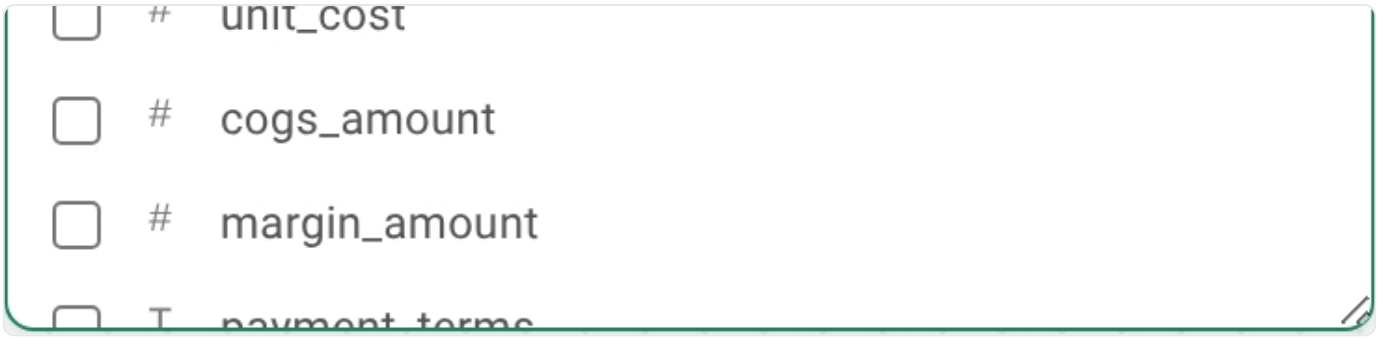
A column that can point to more than one table, such as the `Name_Id` column of a QuickBooks Online report, which can hold the id of a customer, a vendor, or an

employee, shows a list instead of a single name. Open each table you need from the list.

Shape each field on the canvas

A ticked row is the view's field, and you can shape it without leaving the canvas. The controls appear when you hover over the row, and a control that is in effect stays visible.



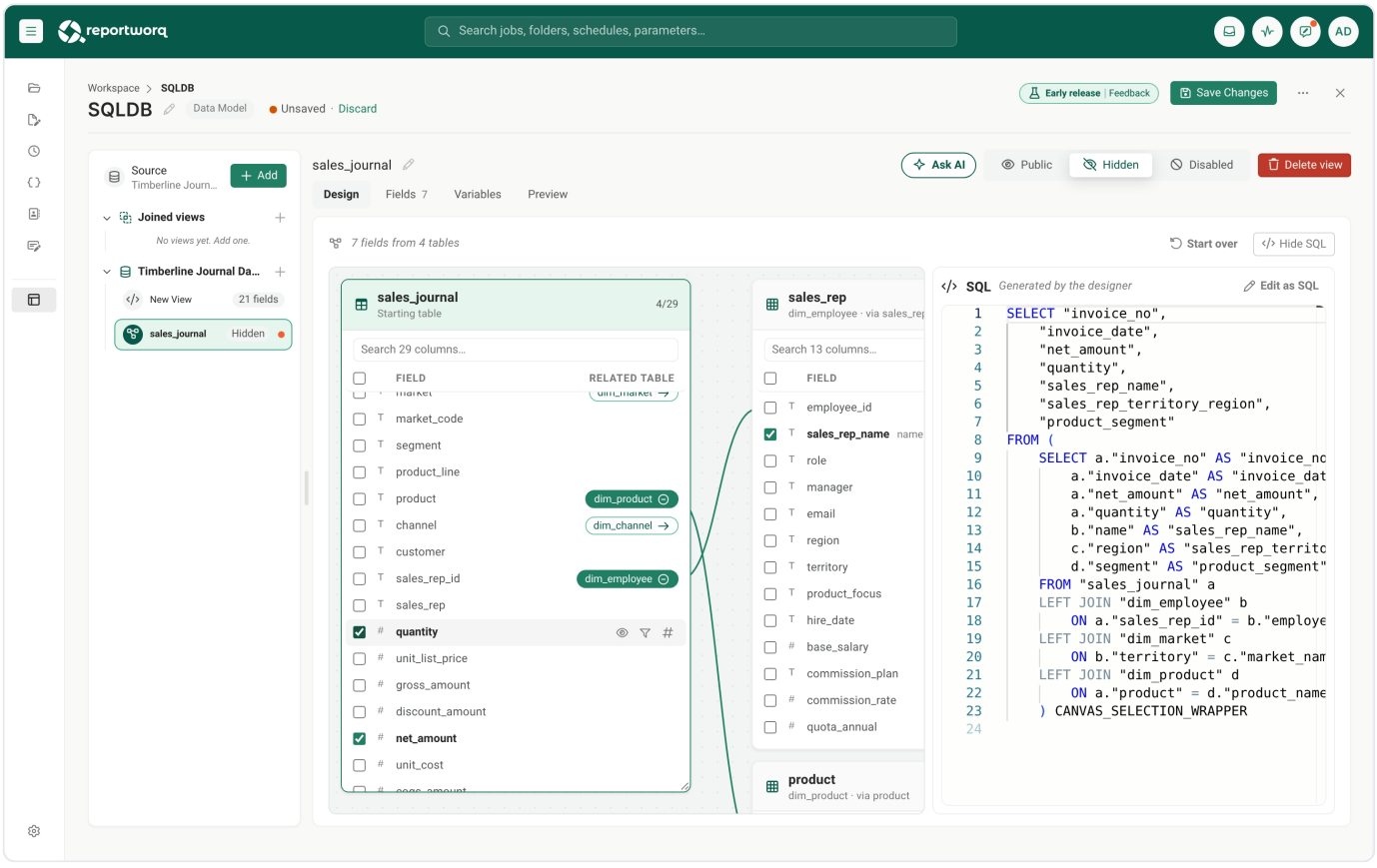


- **Rename.** Select the field's name, type the new name, and press Enter. The source column's name stays visible beside it. Field names must be unique within the view.
- **Hide from reports.** Select the eye. A hidden field is still queried, so it can drive a filter, but report authors do not see it.
- **Filter.** Select the funnel to add a condition to the field. The funnel turns solid while a condition is set.
- **Type and format.** Select **#** to set the field's type and display format.

These are the same settings as the **Fields** tab, which also lets you reorder, add, and delete fields.

Check the SQL

Select **Show SQL** to see the query the designer generates. The panel is read-only and follows the canvas as you change it.



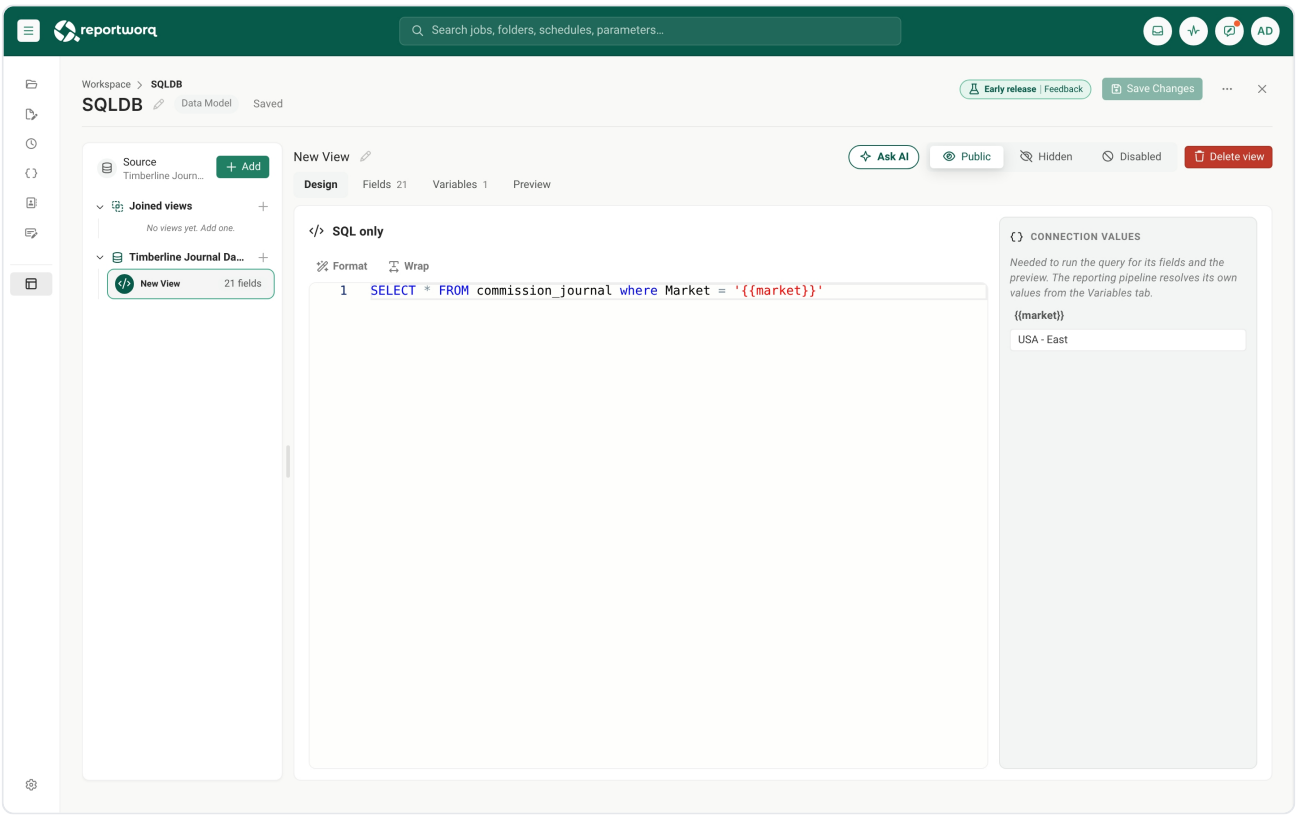
To take the query further by hand, select **Edit as SQL**. The view becomes a SQL-only view that starts from the generated query.

> **Warning: Edit as SQL** cannot be undone. The designer does not read SQL back, so a view converted to SQL stays a SQL-only view. Keep building on the canvas for as long as the designer can express what you need, and convert only when you must.

Write the view as SQL instead

Write a view as SQL when it needs something the designer does not do, such as totals, a union of tables, an inner join, or every child row of a parent.

1. Select **+ Add** in the rail, then the SQL source.
2. On the empty canvas, select **Write SQL instead**.
3. Enter the query in the SQL editor. Use **Format** to tidy it.
4. To let reports supply a value at run time, put a `{{variable}}` in the query. Put a text value inside quotes, for example `WHERE market = '{{market}}'`; Reportworq fills the value safely. See How a value is filled into a query.
5. If the query has variables, enter a sample value for each under **Connection values**. Reportworq needs a value to run the query and work out the view's fields. The sample is only for authoring: set the value reports use as the variable's default on the **Variables** tab (see Data model variables).

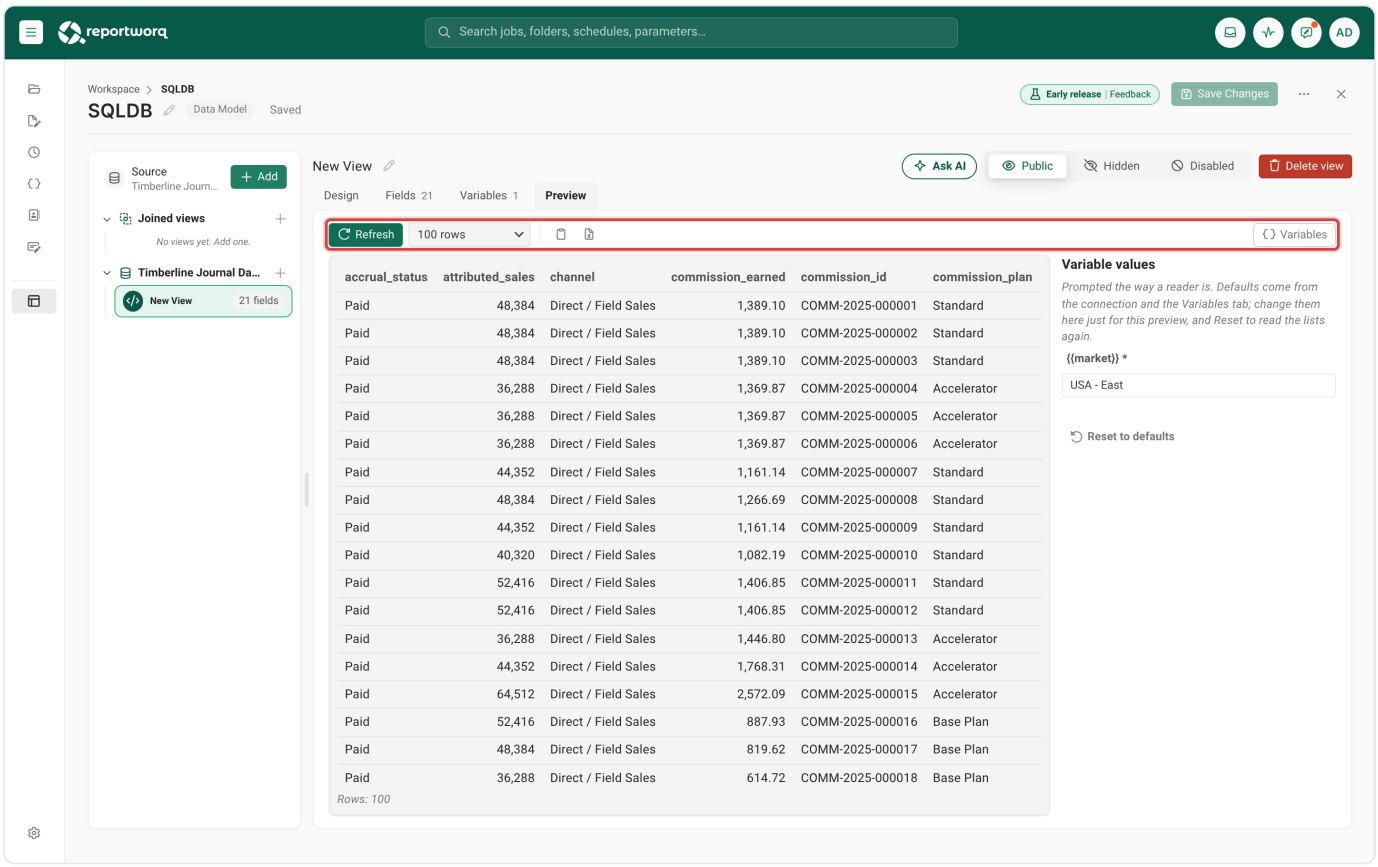


6. Select the **Fields** tab. Reportworq runs the query and builds the field list from its columns.

The Design tab of a SQL-only view is labeled **SQL only**, and the rail reads **SQL Query** for it. Each time you change the query and leave the tab, or save, the fields are brought up to date; a new column becomes a new field, and a field whose column has gone is marked unmapped.

Preview the view

1. Select the **Preview** tab.
2. Choose how many rows to fetch: **10 rows**, **100 rows**, **1,000 rows**, or **All rows**. The default is 100.
3. If the view has variables, the **Variable values** pane is open on the right. Enter values for this preview; they do not change the view's defaults, and **Reset to defaults** puts the defaults back.
4. Select **Run preview**. After the first run the button reads **Refresh**.



From the toolbar you can copy the rows with a header row, ready to paste into Excel, or download them as an Excel workbook. The **Variables** button shows or hides the pane; it is unavailable when the view has no variables. The grid shows up to 5,000 rows; a copy or a download includes every row fetched.

A preview keeps its last result while you work, so you can switch to **Design**, adjust the view, and come back to compare before you select **Refresh**.

Save the view

Select **Save Changes** to save the model, including every view and any relationships you added. A new view starts as **Hidden**; set it to **Public** when report authors should see it in the Excel add-in. See Choose who sees a view.

Troubleshooting

Symptom	Cause and fix
The view reads "Fields may not match the query yet"	The query could not run when you left the tab or saved. Read the warning that appeared, fix the query or enter the missing sample value under **Connection values**, then change tab or save again.
A column has no related-table link you expected	The database does not declare that link. Add it as a manual relationship on the model's **Source** screen. See Table relationships.
A related-table link shows a no-entry icon	The link would loop back to a table already on the path, go deeper than four tables, or pass 20 tables. Hover over it for the reason.
Preview warns "No value for required variable(s)"	Enter a value in the **Variable values** pane, or set the variable's default on the **Variables** tab.
An older view opens as SQL only with an empty editor	It was built with the earlier query composer in a way the designer cannot show, for example with an inner join. Select **Use the designer** to see why. The view still runs as saved.

Related topics

- Data models
- Table relationships
- Connect SQL Server, OLE DB, and ODBC
- Connect SQLite

List reports

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A list report is rows from one view of a data model. You choose the fields that become columns, decide how to sort and group them, add totals, and filter what is shown. The list draws into the cells where you place it and stays connected to the view, so it refreshes with current data.

When to use it. You want the detail behind a number, not a summary. For example, every invoice line from the **Sales Details** view of the Timberline **Sales Journals** data model for one region, sorted by net amount, with a total at the bottom. If you want numbers grouped by two or more fields, build a crosstab instead.

You build a list the same way in Excel and in the web report studio, because both host the same Reportworq pane. The only differences are where you click to start and how much room the sheet has. See The Excel add-in and the studio.

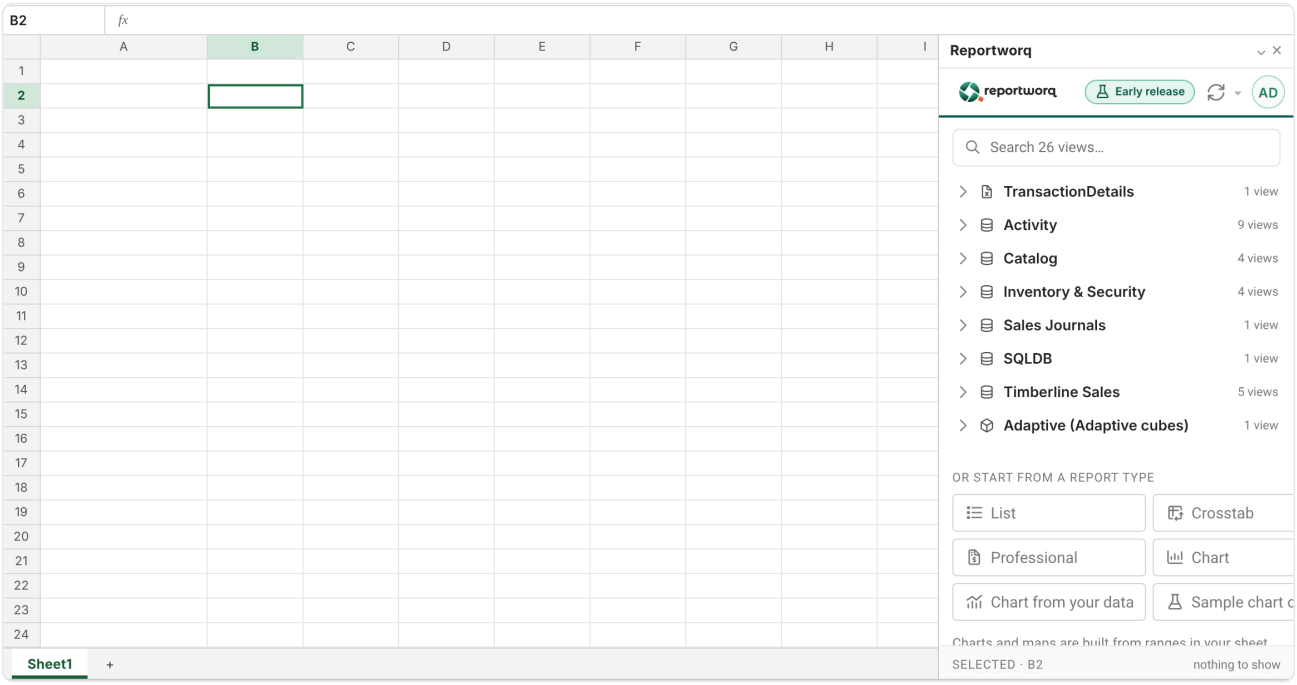
Before you start

- You are signed in as an author or a system administrator.
- A data model with a **Public** view you can see. A list reads a view of a data model. It does not read a cube. A **Hidden** view is not listed in the pane and a **Disabled** view never runs. See View visibility.
- In Excel, a version of Excel that supports the Reportworq pane. In the studio, a web report open in **Editing** mode. See The web report studio.

Start a list

1. Click the cell where the list's top-left corner should go.
2. Start the list:
    - **In Excel:** on the **Reportworq** tab, in the **New report** group, select **List**. Or open the pane and, on **Home**, pick a view from the model list, or select **List** under **Or start from a report type**.

- **In the studio:** on the **Reportworq** tab, in the **New** group, select **List report**. The **Insert** tab has the same button.

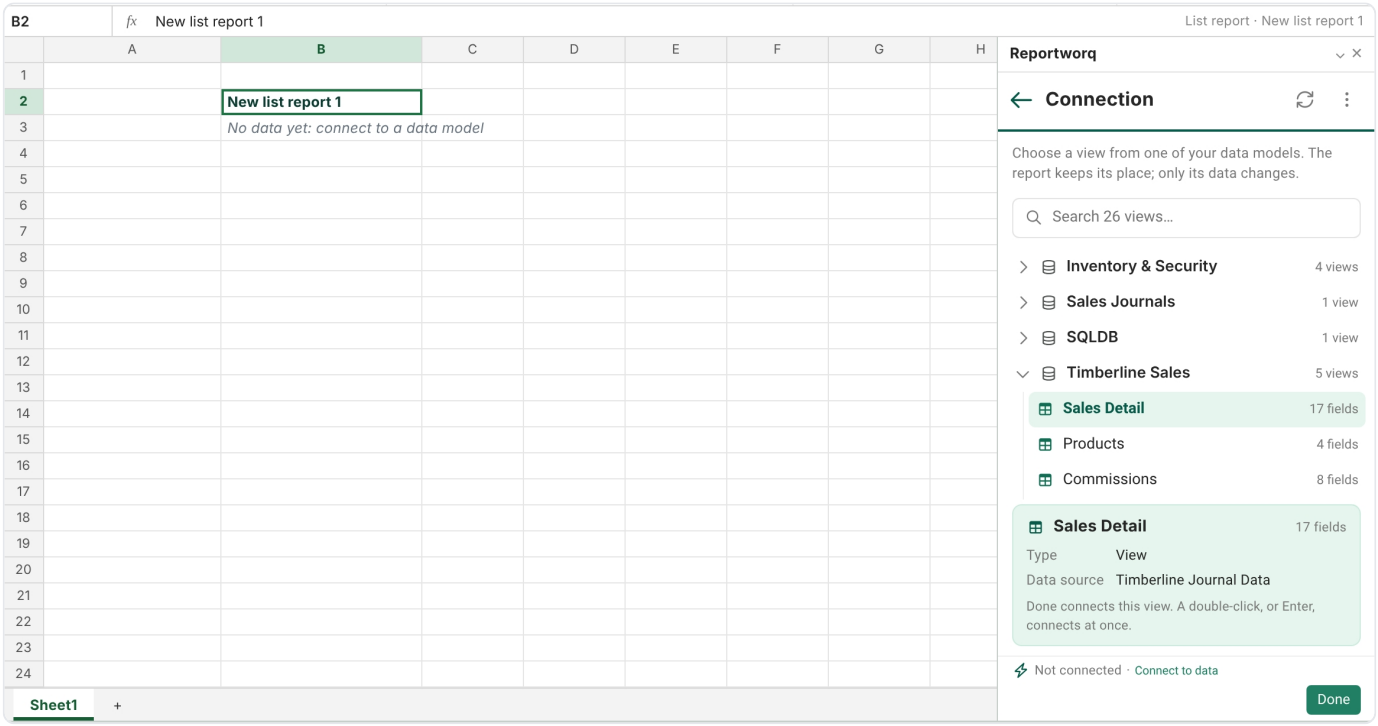


On **Home**, each view in the model tree also has quick buttons that start a list or a crosstab already connected to that view. Double-click a view, or press Enter on it, to start a list from it.

The new list opens on its **Connection** page. If you started from a view, it skips ahead to the field picker. If your selection was on a report or picture already, the list goes two rows below the lowest report on the sheet instead, so it never covers another report.

Choose the view

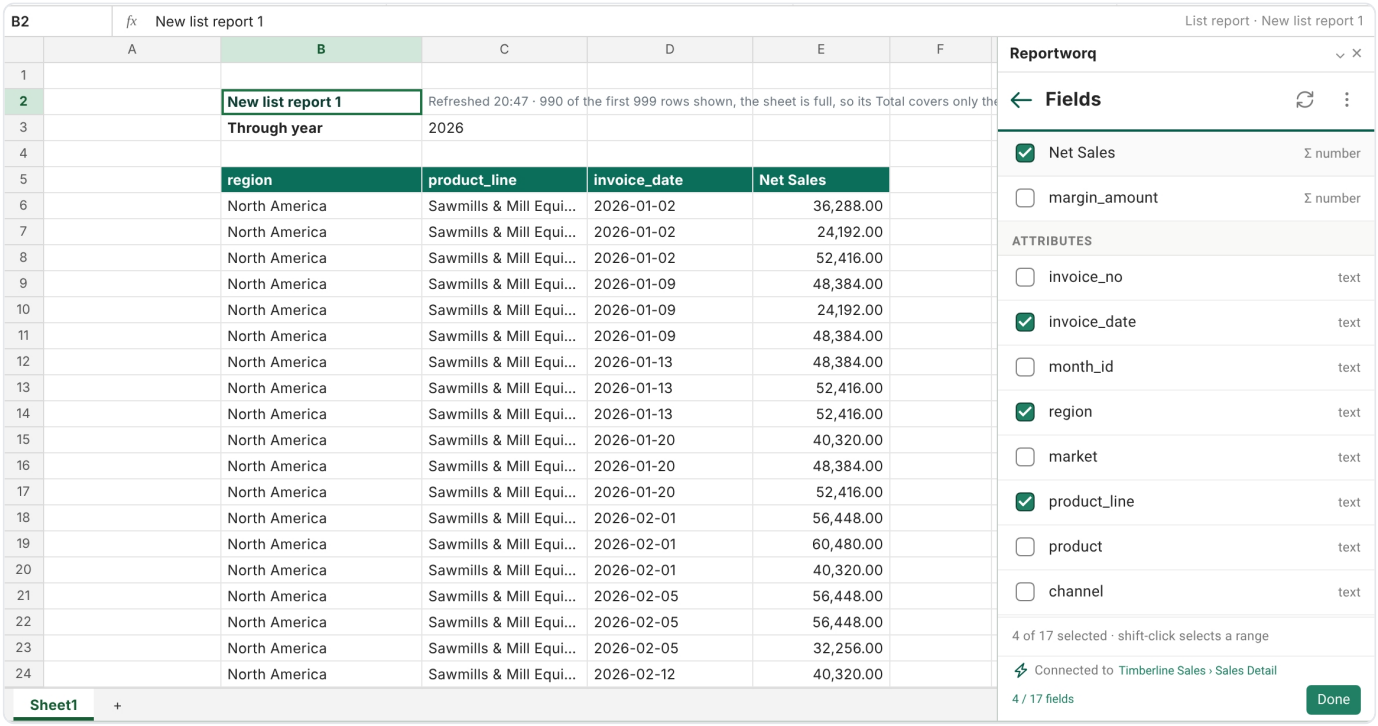
1. On the **Connection** page, find the view. Models are rows you open and close; each shows its number of views, and each view its number of fields. Type in the box ("Search 24 views...") to narrow the list.
2. Select a view. Its details appear below the tree: **Type** (View, Joined view or Cube), **Data source(s)** and **About**.
3. Select **Done**, or double-click the view, to connect.



Selecting a view does not connect it; **Done** does. **Back** leaves the list unchanged. To change the view later, open the same page from the list's menu or the **Connection** button on the List tab. Fields the new view also has are kept.

Pick the fields

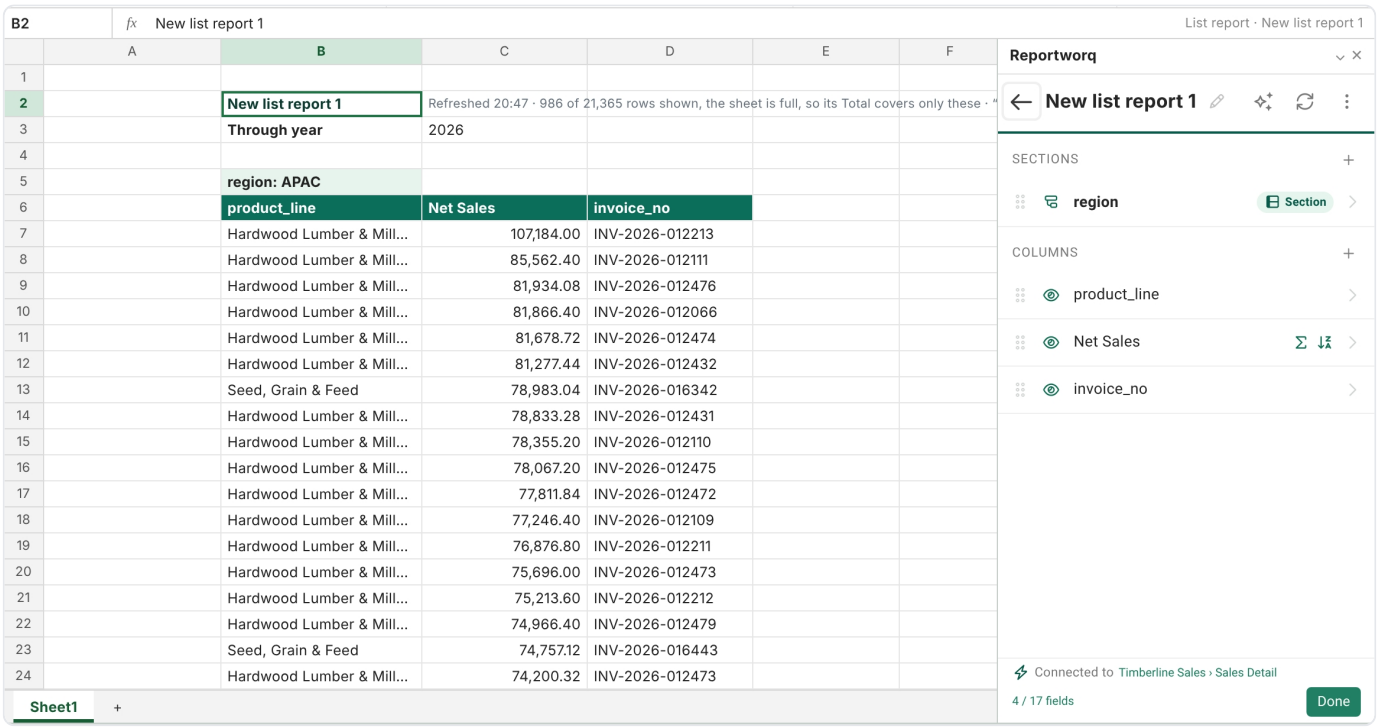
1. On the list's main page, select **Pick fields**. You can also select the **+** beside **Columns** or **Sections**.
2. Tick the fields you want. The list draws as you tick. Type in the box ("Search 30 fields...") to find a field. **Select all** and **Select none** act on the fields the search shows.



Select none leaves a field in place if it is hidden and only used for a filter or a Top N, so one click never removes a filter by accident.

Back on the main page, the list has up to three areas: **Filters** (shown when a field has a filter), **Sections** and **Columns**.

- **Drag a row by its grip** to move it. Hold **Ctrl** (or **Cmd**) and click, or **Shift** and click, to select several rows, then drag them together. Drop a field on the **Sections** heading to make it a section, or on **Columns** to make it a column.
- **The eye** on a column hides it from the output but keeps it available for sorting and filtering. Its tooltip reads "Hide (keep for sorting)".
- **Click a field's name** to open its page.



Set up a field

A field's page has three parts: **Header**, **Data** and **Filter**.

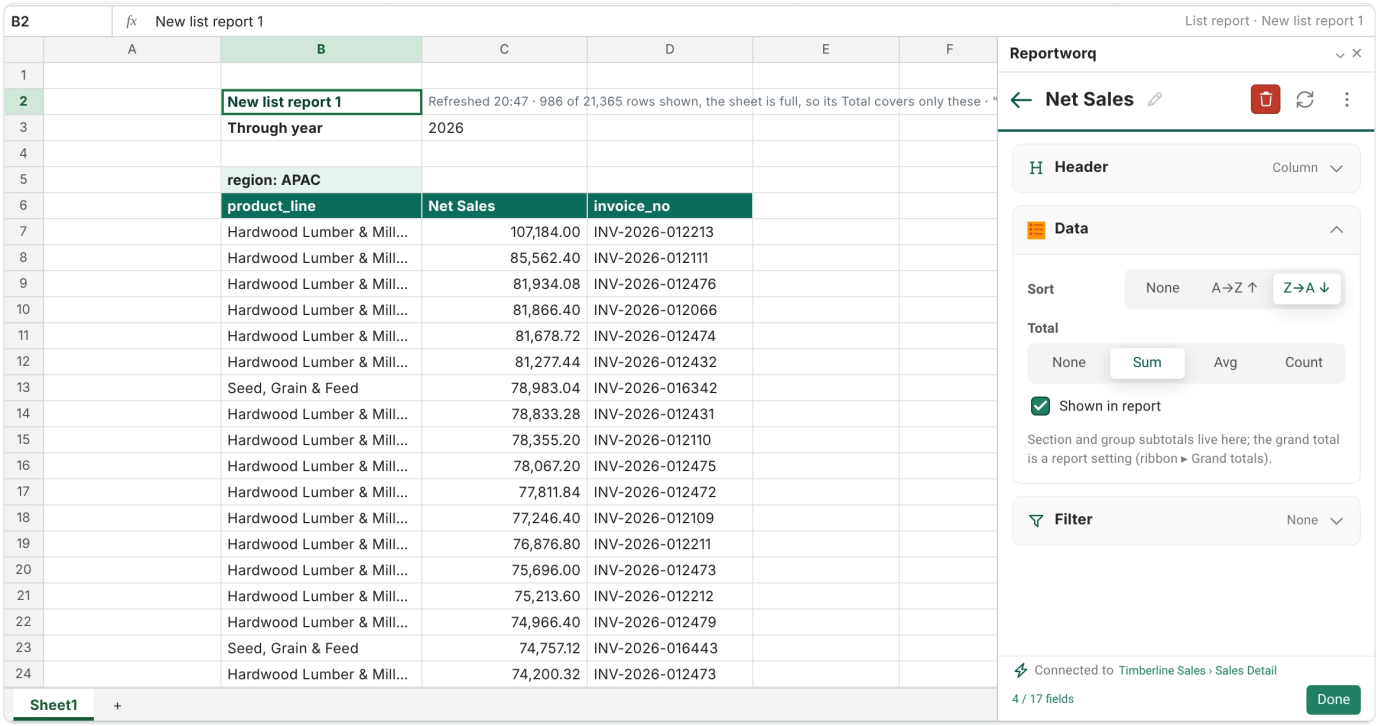
Header

- **Column**, **Section** or **Group**. A **Section** starts a new section for each value, with its own column headers. A **Group** puts rows under a heading for each value inside a section. A plain **Column** is the default.
- **Subtotals**, for a section or group: adds a subtotal row for the measures that have a total.
- **Don't repeat value**, for a text column: shows a repeated value only once, down the column.

Data

- **Sort**: **None**, **A→Z** or **Z→A**. With more than one sorting column, the order you chose them in decides which sorts first, and the page shows "(sort 1)", "(sort 2)". Dragging a column does not change the sort order.
- **Total**, for a number field: **None**, **Sum**, **Avg** or **Count**.
- **Transform**, for a text field: **None**, **Upper**, **Lower** or **Trim**.
- **Shown in report**: clear it to hide the field and still use it.

Filter: see the next section.



Sections and groups sort by their field's own sort: high to low when it is **Z→A**, low to high when it is **A→Z** or has none. Numbers compare as numbers, dates as dates, then text.

Filter a list

Open a field and expand **Filter**. Pick one of up to three ways to filter:

Mode	For	What you do
Select values	Any field	Tick the values to keep. Type in **Search values** to find one. **Select all** and **Clear** change every box.
Label	Fields that are not numbers	**Show items whose label** equals, does not equal, begins with, does not begin with, ends with, does not end with, contains, does not contain, is greater than, is less than, is between, or is not between a value.
Value	Number fields	**Show items whose value** is =, not equal, >, >=, <, <=, between, not between, **top N** or **bottom N**.

The checklist lists at most 200 values at a time with a search over all of them, and says "Showing 200 of N values. Search to find the rest." If you untick a value from an all-ticked list, Reportworq stores it as left out, so a value that appears later still shows. After you choose **Clear**, ticking keeps only the values you tick.

Top N and bottom N keep the highest or lowest N rows for a number field, within each section if the list has sections. A row with no value for that field is never ranked. Ties keep the row the list sorts first, so exactly N rows are kept. You can have both a

comparison and a Top N on one field. The list shows "Top 5 by net_amount, in each section" under **Filters**, with a button to clear it.

Text matches ignore case and leading and trailing spaces: "Low", "low" and "low " are one value. A text field whose values are all numbers or all dates tells you on its page and where to change its type, in the data model's field settings.

Each filter shows as a line above the list on the sheet and as a badge on the field.

Filters that apply to the whole list

The list's **Report filters** page applies before anything else. Open it from the **Report filters** button on the List tab or the list's menu.

1. Select **Add report filter**.
2. Under **Field**, pick the field. Changing the field clears the value.
3. Under **Equals**, pick a value, or under **Or variable** pick a variable that reads its value from a cell.

To use a cell as the value, open the **Variables** page, select **Add variable**, give it a **Name**, select the cell on the sheet, and select the **Use** button (it shows the cell address). Change the cell and refresh the list. For more on cell-driven values, see Cell-driven variables.

A view can also define its own variables, which appear on the list. See Data model variables.

Add totals

1. On the field's page, under **Data**, set **Total** to **Sum**, **Avg** or **Count**.
2. For a section or group subtotal, set **Subtotals** under **Header**.
3. For a total at the bottom of the whole list, open the **Totals** page (the **Grand totals** button on the List tab) and tick **Grand total row**. A total needs at least one value field with a **Total** set.

A **Count** total shows as a whole number. A cell with no value is left out of sums, averages and counts; it is not treated as zero. If a list has no rows, the total row shows zeros for **Sum** and **Count** and blanks for **Avg**.

Place the list and limit its rows

Place (**Place** button on the List tab) sets where the list sits and how it behaves when it grows:

- **Location** shows the anchor cell. **Move to** *Sheet!Cell* moves the list to the cell you have selected.
- **When the report grows**: **Insert rows (default)**, **Insert cells (only the report's columns move)** or **Overwrite (scratch sheets)**. The page names any content below or beside the report that would be pushed.
- **Import as an Excel table** writes the list as an Excel table.

Limit rows (the list's menu or the **Limit rows** button) cuts the list after sorting and filtering. Choose **No limit**, 10, 25, 50, 100, 500 or 1,000, or type a whole number in **Rows**. An empty box means no limit. A list in the pane draws at most the first 10,000 rows when it has no limit of its own; the chip reads "No limit (first 10,000 rows)".

Rows read from the source sets how many rows the list asks the view for. Choose **Default (10,000)**, 1,000, 10,000 or **All rows (up to 1,000,000)**. See the limits below.

Refresh the list

The list refreshes as you edit. Use the **Refresh** button in the pane's header, or **Refresh** on the List tab, to read the view again. **Clear data** removes the values and keeps the definition. If a list is slow to draw, Reportworq switches **Auto-update** off for that report and tells you; refresh it by hand after that.

If the list is cut short, a **Not every row** note appears in the editor, for example "This list read only the first 10,000 rows of its source." Select **Load all rows** to read up to 1,000,000 rows.

Notes and limits

Limit	Value	What happens past it
Rows read from the source, any list	1,000,000	A list that sorts, filters, ranks or has sections or groups is **not shown** and says why. Nothing is cut. Add a filter in the view or a data model variable, or set **Rows read from the source** to 1,000,000 or fewer.
Rows read, plain list with no limit	The first 10,000 in the pane	The list says it shows the first 10,000 rows. **Load all rows** reads up to 1,000,000. A job ignores this default.
Cells in one list	2,000,000 by default (an administrator can set 1 to 5,000,000)	Nothing is written. The list's status is **Too big for Excel** with the cell count and a **Limit to N rows** button.
Ask before writing	Above 100,000 cells (Excel only; an administrator can change it, 0 never asks)	The card reads "Waiting to write" until you choose **Write N cells** or **Cancel**.
Cells in one workbook	5,000,000, over all reports in a workbook in Excel	A report that would pass it writes nothing.
Sheet size in the studio	1,000 rows by 52 columns	A list stops at the last row and says "S of M rows shown, the sheet is full, so its Total covers only these." The list's own title, header and hidden format rows share the sheet, so it has room for fewer than 1,000 data rows.
Sheet size in Excel	Excel's last cell, XFD1048576	A list that would pass it draws the rows that fit and keeps its totals.
Read time	120 seconds	The list fails with "The source took too long." Add a filter or a lower row limit.
Checklist values	200 shown	Search finds the rest.

- **A cube has no rows for a list.** A list on a cube says "A list reads a data model's views, not a cube, so there are no rows to show. To report on a cube, use a crosstab."
- **Order of work.** The list reads its rows, applies field filters and then report filters, sorts, applies Top N within each section, applies **Limit rows**, and finally stops at the end of the sheet. Totals are kept when the sheet cuts the list.

- **Distribution jobs.** A job that refreshes a workbook with a list has no 10,000-row default, never cuts a list, and fails the sheet with a message if the list is too large. See Excel: the analyst pane.

> **Going deeper.** For a formatted, formula-driven report, see Professional reports. To describe a list in plain language and let Reportworq build it, see AI insights.

Key-cell formats and what a job keeps

explanation author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

When a job refreshes a list or crosstab inside a workbook, it has to draw new values in the formats you chose, even when the report grows or shrinks. Reportworq does this with **key cells**: one cell per kind of cell in the report (a heading, a value, a subtotal, a total) that holds the format for every cell like it. This page explains where key cells live, which formats a job keeps, and the one thing you must not do in a report's area.

Why it matters. An analyst formats the Revenue column of a list as `0.0%` in Excel and saves the workbook. A converted job must then send percentages with one decimal, not the default number format. Key cells are how that choice survives each run.

Where the key cells are

Under every list or crosstab, Reportworq keeps a few **hidden rows** called the format range. The format range holds the report's caption and one key cell for each kind of cell the report draws. When you format a report cell in Excel, Reportworq captures the format and applies it to the matching key cell, and to every cell like it. A key cell holds a placeholder or nothing, never the report's value.

In the Excel add-in, the format range is hidden by design. To see it, select **Show format range** in the report's View group; **Hide format range** puts it away again. Rows with no formats yet go out blank and hidden.

Screenshot coming soon

A list in Excel with its hidden format rows shown beneath it, key cells formatted

Which workbooks a job reads formats from

Workbook	Where the formats are	What the job does
Made by **Convert to Excel**, **Export as Excel** or **Download**, **Excel, connected** from the web report	In the report's stored definition	Draws the refreshed report in those formats. The format rows are blank and hidden.
Saved from the Excel add-in's pane, after you formatted key cells in Excel	In the key cells of the hidden format rows	Reads each key cell's Excel format and draws the refreshed report with it. A format you set in Excel is kept on the first run and on every later run.

So a job over a workbook the pane saved keeps the number formats, fonts, fills and borders you gave the key cells, such as `0.0%` on a value or a fill and bold on a column heading.

Number formats the report cannot carry

The report engine carries the common number formats. A few Excel number formats are not carried, for example Accounting, `#,##0.0`, `0.0` and `#,##0;(#,##0)`. For a key cell that uses one, the job keeps the rest of your look (font, fill, borders, alignment) and uses the key's default number format. A value's default is `#,##0`, or the data model field's own format.

What else a job keeps

- Your own formulas, names, merged areas, charts and conditional formats around a report move as Excel moves them when the report adds or gives back rows or columns.
- A column you hid stays hidden. A report's columns only widen to fit.
- Maps, AI insights and professional reports go out as last saved, where a job does not redraw them. The job freezes a professional report's cells to the values saved with them.
- The report definitions and the format rows' placeholders are removed from the output, so recipients see a clean workbook.

What makes a run fail instead

A job never overwrites your content to draw a report. If your content is in the report's area, the run fails with a message and sends nothing.

Message	Cause and fix
`<cell> is in rows <first>-<last>, which “<report>” keeps for its formats (hidden), so a refresh would overwrite it. Nothing was distributed. Move it below row <last>, then run the job again.`	You typed something into the hidden format rows under a report. Move it below the report.
A message naming a cell a growing report "would write over"	The report would grow into your cells. Move your content, or set the report's **When the report grows** option to **Overwrite** in the editor, which lets the report take the cells.
`“<report>” has grown larger than a job can refresh in place (it must end by row 1,048,576 and column XFD). Nothing was distributed.`	The report would pass Excel's last row or column. Reduce what it reads.
An insert or delete "would cut through a merged area"	A merged area is in the way of the rows a report adds or gives back. Unmerge or move it.

Only a report set to **When the report grows: Overwrite** may take cells that are not its own.

Notes and limits

- The storage names inside a converted workbook are a stable format. Do not edit a report's hidden definitions by hand: a job reports a damaged definition as one that "can't be read" and fails, instead of guessing.
- Copying a report's sheet inside the workbook is supported. A sheet copied in from another workbook is rebuilt only when its stored kit is intact. Otherwise it goes out as saved values and the run log says so.

> **Going deeper.** For converting, see Convert a web report to Excel. For the pane itself, see Excel: the analyst pane.

Join views across sources

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **joined view** combines the rows of other views of the same data model into one view. Its inputs are usually views of different sources: a forecast from a planning cube beside actuals from the ERP, or the sales reps in a CRM export beside the employees in an HR workbook. Report authors use a joined view exactly like any other view.

A joined view does not query a source itself. Each input view runs on its own, as last saved, and then Reportworq matches the rows. That is why a joined view can combine sources that could never be joined in one SQL statement.

> **Joining tables of one SQL database?** Use the view designer instead. It joins tables of one source inside the database, which is faster. See Build a view with the designer and Table relationships.

How a joined view works

A few rules shape everything else on this page:

- **A joined view reads only views of its own data model.** A data model is self-contained: anyone who can see the model can report on it, so a joined view never reaches into another model.
- **The views are related once, on the model.** A **relationship between views** says which fields match and how: its keys, its matching options, and any value mapping. It is stored on the data model, and every joined view that joins the same two views uses it. You create, edit, and delete these relationships where you use them, in a joined view's editor.
- **Which rows to keep belongs to each joined view.** Two joined views can follow the same relationship and keep different rows: one keeps every customer, another only the customers with no orders. Changing the keys or matching options of a relationship changes every joined view that uses it; changing **Keep** changes only the joined view you are editing.

- **Inputs run as last saved.** Save a change to an input view before you rely on it in a joined view.
- **An input can be Hidden, but not Disabled.** A Public joined view reads a **Hidden** input like any other, so you can keep its building blocks out of the Excel add-in. A joined view over a **Disabled** input fails with a message that names it, for example ""“CRM Accounts” is disabled", even in **Preview**; it never runs without that input's columns. See Choose who sees a view.
- **An exact match reads less.** When a match is exact, Reportworq asks the related view's source for only the rows whose keys it has already found. It never does this to a view whose rows with no match are the answer: with **Only in right**, Reportworq reads the related view first, in full, and asks the starting view's source for only the rows that can match it instead. The matching options that make a match fuzzy (ignoring case or surrounding spaces, or letting empty values match; ignoring the time of day too, when a key is text that holds dates), and turning off **Fetch only rows that can match**, make Reportworq read every row of the related view and compare the rows itself.
- **A joined view must be saved to run unattended.** A scheduled job or an API call runs a joined view only once it is saved in the data model.

Before you begin

- A data model with at least two views to join, saved. See Data models.
- You have permission to author data models in the workspace.

Build a joined view

1. Open the data model. In the rail, select **+ Add**, then select **Joined view**. You can also select the plus sign beside the **Joined views** folder in the rail.

   A new view named **Joined view** opens on its canvas.

2. Under **Pick a view…**, select the view to start from. Its rows are the ones every join hangs off, for example the sales journal when each row of the result is one sale. The list groups the views under the source they read, with other joined views under **Joined views**, and a long list has a search box. A view that reads this joined view, directly or through another joined view, is not offered, because joining it would make a loop.

3. On the starting view's card, select **Join a related view**, then select the view to join. The list shows the views the model already relates to this one, grouped by source, and leaves out any view that would make a loop.

4. If the view you want is not listed, select **Relate another view…**. The relationship page opens with this view filled in. Relate the two views (see Relate two views), select **Save and Close**, and the view joins.

5. On each card, select the check boxes of the fields to carry into the joined view. A card's key fields are always carried.

6. Repeat steps 3 to 5 on any card to join more views.

7. Select **Save Changes**.

A joined card's strip shows the keys it joins on and a pencil that opens the relationship. A connector line runs from the starting card to the field the card is joined on. A strip can also carry a short tag, described in the following table.

Tag	What it means	What to do
reads all rows	A matching option makes this match fuzzy, or **Fetch only rows that can match** is off, so Reportworq reads every row of the view and compares the rows itself. Point to the tag to see why.	Fine for small views. For a big one, run **Explain** to see whether the option changed any match.
key missing	A key field is no longer output by its view: removed, renamed, hidden, or excluded.	Edit the relationship and pick another key.
relationship deleted	The relationship this join used was deleted.	Edit it to relate the views again, or remove this view from the joined view.
this view only	This join is not one of the model's relationships between views yet.	Edit it to make it one, so other joined views can reuse it.

Relate two views

A relationship between views says how a row of one view finds rows of another. Every joined view that joins those two views uses it, so set it up once and check it on real rows before you save.

The relationship opens as a full page in the data model editor, from **Relate another view…** or from the pencil on a joined card.

1. Under **View** and **Related view**, check the two views. Both lists group the views under their source.

2. Under **Match on**, select a field of each view that holds the same value, for example **Sales Rep ID** and **Employee ID**. For a key made of several fields, select **Add another pair**.
3. Under **Keep**, select which rows the joined view keeps (see the table below). Opened from a joined view's card, the page says which other joined views share the relationship, and **Keep** applies to this joined view only. When you create a relationship, **Keep** is also the starting choice for the next joined view that joins the two views; each joined view can then change its own.
4. Under **Each row finds**, select **At most one match** when the related view is a lookup (one row per key), or **Many matches** when a row can match several. With **At most one match**, **Test match** warns when the key repeats in the related view, which would double-count totals.
5. Under **Matching**, select only the options the data needs (see the table below).
6. Optional: check the match with **Test match**, or let Reportworq find the key with **Discover relationship** (see Check a relationship on real rows).
7. Select **Save and Close**, then select **Save Changes** to save the data model.

The **Keep** options are the following.

Option	Rows the joined view keeps
Matches only	Only rows that match in both views.
All from left	Every row of the first view. The related view's columns are empty where nothing matches.
All from right	Every row of the related view. The first view's columns are empty where nothing matches.
Everything	Every row of both views, matched where possible.
Only in left	Only the rows of the first view that have no match in the related view, for example customers with no orders. Only the first view's columns are kept.
Only in right	Only the rows of the related view that have no match in the first view, for example orders whose customer is missing. Only the related view's columns are kept.

Only in left and **Only in right** find what is missing. A row has no match when it would find none under **Matches only** with the same keys and **Matching** options, so the value mapping, ignoring case, spaces or the time of day, and **Empty values match each other** count exactly as they do for the other options. A row whose key is empty has no match (and is kept), unless **Empty values match each other** is on and the other view has an empty key too. Because only one view's columns are kept, a later join can't start from the view whose columns were dropped: after **Only in left**, join further views from

the first view's side; after **Only in right**, join them from the related view. On the canvas, a card whose columns were dropped shows no fields carried, and its **Join a related view** button is unavailable; point to it to see which join dropped them. **Each row finds** doesn't apply to these two options, because no row is ever repeated.

> **Upgrading and older versions. Only in left** and **Only in right** are new in this release. A data model that uses them must not be opened in, imported into, or run by an older version of Reportworq: an older version doesn't know these two options and runs such a join as **Matches only**, without a warning, so it returns the matching rows instead of the rows with no match. Upgrade every server that shares the data model before you use them.

The **Matching** options are the following. The first three, and **Ignore time of day** on a text key, are marked less efficient: each one makes the match fuzzy, so Reportworq reads every row of the related view instead of asking its source for only the rows that can match. The page says so under the options when that applies.

Option	What it does
Empty values match each other	Rows whose key is empty match one another. By default an empty key never matches.
Ignore upper/lower case	`NORTHERN BROTHERS` matches `Northern Brothers`.
Ignore surrounding spaces	`Northern Brothers ` matches `Northern Brothers`.
Ignore time of day	For date keys: dates match on the day alone, so an order placed at `2026-03-02 14:30` matches the calendar day `2026-03-02`. A date with a time zone is compared in UTC, so its day can differ from the day in its own time zone. Shown only when a key pair has a date field (or the option is already on), because it changes nothing else. It costs no speed on a date key, because Reportworq never asks a source for just the matching dates; it makes Reportworq read every row only when it is on a text key.
Fetch only rows that can match	On by default. Reportworq asks the related view's source for only the rows whose keys are already there, when that is safe.

To delete a relationship, open it from a joined card's pencil and select **Delete this relationship** at the bottom of the page. The page says what else the delete changes:

the related view (and anything joined to it) leaves this joined view, and any other joined view that uses the relationship stops working until the views are related again.

Line up values written differently

Two systems often write the same thing differently: the ERP has a customer called `Northern Brothers` and the CRM calls it `Northern Bros.`. A **value mapping** lines the two up for one key pair, so those rows match. It belongs to the relationship, so every joined view that joins the two views follows it. Values that are not in the mapping match as they are.

1. On the relationship page, under the key pair, select **Value mapping**.
2. Add the mapping in any of these ways:
    - Select **Add a line**, then enter the value as the first view writes it and as the related view writes it.
    - Select **Suggest from the data**. Reportworq reads the first 10,000 rows of each view, finds the values that have no match, and lists the value each most likely means, with how alike the two are and how many rows it affects. Select **Add** on a suggestion, or **Add all**. Check each suggestion first: a close spelling is not always the same thing. Values that still have no match are listed so you can map them by hand.
    - Select **Paste** to replace the mapping with two columns copied from Excel or a text editor. **Copy** copies the mapping out the same way, so you can keep it in a workbook. If the browser does not allow Reportworq to use the clipboard, a text box opens for you to paste into or copy from by hand.
3. Select **Save and Close**.

The **Value mapping** heading shows how many values are mapped. **Test match** reports how many rows match only through the mapping.

Check a relationship on real rows

The relationship page has two checks beside the form, under **Check it on real rows**. Both read each view the way a report would, with its fields, filters, and default variable values.

Test match

Test match runs the match on the first 10,000 rows of each view, without building the joined rows, and reports:

- How many rows of the first view find a match, and the percentage.
- How many rows of the related view are matched by something.

- How many rows of the first view find more than one match and so repeat, and how many rows come out as a result. If each row should find one match, the key is not unique in the related view.
- How many rows match only through the value mapping.
- How many rows have an empty key and so never match.
- With **Only in left** or **Only in right**, how many rows have no match, which are the rows the joined view keeps.

A table then compares the matching combinations side by side, marks the current one, and offers **Use** on the others. For text keys it has four rows (exact, ignoring spaces, ignoring case, and both). When every key is a date, it has two rows: exact matching and **Ignore time of day**. When the keys mix a date with text, it has eight: the four text combinations, each with and without ignoring the time of day. A relationship saved with an option that its keys no longer offer still shows that combination, marked as current. Advice under the table says when a fuzzy option changes nothing, when another combination finds more matches, or when the current fuzzy option finds matches that exact matching misses. When a fuzzy option changes nothing, the advice is to turn it off. Exact matching is then also faster, because only the rows that can match are fetched, except when a key is a date: Reportworq never sends date keys to a source, so every row of the related view is read either way, and the advice says so.

Discover relationship

Discover relationship finds the key for you. It reads the first 1,000 rows of each view and tries every pair of fields, one from each view. It skips yes/no fields and fields that share only a few repeated values, such as a region or a currency code, because matching on those repeats rows.

It shows the four best pairs, each with its match rate, whether the field is unique in the related view, and whether it needs a fuzzy option (for two date fields that match only on the day, **Ignore time of day**). Select **Use this** on a pair to fill in the key, the matching options, and **Each row finds**, and Reportworq then runs **Test match** on the larger sample.

If no pair of fields looks like a key, the two views may not share one, or the key may be in a field one of them does not output.

Read the Explain plan

Explain on a joined view's canvas runs the joins, with the starting view capped at 100 rows, and shows **How this runs**: each numbered step, what it read, and what it matched. Each finding is graded with a mark, and the summary at the top counts the findings of each grade:

- **Helping** (a check mark): the step works in your favor, for example only the matching rows were read, or every row found a match.
- **To check** (a warning triangle): worth a look, though not necessarily something you did, for example rows that found no match, or empty keys.
- **Hurting** (a cross): the step costs you, for example rows repeated by a key that matched more than once (watch totals), or a fuzzy matching option that made Reportworq read every row and changed no match.

For **Only in left** and **Only in right**, the step says how many rows have no match, for example `212 rows of 'Customers' have no match in 'Orders'.` With **Only in right** as the first join, the 100-row preview cap applies to the related view, whose rows are kept, so every row shown really has no match.

Use **Explain** after you build a joined view and whenever a report on it runs slowly.

Variables of a joined view

A joined view lists the variables its input views use, under **From the joined views** on its **Variables** tab, and the variables its own filters use. These rules apply:

- **One name is one variable.** When two inputs use a variable with the same name, the joined view has one variable, and one value drives every input that uses it. A line on the variable's card names the inputs: **Used by** *view* for one, **One value for** *view, view* for several.
- **Only the variable's existence is inherited.** Its default, **Required**, caption, help text, and choices belong to the joined view, set on its own **Variables** tab. Nothing is copied from the inputs, and inside a joined view an input never falls back to its own default.
- **Use another view's settings when only one input uses the variable.** Select **Use the settings from** *input view* on the variable's card to apply that input's settings (its default, **Required**, and prompt) instead of your own. The card then reads, for example, **Uses the settings from Sales Journal: default 2024**. The option is available only while exactly one input uses the variable, because the settings of several inputs could disagree.
- **If a second input starts using the variable later, the variable has no default.** This happens even when the second input is edited elsewhere in the model. The variable then has no default, and is required only if an input pastes it into its query text. The joined view's own default and prompt that were hidden behind **Use the settings from** *input view* do not come back, so set them again on the **Variables** tab if you want them.
- **A joined view's own filters can use a variable too.** Add a `{{Token}}` to a filter on one of the joined view's fields, and the variable shows on its **Variables** tab with

the same settings as any filter variable: a default, **Required**, a caption, help text, and choices. It is optional unless you select **Required**, or an input uses the same name in its query text (see below). Left without a value, an optional one skips its filter, so every row the join returns is kept. When an input uses a variable with the same name, it is one variable: you are asked once, and the one value goes to the input and to the joined view's filter. Its card then reads, for example, **Used by Sales Journal and this view's filters**. That includes **Use the settings from** *input view*: the input's default then fills the joined view's own filter too, so a run with no value filters on it.
- **A variable used in a query's text is always required.** When any input pastes the variable into its query text (a `{{Token}}` in its SQL, MDX, or DAX), the joined view shows it as **Required** and cannot make it optional, because a blank value would break that query.

A joined view never asks for sample values. Its fields come from its inputs: each variable takes the joined view's own default, or else the sample or default of an input that uses it. A variable with none of these is reported by name when the fields are read.

For defaults, choices, and the rest of the **Variables** tab, see Data model variables.

Limits

A joined view holds its inputs in the server's memory while it matches them, so four limits keep one mistaken join from exhausting the server. When a run goes past one, it stops with a message that names the input and the limit, so the author can filter the input or carry fewer fields.

The limits and their defaults are the following.

Limit	Default	What it caps
Rows Read per Joined Input	1,000,000 rows	The most rows a joined view reads from any one input view.
Joined View Memory Budget	20,000,000 cells	The cells (rows times columns) held across all of a joined view's inputs in one run. A cell takes about 50 bytes, so the default is roughly 1 GB.
Rows Returned by a Joined View	2,000,000 rows	The most rows a joined view returns, so a key that matches many rows cannot multiply the output without bound.
Join Keys Sent to a Source	1,000 keys	The most key values Reportworq sends to a source when it asks for only the matching rows. Above it, that input is read whole.

An administrator sets these for every data model on the server, in **Settings** > **Configuration** > **Performance** > **Data Models**. A value of 0 turns that limit off. See Server configuration.

Inputs that are read in full. Reportworq asks a source for only the matching rows only when the source can filter rows itself, which today means a SQL database. A Pigment or Planning Analytics (TM1) input is always read in full: no key values are sent to it, and **Explain** says so on that step. Its rows still count toward the limits above, so keep such an input narrow with the input view's own filters and variables.

When a joined view hits a limit:

- **Too many rows from one input.** Filter the input view, give it a variable, or join a smaller view.
- **Past the memory budget.** Remove fields you do not need from the input views, or filter the inputs.
- **Too many rows returned.** Check **Each row finds** and run **Test match**: a key that is not unique in the related view repeats rows.

Related topics

- Data models
- Data model variables
- Build a view with the designer
- Table relationships

Crosstabs

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A crosstab adds up a measure across row groupings and column groupings, like a PivotTable that stays connected to your data model. You choose which fields go down the side, which go across the top, and which number to add up. Reportworq draws the grid into your sheet and refreshes it from the view.

When to use it. You want a summary. For example, **net_amount** from the Timberline **Sales Details** view by **product_line** down the side and **region** across the top, with a grand total row and column. If you need the individual invoice lines behind a number, use a list report, or drill into the crosstab cell.

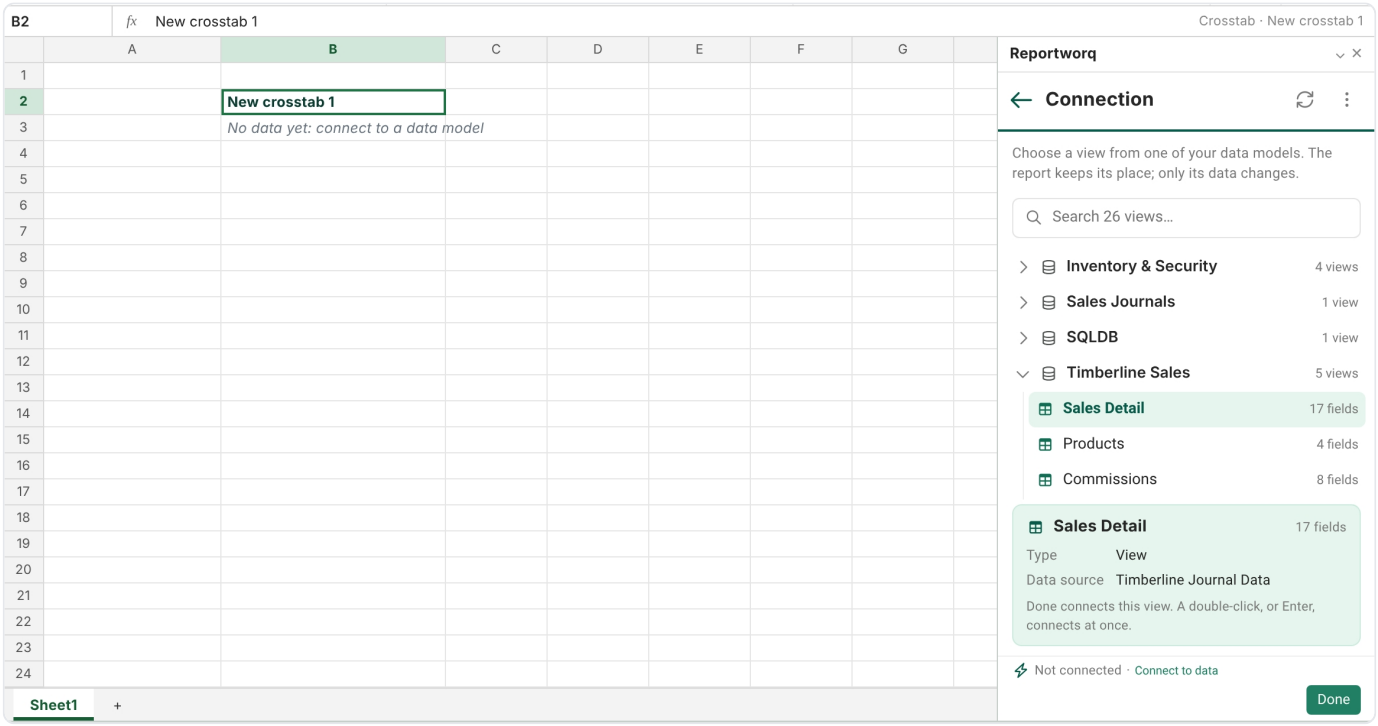
You build a crosstab the same way in Excel and in the web report studio, because both host the same Reportworq pane. See The Excel add-in and the studio.

Before you start

- You are signed in as an author or a system administrator.
- A data model with a **Public** view you can see, or, for a crosstab on a cube, a saved connection to TM1, Adaptive, Vena or Pigment. A **Hidden** view is not listed and a **Disabled** view never runs. See View visibility.
- In Excel, a version that supports the Reportworq pane. In the studio, a web report open in **Editing** mode. See The web report studio.

Start a crosstab

1. Click the cell where the crosstab's top-left corner should go.
2. Start it:
    - **In Excel:** on the **Reportworq** tab, in the **New report** group, select **Crosstab**. Or open the pane and, on **Home**, use the crosstab quick button on a view in the model list, or **Crosstab** under **Or start from a report type**.
    - **In the studio:** on the **Reportworq** tab, in the **New** group, select **Crosstab**. The **Insert** tab has the same button.
3. On the **Connection** page, choose a view and select **Done** (or double-click it). Selecting a view does not connect it until you do.

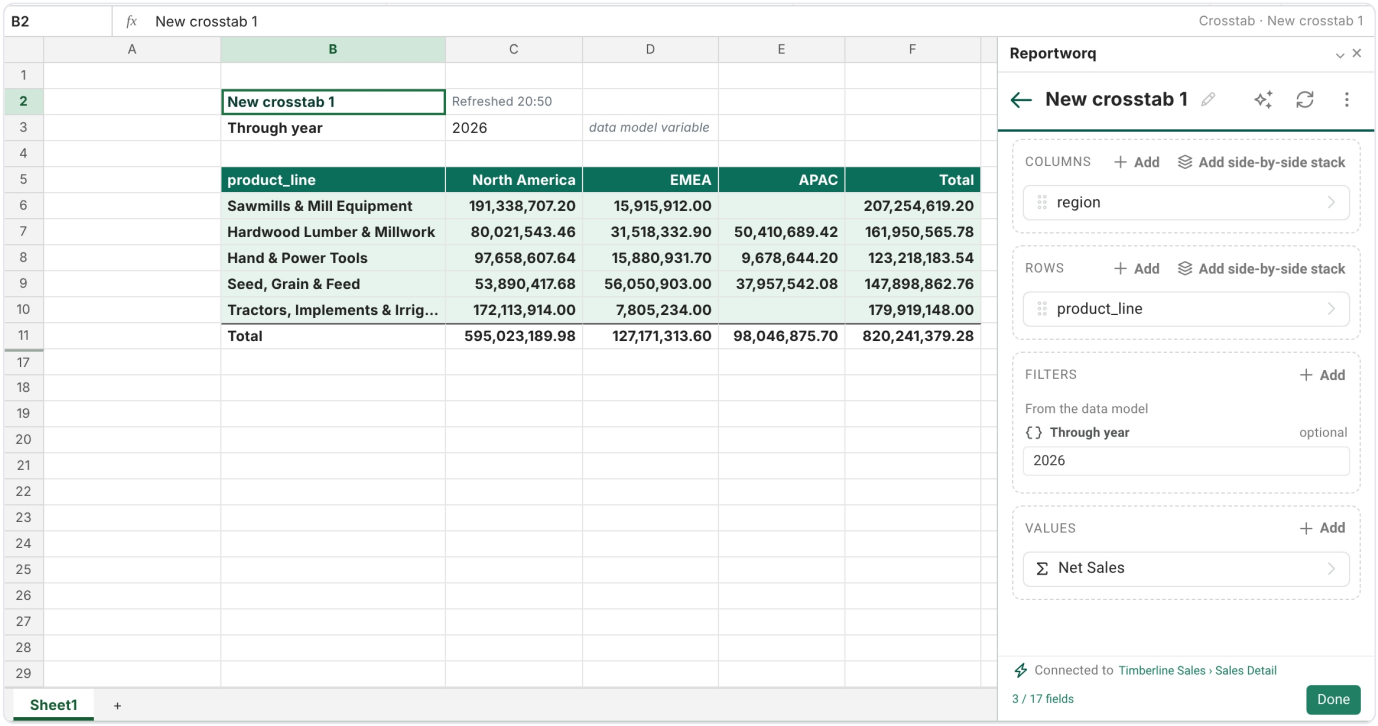


A new crosstab starts with one empty row grouping and says "Add a value (e.g. Revenue) to see data" on the sheet until it has a value.

Arrange the crosstab

The crosstab's main page has four zones: **Columns**, **Rows**, **Filters** and **Values**.

1. Under **Values**, select the **+** and choose a number field, for example **net_amount**.
2. Under **Rows**, select the **+** (tooltip "Add a row field") and choose a field, for example **product_line**.
3. Under **Columns**, select the **+** ("Add a column field") and choose a field, for example **region**. The Rows and Columns pickers offer text fields; a number such as **year** is offered under **Values** instead.
4. Optional: under **Filters**, select the **+** ("Add a filter") and choose a field to filter by. See "Filter a crosstab" below.



The grid renders live as you add fields. Add more than one field to **Rows** or **Columns** to nest them; the first is the outer level. Rows render expanded to level 2, with expand and collapse glyphs in the cells. Select a glyph on the sheet to open or fold a row.

A value is always added up with **Sum**. Add several values and they sit side by side under each column. A value's page offers **Move up**, **Move down** and **Remove value**.

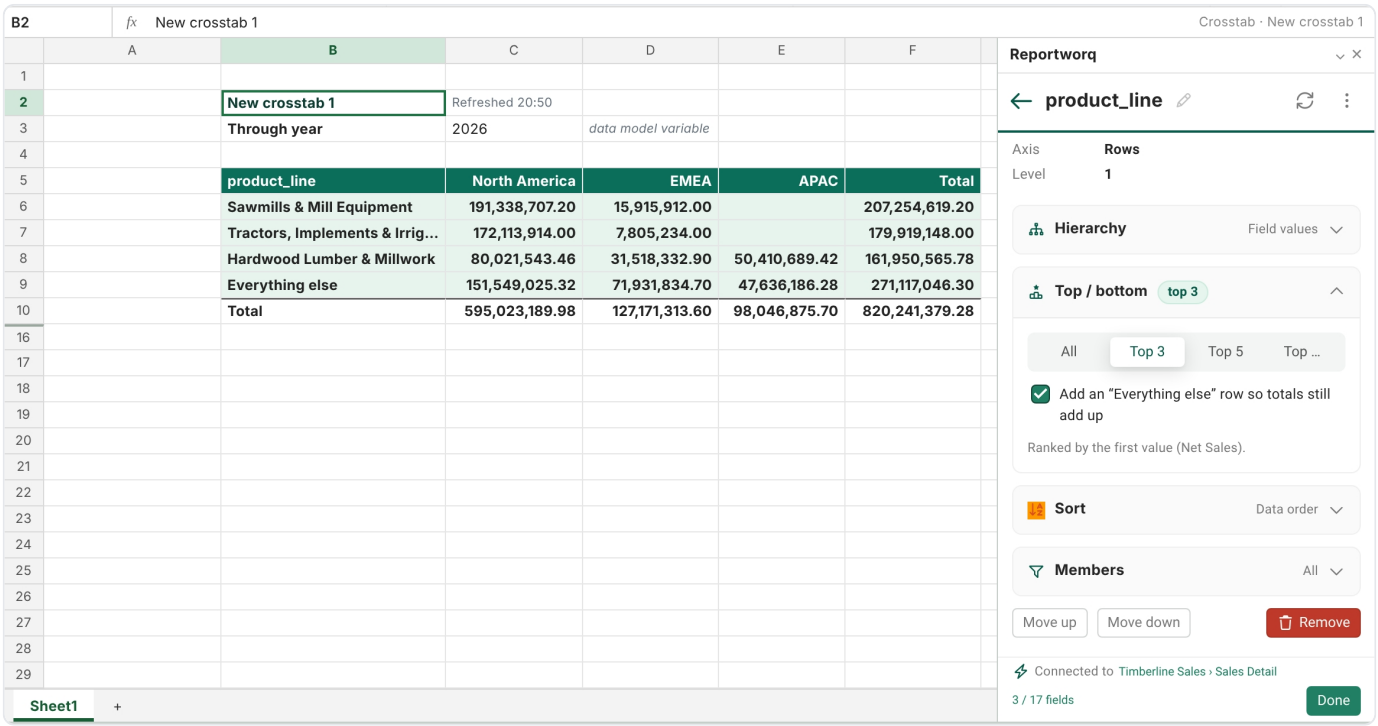
To put two independent sets of rows or columns next to each other over the same other axis, select **Add side-by-side stack** on **Rows** or **Columns**. To nest more fields inside a stack, use that stack's own **+**. Drag a stack by its grip to reorder stacks.

Set up a row or column field

Select a field under **Rows** or **Columns** to open its page. It has these sections:

Section	What it does
Hierarchy	**Show as** the field's own values (**None (field values)**) or through a hierarchy defined for the model, and **Expand to level**. For a cube field the hierarchy is the cube's own. See Hierarchies.
Top / bottom	**All**, **Top 3**, **Top 5** or **Top 10** of this field, ranked by the first value. Tick **Add an "Everything else" row so totals still add up** to keep the grand total correct.
Sort	**A → Z** or **Z → A**. Not available on a cube field, which keeps the cube's order.
Members	A checklist of the field's members. It narrows the field to the members you tick.
Move up, Move down, Remove	Reorders the field within its stack, or takes it out.

The **Members** summary reads "All" until you narrow it. With a member filter on a row or column field, the field also shows in the **Filters** zone, marked "on rows" or "on columns", and the filter shows above the crosstab on the sheet.



Filter a crosstab

There are three kinds of filter, and all of them narrow every total, Top N, "Everything else" row and grand total.

- **Page filters** (the **Filters** zone). Pick a field and its members on the filter's page: search, **Select all**, **Select none**, then **OK** or **Cancel**. Select **Remove filter** to take it away. On the sheet, a page filter is a drop-down above the crosstab that reads **(All)**, one member, or **(Multiple items)**. Pick from the drop-down to re-render. A drop-down shows up to 200 items. In Excel, a list longer than 255 characters in all, or with an item that contains a comma, gets no drop-down; the cell is plain text.
- **Member filters** on a row or column field, from its **Members** section.
- **Report filters**, from the **Report filters** button on the Crosstab tab or the crosstab's menu. Select **Add report filter**, pick a **Field**, then either a value under **Equals** or a variable under **Or variable**. These apply before anything else, and each shows as a line above the crosstab. A crosstab on a cube does not use report filters; use a dimension as a page filter instead.

To clear a filter, pick **(All)** in its drop-down. To clear an optional variable, pick **(Any)**. Pressing Delete does not clear either.

If the view has data model variables, they show in the **Filters** zone under **From the data model**, and as drop-downs on the sheet. A required variable with no value shows as not set. See Data model variables and Cell-driven variables.

Text matches ignore case and leading and trailing spaces: "East", "East " and "EAST" are one row or column.

Totals

Open the **Totals** page (the **Totals** button on the Crosstab tab) and tick **Grand total row and column**. The crosstab then adds a total across each row and down each column.

Place the crosstab and limit what it reads

Place sets the anchor cell and what happens when the crosstab grows: **Insert rows (default)**, **Insert cells (only the report's columns move)** or **Overwrite (scratch sheets)**. **Import as an Excel table** is also there.

A crosstab has no **Limit rows**. It adds up every row it reads, so a limit applies to the source. Under **Rows read from the source** choose **Default (1,000,000)**, 1,000, 10,000 or **All rows (up to 1,000,000)**, or type a number. If you pick a number below the source's size, the crosstab adds up only those rows and says "first N rows" and that its totals cover only them.

Work with the numbers

- **Drill.** Select a cell and use **Drill to transactions** to see the rows behind it, or **Drill to crosstab** to start a new crosstab on a new sheet filtered to that cell, with the next level on rows. **Put this into a new sheet** turns the rows into a live list.
- **Swap rows and columns** on the Crosstab tab flips the axes.
- **To professional report**, on the Crosstab tab, converts the crosstab into a professional report where the studio offers it. It refuses a crosstab with report filters, because the professional report cannot carry them. See Professional reports.

Notes and limits

Limit	Value	What happens past it
Rows read from the source	1,000,000	A crosstab on a data model needs every row, so above this it is **not shown** and says why. Nothing is cut. Add a data model variable or a filter on the view, or set **Rows read from the source** to 1,000,000 or fewer.
Cells in one crosstab	2,000,000 by default (an administrator can set 1 to 5,000,000)	Nothing is written. The status reads **Too big for Excel** with the cell count and a **Limit to N rows** button that sets the rows read from the source.
Ask before writing	Above 100,000 cells (Excel only; an administrator can change it, 0 never asks)	The card reads "Waiting to write" until you choose **Write N cells** or **Cancel**.
Cells in one workbook	5,000,000 over all reports	A report that would pass it writes nothing.
Sheet size in the studio	1,000 rows by 52 columns	A crosstab reads rows to fit the grid and says "first 1,000 rows" beside its refresh time, with its total covering the rows shown.
Sheet size in Excel	Excel's last cell, XFD1048576	A write past it is refused: "Reportworq can't write past the last cell Excel has (XFD1048576)".
Drop-down items	200 per drop-down	Narrow the field with a member filter.
Read time	120 seconds	The crosstab fails with "The source took too long."

- **Cube crosstabs.** A cube's views are its cubes; its fields are the dimensions plus one value. Only authors and administrators see cube connections, because the cube is read with the connection's stored account. TM1 is read for self-hosted

native or CAM logins only.

- **Distribution jobs.** A job that refreshes a workbook with a crosstab refreshes it in place and sends a static copy. It ignores the pane's defaults, reads up to 1,000,000 rows, and fails the sheet with a message if the crosstab is too large. See Excel: the analyst pane.

> **Going deeper.** To chart a crosstab, see Charts and maps. To place a list and a crosstab on one page, see Combine report types.

Table relationships

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **relationship** tells the view designer that a column of one table holds the key of a row in another table: each invoice line's `sales_rep_id` is the `employee_id` of one employee. The designer uses relationships to offer related tables on the **Design** canvas, so you can add the rep's name to an invoice-line view without writing a join. See Build a view with the designer.

Where relationships come from

A data model on a SQL datasource draws relationships from three places. The designer offers all of them side by side.

Source	What it is	Shown as
The database or driver	Foreign keys the database declares, read from its metadata.	**Declared by the driver**
Reportworq	Links Reportworq works out from a source's naming rules, for QuickBooks Online report views, where a column such as `Class_Id` is paired with the `Class` table.	**Inferred**
The data model	Relationships you add yourself on the model's **Source** screen. They are saved with the data model and apply only to it.	**Manual**

Which sources supply relationships on their own:

- **SQLite** databases supply the foreign keys they declare, for single-column keys. Database views declare none.
- **CData-based sources**, such as Salesforce, NetSuite, and Dynamics 365, supply the foreign keys their driver reports. A flat source such as an Excel workbook usually reports none.
- **QuickBooks Online** report views supply inferred links from their id columns to the Accounts, Class, Customers, Vendors, Employees, Items, Departments, and PaymentMethods tables.

- **SQL Server, OLE DB, and ODBC** sources do not supply relationships yet, so every link on those sources is a manual relationship.

When you need a manual relationship

Add a manual relationship when a column holds another table's key but the designer does not offer that table. Typical cases:

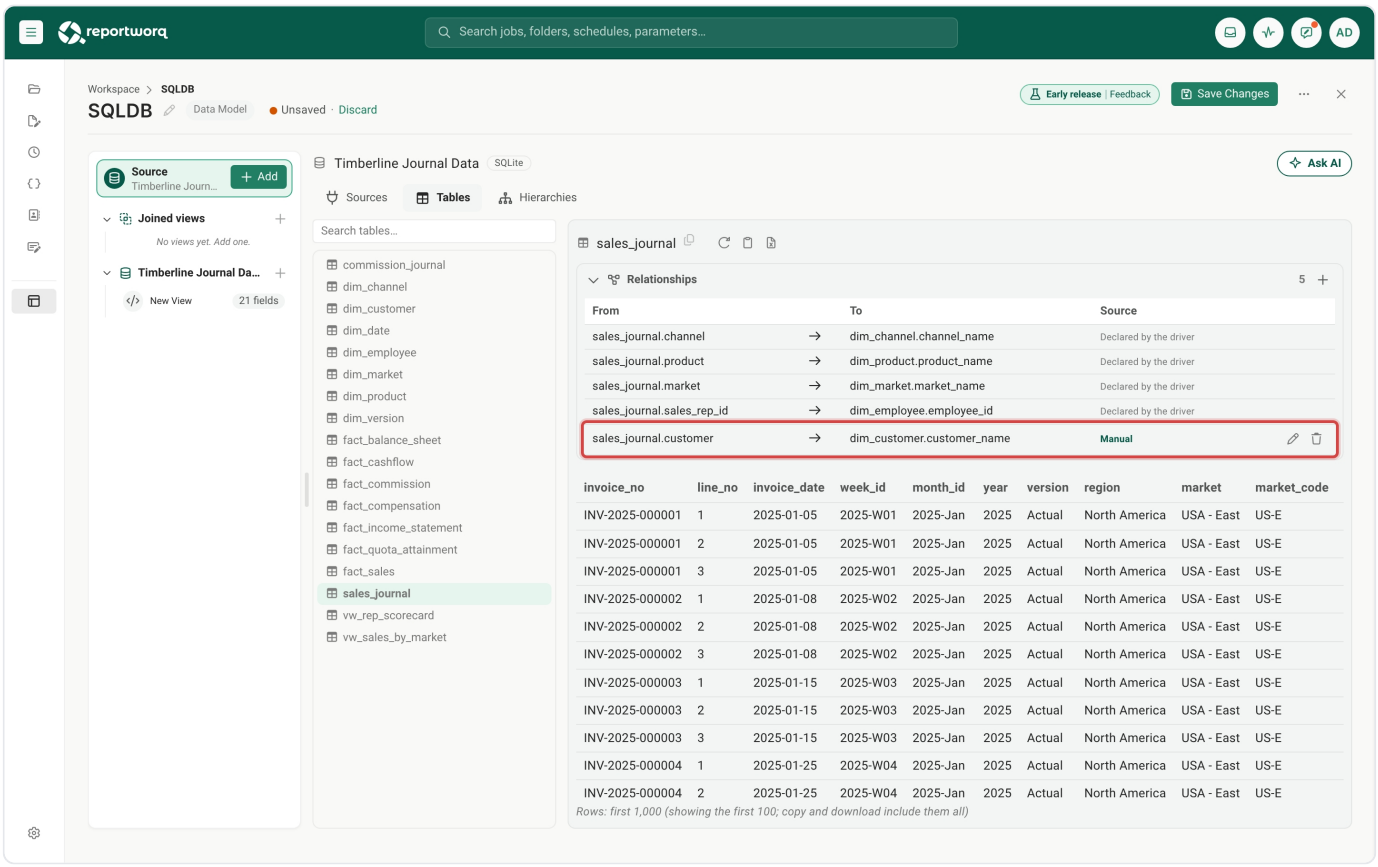
- The database does not declare foreign keys, which is common in reporting databases and data warehouses, and is always the case for SQL Server, OLE DB, and ODBC sources in Reportworq today.
- The key is a name or a code rather than an id, for example an invoice line's `customer` column that holds the customer name, which `dim_customer` uses as its key.
- The starting table is a database view, which never declares keys.

A relationship must point at a key: each value in the first table's column must match at most one row of the second table. If it can match several, the view would repeat rows; write that view as SQL instead.

Add a manual relationship

1. Open the data model and select **Source** in the rail.
2. In the table list, select the table that holds the key, for example `sales_journal`.
3. Above the table's rows, open the **Relationships** section. The table's existing links are listed with their source.
4. Select the plus button beside **Relationships** to add a relationship from this table.
5. Check **From table**, which starts as the table you selected, and select the **Column holding the id**, for example `customer`.
6. Select the table the id **Points to**, for example `dim_customer`, and its **Key column**, for example `customer_name`.
7. Select **Save relationship**.
8. Select **Save Changes** to save the data model with its new relationship.

The relationship is listed as **Manual**, and the designer now offers the related table beside that column on the **Design** canvas. Its tooltip on the canvas reads "defined on the data model".



Each table's **Relationships** section lists the links from that table, both the ones the source supplies and the manual ones.

Edit or delete a manual relationship

- To change a manual relationship, select the pencil on its row, change the columns or tables, and select **Save relationship**.
- To delete one, select the trash can on its row and confirm.

Deleting a manual relationship does not change views that already join through it; they keep their joins. It only stops the designer from offering the link for new joins. Declared and inferred relationships are read-only.

Related topics

- Build a view with the designer
- Data models
- Connect QuickBooks Online
- Connect SQLite

Charts and maps

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A chart or map is a report type of its own. You build it from cells on the sheet, or from a list or crosstab you already have, and it is drawn live while you work. When a distribution job delivers the workbook, charts are redrawn as pictures from the data of that run.

When to use it. You have a Financial Reports crosstab of revenue by quarter and want a column chart beside it, or a table of store locations and want bubbles sized by sales on a map.

Where you can build one

	Web report studio	Excel add-in
Charts (13 kinds) and maps (2 kinds)	Yes	Yes, in the analyst pane (the default pane)
Legend, data labels and gridline settings	Yes	Yes, in the analyst pane. The older companion pane has no controls for them
Map color, size and label by column	Yes	Yes, in the analyst pane only
Live chart with tooltips on the sheet	Yes	The chart is placed as a picture on the sheet

If an administrator turns off **Use the analyst pane in Excel**, the add-in shows the older companion pane. It has its own **New chart** builder with the same chart and map types, but it keeps the legend, labels and map styling on their defaults.

Before you start

- Have the data on the sheet: a table of cells, a list report or a crosstab. A chart does not run a query itself.
- Optional: an enabled AI connection if you want **Fill in from** (the AI suggests which columns are labels and which are series). Without one, a built-in rule does the same

job. See AI insights for what AI features send.

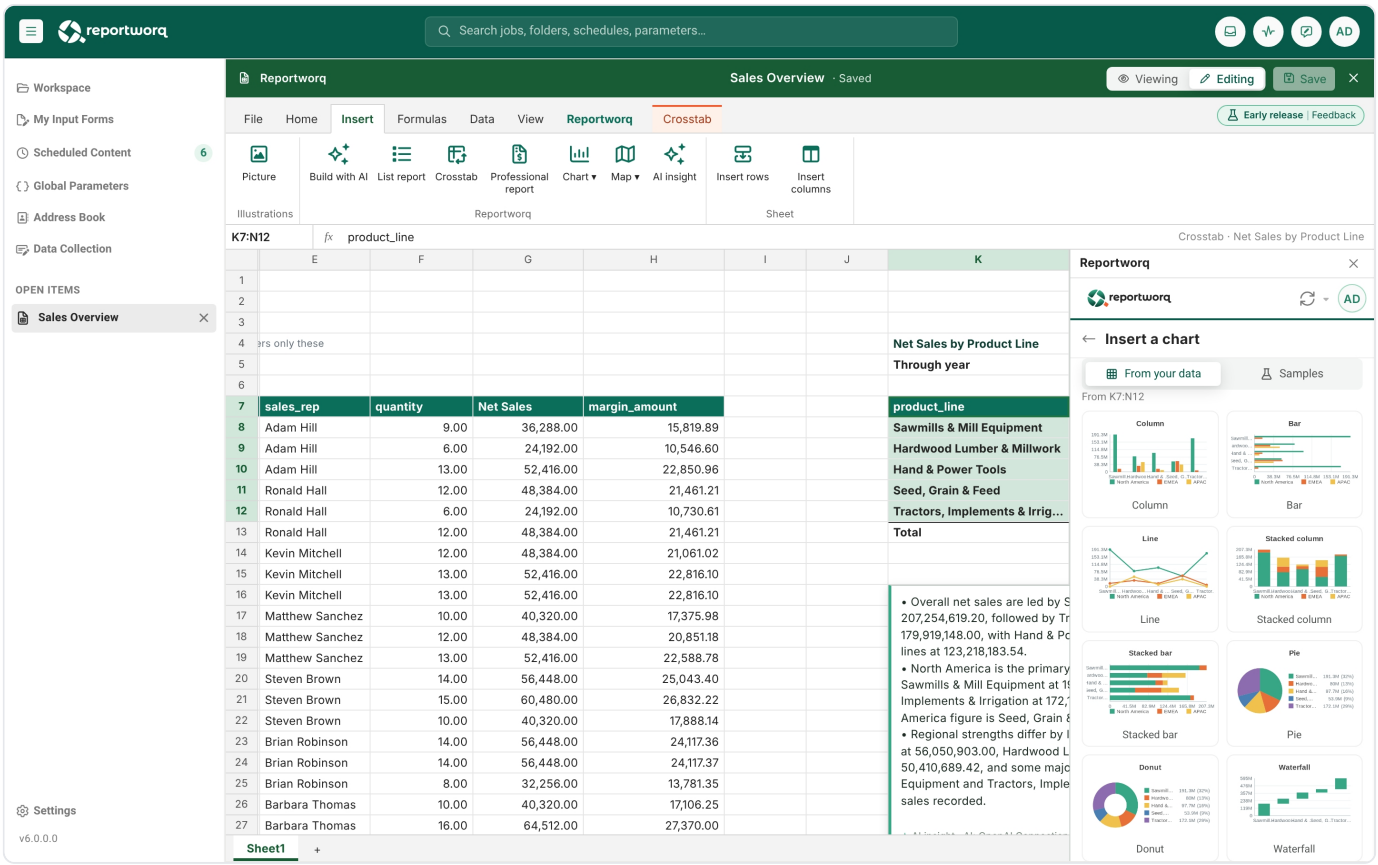
Chart and map kinds

Kind	What it draws	Notes
Column, Bar, Line	One series per value column	Bar puts the value axis along the bottom
Stacked column, Stacked bar	Series stacked	Positive and negative values stack apart
Pie, Donut	Share of a whole	A donut is a pie with a 50 percent hole
Waterfall	A running bridge	Up and down steps are colored differently
Combo	Each series is Column, Line or Area	Tick **2nd axis** to put a series on a right-hand axis, for example revenue columns and a margin percent line
Range bars, Range area	Each category spans a low and a high value	A range chart can label each band with the difference
Heatmap	A matrix colored by value	Reads a crosstab block or a long table of row, column, value
Sankey	Flows between nodes	Reads a table of source, target, value
Map: Points	Markers on a map	Reads a table of label, latitude, longitude
Map: Bubbles	Circles sized by a value	Same table plus a value column

Add a chart

1. Select the cells that hold the data, with a header row if you have one. The first column is used as the labels when it is text or years. A block with no text column is labeled 1, 2, 3 by row.
2. On the **New** group of the ribbon (web report studio) or on **Home** in the pane (Excel), choose **Chart**, then **Chart from your data**.
3. On the **From your data** tab, every chart and map kind is previewed with your selection. Pick the one that reads best. The previews use the top-left 30 rows by 7 columns of the selection.
4. The chart is placed at 480 by 300 pixels and never on top of another picture. Reportworq looks for room beside your selection; when a report is in the way it can land elsewhere on the sheet, and a message names the cell, for example "Chart placed at Sheet1!A40".
5. Open the chart to review its set-up pages: **Data**, then **Legend & labels**. A heatmap has **Matrix**; a Sankey has **Flows** and **Appearance**.

To start from an example instead, choose **Sample chart or map**. The samples write their own data at the active cell and ask "Write sample data?" before overwriting cells.

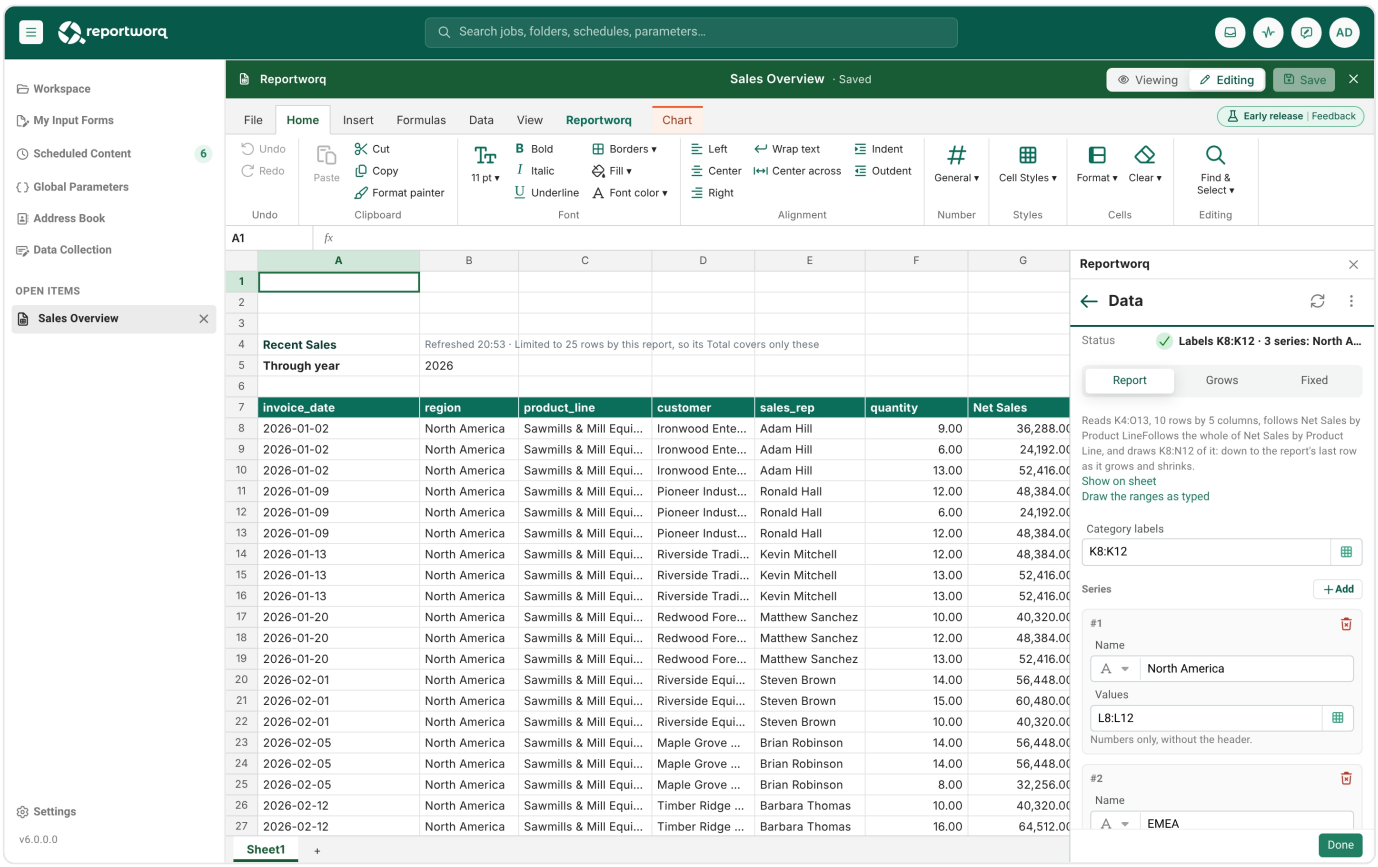


Set up the data

On the **Data** page:

- **Category** is the range that holds the labels.
- Each **series** has a name (typed, or taken from a cell) and a values range.
- On a Combo, each series also has a type (Column, Line or Area) and a **2nd axis** tick.
- A chart title can be typed or taken from a cell. Use the small menu inside a field to switch between **Type a value** and **From a cell**.

On **Legend & labels**, set the legend (Auto, Off, Bottom, Right or Top), data labels (Auto, Off or Values, plus Percent on a pie) and gridlines (not for a pie).



Keep a chart in step with its report

A chart made from a selection remembers what it was made from. The **Data** page shows a **Reads** line and a source mode:

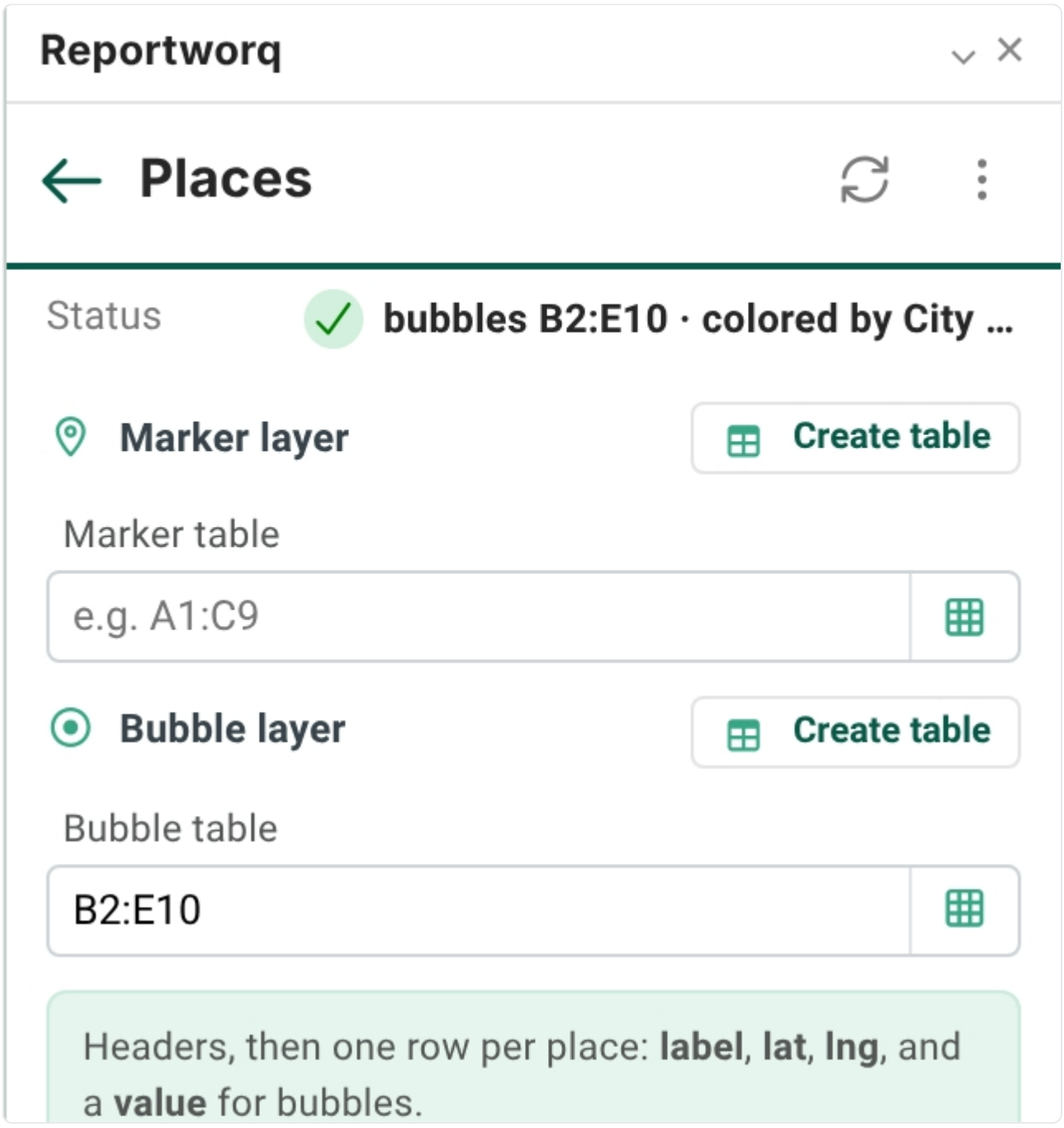
Mode	What it does
Report	Follows a list or crosstab. As the report grows, shrinks or gains filter lines, the chart draws every data row and never the Total row (unless you charted it)
Grows	Starts at the cells you chose and grows down and across while the cells are filled
Fixed	Draws exactly those cells

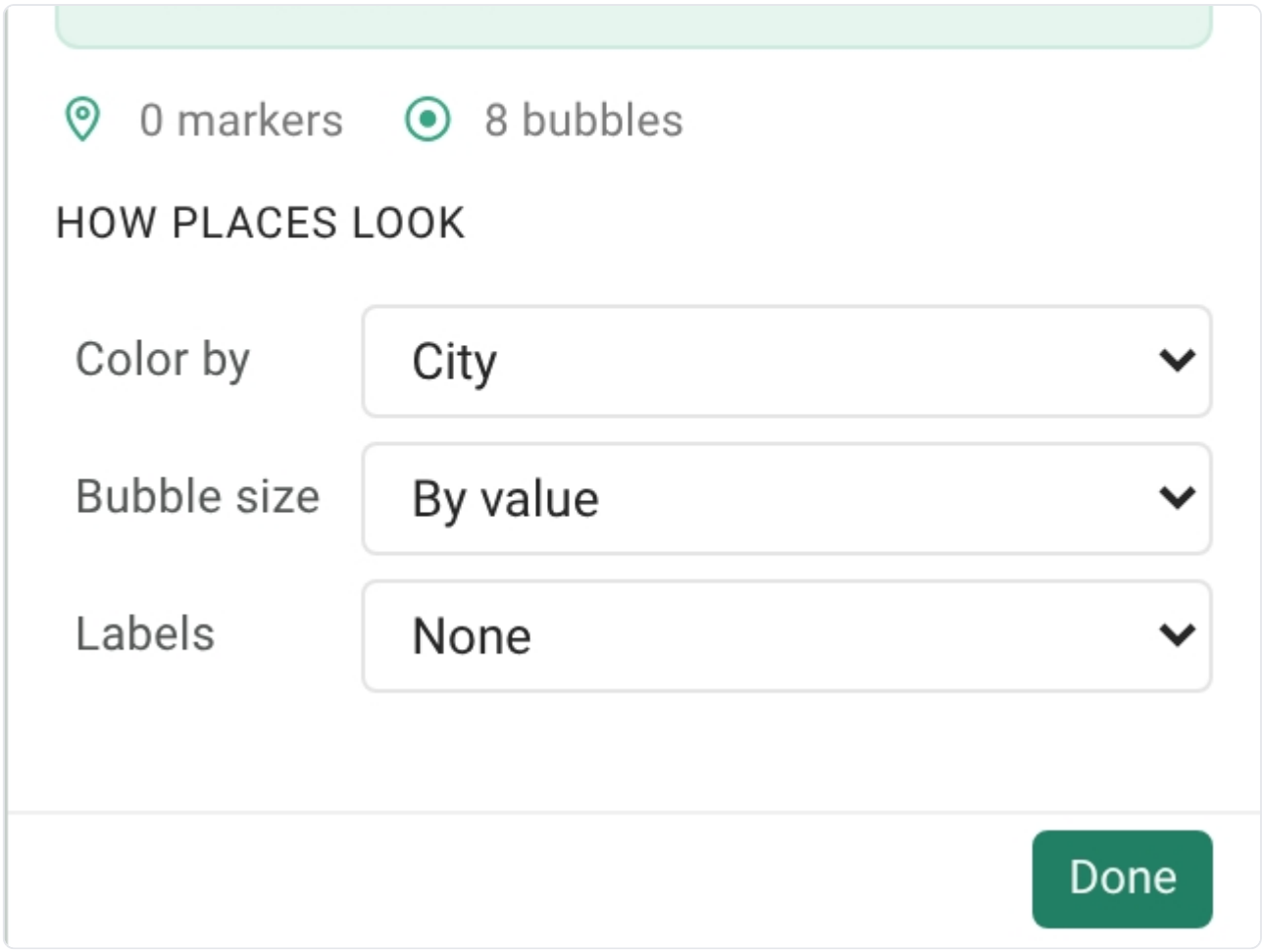
Show on sheet selects the cells the chart reads. If a chart says "Draws fixed cells", choose **Follow these cells** to make it follow its report. **Draw the ranges as typed** drops the source, which is the way out when the report it followed was deleted.

A chart over a crosstab draws each series from the column that holds its name. If a column is filtered away, that series is left out. If the report has no rows for a part of a burst, the chart draws empty under its own title.

Add a map

1. On the sheet, make a table with a header row. For markers use `label`, then latitude and longitude. For bubbles add a `value` column. **Create table** writes a ready-made template at the active cell.
2. Choose **Map** on the **New** group, then **Points** or **Bubbles**.
3. On **Places**, pick the **Marker table** and/or the **Bubble table**. Optionally set **Color by** a column, **Marker size** (or **Bubble size**) by a column, and **Labels** from a column.
4. On **Background & position**, choose a tile map or a picture, then frame it by place search or by latitude, longitude and zoom. Framing cells must be single cells on the map's own sheet.





Notes and limits for maps

- A tile map is a static image: it is not pannable inside Excel. Frame it in the pane.
- Tiles come from OpenStreetMap by default. The server fetches tiles only over https from a host that resolves to public addresses. A tile source that fails the check, or cannot be reached, leaves the map drawn without tiles; the render or job does not fail. A server with no outbound internet draws maps without tiles.
- A picture background can be PNG, JPEG, GIF, WebP or BMP. SVG is not accepted. **Load sample** offers five floor plans: Factory floor, Warehouse, Retail store, Office floor, Farm and mill.
- A place with a latitude outside -90 to 90 or a longitude outside -180 to 180 is left off the map. The Places page says how many were left off. A place with a blank latitude or longitude is kept but not drawn.

In a distribution job

A chart is stored with its definition, so when a job runs a workbook that holds one, the chart is redrawn from that run's data and delivered as a picture at the size it has on the sheet. A map goes out as saved: its places are not re-read, so a job that narrows a web report refuses to run if that would send every recipient all the places. See A web report as a job source.

Notes and limits

- Rows or columns you insert or delete inside the block, below its header, do not move a range that starts below them. In the Excel add-in your own edits there are followed name by name.
- A Sankey drops rows with a blank end, a self-loop, or a zero or negative value. A range chart drops half-pairs.
- A heatmap draws a missing cell blank and outlined, never as zero.
- Chart pictures from a job are rendered at three times the on-sheet size so they stay sharp in PDF and PowerPoint.
- **Fill in from** sends up to 12 rows of the selected block, all their columns, to your AI provider. A line under the button says so. With no enabled AI connection nothing is sent.

> **Going deeper.** To drive a chart's title or a map's center from a cell, see Cell-driven variables. To put a chart on the same sheet as other report types, see Combine report types.

Hierarchies

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **hierarchy** rolls the values of a field up into groups: sales reps into teams, markets into regions, products into product lines. You define it once on the data model, attach it to a field, and every crosstab or professional report that uses the field can show the groups with their subtotals. Because it lives in the model, it travels with the model when you copy, export or import it.

When to use it. Use a hierarchy when the source has the detail but not the grouping you report on, or when the grouping changes by report. At Timberline Harvest Co., the **Sales Details** view lists 36 products, each with a `product_line`. A **Levels** hierarchy of `product` up to `product_line` lets a crosstab show each product line with its products beneath, and the line totals stay correct as products are added. A grouping that exists only in your head, such as **Domestic** and **International** markets, is a **Manual** hierarchy.

Two kinds of hierarchy

	Manual	Levels
You provide	The members and how they roll up, typed as an indented outline	A view of the model and the fields it rolls up through, most detailed first
Members come from	What you typed	The view's data, read each time a report uses the hierarchy
Stays current as data changes	No, you edit the outline	Yes
Best for	A small, stable grouping that is not in the data	A grouping that is already columns in a view

Before you start

- You can edit the data model and it has at least one source.
- Hierarchies are part of web reporting. If the **Hierarchies** tab is missing from the **Source** screen, web reports are not available on your installation; ask your

administrator.

- For a **Levels** hierarchy, a view of the model that has a field for every level. Build it first with Build a view with the designer. Sort the view the way you want the hierarchy sorted (see below).

Create a Manual hierarchy

1. Open the data model, select **Source** in the rail, then the **Hierarchies** tab. It lists every hierarchy of the model with its **Type**, what it is **Built from**, its **Size** and the fields it is **Attached to**. An empty list reads **No hierarchies yet**.

2. Select **New hierarchy**. The hierarchy opens on its own screen.

3. Enter a **Name**. It must be unique within its source.

4. Leave **Type** on **Manual**. If the model has several sources, choose the source under **Built from**; it cannot be changed after you save.

5. Type the outline under **Members**: one member per line, indented two spaces or one tab for each level.

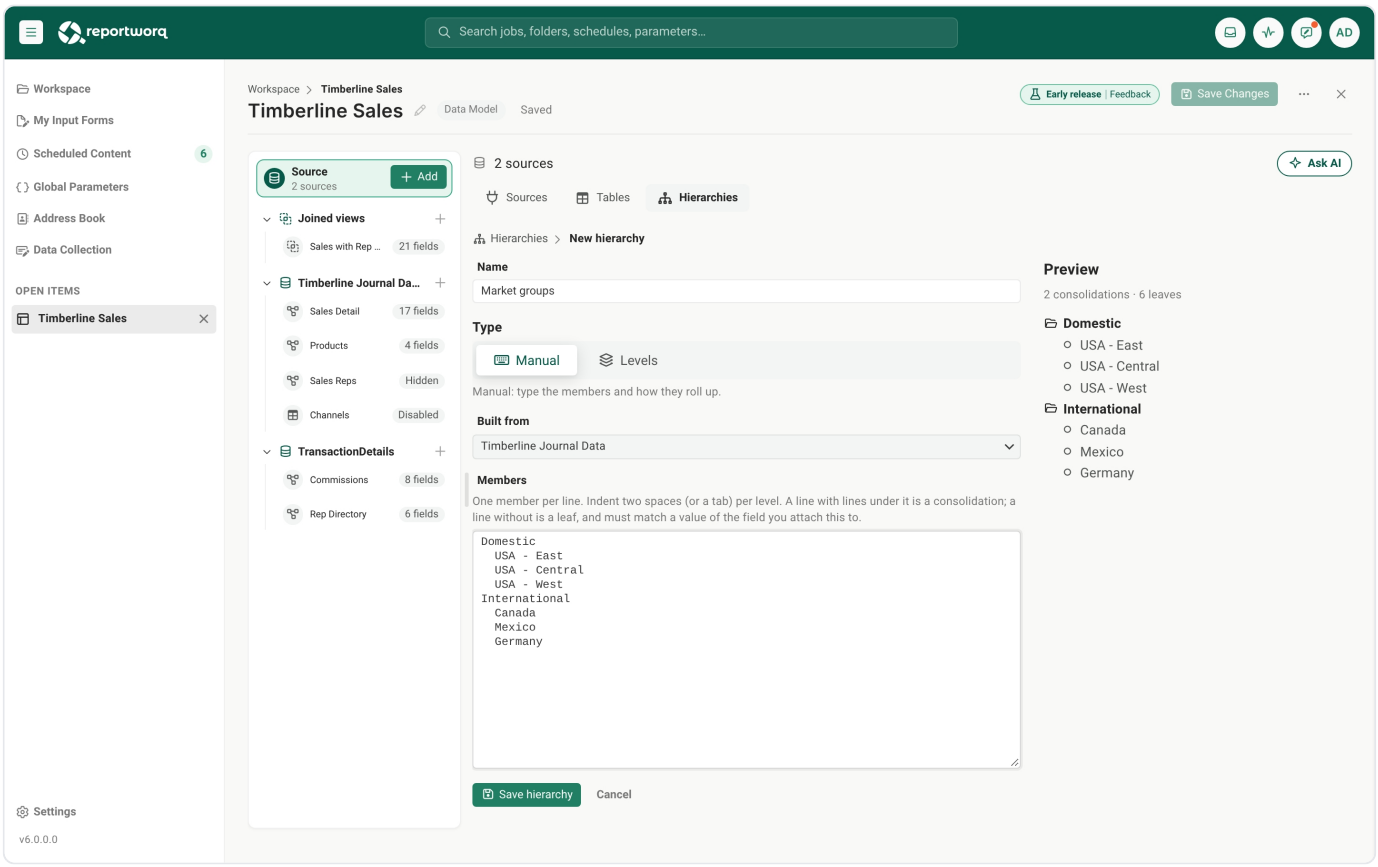
   ```
   Domestic
     USA - East
     USA - Central
     USA - West
   International
     Canada
     Mexico
     Germany
   ```

   A line with lines under it is a **group**; the editor's hint and preview call it a **consolidation**. A line with none is a **leaf**, and it must match a value of the field you attach the hierarchy to, spelled the same way.

6. Check the tree that appears beside the outline. If the outline cannot be used, the preview reads **Reports can't use this yet** and lists each problem with its line number. Typical causes: a top-level line with nothing under it, a value listed twice (`a value has one place`), an indent that is not two spaces or one tab per level, or an indent that jumps more than one level at a time.

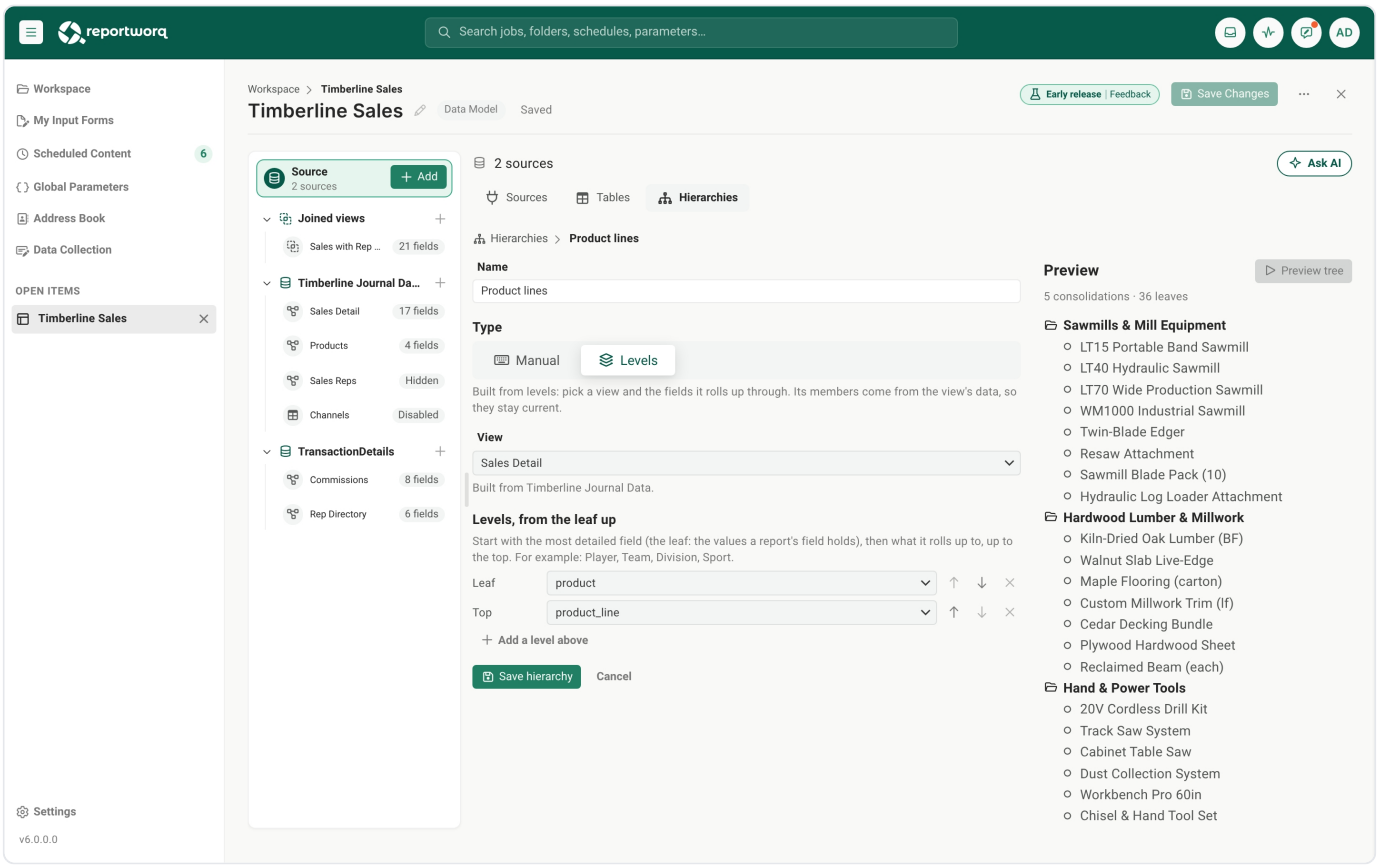
7. Select **Save hierarchy**. You return to the list with a note, `Updated “<name>” at hh:mm:ss. Save the data model to keep it.`

8. Select **Save Changes** to save the data model. Until then the hierarchy exists only in your editing session.



Create a Levels hierarchy

1. On the **Hierarchies** tab, select **New hierarchy** and enter a **Name**.
2. Set **Type** to **Levels**.
3. Under **View**, choose the view to read. The list offers every view of every source in the model, except joined views. When the model has several sources, each entry shows its source in brackets, for example `Sales Details (TransactionDetails)`.
4. Under **Levels, from the leaf up**, choose a field for each level. The first row is the **Leaf**, the most detailed level: the values your report's field holds. The last row is the **Top**. For `Product lines`, the leaf is `product` and the top is `product_line`. Select **Add a level above** for another level, and use the arrows and the remove button to arrange them. A hierarchy needs at least two levels, and a field can be used only once.
5. Select **Preview tree**. Reportworq reads the view as you have it in the editor (unsaved changes included, variables at their defaults) and shows the members, their counts, and anything the data says about them under **What the data says**.
6. Select **Save hierarchy**, then **Save Changes** to save the data model.



How a Levels hierarchy reads its view

- **Each value is one member**, however many rows hold it, matched without regard to case. Leading and trailing spaces are trimmed.
- **The order is the order the view returns its rows.** Sort the view to sort the hierarchy.
- **A member with two parents is reported, not merged.** It is shown under the parent of its first row only, so its totals are counted once. **What the data says** names every parent, for example `Team “Red” rolls up to more than one parent in the view: “East”, “West”. It is shown under “East” only, so its totals are counted once.` Fix the data or the view if that is not what you want.
- **Blank levels are skipped.** A member whose parent is blank sits under the next level up that has a value. A row with a blank leaf gives nothing, and a leaf with every level above blank is left out (a report shows its rows under **Unassigned**).
- **A value used at two levels** is kept at both and reported.
- **Variables follow the report.** If the view has variables, a report reads the hierarchy with its own values, so a report for one season shows that season's members. A report that sets no value for a required variable gets no members.
- **At most 1,000,000 rows are read.** If the view has more, or cannot be read when a report runs, the field shows its values without the hierarchy and the crosstab says so, for example `hierarchy “Product lines” (its view has more than 1,000,000 rows) unavailable, shown flat`. A hierarchy is never built from only some of the

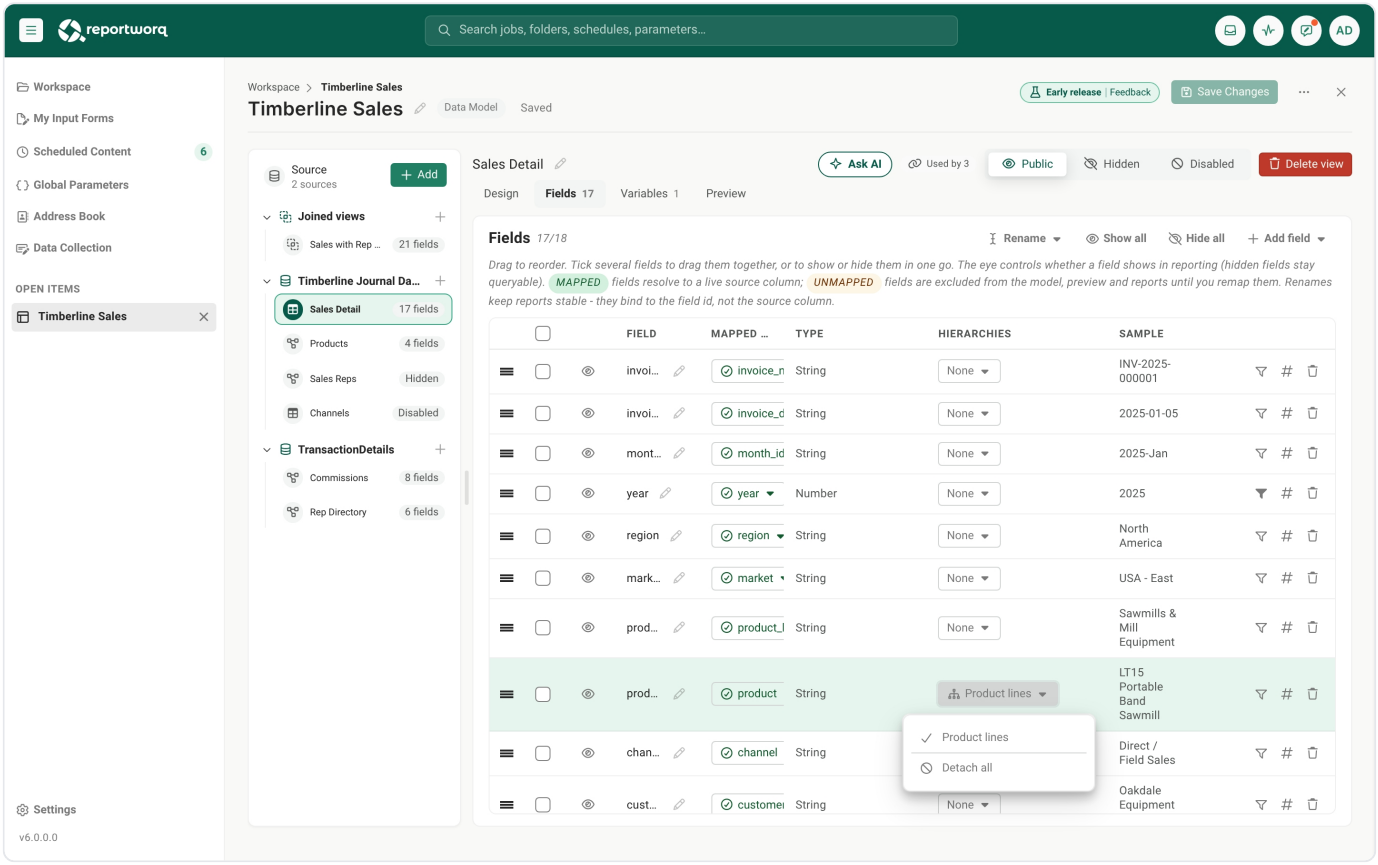
rows. Narrow the view, or give it a variable.

- **Preview tree lists at most 500 members.** It applies the same 1,000,000-row rule as a report.

Attach a hierarchy to a field

A hierarchy does nothing until it is attached to a field.

1. Open a view of the hierarchy's source and select its **Fields** tab.
2. In the **Hierarchies** column of the field, open the menu and tick the hierarchy. For a **Levels** hierarchy, attach it to its **leaf** field (`product` in the example), because the leaves are that field's values. **Detach all** removes every hierarchy from the field.
3. Select **Save Changes**.



Notes on attaching:

- **A hierarchy is attached to a field name on its source**, so every view of that source with a field of that name gets it.
- **A field can carry several hierarchies.** A crosstab lets the author choose one, and a professional report uses the first one attached.
- The **Hierarchies** column appears on views of a single source. A joined view does not have one, and a joined view cannot be a Levels hierarchy's view.

Where a hierarchy shows up

- **Crosstabs in web reports and Explore.** On a row or column field in the crosstab editor, open **Hierarchy**. Under **Show as**, choose **None (field values)** (the field on its own, the default) or one of its hierarchies. A field uses a hierarchy only where the author opts in. Groups are then shown with subtotals, and **Expand to level** controls how many levels are open. See Crosstabs.
- **Professional reports.** A member that names a group means its leaves together, and an expand lists its members in outline order. See Professional reports.
- **Distribution jobs that run a web report** use the same hierarchies, so a rolled-up crosstab looks the same in the output. See A web report as a job source.
- **A burst keeps recipients apart.** Each output lists only the members that have rows for that recipient, so one recipient never sees another's members.
- **A value the hierarchy does not name is not lost.** If a row has a value no leaf names, or a blank, it appears under **Unassigned**, and its numbers are still in every total above it.

Edit, move and delete

- **Edit** on a row of the **Hierarchies** tab reopens it. **Cancel**, or the **Hierarchies** link at the top, asks `Leave this hierarchy without saving your changes?` if you have edits.
- **Rename a field and the level follows.** If you rename a field of the view a Levels hierarchy reads, the level is renamed with it.
- **Move a Levels hierarchy to another source** by editing it and choosing a view of that source. Reportworq refuses, and says why, while the hierarchy is attached to fields (detach it on the **Fields** tab first), when the other source already has a hierarchy of that name, or when that source has a problem. A Manual hierarchy stays on the source it was created on.
- **Delete** asks `Delete the hierarchy “<name>”? Fields it is attached to lose it.` and removes the hierarchy with its attachments.
- **Delete a view a hierarchy reads** and the hierarchy keeps its definition but has no members until you edit it and pick another view. The delete confirmation says so.
- **Copy, export and import** carry hierarchies with the model. A Levels hierarchy follows its view to the copy's own view.

Ask AI can propose hierarchies

Ask AI on the **Source** screen can propose Levels hierarchies with its suggested views: a dedicated view of one table and its level columns, the levels leaf first, and the fields to attach it to. **Add** creates all of it unsaved in your editing session, exactly as if you had

made it here. The assistant never writes SQL for a hierarchy's view. Preview the tree before you save, because the assistant does not compare leaf values with your field's values. See Data models.

Notes and limits

- **Hierarchies roll up in web reports, Explore, professional reports and the jobs that run them.** The Excel add-in's own list reports do not use them, and crosstabs on a cube use the cube's own hierarchies.
- **A joined view cannot carry a hierarchy.** Attach it on a view of a single source, or build the Levels hierarchy from the input view.
- **Mixed versions.** An older version of Reportworq that saves a model drops a Levels hierarchy's definition. Upgrade every server that shares the model before you build one.
- **Manual leaves must match the field's values.** A leaf the data does not hold is harmless, and a value the outline does not name falls under **Unassigned**.
- **Hierarchies are saved with the model.** Nothing is stored until you select **Save Changes** on the model.

> **Going deeper.** To join sources first and group afterward, see Join views across sources. For the full tour of the Source screen, see Data models.

Professional reports

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

> **Not released yet.** Professional reports are not currently released. This page is in preparation for the next release. It describes what to expect, and the feature is not available on a standard license today.

A professional report is a sheet you lay out freely: your own labels, blank lines, Excel formulas between report rows, and any formatting. Reportworq owns only the report's own formulas, one per row or column header, and fills in every number where a row meets a column.

When to use it. A monthly Timberline P&L where the account rows are fixed, a "Top 5 product lines under Revenue" row grows and shrinks on every refresh, your own margin formulas sit between rows, and a quarter filter is a drop-down on the sheet. A crosstab cannot be formatted line by line like that.

License

Professional reports are a licensed feature. They are not part of a standard license yet; today they come with internal and QA licenses only. Web reports themselves need no license, and neither do lists, crosstabs, charts, maps or AI insights. See Licensing.

A server without the license behaves like this:

- **Controls stay, disabled. Professional report** in the New group, **Convert to professional**, **Add dynamic row**, **Add row member** and **Add column member** are disabled with the reason `Professional reports are coming soon.` A command that reaches the pane anyway shows the same sentence and creates nothing.
- **A web report that already holds a professional report** opens with that report's value cells blank, only in that session, and it cannot be saved, saved as, exported or downloaded. The stored report is not changed. Once the server is licensed it opens normally.

- **A distribution job over it stops and distributes nothing.** The job fails with this message, where the name is the web report the job uses: `Nothing was distributed: the web report '<name>' uses a professional report, which this license does not include.` The message is quoted exactly as the product words it. If a workbook holds the report's formulas, the same sentence names that source.
- **AI insights** are not asked over a professional report's cells.

The license does not show in the **Capabilities** block of the License tab, so the messages above are the signal. To get it, ask your Reportworq contact.

Screenshot coming soon
A New menu with Professional report disabled and the reason Professional reports are coming soon

Where you can build one

	Web report studio	Excel add-in
Start blank or convert a crosstab	Yes (licensed)	No. The controls are disabled with `Professional reports aren't in Excel yet.` and a toast adds "Build them in the web report studio."
Open a workbook that already holds one	Yes	Listed and handled, but its formulas do not calculate in Excel itself

Build professional reports in the web report studio. Then convert the web report to Excel or use it as a job source; a professional report goes out of a job as saved values.

Before you start

- A licensed server (above).
- A data model view with at least one measure. See Data models.
- A web report open in the studio, with an empty cell where the report should start.

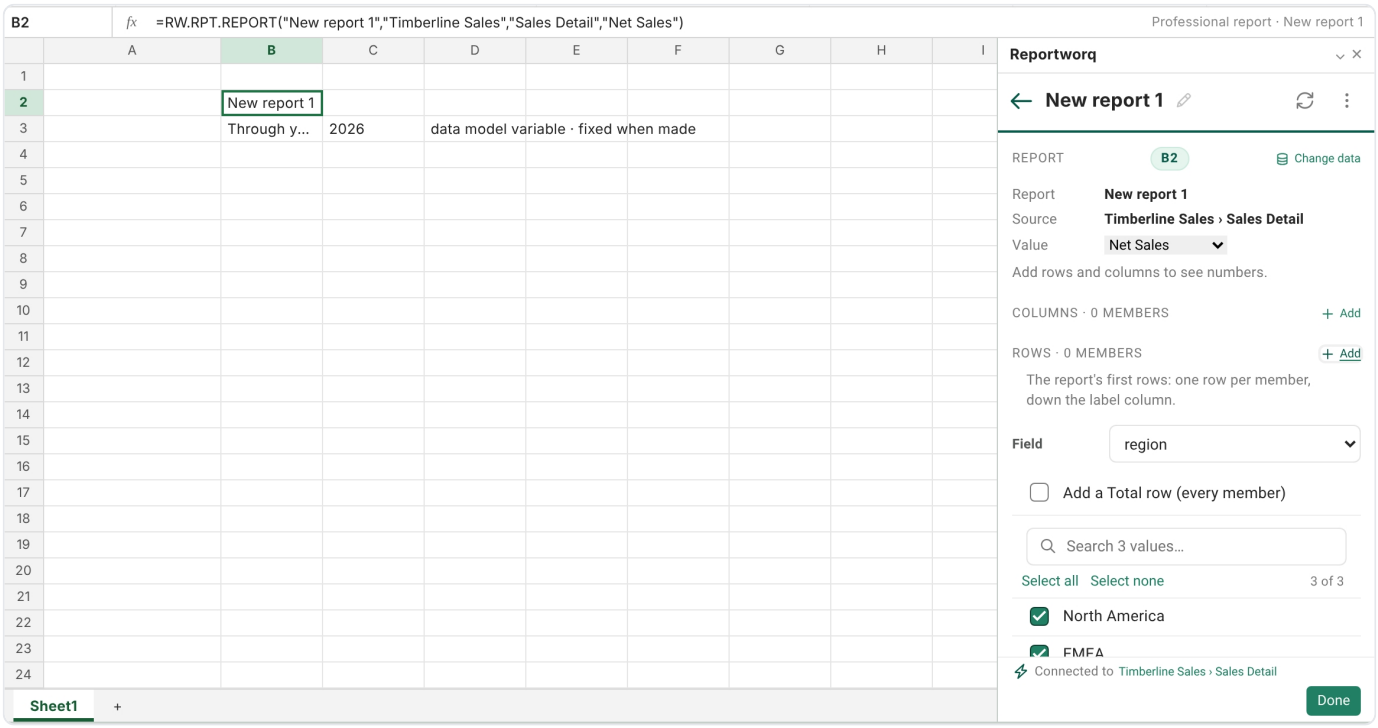
Start a professional report from blank

1. Click an empty cell where the report should begin.
2. In the studio ribbon choose **New**, then **Professional report**. (The Home start button **Professional** does the same on an empty sheet.)
3. On the report's **Connection** page, pick the data model and the view. Nothing is written to the sheet until you pick. The pick writes the report's anchor cell with its

first measure.

4. Under **Rows**, choose **Add**. In the form, pick a **Field**, tick its **Members**, and optionally tick **Total**. **OK** writes one row header per member down the label column (up to 200 per Add).
5. Under **Columns**, choose **Add** and do the same across the header row. The first columns replace the temporary value column that shows each row's total.
6. Choose the **Value** in the Report section if you want a different measure.

The pane shows the numbers as soon as the report has rows and columns. Before then it says "Add rows and columns to see numbers."



Refused when you start: a cell that is not empty ("Select an empty cell"), a view without a measure, a view with a required variable that has no default, a cube view (start from a crosstab and convert instead), and a place that would land on another report.

Convert a crosstab

1. Build and check the crosstab first. See Crosstabs.
2. Select the crosstab and choose **To professional report** (the editor menu or the Crosstab ribbon tab).
3. Reportworq rewrites the crosstab in place as formulas, keeps its formats, then compares every value and total with the crosstab. If anything differs, the convert is undone and refused with the count.

A crosstab converts only with a model, a row field, one column field, one column stack and one value. Several row stacks convert to one professional report per stack, layered

down the sheet. Filters become report filters. Data model variables are carried as fixed values: those cells show the value the report reads but take no typing. To read with another value, change it on the crosstab and convert again.

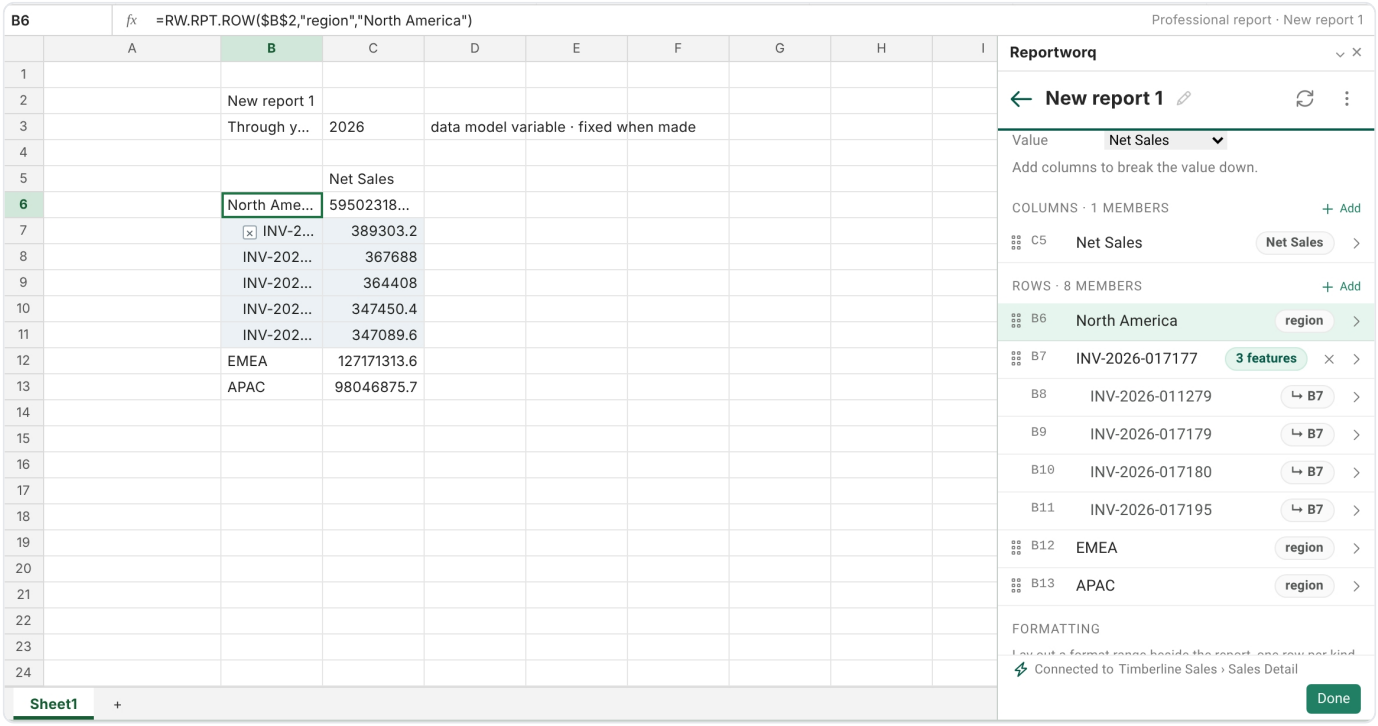
A chart, map or AI insight that followed the crosstab follows the new report. After a stacked convert the toast tells you which chart still reads the old crosstab so you can pick its data again.

Convert refuses, naming the field, for a Top N on a row field or an inserted row, an "Everything else" row under an inserted row with left-out members, and any row or column member filter on a cube.

Make a row dynamic

Select a row member and use **Add dynamic row**, then **Top N**. A temporary block grows or shrinks on every refresh and ranks by the report's value. In the **Add** menu a member can also take a label filter (equals, does not equal, contains, begins with, ends with) or **Expand**, which lists the member values in scope. On a cube, Expand on a consolidation walks the hierarchy.

Drag members and their generated blocks to reorder; references follow, as when you insert cut cells in Excel.



Notes and limits

- Only Top and Bottom N, label filters and Expand are offered in the pane. Value filters, an "everything else" row on a Top N, subtotal planning and sort are not

available yet; formulas that name them are kept but have no effect.
- A dynamic column is not available in the pane. Top, Filter and Expand work on row members.
- One value per report. Add writes siblings only, not child rows. Report filters cannot be added from the pane yet; a converted crosstab carries its filters.
- Values are rounded to two decimals and show unformatted until you format them with Excel's own tools.
- A cell claimed by two reports shows `#COLLISION`. If a growing block would run through a neighbor, the refresh inserts cells across this report's columns instead, or is refused.
- A member wrapped in your own IFERROR or IF still refreshes, but the pane shows it read-only.
- A refresh is held to the per-report cell cap and, in Excel, to the workbook total.

Field filtering

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **field filter** is one or more conditions on a field of a view. Every row the view returns, in **Preview**, in the Excel add-in, in a web report and in a distribution job, has to pass them. You set the condition once on the model and every report that uses the view gets the narrowed rows without repeating it.

When to use it. Use a field filter to keep a view to the rows people should always report on. At Timberline Harvest Co., the **Sales Details** view holds every invoice line since 2022. A filter on `year` of `>=` `2025` keeps a view to the current years, and a filter on `paid_status` of `=` `Open` makes a view of unpaid invoices for collections reports. To let each report choose the value, put a `{{variable}}` in the filter instead of a fixed value.

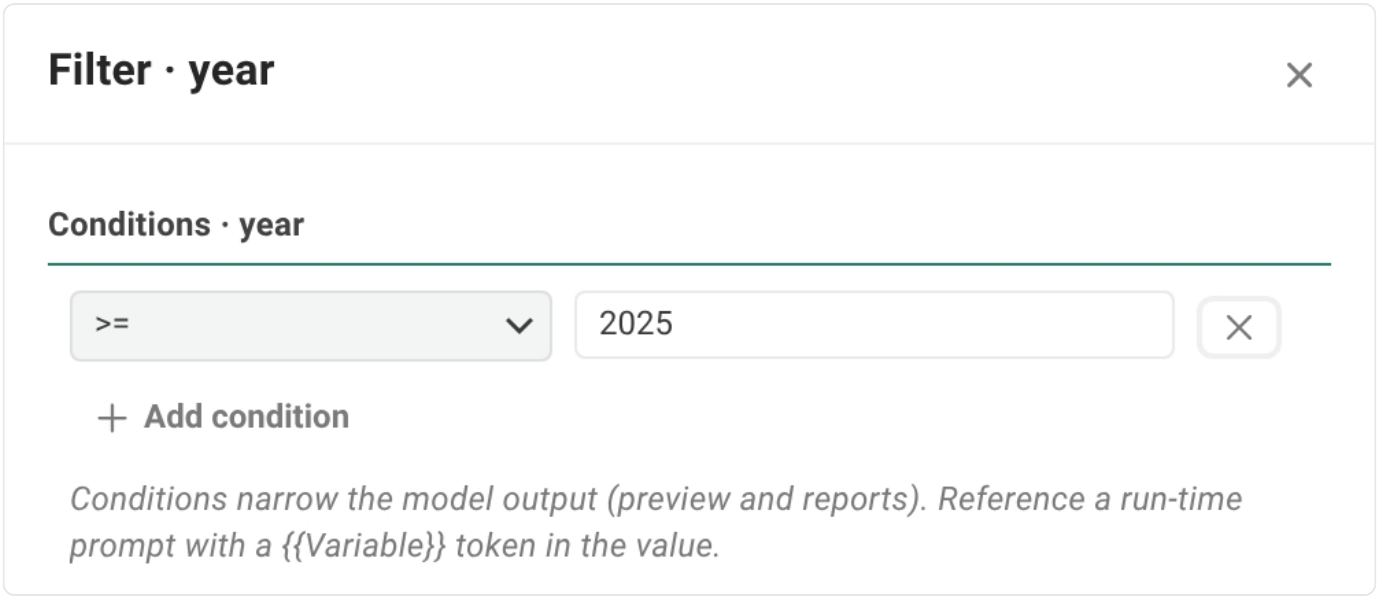
Before you start

- You can edit the data model, and the view has a **Fields** tab with the field you want to filter. A field must be **mapped** (it shows the green **mapped** chip) to have a filter. An unmapped field is left out of the model, so it has no filter button.
- To use a variable in the filter, read Data model variables first. It covers defaults, required values and how a value is written into the query.

Filter a field

1. Open the data model, select the view in the rail, and select its **Fields** tab.
2. On the field's row, select the filter button. Its tooltip reads **Filter this field**. The **Filter** panel opens at the side, titled with the field's name, with a **Conditions** section.
3. Select **Add condition**.
4. Choose the **operator** from the list on the left, and type the **value** on the right. The box shows the hint `Value or {{Variable}}`. Operators such as **Is Null** show **no value needed**.
5. To add more conditions on the same field, select **Add condition** again. Once there are two or more, a **Match** control appears with **all** and **any**: **all** keeps a row only if it passes every condition, **any** keeps it if it passes at least one.

6. To remove a condition, select the **x** beside it (**Remove condition**).
7. Close the panel and select **Save Changes**. Check the result on the view's **Preview** tab.



A field that has conditions shows a filled filter icon, and its tooltip reads `1 filter condition(s). Edit`. A condition that needs a value but has none is dropped when you close the panel, so a half-finished condition never narrows anything.

Value rules

- **Type the value as it is.** Write `Open`, `USA - East` or `2025`, with no quotes. Reportworq writes the value into the source's query for you, using the field's type. A value typed as `'USA - East'` is searched for with its quotes and matches nothing.
- **Use a variable for a value chosen at run time.** Write `{{Region}}` in the value box, with double braces and no quotes. The variable then appears on the view's **Variables** tab, where you set its default and whether it is required. A condition on an optional variable left blank is dropped, so the report sees all rows. See Data model variables.
- **Is Contained In takes a list.** Write `EMEA, APAC`. Items are separated by commas, and an item that has a comma in it goes in quotes.
- **Dates and numbers** are checked against the field's type. A value that is not a number for a number field is refused with its own message, not silently ignored.

Operators

Operator	Keeps a row when the field...	Needs a value
`=`	equals the value	Yes
`<>`	differs from the value	Yes
`<`, `<=`, `>`, `>=`	is less than, at most, greater than, or at least the value	Yes
Starts With, **Ends With**	starts or ends with the text	Yes
Contains	contains the text	Yes
Does Not Contain	does not contain the text	Yes
Is Contained In	is one of a list of values	Yes, a list
Is Null, **Is Not Null**	has no value, or has a value	No
Is Empty, **Is Not Empty**	is an empty text, or is not	No
Is Null Or Empty, **Is Not Null Or Empty**	is missing or empty, or is not	No

The list shows every operator for every type of field. On a number, date or true/false field a value is never empty, so **Is Empty** keeps nothing and **Is Not Null Or Empty** is the same as **Is Not Null**.

How filters combine

- **Within one field**, the **Match** control decides: **all** or **any**.
- **Across fields**, a row must pass the filters of every field that has one. Filtering `year` and `paid_status` keeps rows that satisfy both.
- **A hidden field can still carry a filter.** Hiding a field removes it from the report author's field list but keeps it usable in a filter, so you can filter on a key column without exposing it.

Where filters run

A filter runs **in the source's own query** wherever the source can express it, so the database returns only the matching rows and less data crosses the network. There is nothing to switch on.

- **SQL sources** (SQL Server, Oracle, PostgreSQL, SQLite, and Excel and SQLite workspace files, among others) filter at the source. The result is the same as

filtering in memory, with the differences listed below.

- **Other sources**, such as Planning Analytics and the planning-tool APIs, cannot express the condition in their own query language, so Reportworq retrieves the rows and filters them in memory. You get the same result, with more data moved.
- **If a filtered read fails at the source,** Reportworq reads the view whole (up to the limit an administrator sets for joined inputs), filters it in memory, and adds the warning `Filtered after reading every row`. The rows are right, but the run reads more than it needs to. If the view is larger than that limit, the run fails and tells you to change the view's SQL or narrow it with a variable.

When the same filter returns different rows

Because the database does the filtering, a few comparisons follow the database's rules and not Reportworq's:

Difference	What to expect
Case and text order	Whether `maine` matches `Maine`, and how text sorts, follow the database's collation. If a filter returns different rows than you expect, check how the source compares text.
Trailing spaces	SQL Server treats text with trailing spaces as equal.
`%` and `_` in Contains or Starts With	They act as wildcards at a SQL source. In memory they are plain characters.
Is Contained In	At a SQL source it matches whole values from the list. In memory it matches pieces of the text. The source is the one to trust.
Access over ODBC	Text empty conditions fall back to the capped in-memory read.

Notes and limits

- **Preview always runs the view fresh**, so it shows a filter you just set.
- **Joined views** can carry their own filters on their fields, and an optional variable in one is handled the same way. See Join views across sources.
- **Calculated fields do not exist.** A filter applies to a mapped source column. See Data models.
- **Filtering is not security.** A filter narrows what a view returns, but anyone who can run the view gets the rows that pass it. To keep data private, use folder permissions on the model. See View visibility.

Going deeper. For defaults, prompts and how a variable's value is written into the query, see Data model variables. For the Fields tab in full (renaming, hiding and reordering), see Data models.

AI insights

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

An AI insight is a report type that explains other cells in words. You point it at a list, a crosstab, a chart or a range, ask a question, and the answer appears in a text box on the sheet or in one cell. It keeps its answer with the report, so readers see the text without anyone asking again.

When to use it. Under a Financial Reports crosstab of revenue by region, a one-paragraph "What stands out this quarter?" for the people who only read the delivered PDF.

> This is not the same as the **AI Generated Insights** section of a job's Distribution step, or the `RWAIINSIGHT` worksheet function. Those are described in AI-generated insights in reports. An AI insight report type lives on the sheet and is part of the web report or workbook.

Where you can build one

	Web report studio	Excel add-in
AI insight report type	Yes	Yes, in the analyst pane
Companion pane (analyst pane turned off)	Not applicable	No. Its only AI tool is **Create with AI** in the chart builder
Build with AI and Edit with AI (the report chat)	Yes	Yes, in the analyst pane

Before you start

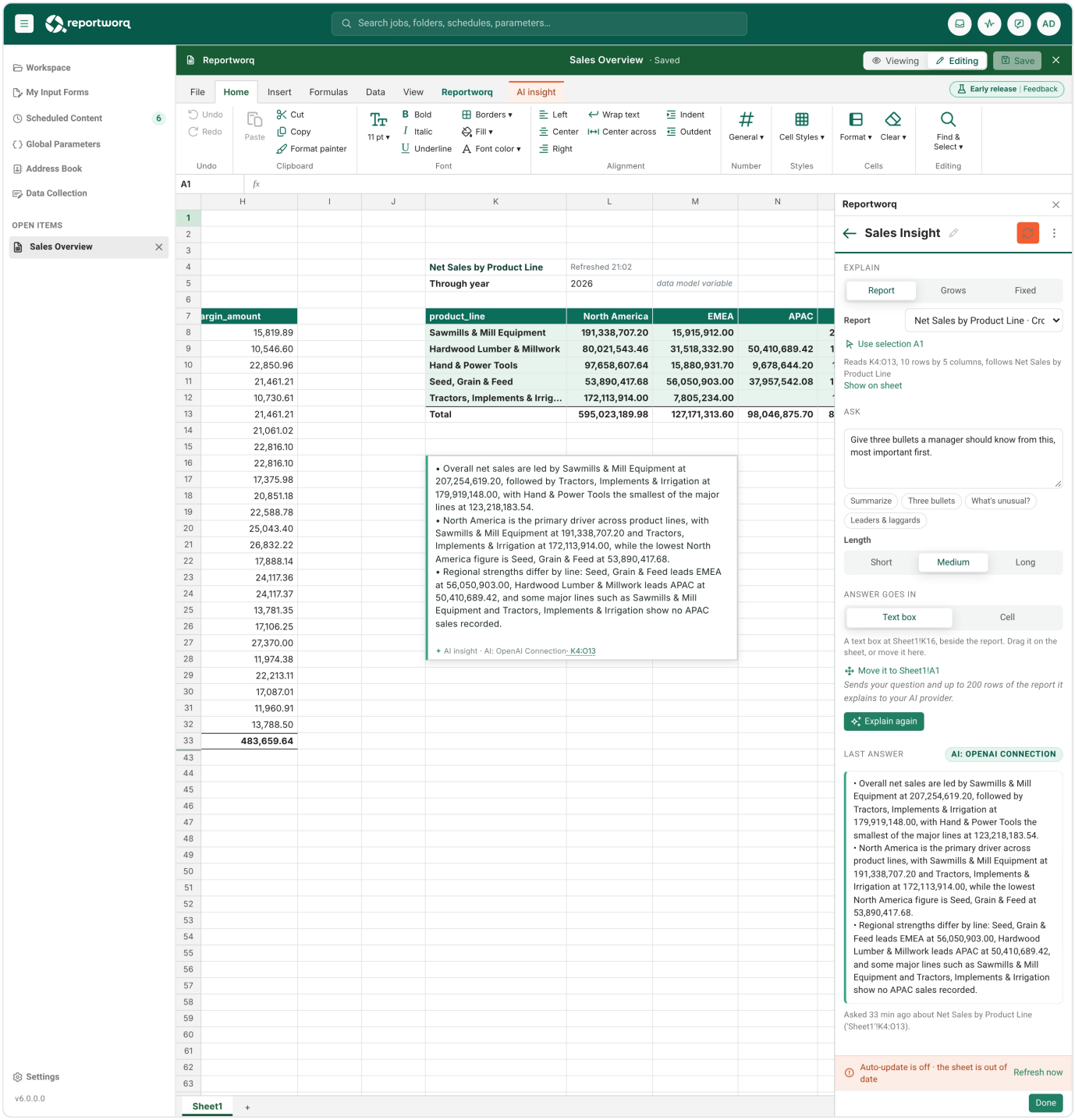
- An administrator has added and enabled an AI connection. See Connect an AI provider. Any enabled provider connection works; if the insight names none, the first enabled one is used.

- Without an enabled connection the insight still works in the pane, but a labeled offline stand-in answers from the numbers (what leads, what is smallest, how many are above average) and marks itself "AI offline: demo stand-in". A distribution job never uses the stand-in.
- A report, chart or range to explain.

Add an AI insight

1. Click the cell where the insight should start. A text box can sit anywhere; do not click inside another report.
2. Choose **New**, then **AI insight** (in Excel, also the **AI insight** button on the List and Crosstab ribbon tabs, or a report card's menu on **Home**).
3. Under **What to explain**, pick a source mode, or choose **Use selection** with the cells selected:
    - **Report** follows a list, crosstab, chart or map as it renders now, however a filter or variable changes it.
    - **Grows** starts at a cell and reads down to the first blank row and across to the first blank column, stopping at any report.
    - **Fixed** reads exactly the cells you typed.
4. Type the question, for example "Which regions are below plan and by how much?"
5. Choose where the answer goes (a text box or one cell), the length (short, medium or long) and, if you have more than one, the AI connection by name.
6. Choose **Explain**.

The pane's **Reads** line says what the insight reads, for example "Reads B5:F12, 8 rows by 5 columns, follows Sales by Region". **Show on sheet** selects those cells.



What is sent to the AI provider

The pane shows a line above **Explain**: "Sends your question and up to 200 rows of the report it explains to your AI provider."

- The cells a reader sees now, as displayed text (`15.3%`, `52.05M`, never the raw number). Rows and columns hidden by hand, by an AutoFilter or by Reportworq, and a report's format range, are left out.
- The report's title and filter lines, its header row, and a marker on subtotal and total rows.

- At most 200 rows or 24,000 characters. If the data is cut, the cut is stated in the request and the total rows are still sent.
- A chart or map sends its data range. A range over two reports is sent in parts, with a rule never to combine figures across parts.
- Cell text that looks like instructions to the model is neutralized so a cell cannot close the data block or open one of its own.
- Nothing is sent without a click on **Explain**, **Refresh** or a refresh-all command, or a job run that reaches the insight.

Treat the provider as having seen the cells shown. Do not point an insight at cells a recipient should not see.

Check the answer

An answer is checked before it is kept. Figures in the answer that cannot be found in the data are listed, and the answer shows "Unverified figures: ... Check them before you rely on this answer." in a callout in the pane, a chip on the report card, and the foot of the text box. A cell answer carries the same line as a note on the cell. A correct figure the model worked out itself is marked too: the mark means nobody checked, not that it is wrong. A refused answer, for example one holding a link, contact or path, is an error and the insight keeps its previous answer.

A model can still state something false in words, or steer tone, if a cell tells it to. Review insights before you send them to customers.

When the cells an insight read change after its answer, it is marked out of date until it is asked again.

In a distribution job

When a job uses a web report as its source, each insight is asked again while the report is converted, using the cells as they are in that run. The answer goes out as a text box or cell in the delivered workbook or PDF.

- If the ask fails (no AI connection, a disabled connection, a timeout, a refused answer), the saved answer is withdrawn rather than sent as this run's. The text box or cell says `AI insight not refreshed in this run.` In a narrowed or burst run it says `AI insight unavailable for this filtered report.` The run completes with a warning that names the insight and the reason; it does not fail.
- On a **Test** run, tick **Skip AI Insights** to spend no AI tokens. A placeholder `[AI Insight skipped for this test run]` appears, and Run History will not let you Send that run to recipients.

- A workbook converted with **Convert to Excel** keeps its insights as saved text. A whole-workbook run that refreshes the reports withdraws their answers with "AI insight not refreshed in this run." because nothing asks them again on that route.
- Rows hidden by a report-suppress step during the run are not hidden yet when the insight reads, so they can be sent.

See A web report as a job source and Convert to Excel.

Notes and limits

- The text box is 420 pixels wide by default; its height follows the answer, from 100 to 640 pixels. A cell answer is one or two sentences.
- A source that cannot be read (its report or cells were deleted) is never asked and says why.
- In Excel, a Grows source over plain cells becomes a Fixed range, because the pane cannot see where your own table ends. Excel's own insert and delete do not move an insight's source until the next sync.
- On the studio canvas a source reads at most 1,000 rows by 52 columns.

Direct queries

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A data model normally hands report authors **views**: curated sets of fields you have named, filtered and made **Public**. **Direct queries** is a model-level option that adds a second path. When it is on, a report author in the Excel add-in can write or build a query against one of the model's sources and use the result's own columns, with no view in between.

When to use it. Use it for a one-off, well-understood or exploratory pull where building a view is more work than the answer is worth. At Timberline Harvest Co., an analyst who already knows the exact SQL for an extract from the **TransactionDetails** workbook can run it as a direct query, or look at a source before deciding whether it deserves a curated view. For reporting that many people share and that must survive changes to the source, build a view instead.

A view or a direct query

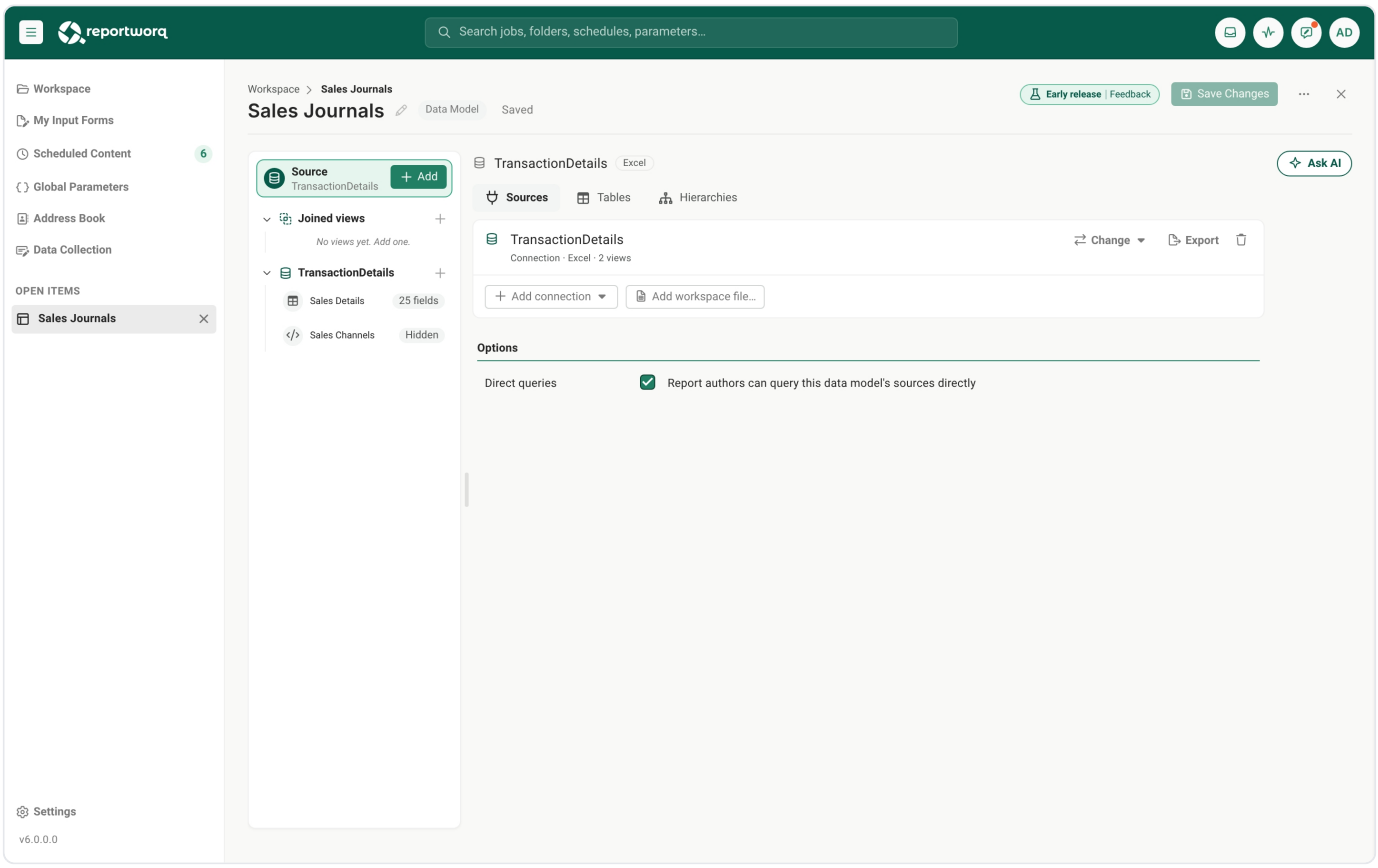
	Public view	Direct query
Who builds it	The model author, once	The report author, in the report
Fields	Curated names, types and mappings that survive a change to the source	The query's own columns, as the source returns them
Where it is stored	In the data model, reusable by many reports, jobs and forms	Inside the one report's workbook
Visibility (Public, Hidden, Disabled)	Applies	Does not apply, because no view is involved
Shared and governed	Yes	No
If the source's schema changes	The model author re-maps the field and reports keep working	The report's query may break

Before you start

- You can edit the data model.
- The model has at least one source. A connection or a workspace file both work. See Bring several sources into one model.

Turn direct queries on

1. Open the data model and select **Source** in the rail, then the **Sources** tab.
2. Under **Options**, tick **Direct queries**. Its label reads **Report authors can query this data model's sources directly**.
3. Select **Save Changes**.



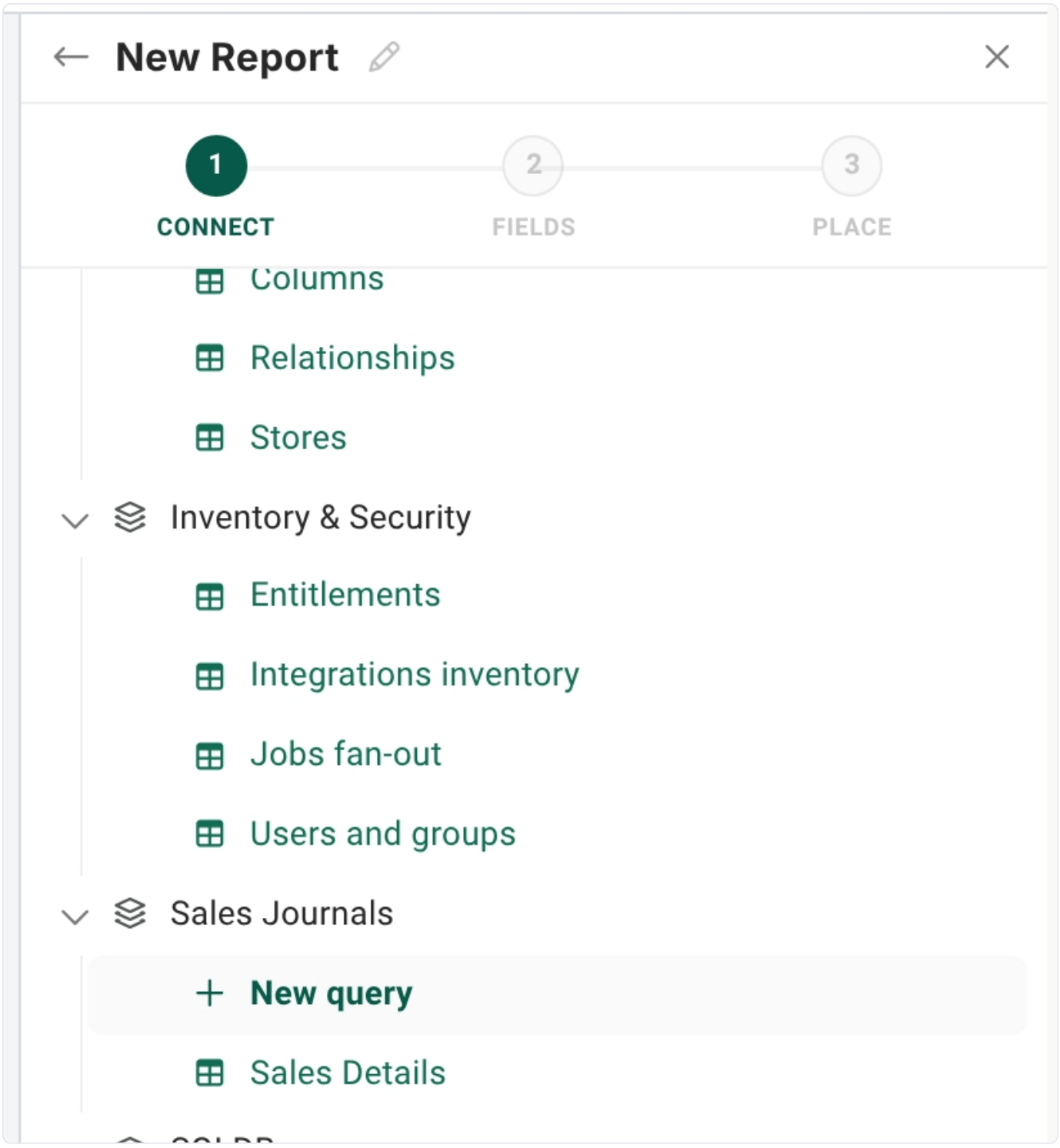
The option is **off** by default, so a model is view-only until someone turns it on. It is one checkbox for the **whole model**: it applies to every source the model reads.

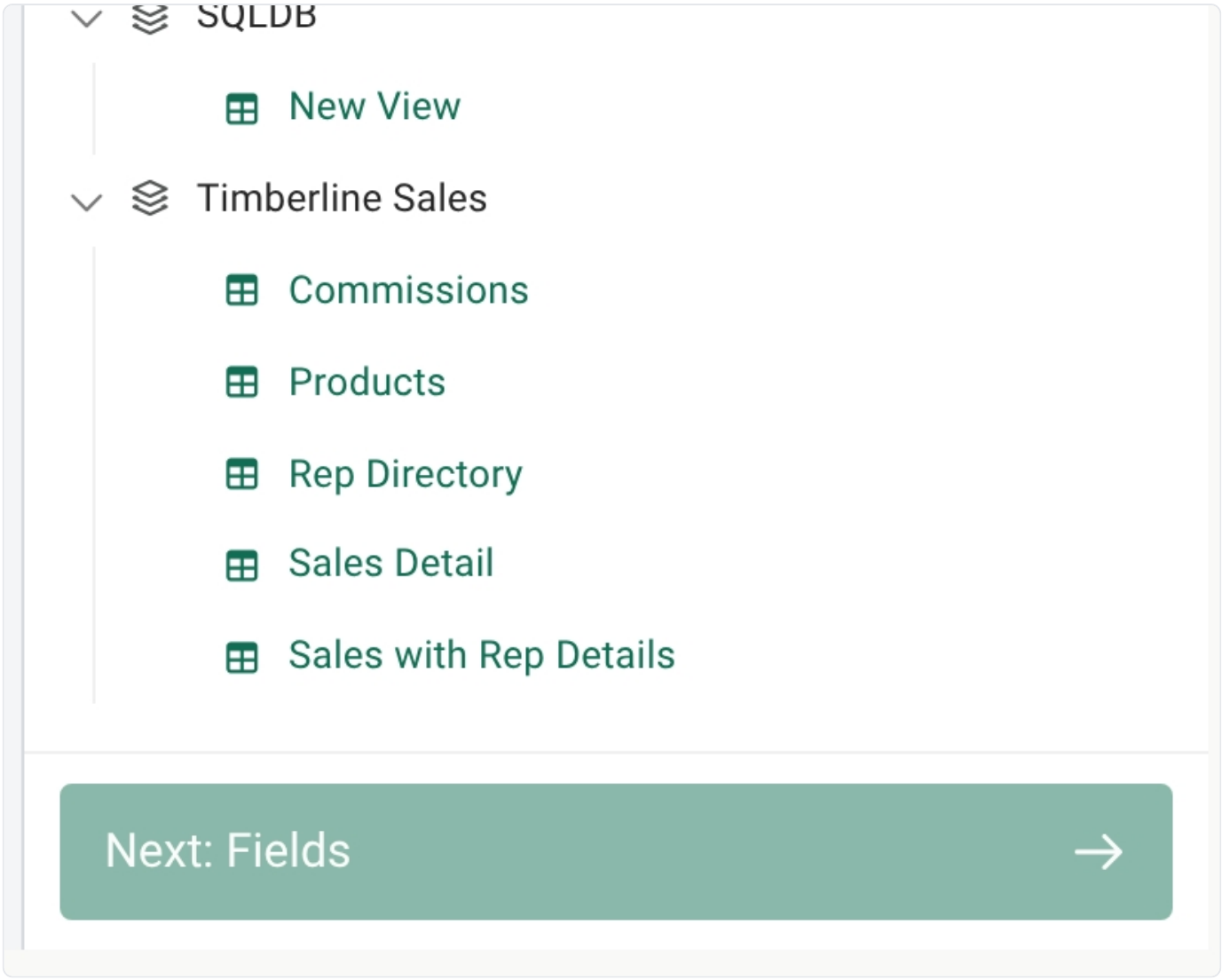
What the report author sees

In the Excel add-in's report builder, on the **Connect** step, the model lists its **Public** views as before. With direct queries on:

1. The model is listed even if it has no Public view yet.

2. Under the model there is a **New query** entry. If the model reads several sources, there is one for each, named for the source: **New query · TransactionDetails**, **New query · Timberline Journal Data**.
3. Choosing it opens **Build a query**, with the same query editor the model author gets for that source: the table designer or SQL for a database or workspace file, the connector's own query settings for other sources. If the query has `{{variables}}`, the author supplies a sample value for each so the query can run.
4. **Build query** runs it and moves on to the **Fields** step, where the author picks and arranges the columns like any other report.
5. When the report is reopened later, the query opens again for editing.





The query is saved **inside the report's workbook**, not on the server, and it is not a view of the model. See The Excel add-in analyst pane for the rest of the report builder.

What is checked when the report runs

The report carries its own query, so Reportworq checks it every time it runs, on the server:

- The query's source must be a source of a data model that the person running it can see **and** that has **Direct queries** on. If not, the run fails with `This report's direct query reads a source that no data model you can see allows direct queries on. Turn on Direct queries on the data model, or bind the report to one of its views.`
- A direct query cannot be a joined view. A report that tries is refused with `A direct query can't be a joined view. Bind the report to a saved view instead.`

Turning the option off therefore stops the direct-query reports that depend on it, with the first message above, until you turn it back on or the authors rebind those reports to views. Views and the reports built on them are not affected.

Notes and limits

- **It opens the source, not just the tables you modeled.** A direct query can read anything the connection can read, so choose which models turn it on with that in mind. The folder permissions on the data model decide who can see the model and so who can use it. See View visibility for the other ways a model limits what people see.
- **No curated fields.** Renames, types, hidden fields and the model's variables belong to views, so a direct query has none of them. A `{{variable}}` in a direct query belongs to the report, which supplies its values when it is placed on the sheet.
- **It cannot join across sources.** A direct query reads one source. To combine sources, build a joined view. See Join views across sources.
- **Direct queries are offered in the Excel add-in's report builder.**

> **Going deeper.** To build the governed alternative, see Build a view with the designer and View visibility.

Combine report types on one sheet

explanation author

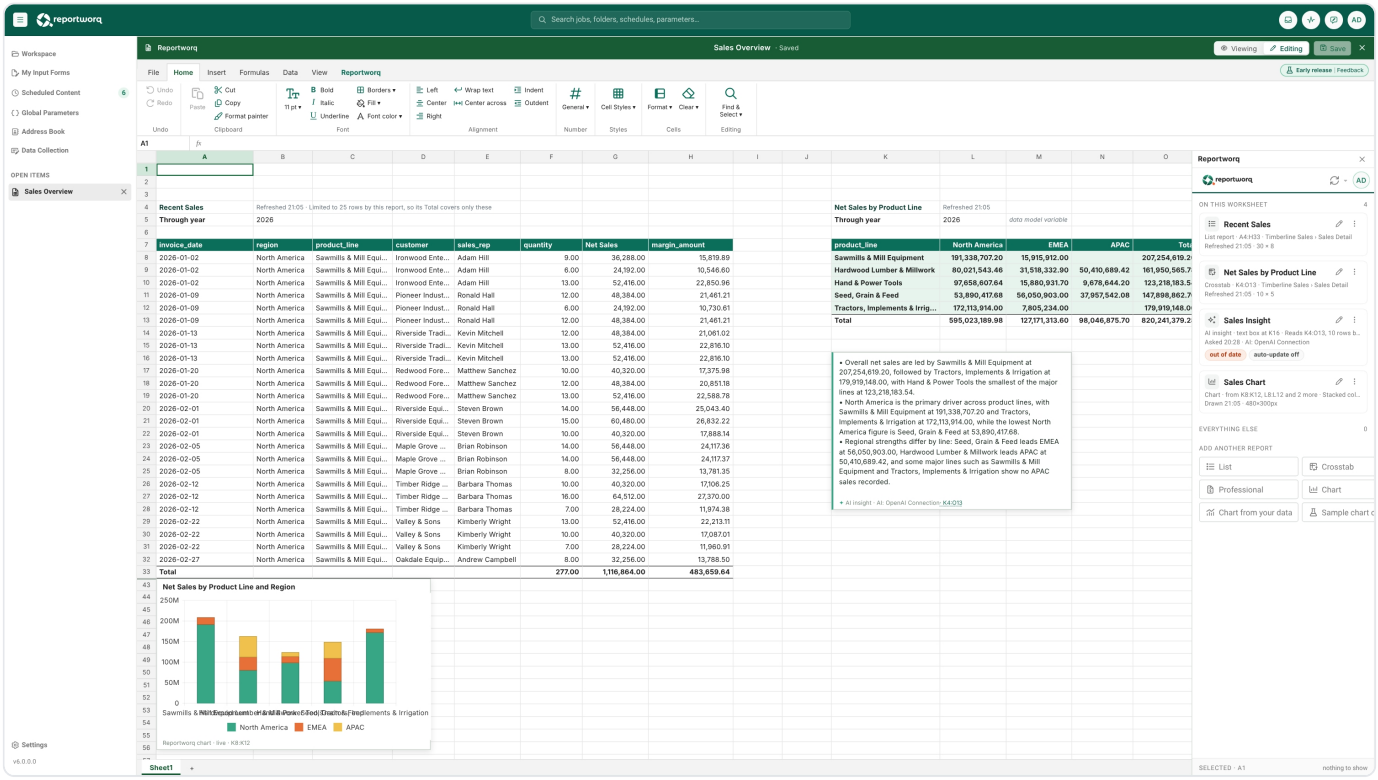
> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A web report or workbook can hold many reports at once, on one sheet or several: lists, crosstabs, professional reports, charts, maps and AI insights. Each is an independent report with its own definition, and several of them can be tied to the same data or cell so the sheet reads as one page.

When to use it. A Timberline management page: a crosstab of revenue by region, a column chart beside it that grows with it, a short AI insight underneath that explains it, and a Period cell at the top that all of them respect.

Combinations that work

Combination	What you get	How
Chart over a list or crosstab	The chart draws every data row, grows and shrinks with the report, skips the Total row, and follows filters and burst parts	Select the report's cells and **Chart from your data**, or choose source mode **Report** on the chart's **Data** page. See Charts and maps
Map over a table of places	Markers or bubbles from a table, optionally on a floor plan	A list report or plain cells can hold the table
AI insight on a report	A written explanation that follows the report, whatever its filters	Source mode **Report** in the insight editor. See AI insights
AI insight on a chart or map	The insight reads the chart's data range, with its header row	Source mode **Report** over the chart
AI insight on top of a chart over a crosstab	One sheet with a table, a picture and a paragraph, each following the one before	Make the crosstab, the chart, then the insight, each with **Use selection**
One cell driving several reports	A Period, Region or Scenario cell that every report on the sheet reads	A report filter bound to a cell in each report. See Cell-driven variables
A job parameter driving several reports	One job value narrows the reports you map it to	Map the parameter once per report; a parameter mapped only to a cell reads "Mapped in: ..."
A crosstab converted to a professional report	A chart, map or AI insight that followed the crosstab follows the new report	Done for you by **To professional report**



How the pieces relate

- **Following.** A chart, map or AI insight made from a selection remembers whether it follows a report (**Report**), grows from cells (**Grows**) or reads a fixed range (**Fixed**). Following is bookkeeping done by the pane; renderers and job runs do not rewrite it.
- **Placement.** A new chart or map is placed two columns to the right of its selection and never on another picture. An AI insight text box is a picture on the page, so a chart made over the same selection moves clear of the box.
- **Refresh order.** Refresh all asks AI insights last, so they explain data that has just been refreshed.
- **Copies.** When you copy a sheet, a chart or insight on the copy follows the copy's report, not the original's.
- **One sheet, one grid.** A report that grows pushes the cells below it down. If doing so would run through another report on the sheet, the refresh is refused or, for a professional report, inserts cells only across its own columns. Leave room, or put reports side by side with a spare column or place them on different sheets.

What does not combine

- A **professional report** is licensed and not built in the Excel add-in. See Professional reports. An AI insight is not asked over a professional report's cells on an unlicensed server.

- Reports on **a hidden sheet** are checked when you convert a web report for a job that narrows it by a filter or a cell. A list or crosstab there must be narrowed by the same filter, and a map or professional report there cannot be narrowed at all, so convert is refused. Move it to a visible sheet or remove it.
- A **map** is delivered as saved in a job. It does not re-read its places for each run, so a job that narrows a web report cannot narrow a map.
- A **chart** whose crosstab no longer has the column a series was named for leaves that series out. A series named "Series 1" is drawn from its column as it is.
- A Grows source stops at any report's cells, so an insight that grows from a cell never reads another report's title.
- A filter reads a cell **on its own sheet only**. Move the cell, or have the job write it there.

In a distribution job

When the job uses the web report, lists and crosstabs are re-queried, charts are redrawn from that run's data, AI insights are asked again (or withdrawn with a warning if they fail), and maps and professional reports go out as saved. For the details see A web report as a job source.

> **Going deeper.** The web report studio and The Excel add-in and the studio explain where each report type can be built.

View visibility

reference author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

Every view in a data model has a **visibility**: **Public**, **Hidden** or **Disabled**. It decides two things: whether report authors are **offered** the view when they pick data, and whether the view is **allowed to run**. It is not a permission. Who may open the data model at all is decided by the folder permissions on the model.

When to use it. Keep a view **Hidden** while you build it, or when it only exists to feed a joined view. Make it **Public** when report authors should find it. Set it **Disabled** to stop it everywhere because it is wrong. At Timberline Harvest Co., the **Sales Journals** model has **Sales Details**, which report authors pick, set to **Public**, and **Sales Channels**, a small list of channels that only supports other views, left **Hidden**.

The three states

	Public	Hidden	Disabled
Listed in the Excel add-in and the web report studio, for everyone, authors included	Yes	No	No
Joined views of the model can read it	Yes	Yes	No. The joined view fails, naming it
Runs (reports, jobs, previews in the add-in)	Yes	Yes	No. It fails, naming it
Editable in the data model editor, and **Preview** works	Yes	Yes	Yes
A new view starts as		Hidden	

The tooltips on the three buttons say the same in a sentence:

- **Public**: `Listed in the Excel add-in for everyone. Joined views can read it, and it runs.`
- **Hidden**: `Not listed in the Excel add-in, for anyone. Joined views can still read it, and reports already using it still run.`
- **Disabled**: `Doesn't run anywhere but Preview here. Not listed, and joined views and reports that use it fail, naming it.`

Who sees what

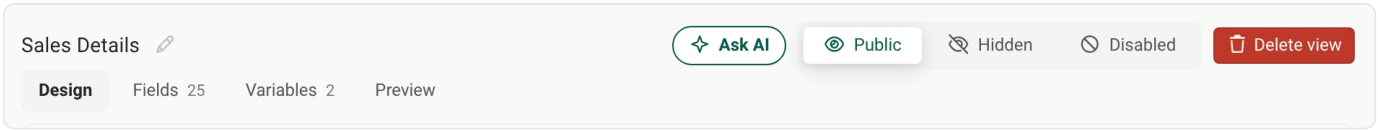
Who	Where	Public	Hidden	Disabled
Model author	The data model editor	Sees and edits it	Sees and edits it	Sees and edits it, and **Preview** runs it
Report author	The Excel add-in, when picking data for a new report	Listed	Not listed	Not listed
Report author	The web report studio and Explore, when picking data	Listed	Not listed	Not listed
Anyone, with a report already built on the view	An existing Excel report, web report or distribution job	Runs	Runs	Fails, naming the view
Reader (a person who only reads what is shared with them)	A web report shared with them	Runs	Runs, if they can see the data model's folder	Fails, naming the view
Reader	The Excel add-in	Sees no views, whatever their visibility	Sees no views	Sees no views

Points that table cannot carry:

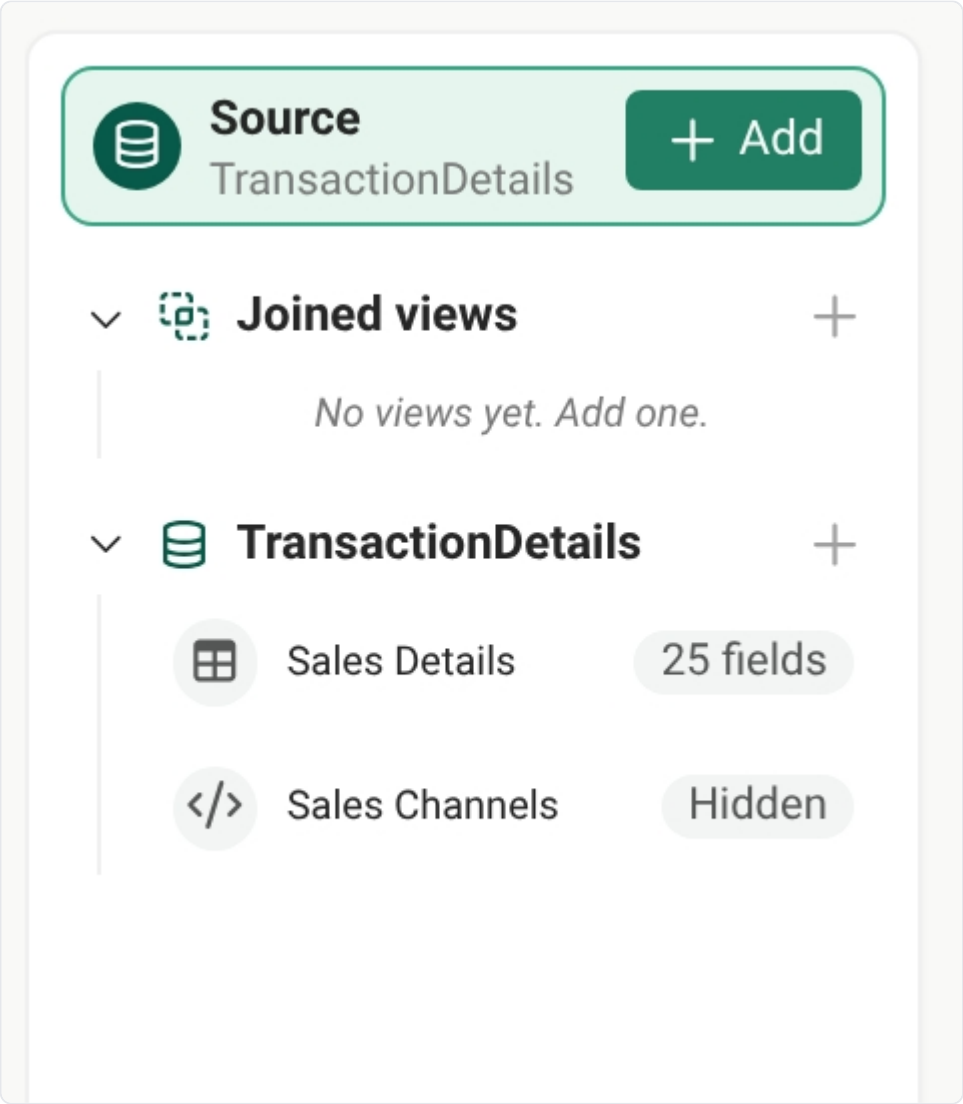
- **Authors get no special list.** The rule is the same for everybody: only **Public** views are listed, authors' own included. To use a **Hidden** view in a new report, set it to **Public**, build the report, and set it back. The report keeps running.
- **A model with no Public view is not listed** in the add-in or the studio, unless it also has Direct queries on, which lists it for **New query**.
- **Readers in the Excel add-in** see the notice `Live reports in Excel are for authors. Ask your administrator for author access.` instead of a list of views. Readers reach data through web reports and through the output of distribution jobs, not by building in the add-in.
- **A reader can run a Hidden view** if they can see the model's folder, for example through a shared web report built on it. See Hidden is not a permission.

Set a view's visibility

1. Open the data model with **Edit** from its row in the workspace, and select the view in the rail.
2. In the view's header, choose **Public**, **Hidden** or **Disabled**. The three buttons sit beside **Delete view**. Hover a button to read what it does.
3. If you choose **Disabled** and something in the model uses the view, Reportworq opens **Disable view** and lists what uses it: joined views, relationships between views, level hierarchies, and views whose variable choices are listed from it. Choose to go ahead or cancel. Reports and jobs already bound to the view are not listed, because the model does not record them.
4. Select **Save Changes**. The new visibility takes effect for everyone when the model is saved.



In the rail, a view that is **Hidden** or **Disabled** says so on the right of its row. A **Public** view shows its field count instead.



What a Disabled view does

A **Disabled** view never runs outside the data model editor's **Preview**. Every other way of running it is refused with a message that names the view:

> “Sales Channels” is disabled. Set it to Public or Hidden in its data model to run it.

The refusal applies to Excel reports, web reports, distribution jobs, a variable's choice list read from the view, and a joined view that has the view as an input, even in **Preview**. A joined view over a Disabled input fails before it reads any source. In the editor, the view's status as you have it unsaved counts, so an input you just set to **Disabled** is refused before you save.

To bring the view back, set it to **Public** or **Hidden** and save.

What a Hidden view does

A **Hidden** view is left out of the lists, and nothing else changes:

- **Reports and jobs already using it keep running.** Hiding a view is how you retire it from the picker without breaking anything.
- **Joined views read it** like a Public view. A Public joined view can sit over several Hidden input views, so authors see one clean view and not the building blocks.
- **Hierarchies and variable choices** can still be built from it.

Hidden is not a permission

Hidden only leaves a view out of the lists in the Excel add-in and the web report studio. **Anyone who can see the data model through its folder permissions can run a Hidden view**, readers included. A web report that is shared with readers shows them the rows of the Hidden view it is built on.

To keep data private:

- Put the data model in a folder that only the right people can see, or
- Set the view to **Disabled**.

Save warnings follow visibility

If a view's query cannot run when you save, the warning depends on its visibility. A **Public** view gets a modal that says it will not run and suggests setting it to **Hidden** or **Disabled**, because report authors can pick it. A **Hidden** or **Disabled** view whose fields could not be read gets only a light notice, because nobody is offered it.

Notes and limits

- **Each view has its own setting.** A model can hold any mix.
- **Direct queries skip views, and so visibility.** See Direct queries.
- **Upgrading from an earlier version.** Views that were **Published** are now **Public**, and views that were **Draft** are now **Hidden**. Nothing that ran before stops running. Authors no longer see Draft (now Hidden) views in the Excel add-in. A reader with access to the model's folder can now run a view that used to be a Draft one, so before you upgrade, protect any such view that holds data readers should not see with folder permissions, or set it to **Disabled** afterward.
- **Rolling back.** An earlier version does not know **Disabled** and treats a Disabled view like a draft one, which runs. Do not roll back past this version while views are Disabled, or first set those views to **Hidden** and expect them to run.

- **Sharing a model across servers.** Export and import carry each view's visibility. A model exported from an earlier version arrives with Draft views as **Hidden** and Published views as **Public**.

> **Going deeper.** The full tour of the model editor is in Data models. For how joined views read their inputs, see Join views across sources.

Cell-driven variables

how-to author

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A report filter or variable normally holds a fixed value. A cell-driven variable instead reads its value from a cell on the sheet, every time the report refreshes. Change the cell and refresh, and the report is queried again with the new value. One cell can feed several reports.

When to use it. A cost-center cell on the Financial Reports sheet that a department head changes, a Scenario cell that flips a forecast between Budget and Forecast, or a Period cell that a distribution job fills with a different value for each run.

Where it works

	Web report studio	Excel add-in
Report filters and variables that read a cell	Yes	Yes. The cell is read from Excel itself on every refresh
Drop-downs above a report for data model variables and filters	Yes	Yes, as in-cell drop-downs
A distribution job writing the cell	Yes (a job's cell parameter)	Yes, when the job runs the workbook

Before you start

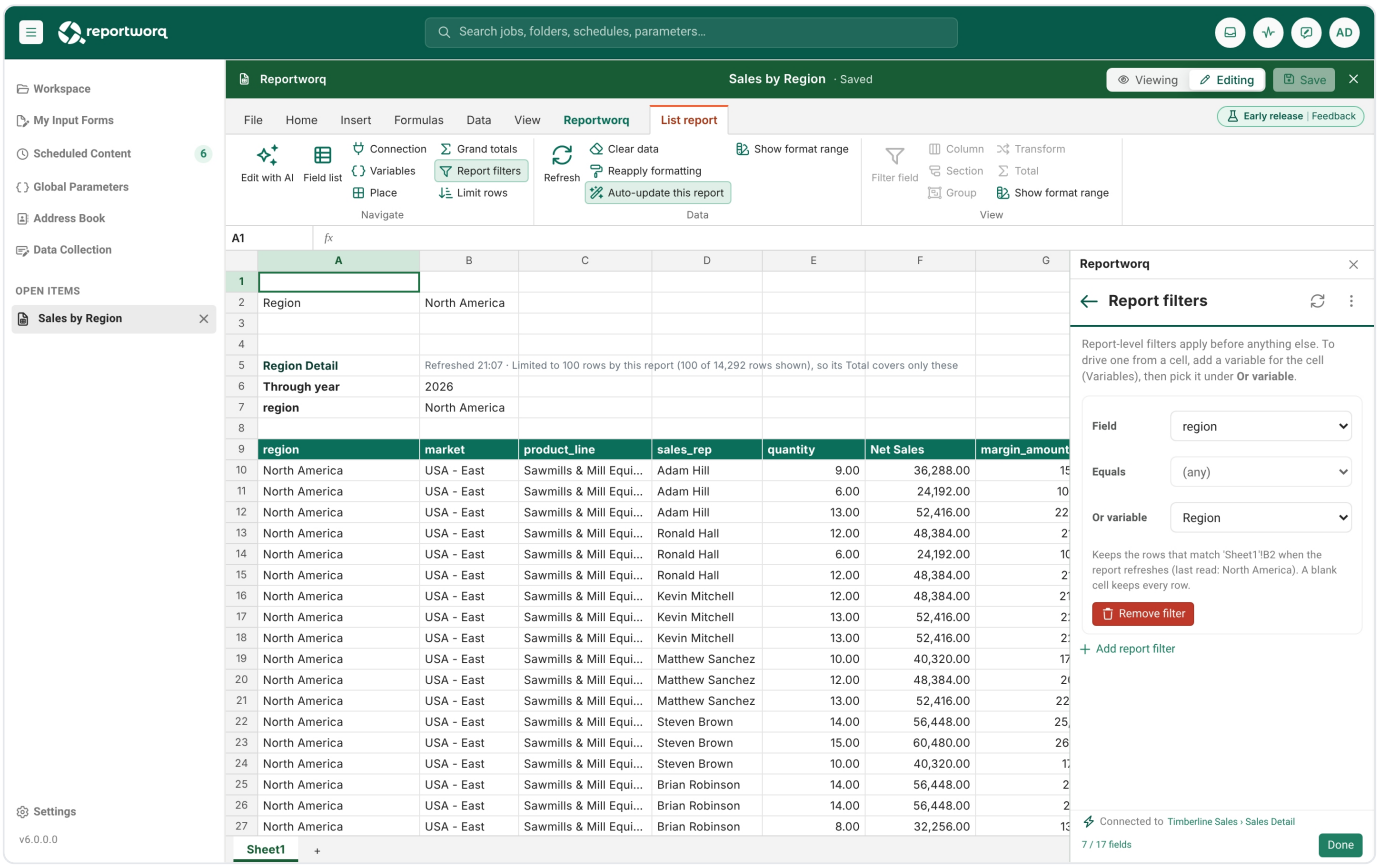
- A list or crosstab on a data model view. See List reports and Crosstabs.
- A cell that will hold the value: typed, a formula, or a drop-down made with data validation. Put it on the same sheet as the report that reads it.
- If the value goes into a data model variable, know its default and whether it is required. See Data model variables.

Bind a filter to a cell

1. Open the list or crosstab and go to **Report filters** (the Crosstab ribbon tab, or the editor menu).

2. Add a filter. Pick the **Field**, leave **Equals** empty, and under **Or variable** choose a variable name, for example `Region`.
3. Go to the **Variables** page. Each variable has a **Name**, a **Cell** and a **Current value**. Enter the cell, or select **Use A1** to use the cell you have selected on the sheet. The top-left cell of a range is the one read.
4. Refresh. The filter now uses the cell's value, and a line above the report shows the filter, its value and a "report filter" note that names the variable.

A blank cell keeps every row: the filter help reads "A blank cell keeps every row." For a data model variable, an empty cell falls back to the model's default for that variable. A variable with no value and no default blocks the report from running until you set one.



Variables are found automatically: from the data model's own variables, and from `{{tokens}}` used in report filters. A variable nothing references any more is removed.

Use the drop-downs above a report

A crosstab writes its filters and data model variables above itself as in-cell drop-downs, in the style of a PivotTable: "(All)", one member, or "(Multiple items)". Picking one re-renders the report. Choose "(All)" to clear a filter and "(Any)" to clear an optional variable; Delete clears neither.

A variable that has a pick list and does not allow typing takes only a value on its list, a blank or the model's default. A pasted value that is not on the list shows "Not on the

list", and the cell goes back to the value in effect. A list longer than 255 characters in all, or with a comma in an item, has no drop-down in real Excel, so the cell is plain text.

Drive the cell from a job

A distribution job can write a value into a cell by parameter, which re-queries every report that reads it. This makes one workbook layout serve every department or period. Details of mapping are in A web report as a job source.

When a job's cell parameter can write a cell that a report filter reads, rules apply to protect recipients:

- A report filter reads a cell **on its own sheet only**. To use a value from another sheet, have the job write the value into a cell on the report's own sheet.
- A formula on the report's sheet that points at another sheet reads as an error, and the report comes out empty.
- If a hidden sheet holds a list or crosstab that does not filter on that cell, converting the web report is refused, so a recipient cannot get all of that report's data by accident.

Notes and limits

- A variable's cell is read on every refresh. If Excel cannot be asked right now (a failed read, edit mode, writes still queued), the refresh stops with a "Try again" message that names the variable, and never filters on an old value.
- A reference that names nothing reads as blank. A reference to several cells takes the first.
- A value no row has shows "No row of the source has this value" beside the variable.
- A professional report's variables are fixed when the report is made and take no typing. See Professional reports.
- A chart's title and a map's center, zoom and tile settings can also be cell-driven. Use the small menu inside the field to choose **From a cell**. Map framing cells must be single cells on the map's own sheet.
- Values go into queries safely: a value can only be a value. See Data model variables.

Data model variables

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **variable** lets a report supply a value to a data model view at run time: the fiscal year of a P&L, the region of a burst, the cost center of a budget submission. You write it as a `{{Token}}`, for example `{{Year}}`, and set how it behaves on the view's **Variables** tab. The values flow downstream to Preview, to reports, to bursting, and to the Excel add-in.

Where variables come from

A view's **Variables** tab shows each variable the view uses as a card, headed by its caption (once you set one) and its `{{Token}}`, in up to three groups:

Group	Where the token is	Required?
Connection - required	In the view's query: its SQL, MDX, or DAX, or a connector's query setting, for example `WHERE fiscal_year = {{Year}}`.	Always. A query cannot run with a hole in it.
Filters	In a field filter's value on the **Fields** tab, for example the **Region** field filtered to `{{Region}}`. A joined view's own filters count too.	Your choice. An optional filter variable left blank drops its filter, so the view returns every region.
From the joined views	In the input views of a joined view.	See Variables of a joined view.

The sample value is not a default

A view whose query has a `{{Token}}` needs a value to run while you build it, so the **Design** or **Connection** tab shows a **Connection values** sidebar for a sample value. Reportworq uses the sample to run the query, work out the view's fields, and fill in Preview while you author.

The sample is only for authoring. It never becomes the variable's default, and reports never see it. Set the default on the **Variables** tab. Preview starts from the default, and from the sample value when a view has no default. A joined view never asks for a sample value: see Variables of a joined view.

Set a default and whether a value is required

1. Open the data model, select the view in the rail, and select the **Variables** tab.
2. In the variable's card, enter its **Default**: the value Preview and reports use when nothing overrides it, for example `2024`. For an optional filter variable the box reads **Optional: leave blank to skip the filter**; otherwise it reads **Default value**.
3. For a filter variable, select or clear **Required** at the top right of the card. A connection variable always shows **Required** and cannot be made optional.
4. Select **Save Changes**.

A report, a burst, or the Excel add-in can override the default with its own value. In Preview, a required variable left blank brings a warning, "No value for required variable(s)", that names it.

Variables in Preview

The **Variable values** pane in Preview starts with each variable's default. For a joined view that is the same value a run would use. Two rules then apply:

- **A box you change keeps your value until you select Reset to defaults.** That includes a box you clear. Editing the model elsewhere does not refill it.
- **A cleared box runs blank.** An optional filter variable with no value is dropped, so you see all rows. A required variable with no value is refused, with a warning that names it.

A box you have not touched follows the view's defaults as you change them.

Set how a report prompts for a variable

Select **Prompt** at the top right of a variable's card to open its prompt settings. They decide what the person running the report sees when asked for the value.

- **Caption** and **Help text** name the variable and explain it in the prompt, for example **Fiscal year** and "The year of the actuals to report".

- **Choices** sets where the values to pick from come from:

    - **Free text**: the person types a value. This is the default.
    - **A view of this data model**: pick a view, then the field whose values become the list, for example a cost-center lookup view and its **Cost Center** field. Any other view of the same model can supply the list.
    - **Parameter list**: select **Set the list…** to open the same parameter editor the job editor uses, titled **Choices for** `{{Name}}`. Every parameter type your installation offers is available: a SQL query, a Planning Analytics subset or

dimension, an Excel range, a list of values, a global parameter, and the rest. See Parameters reference. An **Excel Range** list can take its values **By column** (pick a worksheet and one of its column headers; the list follows the data as rows are added), or **By range** (a named range or an address, with **My data has headers**); see Excel Range lists. The card then reads **List from:** and the type, with **Edit the list…** beside the drop-down to change it.

- **Also allow typing a value that isn't on the list** appears once **Choices** is a view or a list. Leave it off to limit people to the list. Turn it on when the list is a convenience rather than the complete set of valid values.

  With it off, the variable is **pick only**. In this release the choices shape the variable's prompt in the data model's **Preview**: a drop-down of the list, or a box you can also type in when the option is on. Preview shows a plain message when a value isn't on the list, for example "East2 isn't one of the choices for Region. Pick a value from its list.", so you can check the rule before readers use it. A report built in the Excel add-in uses its own value as given, whether typed or read from a cell. **A job is never refused**: a value that a job supplies, from a parameter or a burst, is used as given, so bursting with any value works.

- **In the query: Raw query text, pasted as is** is the opt-in described in Use a value as query text. It appears for a connection variable on a view whose query text Reportworq fills: a SQL query, a Planning Analytics MDX query, a Power BI or Analysis Services MDX or DAX query, or a Pigment export view. In a Pigment export, values in the URL and the JSON body are otherwise encoded; the opt-in pastes the value unencoded.

Select **Save Changes** when you are done.

How a value is filled into a query

People type values into prompts and Excel cells feed values into variables, so Reportworq makes each value safe where it is pasted into a SQL, MDX, or DAX query. A value can never change what the query does: it can only be a value. Reportworq looks at where each `{{Token}}` sits in the query text and fills it accordingly.

The rules by position are the following.

Where the token sits	What happens to the value
Inside a quoted string, such as `'{{Market}}'`	The quote character is doubled, so `O'Brien` becomes the text `O'Brien`, not the end of the string.
Inside a quoted name, such as `[{{Column}}]`, `"{{Column}}"`, or a DAX `'{{Table}}'`	The closing character is doubled in the same way.
Bare, not inside quotes	Only a number is accepted.
Inside a comment	A value that would end the comment, or start a new one, is refused.

Two kinds of bare value are accepted as well as a number:

- **A list, in SQL**, when the token sits directly inside parentheses: `IN ({{Regions}})` with the value `'East', 'West'`, or `IN ({{Years}})` with `2023, 2024`.
- **Member names, in MDX**, such as `[Region].[East]` or `[Region].&[12]`, and comma lists of them, so bursting on members still works.

A value that breaks a rule fails the run, and nothing is sent to the source. The message names the variable and says what to do. For a bare token in SQL it reads:

> The value for “Region” can't be used: it is pasted into the query without quotes, so it only takes a number (or, inside IN ( ), a list such as 1, 2, 3 or 'East', 'West'). For a text value, the data model's author should put quotes around it in the query: '{{Region}}'. Only as a last resort, for query text such as a column name, can the data model's author mark the variable as raw query text, which turns this protection off and pastes readers' values unchecked.

In MDX the message asks for brackets instead (`[{{Region}}]`), and in DAX for double quotes (`"{{Region}}"`). On a connection whose query uses syntax that databases read differently (see Database rules), the message says that the variable can only take a plain number unless the author marks it as raw query text, and suggests rewriting that part of the query.

What fails and how to fix it

What you see	Why	Fix
The message above, for a text value such as `East`	The token sits bare in the SQL (`WHERE region = {{Region}}`), so only a number is accepted.	Quote the token in the query: `WHERE region = '{{Region}}'`.
The same message for a date such as `2024-01-01`	A bare date is not a number.	Quote it, `'{{Day}}'`.
The message, for a list such as `East, West`	A list is accepted only directly inside parentheses, and each text item needs its own quotes.	Write `IN ({{Regions}})` and supply `'East', 'West'`.
The message, for a column or table name (`ORDER BY {{SortColumn}}`)	A name is query text, not a value.	Select **Raw query text, pasted as is** for that variable (see Use a value as query text), and limit the values to a list.
A message that says the variable can only take a plain number	The connection is ODBC, OLE DB, or CData, or the query runs a string as SQL, or has a lone carriage return.	Rewrite that part of the query, move to a SQL Server or SQLite connection, or opt in as above.
No error, but no rows (`IN ('{{List}}')` with `East','West`)	The value is now one text string, not several items.	Write `IN ({{List}})` with the value `'East', 'West'`.

Write tokens so they fill cleanly

The following patterns cover most queries.

To pass	Write	Example value
A text value in SQL	`WHERE market = '{{Market}}'`	`Pacific Northwest`
A number in SQL	`WHERE fiscal_year = {{Year}}`	`2024`
A date in SQL	`WHERE posted_on >= '{{From}}'`	`2024-01-01`
Several values in SQL	`WHERE region IN ({{Regions}})`	`'East', 'West'`
A member in MDX	`[Region].[{{Region}}]`	`East`
Members in MDX	`{ {{Regions}} }`	`[Region].[East], [Region].[West]`
Text in DAX	`"{{Region}}"`	`East`

Database rules

How a database reads quotes and comments differs, so Reportworq applies the rules of the view's connection:

- **SQL Server**: a backslash in a string is an ordinary character, and block comments can nest.
- **SQLite**: a backslash is an ordinary character, and block comments do not nest.
- **ODBC, OLE DB, and CData** connections can reach any database (MySQL, PostgreSQL, Oracle, Snowflake, and others), so they get the strictest rules. A backslash inside quotes is refused. And if the query itself uses syntax that some of those databases read differently, every token in it takes only a plain number. That syntax is a backslash inside a literal, `#` or `//` comments, `$` (dollar) quoting, `[brackets]`, triple quotes, Oracle `q'` quoting, `E'` strings, `/*!` comments, nested `/*` comments, or `--` with no space after it. Rewrite that part of the query, or connect with a SQL Server or SQLite connection where one applies.
- **SQL that runs a string as SQL**, such as `EXEC('…')`, `EXECUTE`, `sp_executesql`, `PREPARE`, or `EXECUTE IMMEDIATE`: a value that is safe as text becomes code once the string runs, so every token in such a query takes only a plain number. Pass values to dynamic SQL as `sp_executesql` parameters instead.
- **Any connection**: a lone carriage return in the query text (an old Mac line ending) makes every token take a plain number. Save the query with normal line endings.

Planning Analytics, Analysis Services, and Power BI

MDX and DAX follow the same idea:

- A bare MDX token takes only a number or member names. A value such as `East` in `{[Region].{{X}}}`, before `.Children`, in a range, or as a set expression is refused. Put the token inside the brackets instead: `[Region].[{{X}}]`.
- An ordinary MDX string is a plain text value: in a comparison such as `.Name = "{{Region}}"` the quote is doubled and any text works. But a string that Planning Analytics and Analysis Services read again as MDX (the argument of `StrToMember`, `StrToSet`, `StrToTuple`, or `StrToValue`, or the body of `WITH MEMBER` or `WITH SET … AS '…'`) gets the bare-token rule, outside a `[name]`.
- A bare DAX token takes only a number. Put it inside `"…"` for text.
- Names that Reportworq passes to Planning Analytics as names rather than MDX refuse `[ ] { } ( )`, quotes, `;`, and line breaks. That covers the dimension, hierarchy, and subset of an **Elements** query, and the cube and view of a **Data view** query.

Field filters

A value that reaches a field filter, and is pushed down to the source as part of its query, is written by its field's type: a number is checked and written as a number; text, dates, and true/false values are quoted with their quotes doubled; and each item of a list is written the same way. Nothing in the filter needs quoting by hand.

Adaptive Planning names

A value placed in an Adaptive Planning sheet, version, period, level, or dimension name is not checked by position: it is filled in as given. Give such a variable a list of **Choices** with **Also allow typing a value that isn't on the list** turned off.

Use a value as query text

Sometimes a variable is meant to change the query itself: a column to sort by (`ORDER BY {{SortColumn}}`), a table or schema name (`FROM {{Schema}}.ledger`), or an MDX set expression. A value like that is not a number, so it is refused unless you opt in.

1. On the view's **Variables** tab, select **Prompt** on the variable's card.
2. Under **In the query**, select **Raw query text, pasted as is**. A note under it reminds you that readers' values are then pasted into the query unchecked, and that quoting the token is the better choice for text.
3. Select **Save Changes**.

The value is then pasted into the query exactly as given, with no checking. The setting is stored on the view's own query, so only the data model's author can turn it on: nothing a report or a caller sends can set it.

> **Warning:** A raw query text variable can change what the query does. Whoever can supply its value can rewrite that part of the query against the source, with the connection's permissions. Turn it on only for a variable that genuinely holds query text, and give it a list of **Choices** with **Also allow typing a value that isn't on the list** turned off, so people pick from known values.

If you are upgrading

Earlier releases pasted a value into the query as raw text. Queries that relied on that can stop working after the upgrade. Check views that match any of these patterns:

- **`IN ('{{List}}')` with a value like `East','West`.** The value used to expand into several items. It is now one string, `East','West`, which matches nothing, **without**

an error. Write `IN ({{List}})` with the value `'East', 'West'` instead.
- **A token used as SQL text**, such as `ORDER BY {{Col}}`, `FROM {{Table}}`, or `{{Schema}}.x`, now fails until you select **Raw query text, pasted as is** (see Use a value as query text).
- **A bare date**, such as `WHERE posted_on >= {{Day}}` with `2024-01-01`, now fails. Quote it in the SQL, `'{{Day}}'`, or opt in as above.
- **On ODBC, OLE DB, and CData connections**, a value with a backslash (`DOMAIN\user`, a file path) is refused inside quotes, and a query that uses `[brackets]`, `#temp` tables, `$` quoting, `//` comments, triple quotes, Oracle `q'` quoting, or `--` with no space after it takes numbers only unless you opt in. Use a SQL Server or SQLite connection where one applies.
- **SQL that runs a string as SQL** (`EXEC('…')`, `sp_executesql`, `PREPARE`) takes numbers only in every token, unless you opt in. Pass values to dynamic SQL as `sp_executesql` parameters instead.
- **In MDX and DAX**, a bare token takes only a number (or member names, in MDX), and so does a token inside a `StrToMember`-style string or a `WITH … AS '…'` body. Move it inside brackets in MDX (`[Region].[{{X}}]`) or inside `"…"` in DAX, or opt in.
- **Planning Analytics names** refuse the characters listed above: the dimension, hierarchy, and subset of an Elements query, and the cube and view of a Data view query.
- **A sample value is no longer copied into an empty default.** A view saved by an earlier release may already carry a default that was copied from its sample value. Check the **Variables** tab and clear any default you did not mean to set.

The run's error names the variable and the reason, so the failing view is easy to find.

Related topics

- Data models
- Join views across sources
- Build a view with the designer
- Reportworq 6 release notes

Global parameters

explanation author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A **global parameter** is a named value that lives at the workspace level instead of inside one job or one report. You define it once, such as the reporting year, the version, or the entity, and everything that needs it points at it. When the value changes, you change it in one place and every user of it follows on its next run.

This page is the reporting view of global parameters: what they are for next to data model variables, and the rules to know. Creating, renaming, and deleting them is covered in full in Global parameters in the User Guide.

When to use it. Use a global parameter when the same value is, or should be, identical across many jobs or many prompts. At Timberline Harvest Co. the `Current Year` global parameter is a subset of the time dimension in the planning model, and the Financial Statements job and the budget campaign read it as their `Year`. A data model variable such as `{{Year}}` can offer the same list of years as its choices. Use a plain variable default instead when only one view needs the value.

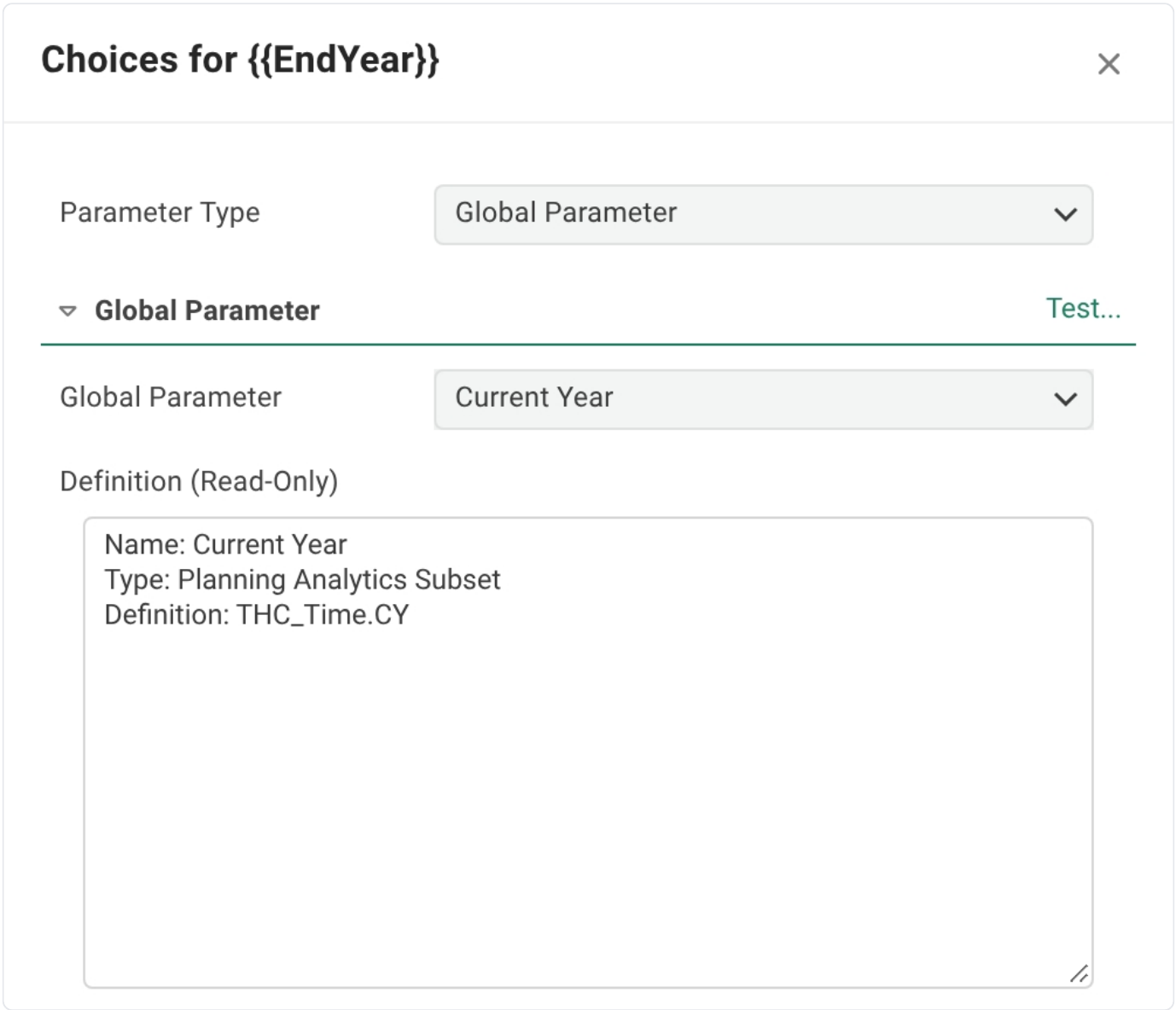
Where a global parameter is used

Place	How it uses the value
A distribution job's parameter	The author binds the job's parameter to the global parameter on the job's **Parameters** step. The job then runs with the global value. See Parameterize a distribution job.
A data model variable's **Choices**	A variable's prompt can take its list of values from any parameter type, **Global Parameter** included. The list a reader picks from then follows the global value.
A schedule or a run parameter	Points at the global parameter the same way a job does.

A web report or a data model view does not read a global parameter directly. The value reaches a view only through a variable: a job or a prompt supplies the variable's value, and Reportworq fills it into the view's query as described in Data model variables.

Use a global parameter as a variable's choices

1. Open the data model, select the view, and select the **Variables** tab.
2. On the variable's card, select **Prompt**.
3. Under **Choices**, select **Parameter list**, then select **Set the list…**. The editor opens, titled **Choices for** `{{Name}}`.
4. Set the parameter type to **Global Parameter** and pick the global parameter.
5. Save the editor, then select **Save Changes** on the data model.



The card then reads **List from:** and the type. Select **Edit the list…** to change it.

Rules to know

- **A global parameter holds its own value.** Its value can be any parameter type your installed connectors offer (a subset, a list of values, a SQL query, and so on) except **Global Parameter**. A global parameter can no longer point to another

global parameter.

- **One hop only.** A job, a schedule, a run parameter, or a data model variable's choices can point at a global parameter. The global parameter itself cannot point onward.
- **Older entries that already point at another global parameter** are not rewritten. A job or prompt that uses one fails with an error that names both parameters. Open the entry on the **Global Parameters** screen and choose a value type other than **Global Parameter**. The list flags such an entry in its **Value** column.
- **Changing a used value asks first.** Renaming, editing, or deleting a global parameter that jobs use shows **Impact Review Required** and lists the affected jobs. The change applies to their future runs.
- **Workspace scope.** A global parameter is not shared across workspaces.
- **Who sees the screen. Global Parameters** is in the left side rail for authors and administrators. A reader-only user does not see it.

> **Going deeper.** For the full steps to create and change a global parameter, and the **Used In** list, see Global parameters. For the parameter types a global parameter can hold, see Parameters reference. To see Timberline's `Current Year` at work, see How Timberline rolls the reporting year forward.

Related topics

- Data model variables
- Data models

Discover your data

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

Before you build a report on a source, look at it. Reportworq gives you several ways to see what a source and a data model hold, from a quick look at one table to a live exploration of every view. Each is read-only: none of them changes the model, the source, or any report.

When to use it. Use these when you meet a source you do not know, when a view returns something you did not expect, or when you want to hand a snapshot of the data to someone. For example, at Timberline Harvest Co. you might browse `sales_journal` to see what one invoice line looks like, then explore the finished view with a pivot to check the totals before you publish it.

Pick the right tool

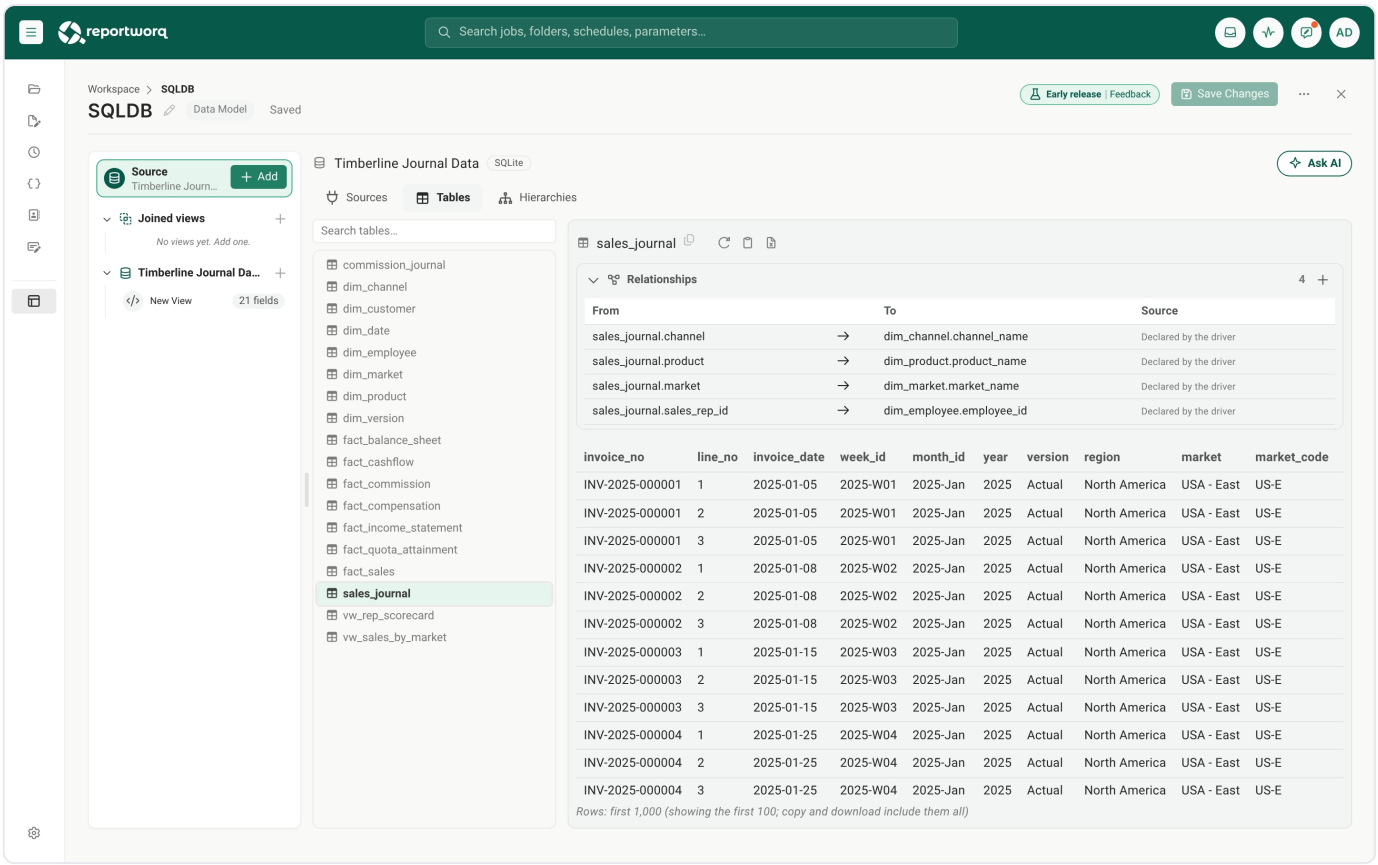
You want to	Use	Reads
See the raw rows of one table, before you model it	Browse a table	Up to 1,000 rows
Check what a view returns, with its variables	Preview a view	The rows you ask for, up to 10, 100, 1,000, or all
Pivot, drill, and total a whole model's views	Explore a model	A live list per view
Copy every table of a source into one workbook	Export a source to Excel	Every row, within a size budget
Check that two views match on a key	Check how views match	Up to 10,000 rows per view (1,000 for **Discover relationship**)
Check a hierarchy's members	Preview a hierarchy	The hierarchy's view, up to 1,000,000 rows
Have the data explained in plain words	Ask the assistant to explain	A small sample of rows

Before you start

- A data model with at least one source. See Data models.
- Permission to open the data model in the editor. **Explore** is for authors and administrators; a reader-only user cannot open it.

Browse a table before you model it

1. Open the data model and select **Source** in the rail.
2. Select the **Tables** tab. It appears when the model has a SQL source to browse, including an Excel or SQLite workspace file.
3. Select a table in the list. **Search tables** narrows a long list.
4. Read the rows. The grid reads at most 1,000 rows, and the row count under it says how many were read. The **Relationships** section above the rows shows the table's links.
5. Optional: use the clipboard button to copy the rows with a header row, ready to paste into Excel, or the Excel button to download a workbook of the table.



Preview a view

Preview runs a view the way a report would and shows the resolved rows after the variables and field choices are applied.

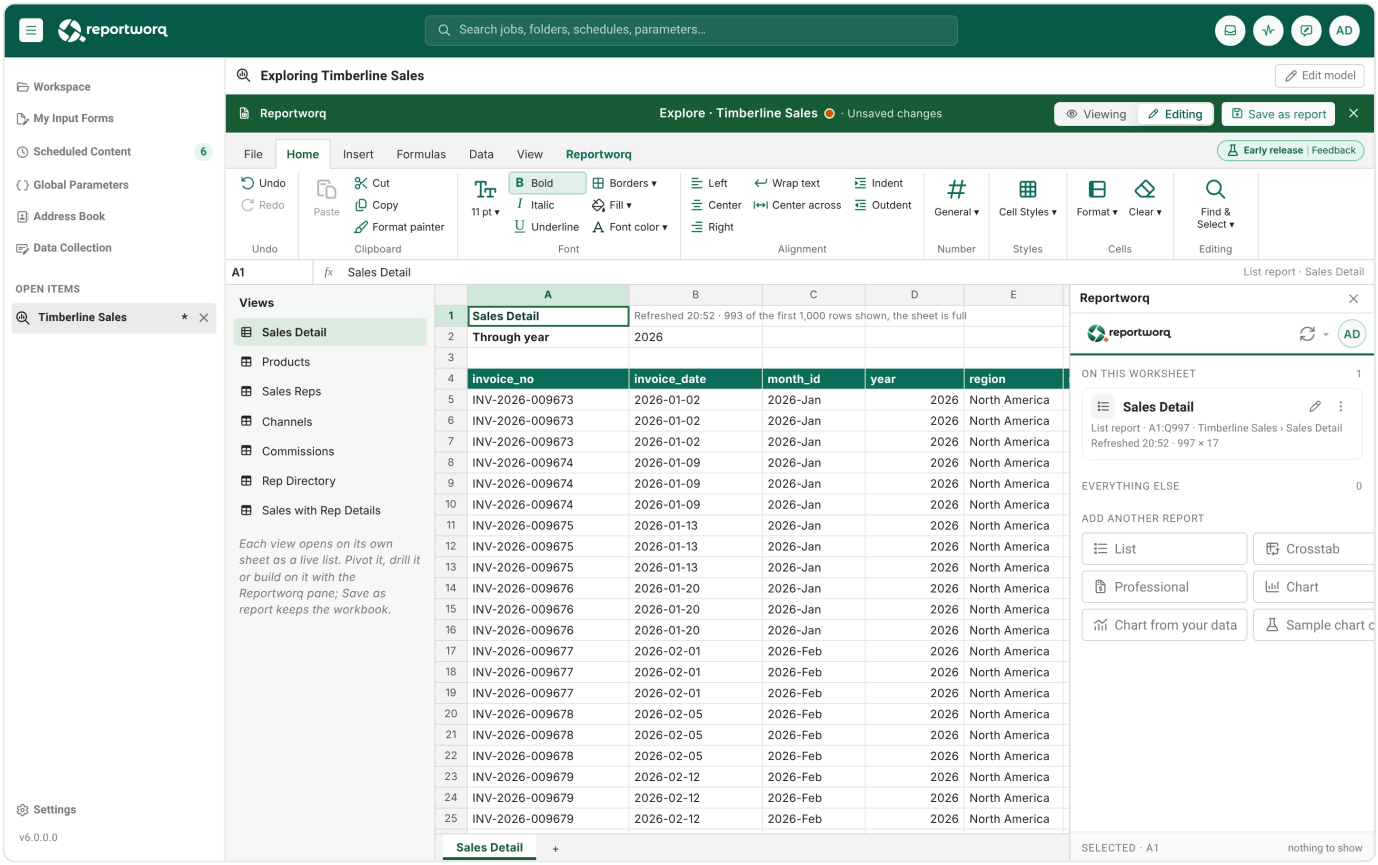
1. Open the view and select the **Preview** tab.
2. Choose how many rows to read: **10**, **100**, **1,000**, or **All rows**. The default is 100.
3. If the view has variables, the **Variable values** pane is open. Enter values for this run only; **Reset to defaults** puts the defaults back.
4. Select **Run preview**. After the first run the button reads **Refresh**.

The grid draws at most 5,000 rows. Copying the rows and downloading them as an Excel file take every row that was read. Nothing runs until you select **Run preview**. The full steps, including what a blank required variable does, are in Build a view with the designer and Data model variables.

Explore a model

Explore opens the web report studio over a whole data model, so you can look at every view as a live list and pivot, drill, and total it, with nothing saved until you say so.

1. In the workspace, select the ⋮ menu on the data model's row and choose **View**. (Selecting the model's name opens the editor instead.)
2. The model opens in a tab headed **Exploring** and the model's name. The **Views** list on the left shows each view of the model that has fields.
3. Select a view. It opens on its own sheet as a live list of all its fields. Select another to open it on a new sheet.
4. Work with the data using the Reportworq pane: build a crosstab, drill into a value, filter, or total.
5. Select **Edit model** to jump to the model editor, or **Save as report** to keep what you built as a web report. Choose a name and a folder in the dialog that opens.



- **Nothing is saved until you choose.** The workbook is unsaved at first, and closing the tab asks whether to save or discard.
- **Explore lists every view with fields, including Hidden ones.** A Disabled view does not run; opening it fails with a message that names it. See Choose who sees a view.
- **Large views.** A list reads the rows that fit the sheet (about 1,000) and says so on the report when the source has more.
- **Hierarchies roll up here.** A hierarchy attached to a field shows subtotals in a crosstab. See Build hierarchies.
- **Explore never writes the model.**

Export a source to Excel

Export copies every table of a source into one workbook, for example to hand a snapshot to someone without access or to read the tables side by side. It checks the size first and shows a verdict (Quick, Medium, or Large). Only a source past 10,000,000 cells is refused; a large table is cut to fit a sheet, never refused.

Select **Source**, open the **Sources** tab, and select **Export** on the source's row. The full steps, the workbook's sheets, and the cut-to-fit rules are in Export a whole source to Excel.

Check how views match

On a joined view, three tools read real rows so you can trust a match before a report depends on it.

Tool	What it tells you	Reads
Test match	How many rows of the left view find a match, how many repeat, and which matching options would find more.	Up to 10,000 rows of each view
Discover relationship	The four best pairs of fields to match on, each with its match rate. **Use this** fills in the key.	Up to 1,000 rows of each view
Explain	How the joined view runs, step by step, with each step graded as helping, worth a look, or hurting.	The starting view capped at 100 rows

See Check a relationship on real rows and Read the Explain plan.

Preview a hierarchy

On the **Hierarchies** tab of the **Source** screen, **Preview tree** reads the hierarchy's view as you have it in the editor, unsaved changes included, and shows the members it gives. Under **What the data says** it reports problems, such as a team that rolls up to two divisions. See Build hierarchies.

Ask the assistant to explain

The data model assistant can describe what a source holds in plain words. On the **Source** screen, select **Ask AI**, then **Analyze my data**, or pick **Explain what is in my sources** under **More things I can do** and finish the request, for example "what each source holds". Nothing changes in the model when you ask for an explanation. To use it, an administrator must have set up an AI provider, and the assistant sends a small sample of real rows to that provider. See Ask AI on a data model for what it reads and sends.

Notes and limits

- **Every check reads a sample.** Browsing reads 1,000 rows, **Test match** 10,000, **Discover relationship** 1,000, and **Explain** caps the starting view at 100. None of them proves the whole data.
- **Credentials and reachability are tested when you read.** The editor only checks that a source's connection or file still exists when the model opens. A wrong password shows when you browse, preview, or explore.

- **Reading is not free on a database.** Browsing, previewing, and exporting run queries against the source. Export warns when it would read more than 2,000,000 rows, and suggests running it outside working hours.
- **There is no separate profiling screen.** Counts and distributions come from previewing the view, or from a crosstab in **Explore**.

Related topics

- Data models
- Build a view with the designer
- Join views across sources
- Ask AI on a data model

Ask AI on a data model

reference author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

Ask AI is an assistant built into the data model editor. It works in two places. On the **Source** screen it looks at all of the model's sources and suggests views, joined views, and hierarchies. Inside a view it changes the view you have open: renames fields, adds filters, sets how a variable is asked for, and more. In both places it proposes changes as a list for you to review. Nothing is applied until you choose, and nothing is saved until you select **Save Changes**.

When to use it. Use it to start a model over an unfamiliar source ("suggest views for monthly sales by region, for finance"), or to do tidy-up work in bulk ("drop the prefixes from every field name and hide the id columns"). For a few precise changes on a view you know well, the view's own tabs are often quicker.

Before you start

- An AI provider set up by an administrator in **Settings** > **Integrations**. See Connect an AI provider. With none enabled, the panel says "No AI model is set up. An administrator can add one in Settings, Integrations, AI." and offers nothing to run.
- The assistant uses the AI connection selected for the in-app chat assistant when one was selected and is enabled; otherwise it uses the first enabled AI connection. See The AI assistant.
- Permission to edit the data model.

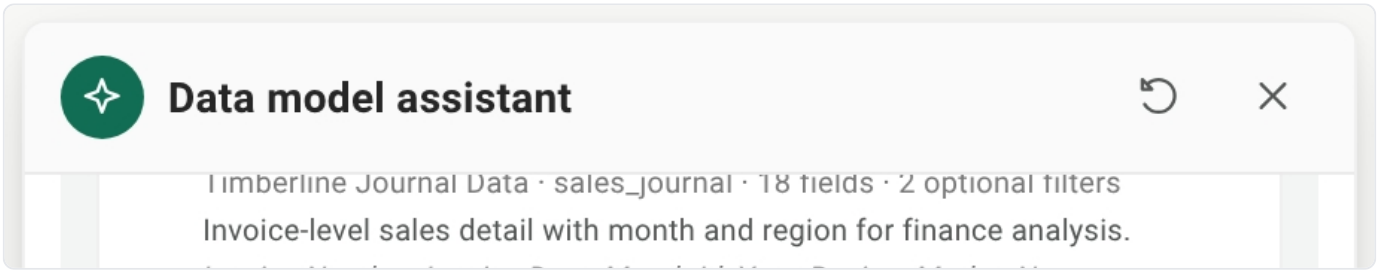
What the assistant reads and sends. It never writes SQL and never changes the model without you. To do its work it does send real information to your AI provider.

- **On the Source screen**, when you select **Analyze my data**, it lists each source's tables (up to 80 per source) with their columns and links, and reads **two sample rows from each of up to 40 tables per source** (12 for a cloud connector, where each read is a call to the service's API). It sends the provider the source names and types, the table and column names and types, the links, the names and fields of your existing views, the names and levels of your hierarchies, and the sample rows, each value cut to 40 characters. The sample reads run against every source, including a connection an administrator has disabled.
- **Inside a view**, it reads no rows. It sends the model and view names, each field's name, type, and the one sample value the field already holds (cut to 40 characters), each filter with its value, each variable with its default and caption, the view's relationships and, for a joined view, its joins, the names and fields of the model's other views, and the conversation so far.

Decide whether that suits the source before you ask.

Ask AI on the Source screen

1. Open the data model and select **Source** in the rail.
2. Select **Ask AI** at the top right. The panel opens over the editor, headed **Data model assistant**. Drag its header to move it; double-click the header to dock it again.
3. Select **Analyze my data**. The assistant reads the sources, explains what the data is and which questions it can answer, and offers options.
4. Say what you want in the box, or pick a starter (see Starter requests). When it proposes views or hierarchies, each has a check box. **Turn that into views** asks it for a concrete set.
5. Untick anything you do not want and select the add button, which names how many views it adds, for example **Add these 4 views**. They are added unsaved.
6. Review them, then select **Save Changes**, or **Discard**.



Invoice Number, Invoice Date, Month Id, Year, Region, Market Name, Customer, Segment, Product Line, Product, Channel, Quantity, Gross Amount…

- [x] **Sales - Monthly by Market**

  Timberline Journal Data · fact_sales · 19 fields · 2 optional filters

  Summarized sales by date, market, product, and channel for finance reporting.

  Date Id, Week Number, Month Id, Month, Quarter, Year, Region, Market Name, Product Name, Segment, Product Line, Channel Name, Version, Units, Gross…

- [x] **Financial - Income Statement by Market**

  Timberline Journal Data · fact_income_statement · 10 fields · 2 optional filters

  Income statement values by account, market, and time.

  Date Id, Month Id, Month, Quarter, Year, Market Name, Region, Version, Account, Value

- [x] **Sales vs P&L - Monthly by Market**

  Joined view · joins Sales - Monthly by Market, Financial - Income Statement by Market

  On Sales - Monthly by Market.Market Name + Date Id + Version = Financial - Income Statement by Market.Market Name + Date Id + Version

  Combined operational sales and income statement values by market, date, and version for finance reconciliation.

+ Add these 4 views

Untick any you don't want in the list above. They're added unsaved.

Suggest views for my reports | Build a joined view | Build hierarchies

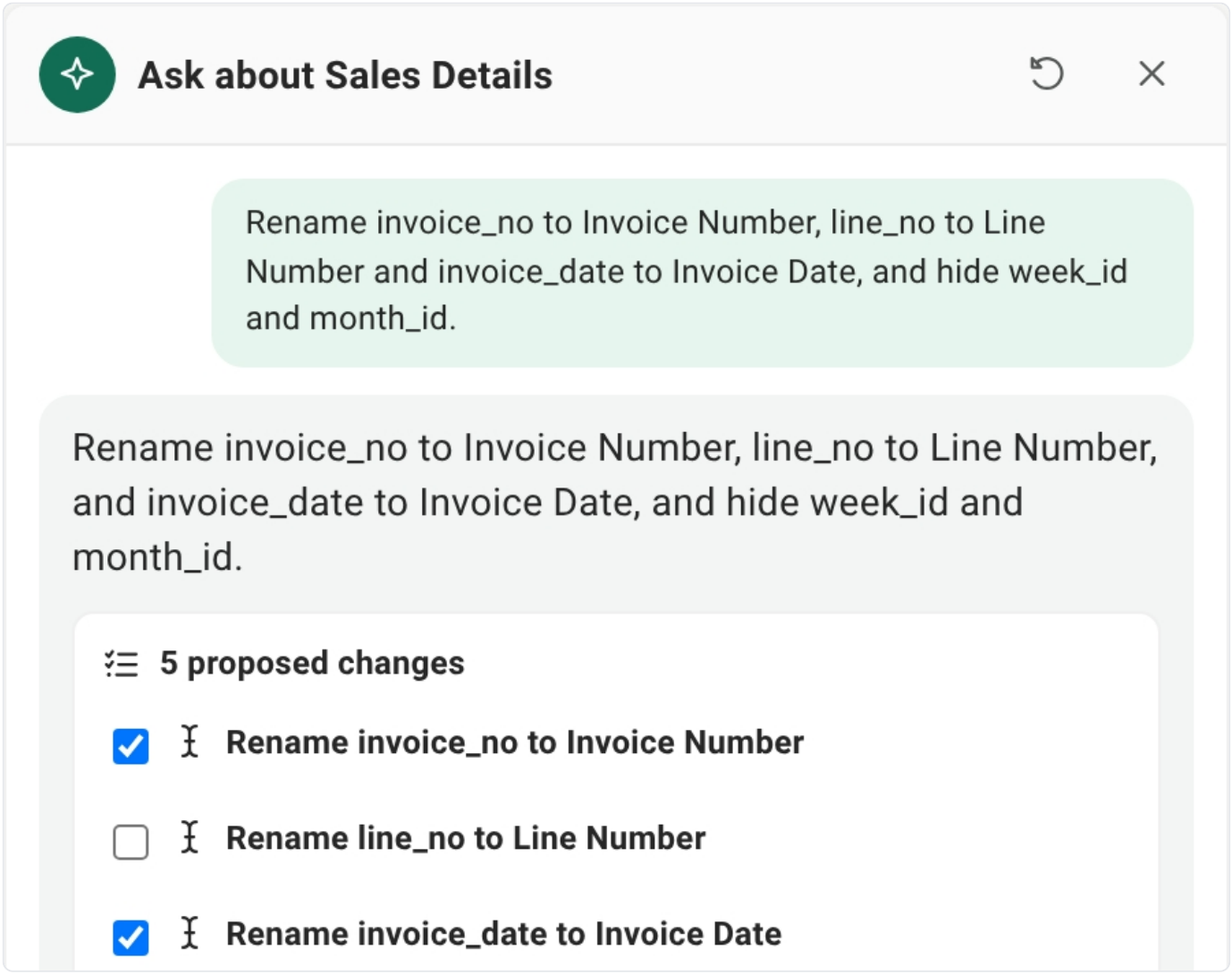
More things I can do

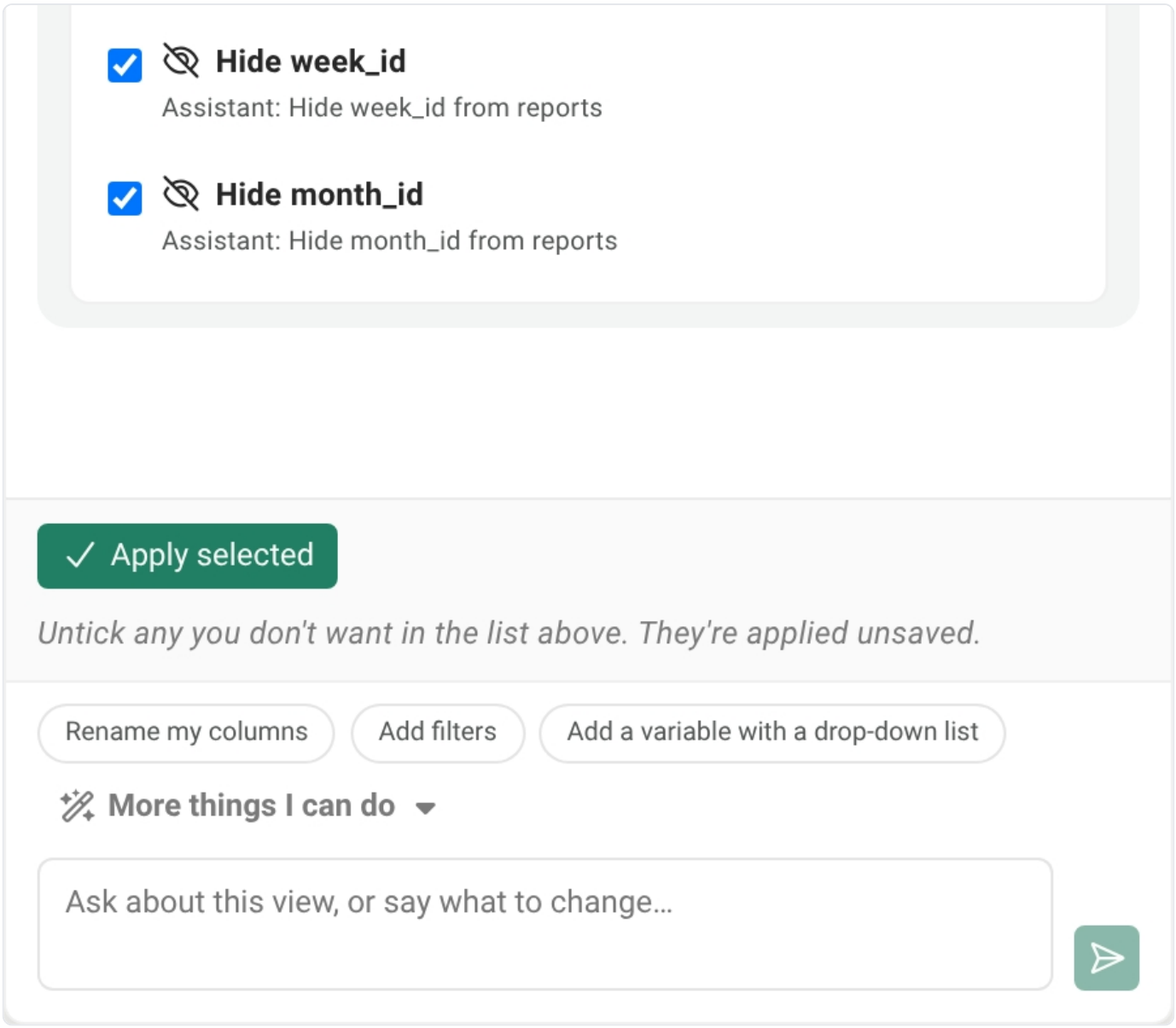
Ask about the data, or say what views you need…

Every proposed view is checked against what the source really has; a table, column, or link that does not exist is left out with a note. A filter the assistant proposes becomes an optional filter variable. **Start over** clears the conversation.

Ask AI inside a view

1. Open a view or a joined view and select **Ask AI** in its header, beside the **Public**, **Hidden**, and **Disabled** choice. The button appears only when an AI provider is set up. The panel is headed "Ask about" and the view's name.
2. Say what to change, for example "Tidy the field names and hide the id columns", or pick a starter and finish it.
3. The assistant lists its changes, each with a check box and a plain line saying what it does, such as "Rename cust_nm to Customer". Clear the check box of any you do not want.
4. Select **Apply selected**. The changes are applied in order and unsaved, exactly as if you had made them on the view's tabs. Each line then says applied, or skipped with the reason, and a summary such as "Applied 5 of 6" heads the list.
5. Review the view, then select **Save Changes** or **Discard**.



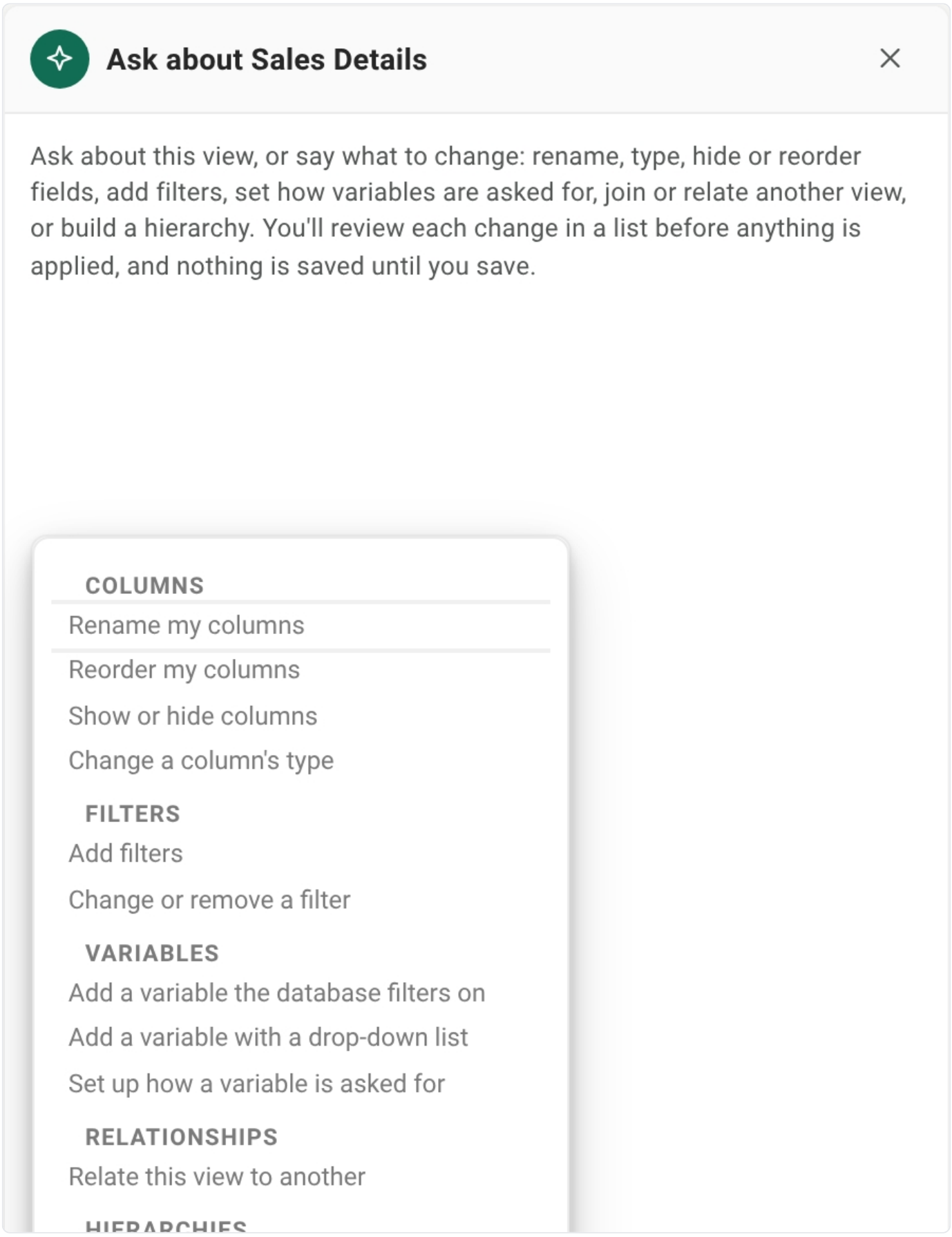


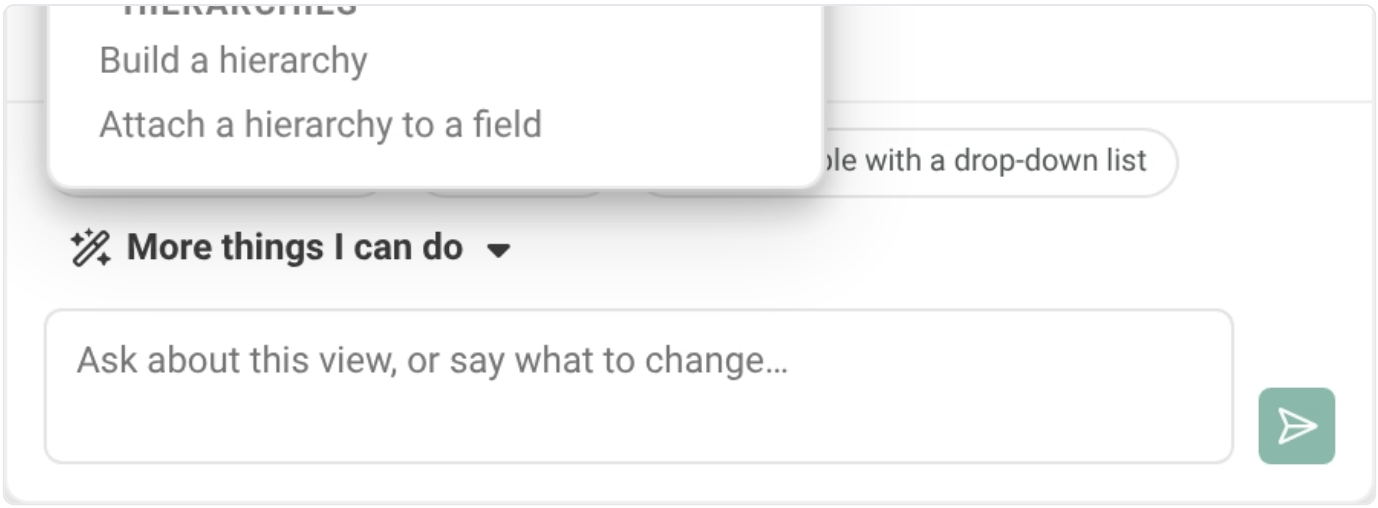
A change that cannot be made is grayed out with its reason and cannot be ticked, so you never apply it by mistake. For example, a change to a variable that the view does not have, or that is pasted into the query as raw text (see Data model variables), is grayed out. A change that cannot be made when you apply it, such as a field the view does not have or a join that would make a view read itself, is skipped and the rest still apply. A change the assistant does not support yet shows as "Not supported yet" and is never applied. Each view keeps its own conversation while the editor is open.

Starter requests

Picking a starter never sends anything. It puts the start of a request in the box with a bracketed part selected, and shows a question above the box with a few example answers. Type over the bracketed part, or select an example, edit as you like, and send. If you had already typed something, it stays and the starter goes on a new line after it. Pressing Enter while the bracketed part is still there selects it again rather than sending half a request.

The first three starters appear as chips under the chat; **More things I can do** opens the whole list. The list works from the keyboard: open it with Enter or Space, move with the arrow keys, press Enter to pick, and Escape or Tab to close it.





On the Source screen

Starters appear once the conversation has started. The chips are marked below.

Group	Starter	What goes in the box	Chip
Views	**Suggest views for my reports**	`Suggest views for: [what your reports cover]`	Yes
Joins	**Build a joined view**	`Build a joined view of: [which data belongs together]`	Yes
Joins	**Find where my sources could join**	`Find where my sources could join: [which sources or tables]`	No
Hierarchies	**Build hierarchies**	`Build hierarchies for: [which roll-ups]`	Yes
Understand the data	**Explain what is in my sources**	`Explain: [what you want to know about the sources]`	No
Understand the data	**Tell it our naming standards**	`We name views and fields like this: [your naming standard]`	No

Views proposed from now on follow the naming standard you give. On the Source screen the assistant does not change a view you already have: to do that, open the view and use **Ask AI** there.

Inside a view

Group	Starter	What goes in the box	Offered on
Columns	**Rename my columns** (chip)	`Rename my columns: [how you want them named]`	Every view
Columns	**Reorder my columns**	`Put my columns in this order: [which come first]`	Every view
Columns	**Show or hide columns**	`Show or hide columns: [which ones]`	Every view
Columns	**Change a column's type**	`Change column types: [which columns, and to what]`	Every view
Filters	**Add filters** (chip)	`Add filters: [which rows to keep]`	Every view
Filters	**Change or remove a filter**	`Change this view's filters: [which filter, and how]`	Every view
Variables	**Add a variable the database filters on**	`Add a variable the database filters on: [which field, and how each report picks it]`	A view of one source
Variables	**Add a variable with a drop-down list** (chip; on a joined view it is in the list, not a chip)	`Add a variable with a drop-down list: [which field, and which view lists the choices]`	Every view
Variables	**Set up how a variable is asked for**	`Set up how a variable is asked for: [which variable, and its default, caption, help or whether it is required]`	Every view
Joins	**Join another view** (chip)	`Join another view: [which view, and on which fields]`	A joined view
Joins	**Change how rows match**	`Change how a join matches rows: [which join, and how]`	A joined view
Joins	**Remove a join**	`Remove a join: [which view to stop joining]`	A joined view
Relationships	**Relate this view to another**	`Relate this view to another: [which view, and on which fields]`	Every view
Hierarchies	**Build a hierarchy**	`Build a hierarchy: [which levels roll up]`	A view of one source

Group	Starter	What goes in the box	Offered on
Hierarchies	**Attach a hierarchy to a field**	`Attach a hierarchy to a field: [which field, and which hierarchy]`	A view of one source

The three chips are **Rename my columns**, **Add filters**, and **Add a variable with a drop-down list**; on a joined view the third is **Join another view**.

Example answers the starters offer include "Title Case, with spaces between words", "hide the internal ids and audit columns", "a date range each report picks", "region, with the choices from a view of regions", and "default the start date to 2026-01-01".

What it cannot do

Ask for something outside the lists above and the assistant says plainly that it cannot do it here and proposes nothing.

- **It never writes SQL** or changes a view's query. Say what to change, and change it on the view's **Design** tab. A suggestion that comes back with SQL in it is left out with a note.
- **It does not add calculated or literal fields.** A field must map to a source column.
- **It does not change a source** or save for you.
- **It never sets a variable that is pasted into the query as raw text.** That setting stays with you.
- **On the Source screen it does not change an existing view.**
- **A hierarchy it proposes is not checked against the data.** It does not compare a hierarchy's values with the field's values, so open it on the **Hierarchies** tab and select **Preview tree** before you save. See Build hierarchies.

Notes and limits

- **It needs an AI provider**, and its answers depend on that provider's model. Review every change.
- **Customized instructions.** An administrator can tune the assistant's instructions in **Settings** > **AI Prompts** (see The AI prompt store). A customized version keeps its text and does not gain the list of starters until it is reset to the default.
- **Defaults it sets** are filled into the query by the same safe rules as a value a reader types. See Data model variables.
- **Samples, not the whole data.** What it learns comes from sample values, not from every row.

> **Going deeper.** The full account of each change type, what a reply looks like when nothing can be made, and the Source screen's proposals is in Data models and Ask AI inside a view.

Related topics

- Data models
- Data model variables
- Discover your data
- AI in Reportworq

Build a data model from a workspace file

how-to author administrator

> **Early release.** This feature is in early release, so its screens and behavior may change. Tell us what you think with **Send feedback**. See Early release features and Send feedback.

A data model does not have to bind to a connection in **Settings** > **Integrations**. When the file it needs to read, an Excel workbook or a SQLite database, already lives in the same workspace as the model, point the model straight at that file. No connection is added to Integrations, so dropping many workbooks or databases into a workspace folder never clutters the datasource list.

A workspace file is one source among possibly several: a model can read any number of workspace files and Integrations connections side by side, and each view draws from one of them. This is the recommended way to model a workbook or database that belongs to one workspace and one purpose. If instead the file should be shared as a connection other workspaces or other models can also use, see Connect Excel and Connect SQLite for **Configure as data source**, the admin path that creates a real Integrations connection from the same file.

Before you begin

- The file is an Excel workbook (`.xlsx` , `.xlsm`) or a SQLite database (`.db` , `.sqlite` , `.sqlite3`), uploaded into the **same workspace** where you are building the data model.
- SQLite databases must be an allowed upload type in this workspace. On an existing install, an administrator may need to add the **Databases** file-type category to the workspace's upload allow-list first, see Notes and limits.
- You have permission to create a data model in the workspace (the same permission **New** > **Data model** requires).
- The workbook or database follows the same layout rules as any Excel or SQLite source: a header row of column names for a worksheet table, or ordinary tables in the database. See Connect Excel and Connect SQLite for the query modes and what each driver exposes.

Two ways to start

From the file, in the workspace browser

1. In the workspace's file tree, find the Excel workbook or SQLite database.
2. Open its ⋮ (more actions) menu and choose **Create data model**.
3. Reportworq creates a data model in the same folder, named after the file, already bound to it, and opens it in the editor.

From an existing data model's Sources tab

1. Open the data model (new or existing), select **Source** in the rail, then the **Sources** tab.
2. Select **Add workspace file…** (on a model with no source yet, this is also how you start).
3. Pick an Excel workbook or SQLite database from this workspace's files. The picker lists only files you can see, filtered to Excel and SQLite.

The file appears in the list of sources, named by its path, with how many views read it. **Change…** on its row picks a different file, and **Remove** takes it out (the confirmation names the views that read it). To start a view on the file, use **+ Add** on the rail's Source node and choose the file.

Look inside the file first

To see what a workbook holds before you model it, select its name in the workspace. It opens full screen in a read-only spreadsheet preview, with its sheets as tabs along the bottom; **Close** or the **Esc** key returns you to the folder. The preview shows the values stored in the file. See Workspace files.

Build the views

Once the file is picked, start a view with **+ Add** in the rail and choose the file. The rest of the data model editor works exactly as it does for a connection in Integrations:

- **Start with a table** on the **Design** canvas lists the workbook's worksheets (and named ranges), or the database's tables. See Build a view with the designer.
- **Write SQL instead** works the same way it does against any relational source, for example:

  ```
  SELECT Region, Product, Sales FROM Sales
  ```

- Fields, **Preview**, filters, and variables all behave as documented in Data models.

Portability

The data model stores the file's path **inside its own workspace**, never a workspace id. That is what makes the model portable: when you export the workspace and import it elsewhere, the model resolves the same path in the new workspace and keeps working, with nothing to reconfigure.

The trade-off is the same one any relative path carries: if the file is later **renamed or moved** to a different folder, the model's reference breaks. Open the model's **Sources** tab and use **Change…** on the file's row to pick its new location.

How your data stays secure

- **No datasource connection is created or stored** for a workspace-file model. The connection is built in memory each time the model is used, previewed, imported, run in a job, or bound to a report, and discarded afterward.
- **A file source is treated like a connection: if you can see the data model, you can report on it.** The picker lists only the files you can see, so you need to see a file to pick it. After that, anyone who can see the data model can preview it, report on it and run jobs from it, without access to the file itself, just as they would for a model on an Integrations connection. The model can only ever read files in its **own** workspace, never another workspace's file, even if you know its path.
- **Scheduled jobs and campaigns do not depend on any one person's access.** A distribution job or contribution campaign that runs on a schedule, or through the background job runner, reads the file the same way, so it keeps running even if the author who built the model later loses visibility into the file.
- **A view's SQL can name only the file's own sheets or tables.** A SQL view over an Excel workbook reads that workbook's sheets, and a view over a SQLite database reads that database's tables. A query that names another file (for example a sheet qualified by a file path, such as `[C:/Finance/Salaries.xlsx].[Sheet1]`), reads a driver system table (`sys_...`), attaches another SQLite database, or does anything other than read with `SELECT` is refused before it runs, with the message "This query names another file (…). A data model reads only its own workspace file." Sheet names with spaces, column headers with `/` or `:`, cell ranges such as `[Data#A1:C20]`, and comments are unaffected. One limit: in SQL you write yourself, a sheet name or alias that contains a dot can't be used in front of a column (as in `[my.sheet].[Region]`), because it reads like a file name; give it an alias without a dot.
- **The connection is always read-only.** Reportworq opens the file for reading only, whichever driver is used underneath. A write-back operation against it, such as a SQL export or `RWSQLUPDATE` targeting the file, fails instead of silently doing nothing. To change the data, edit the file itself and upload the new version into the

workspace.
- The server keeps a private, working copy of the file for the drivers to read from. That copy refreshes automatically the next time the model is used after you upload a new version of the file, and it is removed when the file or the workspace is deleted.

If the source breaks

The editor never fails to open just because a file source is broken. If the file was renamed, moved or deleted, that source's row on the **Sources** tab shows a problem, and the rest of the editor still opens normally. Use **Change…** on the row to pick the file again, either the same file at its new location, or a replacement.

What happens on import

When you import a workspace (or a data model) that was exported from elsewhere:

- The model keeps its source pointer, whether or not you can see the file. A model is not stopped from importing because of what the importer can see in the target workspace.
- If the source file **hasn't been uploaded yet**, the model starts working as soon as you upload the file to the same path. Until then the model's **Sources** tab shows the problem.
- Either way, the imported model is treated like a fresh copy: it gets its own new internal identifier, so it never accidentally shares its source connection with the model it was exported from.

Notes and limits

- **SQLite uploads may need an admin step first.** A fresh Reportworq install permits `.db` / `.sqlite` / `.sqlite3` uploads out of the box. An install upgraded from an earlier version keeps its existing upload allow-list, so a `.db` upload may be refused with "These file types are not permitted…" until an administrator adds the **Databases** category. That setting lives in **Settings** > **Security** > **Workspaces** > *the workspace* > **Workspace Files**, under **Allowed File Types**, use the **+ Databases** quick-add chip.
- **Preview needs a saved model, under one specific setting.** If **Settings** > **Performance** is configured to run reporting queries out of a separate process, previewing a file you just added but have **not yet saved** will not find it, the out-of-process worker only sees a model's saved source. Save the model once after adding the file, then preview.
- Excel workbook and SQLite rules otherwise match the connector pages: see Connect Excel and Connect SQLite for worksheet/table layout, query modes, and

driver behavior.

Related pages

- Data models for curating fields, filters, variables and hierarchies, and for choosing who sees a view (Public, Hidden or Disabled), once the source is added.
- Connect Excel and Connect SQLite for the same files used as a shared Integrations connection instead (**Configure as data source**), and for the query modes both paths share.
- Connect a data source for the general Integrations add-and-test flow.